

Tru64 UNIX

ユーザーズ・ガイド

Part Number: AA-RK3LC-TE

2002 年 11 月

ソフトウェア・バージョン: Tru64 UNIX Version 5.1B

本書では、Compaq Tru64™ UNIX Version 5.1B オペレーティング・システムにおけるコマンドおよびシェルの使用方法を説明するとともに、他のネットワーク・ユーザとの通信の概要を紹介します。また、ファイル・システムの概要についても説明しています。

日本ヒューレット・パッカード株式会社

© 2002 日本ヒューレット・パッカート株式会社

本書の著作権は日本ヒューレット・パッカート株式会社が保有しており、本書中の解説および図、表は日本ヒューレット・パッカートの文書による許可なしに、その全体または一部を、いかなる場合にも再版あるいは複製することを禁じます。

また、本書に記載されている事項は、予告なく変更されることがありますので、あらかじめご承知おきください。万一、本書の記述に誤りがあった場合でも、弊社は一切その責任を負いかねます。

日本ヒューレット・パッカートは、弊社または弊社の指定する会社から納入された機器以外の機器で対象ソフトウェアを使用した場合、その性能あるいは信頼性について一切責任を負いかねます。

本書で解説するソフトウェア(対象ソフトウェア)は、所定のライセンス契約が締結された場合に限り、その使用あるいは複製が許可されます。

COMPAQ, Compaq ロゴ, Digital ロゴは U.S. Patent and Trademark Office に登録されています。以下は、Digital Equipment Corporation の商標です: ALL-IN-1, Alpha AXP, AlphaGeneration, AlphaServer, AltaVista, ATMworks, AXP, Bookreader, CDA, DDIS, DEC, DEC Ada, DECevent, DEC Fortran, DEC FUSE, DECnet, DECstation, DECsystem, DECterm, DECUS, DECwindows, DTIF, Massbus, MicroVAX, OpenVMS, POLYCENTER, PrintServer, Q-bus, StorageWorks, Tru64, TruCluster, TURBOchannel, ULTRIX, ULTRIX Mail Connection, ULTRIX Worksystem Software, UNIBUS, VAX, VAXstation, VMS, XUI。このドキュメントに記載されているその他の会社名および製品名は、各社の商標または登録商標です。

NFS は米国 Sun Microsystems 社の登録商標です。Open Software Foundation, OSF, OSF/1, OSF/Motif, および Motif は Open Software Foundation 社の商標です。UNIX は The Open Group の米国ならびに他の国における登録商標です。The Open Group は The Open Group の米国ならびに他の国における商標です。

このドキュメントに記載されているその他の会社名および製品名は、各社の商標または登録商標です。

原典: Command and Shell User's Guide (AA-RH91C-TE)
Copyright ©2002 Hewlett-Packard Company

目次

まえがき

1 基本操作

1.1	ログイン	1-2
1.1.1	ログインの制限	1-6
1.2	ログアウト	1-7
1.3	コマンドの使用	1-7
1.4	コマンド実行の中止	1-9
1.5	パスワードの設定	1-9
1.5.1	パスワードの期限	1-10
1.5.2	パスワードに関するガイドライン	1-11
1.5.3	ユーザによるパスワードの決定	1-12
1.5.3.1	システムが生成するパスワードの選択	1-13
1.5.3.2	ユーザ選択パスワード	1-13
1.5.4	パスワード設定の手順	1-15
1.6	ヘルプの利用	1-16
1.6.1	オンライン・リファレンス・ページの表示と印刷 (man コマンド)	1-17
1.6.2	キーワードを使用したコマンドの検索	1-19

2 ファイルとディレクトリの概要

2.1	テキスト・エディタの概要	2-1
2.2	vi テキスト・エディタによるサンプル・ファイルの作成	2-2
2.3	ファイル, ディレクトリ, およびパス名	2-6
2.3.1	ファイルおよびファイル名	2-6

2.3.2	ディレクトリおよびサブディレクトリ	2-8
2.3.3	現在のディレクトリ (作業ディレクトリ) の名前の表示 (pwd コマンド)	2-9
2.3.4	ツリー構造ファイル・システムおよびパス名	2-10
2.4	パターン照合によるファイル指定	2-14

3 ファイルの管理

3.1	ファイルのリスト (ls コマンド)	3-2
3.1.1	現在のディレクトリの内容のリスト	3-2
3.1.2	他のディレクトリの内容のリスト	3-3
3.1.3	ls コマンドのフラグ	3-4
3.2	ファイル内容の表示	3-6
3.2.1	フォーマットを伴わないファイル表示 (pg, more, cat コマンド)	3-6
3.2.2	フォーマットを伴うファイル表示 (pr コマンド)	3-8
3.3	ファイルの印刷 (lpr, lpq, lprm コマンド)	3-12
3.4	ファイルのリンク (ln コマンド)	3-16
3.4.1	ハード・リンクとソフト・リンク	3-16
3.4.2	リンクとファイル・システム	3-17
3.4.3	リンクの使用	3-18
3.4.4	リンクの仕組み (ファイル名とファイル通し番号)	3-19
3.4.5	リンクの削除	3-20
3.5	ファイルのコピー (cp コマンド)	3-22
3.5.1	現在のディレクトリ内でのファイルのコピー	3-23
3.5.2	別のディレクトリへのファイルのコピー	3-24
3.6	ファイル名の変更およびファイルの移動 (mv コマンド)	3-25
3.6.1	ファイル名の変更	3-26
3.6.2	別のディレクトリへのファイルの移動	3-27
3.7	ファイルの比較 (diff コマンド)	3-28

3.8	ファイル内容のソート (sort コマンド)	3-30
3.9	ファイルの削除 (rm コマンド)	3-31
3.9.1	単一のファイルの削除	3-31
3.9.2	複数のファイルの削除 (パターン照合)	3-32
3.10	ファイル・タイプの参照 (file コマンド)	3-34
4	ディレクトリの管理	
4.1	ディレクトリの作成 (mkdir コマンド)	4-2
4.2	ディレクトリの変更 (cd コマンド)	4-4
4.2.1	現在のディレクトリの変更	4-4
4.2.2	相対パス名の使用	4-5
4.2.3	シンボリック・リンクを介したディレクトリへのアクセス	4-7
4.3	ディレクトリの表示 (ls -F コマンド)	4-8
4.4	ディレクトリのコピー (cp コマンド)	4-9
4.5	ディレクトリ名の変更 (mv コマンド)	4-10
4.6	ディレクトリの削除 (rmdir コマンド)	4-11
4.6.1	空ディレクトリの削除	4-12
4.6.2	複数のディレクトリの削除	4-13
4.6.3	現在のディレクトリの削除	4-13
4.6.4	ファイルとディレクトリの同時削除 (rm -r コマンド)	4-14
5	ファイルとディレクトリへのアクセス制御	
5.1	パスワード・セキュリティ・ファイルとグループ・セキュリティ・ファイル	5-2
5.1.1	/etc/passwd ファイル	5-2
5.1.2	/etc/group ファイル	5-4
5.2	ファイルとディレクトリの保護	5-5
5.3	ファイル許可およびディレクトリ許可の表示 (ls コマンド)	5-7
5.4	ファイルおよびディレクトリ許可の設定 (chmod コマンド) ...	5-9

5.4.1	英字と演算記号による許可の指定	5-10
5.4.1.1	ファイル許可の変更	5-11
5.4.1.2	ディレクトリ許可の変更	5-12
5.4.1.3	パターン照合文字の使用	5-12
5.4.1.4	絶対許可の設定	5-13
5.4.2	8進数による許可の指定	5-13
5.5	ユーザ・マスクを使用した省略時の許可の設定	5-15
5.5.1	umask の設定	5-18
5.6	ファイルにアクセスするための ID の変更	5-20
5.7	スーパーユーザの概念	5-21
5.8	所有者とグループの変更 (chown および chgrp コマンド)	5-23
5.9	セキュリティに関する追加情報	5-24

6 プロセスの使用

6.1	プログラムとプロセス	6-1
6.2	標準入力，出力，およびエラー	6-2
6.2.1	入力と出力のリダイレクト	6-2
6.2.1.1	ファイルからの入力の読み取り	6-3
6.2.1.2	出力のリダイレクト	6-4
6.2.2	標準エラーのファイルへのリダイレクト	6-5
6.2.2.1	Bourne シェル，Korn シェル，および POSIX シェルのエラー・リダイレクション	6-5
6.2.2.2	C シェルのエラー・リダイレクション	6-6
6.2.3	標準エラーと標準出力の両方のリダイレクト	6-6
6.3	複数プロセスの同時実行	6-7
6.3.1	フォアグラウンド・プロセスの実行	6-8
6.3.2	バックグラウンド・プロセスの実行	6-8
6.4	プロセスのモニタと終了	6-10
6.4.1	プロセス状態のチェック	6-10

6.4.1.1	ps コマンド	6-10
6.4.1.2	jobs コマンド	6-12
6.4.2	フォアグラウンド・プロセスの取り消し (Ctrl/C)	6-13
6.4.3	バックグラウンド・プロセスの取り消し (kill コマンド) ...	6-14
6.4.4	フォアグラウンド・プロセスの中断と再開 (C シェルのみ)	6-15
6.5	ユーザとユーザのプロセスに関する情報の表示	6-16

7 シェルの概要

7.1	シェルの目的	7-1
7.2	C シェル, Bourne シェル, Korn シェル, および POSIX シェルの機能の要約	7-2
7.2.1	C シェルおよび Korn シェルまたは POSIX シェルの機能に関する詳細情報	7-3
7.2.2	制限付き Bourne シェル	7-5
7.3	シェルの変更	7-5
7.3.1	実行しているシェルの確認	7-6
7.3.2	シェルの一時的変更	7-6
7.3.3	シェルの永久的変更	7-7
7.4	コマンド入力支援機能	7-8
7.4.1	複数コマンドとコマンド・リストの使用	7-8
7.4.1.1	セミコロン (;) を使用したコマンドの順次実行	7-8
7.4.1.2	条件付きのコマンド実行	7-9
7.4.2	パイプとフィルタの使用	7-10
7.4.3	コマンドのグループ化	7-12
7.4.3.1	カッコ () の使用	7-12
7.4.3.2	中カッコ { } の使用	7-13
7.4.4	引用	7-13
7.4.4.1	バックスラッシュの使用 (\)	7-14
7.4.4.2	一重引用符の使用 (')	7-14

7.4.4.3	二重引用符の使用 (" ")	7-15
7.5	シェル環境	7-15
7.5.1	ログイン・プログラム	7-15
7.5.2	環境変数	7-16
7.5.3	シェル変数	7-18
7.6	ログイン・スクリプトとユーザ環境	7-19
7.7	変数の使用	7-21
7.7.1	変数の設定	7-22
7.7.1.1	Bourne シェル , Korn シェル , および POSIX シェルの シェル変数	7-22
7.7.1.2	C シェルの変数	7-24
7.7.1.3	すべてのシェルにおける変数の設定	7-24
7.7.2	変数の参照 (パラメータ置換)	7-25
7.7.3	変数値の表示	7-25
7.7.4	変数値のクリア	7-26
7.8	シェルによるコマンドのを見つけ方	7-27
7.9	ログアウト・スクリプトの使用	7-28
7.9.1	ログアウト・スクリプトとシェル	7-29
7.9.2	.logout ファイルの例	7-29
7.10	シェル・プロシージャの使用 (スクリプト)	7-30
7.10.1	シェル・プロシージャの作成と実行	7-31
7.10.2	実行シェルの指定	7-32

8 シェルの機能

8.1	C , Bourne , Korn , および POSIX シェルの機能の比較	8-1
8.2	C シェルの機能	8-2
8.2.1	.cshrc スクリプトおよび .login スクリプトの例	8-2
8.2.2	メタキャラクタ	8-5
8.2.3	コマンド・ヒストリ	8-7

8.2.4	ファイル名補完	8-9
8.2.5	別名	8-9
8.2.6	組み込み変数	8-11
8.2.7	組み込みコマンド	8-12
8.3	Bourne シェルの機能	8-13
8.3.1	.profile ログイン・スクリプトの例	8-14
8.3.2	メタキャラクタ	8-15
8.3.3	組み込み変数	8-17
8.3.4	組み込みコマンド	8-18
8.4	Korn または POSIX シェルの機能	8-19
8.4.1	.profile および .kshrc ログイン・スクリプトの例	8-19
8.4.2	メタキャラクタ	8-22
8.4.3	コマンド・ヒストリ	8-24
8.4.4	fc コマンドを使用したコマンド行編集	8-26
8.4.4.1	コマンド行編集の例	8-27
8.4.5	ファイル名補完	8-28
8.4.6	別名	8-29
8.4.7	組み込み変数	8-31
8.4.8	組み込みコマンド	8-33
9	System V 動作環境の使用	
9.1	環境の設定	9-2
9.2	環境設定スクリプトによる PATH 環境変数の設定	9-3
9.3	シェル・スクリプトの互換性	9-5
9.4	System V のコマンド概略	9-5
10	ネットワーク・ユーザおよびホストに関する情報の収集	
10.1	ローカル・ホスト上のユーザの確認	10-2
10.2	ネットワーク・ユーザに関する情報の取得	10-3

10.2.1	特定のユーザに関する情報の取得	10-3
10.2.2	リモート・ホスト上のユーザに関する情報の取得	10-4
10.2.3	リモート・ホスト上の各ユーザに関する情報の取得	10-4
10.2.4	finger コマンドからの出力のカスタマイズ	10-5
10.3	リモート・ホストおよびユーザに関する情報の取得	10-5
10.4	リモート・ホスト上のユーザに関する情報の取得	10-8
10.5	リモート・ホストがオンラインであるかどうかの判定	10-10
 11 メッセージの送受信		
11.1	メール・メッセージのアドレス指定	11-2
11.2	mailx を使用したメール・メッセージの送信	11-3
11.2.1	メッセージの編集	11-4
11.2.2	メッセージの打ち切り	11-5
11.2.2.1	Ctrl/C を使用したメッセージの打ち切り	11-5
11.2.2.2	エスケープ・コマンドによるメッセージの打ち切り ..	11-6
11.2.3	メッセージへのファイルの取り込み	11-6
11.3	メール・メッセージの受信	11-8
11.3.1	メッセージの削除	11-11
11.3.2	メッセージへの応答	11-11
11.3.3	メッセージの保存	11-13
11.3.3.1	ファイルへのメッセージの保存	11-13
11.3.3.2	フォルダへのメッセージの保存	11-14
11.3.4	メッセージの転送	11-16
11.4	mailx からのヘルプ情報の参照	11-17
11.5	メールの終了	11-17
11.6	メール・セッションのカスタマイズ	11-18
11.6.1	メールの別名の作成	11-19
11.6.2	メール変数の設定	11-20
11.7	MH プログラム	11-21

11.8	write を使用したメッセージの送受信	11-25
11.9	talk を使用したメッセージの送受信	11-28
12	他のホストへのファイルのコピー	
12.1	ローカル・ホストとリモート・ホスト間でのファイルのコピー	12-1
12.1.1	rcp を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー	12-2
12.1.2	ftp を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー	12-3
12.1.3	mailx を使用したローカル・ホストとリモート・ホスト間での ASCII ファイルのコピー	12-10
12.1.4	write を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー	12-11
12.2	ローカル・ホストとリモート・ホスト間でのディレクトリのコピー	12-11
12.3	2 つのリモート・ホスト間でのファイルのコピー	12-12
13	リモート・ホストでの作業	
13.1	rlogin を使用したリモート・ホストへのログイン	13-1
13.2	rsh を使用したリモート・ホスト上でのコマンドの実行	13-3
13.3	telnet を使用したリモート・ホストへのログイン	13-3
14	UUCP ネットワーク・コマンド	
14.1	UUCP パス名規約	14-1
14.2	UUCP をサポートするホストの検索	14-2
14.3	リモート・ホストへの接続	14-3
14.3.1	cu を使用したリモート・ホストへの接続	14-3
14.3.1.1	cu を使用した名前によるリモート・ホストへの接続	14-4
14.3.1.2	cu を使用した直接接続のリモート・ホストの指定	14-5

14.3.1.3	cu を使用した電話回線によるリモート・ホストへの接続	14-5
14.3.1.4	ローカルの cu コマンド	14-7
14.3.1.5	cu を使用したローカル・コンピュータの複数のリモート・コンピュータへの接続	14-9
14.3.2	tip を使用したリモート・ホストへの接続	14-10
14.3.2.1	tip を使用した名前によるリモート・ホストへの接続	14-11
14.3.2.2	tip を使用した電話回線によるリモート・ホストへの接続	14-12
14.3.2.3	ローカルの tip コマンド	14-14
14.3.2.4	tip コマンドを使用したローカル・ホストから複数のリモート・ホストへの接続	14-16
14.3.3	ct を使用したモデムによるリモート端末への接続	14-18
14.4	uux を使用したリモート・ホスト上でのコマンドの実行	14-21
14.4.1	Bourne シェル, Korn シェル, または POSIX シェルからの uux の使用	14-23
14.4.2	C シェルからの uux の使用	14-23
14.4.3	その他の uux の機能およびヒント	14-23
14.5	UUCP を使用したファイルの送受信	14-25
14.5.1	Bourne シェル, Korn シェル, および POSIX シェルでの UUCP を使用したファイルのコピー	14-26
14.5.2	C シェルでの UUCP を使用したファイルのコピー	14-27
14.6	uuto と uupick を使用したファイルのコピー	14-28
14.7	uuto を使用したファイルのローカル送信	14-30
14.8	UUCP ユーティリティのジョブ状態の表示	14-31
14.8.1	uustat コマンド	14-31
14.8.1.1	uustat のオプションを使用した保留状態のキュー出力の表示	14-32
14.8.1.2	uustat オプションを使用した現在のキュー出力の表示	14-34
14.8.2	uulog コマンドを使用した UUCP ログ・ファイルの表示	14-35

14.8.3	UUCP 状態のモニタ	14-37
A vi 入門		
A.1	入門	A-2
A.1.1	ファイルの作成	A-2
A.1.2	既存のファイルのオープン	A-5
A.1.3	ファイルの保存と vi の終了	A-5
A.1.4	ファイル内での移動	A-7
A.1.4.1	カーソルの移動 (上, 下, 左, 右)	A-7
A.1.4.2	カーソルの移動 (語, 行, 文, パラグラフ)	A-8
A.1.4.3	ファイル内でのカーソルの移動とスクロール	A-9
A.1.4.4	移動コマンドのまとめ	A-9
A.1.5	新しいテキストの入力	A-10
A.1.6	テキストの編集	A-14
A.1.6.1	単語の削除	A-14
A.1.6.2	行の削除	A-15
A.1.6.3	テキストの変更	A-15
A.1.6.4	テキスト編集コマンドのまとめ	A-16
A.1.7	コマンドの取り消し	A-16
A.1.8	編集の終了	A-17
A.2	高度なテクニックの使用	A-17
A.2.1	文字列の検索	A-17
A.2.2	テキストの削除と移動	A-18
A.2.3	テキストのコピー	A-19
A.2.4	その他の vi の機能	A-19
A.3	ex コマンドの利用	A-20
A.3.1	置換	A-21
A.3.2	ファイル全体または一部の書き込み	A-23
A.3.3	テキスト・ブロックの削除	A-24

A.3.4	ユーザ環境のカスタマイズ	A-24
A.3.5	カスタマイズした内容の保存	A-26

B ed によるファイルの作成および編集

B.1	テキスト・ファイルと編集バッファ	B-1
B.2	テキスト・ファイルの作成と保存	B-2
B.2.1	ed プログラムの起動	B-2
B.2.2	テキストの入力 - a (追加) サブコマンド	B-3
B.2.3	テキストの表示 - p (プリント) サブコマンド	B-4
B.2.4	テキストの保存 - w (書き込み) サブコマンド	B-5
B.2.4.1	同じファイル名でのテキストの保存	B-5
B.2.4.2	別のファイル名でのテキストの保存	B-5
B.2.4.3	ファイルの一部の保存	B-6
B.2.5	ed プログラムの終了 - q (終了) サブコマンド	B-6
B.3	ファイルの編集バッファへのロード	B-7
B.3.1	ed (編集) コマンド	B-8
B.3.2	e (編集) サブコマンドの使用	B-8
B.3.3	r (読み取り) サブコマンドの使用	B-9
B.4	現在の行の表示と変更	B-10
B.4.1	バッファ内での位置付け	B-11
B.4.2	バッファ内での位置の変更	B-12
B.5	テキストの探索	B-13
B.5.1	バッファ内での順方向探索	B-14
B.5.2	バッファ内での逆方向探索	B-14
B.5.3	探索方向の変更	B-15
B.6	置換 - s (置換) サブコマンド	B-15
B.6.1	現在の行での置換	B-16
B.6.2	特定の行での置換	B-16
B.6.3	複数行での置換	B-17

B.6.4	すべての該当文字列の置換	B-17
B.6.5	文字の削除	B-18
B.6.6	行頭と行末での置換	B-18
B.6.7	文脈探索の使用	B-19
B.7	行の削除 - d (削除) サブコマンド	B-19
B.7.1	現在の行の削除	B-20
B.7.2	特定の行の削除	B-20
B.7.3	複数行の削除	B-21
B.8	テキストの移動 - m (移動) サブコマンド	B-21
B.9	テキスト行の変更 - c (変更) サブコマンド	B-22
B.9.1	1 行のみの変更	B-23
B.9.2	複数行の変更	B-23
B.10	テキストの挿入 - i (挿入) サブコマンド	B-24
B.10.1	行番号の使用	B-24
B.10.2	文脈探索の使用	B-25
B.11	行のコピー - t (転送) サブコマンド	B-26
B.12	ed からのシステム・コマンドの使用	B-27
B.13	ed プログラムの終了	B-27

C 国際化機能の使用

C.1	ロケール	C-1
C.2	ロケールがデータの処理と表示に与える影響	C-3
C.2.1	照合	C-3
C.2.2	日付/時刻フォーマット	C-4
C.2.3	数字と通貨のフォーマット	C-5
C.2.4	メッセージ	C-6
C.2.5	Yes/No プロンプト	C-6
C.3	ロケールが設定されているかどうかの判断	C-6
C.4	ロケールの設定	C-7

C.4.1	ロケール・カテゴリ	C-9
C.4.2	ロケール設定の制約	C-11
C.4.2.1	ロケール設定が有効でない	C-11
C.4.2.2	ファイル・データがロケールに対応していない	C-11
C.4.2.3	LC_ALL の設定は他のロケール変数を上書きする	C-11

D mailx セッションのカスタマイズ

E mailx セッションでのエスケープ・コマンドの使用

F mailx コマンドの使用

G アクセス制御リスト (ACL)

G.1	ACL の構造	G-2
G.2	アクセス決定のプロセス	G-4
G.3	ACL の継承	G-5
G.3.1	ACL 継承の例	G-7
G.4	ACL の管理	G-9
G.4.1	dxsetacl インタフェースの使用	G-10
G.4.2	getacl コマンドの使用	G-10
G.4.3	setacl コマンドの使用	G-10
G.5	コマンド, ユーティリティ, およびアプリケーションと ACL の相互作用	G-11
G.5.1	pax および tar コマンド	G-12

索引

例

1-1	典型的なログイン画面	1-5
1-2	date コマンドのリファレンス・ページ	1-17
3-1	ディレクトリの詳細なリスト (ls -l)	3-5

3-2	pg コマンドの出力 (1 つのファイル)	3-6
3-3	pg コマンドの出力 (複数のファイル)	3-7
3-4	lpr コマンドの使用	3-13
3-5	ファイルのリンク	3-19
5-1	絶対許可の設定	5-13
5-2	絶対許可の削除	5-13
5-3	su コマンドの使用	5-21
6-1	ps コマンドからの出力	6-11
6-2	who コマンドからの出力	6-17
6-3	who -u コマンドからの出力	6-17
6-4	w コマンドからの出力	6-18
6-5	ps au コマンドからの出力	6-18
8-1	ksh ヒストリの出力例	8-24
11-1	dead.letter ファイルへの取り込み	11-7
11-2	mailx コマンドを使ったファイルの取り込み	11-8
11-3	mailx 環境に入る	11-8
11-4	mailx プロンプトでのメッセージ開封	11-9
11-5	mailx プロンプトでの他のメッセージの開封	11-10
11-6	メール・メッセージへの応答	11-12
11-7	Forwarding a Message	11-16
11-8	mailx の help コマンドからの出力	11-17
11-9	.mailrc ファイルの例	11-18
12-1	ftp を使用したファイルのコピー	12-5
13-1	telnet コマンドの使用	13-4
D-1	mailx 詳細モード	D-6
G-1	ファイルの ACL の表示	G-10
G-2	ファイルの ACL の設定	G-11

図

1-1	シェルによるユーザおよびオペレーティング・システムとの対話	1-8
2-1	典型的なファイル・システム	2-10
2-2	相対パス名と完全パス名	2-13
3-1	リンクとファイルの削除	3-21
4-1	ディレクトリとサブディレクトリとの関係	4-3
4-2	ディレクトリ・ツリーのコピー	4-10
5-1	ファイルおよびディレクトリの許可フィールド	5-8
7-1	パイプラインを通る流れ	7-10
9-1	System V 機能	9-2

表

2-1	パターン照合文字	2-15
2-2	国際化パターン照合文字	2-16
3-1	ls コマンドのフラグ	3-4
3-2	pr コマンドのフラグ	3-10
3-3	lpr コマンドのフラグ	3-13
5-1	ファイル許可とディレクトリ許可の違い	5-6
5-2	許可の組み合わせ	5-14
5-3	8 進数と許可フィールドとの関係	5-15
5-4	umask 許可の組み合わせ	5-16
6-1	入力読み取りと出力リダイレクトのためのシェル表記	6-3
7-1	シェルのファイル名と省略時のプロンプト	7-6
7-2	複数コマンド演算子	7-8
7-3	コマンドをグループ化する記号	7-12
7-4	シェル引用規約	7-14
7-5	主なシェル環境変数	7-17

7-6	システムおよびローカルのログイン・スクリプト	7-20
7-7	シェル・スクリプト例の説明	7-31
8-1	C , Bourne , Korn , および POSIX シェルの機能	8-1
8-2	.cshrc スクリプト例の説明	8-3
8-3	.login スクリプト例の説明	8-5
8-4	C シェルのメタキャラクタ	8-6
8-5	ヒストリ・バッファ内のコマンドの再実行	8-8
8-6	組み込み C シェル変数	8-11
8-7	組み込み C シェル・コマンド	8-12
8-8	Bourne シェルの .profile スクリプト例の説明	8-14
8-9	Bourne シェル・メタキャラクタ	8-15
8-10	組み込み Bourne シェル変数	8-17
8-11	組み込み Bourne シェル・コマンド	8-18
8-12	Korn または POSIX シェルの .profile ログイン・スクリプトの 例の説明	8-20
8-13	.kshrc ログイン・スクリプトの例の説明	8-22
8-14	Korn または POSIX シェルのメタキャラクタ	8-23
8-15	ヒストリ・バッファ内のコマンドの再実行	8-25
8-16	組み込み Korn または POSIX シェル変数	8-32
8-17	組み込み Korn または POSIX シェル・コマンド	8-33
9-1	ユーザ・コマンドのまとめ	9-5
10-1	finger コマンドのオプション	10-5
10-2	ruptime コマンドのオプション	10-8
11-1	MHプログラムのコマンド	11-22
12-1	ホストへの接続とファイルのコピーを行う ftp サブコマンド .	12-6
12-2	ディレクトリおよびファイルを操作するための ftp サブコマン ド	12-9
12-3	ヘルプおよび状態情報を得るための ftp サブコマンド	12-9
13-1	telnet サブコマンド	13-6

14-1	cu コマンドのオプション	14-6
14-2	ローカルの cu コマンド	14-10
14-3	tip コマンドのオプション	14-14
14-4	ローカル tip コマンド	14-17
14-5	ct コマンドのオプション	14-20
14-6	uux コマンドのオプション	14-24
14-7	UUCP コマンドのオプション	14-28
14-8	uupick コマンドのオプション	14-29
14-9	uuto コマンドのオプション	14-30
14-10	uustat コマンドのオプション	14-34
14-11	uulog コマンドのオプション	14-37
A-1	write および quit コマンドのまとめ	A-7
A-2	カーソル移動コマンド一覧	A-10
A-3	テキスト挿入コマンド一覧	A-13
A-4	テキスト編集コマンドのまとめ	A-16
A-5	vi の一般的な環境変数	A-24
C-1	ロケール名	C-8
C-2	ロケール関数に影響を及ぼす環境変数	C-9
D-1	mailx セッションをカスタマイズする変数	D-1
E-1	メール用エスケープ・コマンド	E-1
F-1	mailx プログラムのコマンド	F-1
G-1	ACL エントリの例	G-3

まえがき

本書では、HP Tru64 UNIX オペレーティング・システムの基本的なコマンドとシェルの使用方法について紹介します。本書では、他のネットワーク・ユーザと通信する方法についても紹介します。

本書の対象読者

本書は、UNIX 互換オペレーティング・システムについて詳しい知識がないユーザを対象に書かれています。本書では Tru64 UNIX オペレーティング・システムに関する重要な概念について説明し、実際に操作しながら理解できるような構成になっています。

本書では、コマンド行からのコマンドの入力および実行について説明します。これらの機能の多くは、グラフィカル・ユーザ・インタフェース (GUI) を使用して実行することもできます。また GUI では、ここで説明する以外の機能も実行することができます。詳細は、ウィンドウ・マネージャあるいは個々のアプリケーションのヘルプを参照するか、またはシステム管理者にお問い合わせください。

新しい機能および変更された機能

本リリースでは 付録 G が追加されています。ここでは、アクセス制御リスト (ACL) を使って、特定のユーザおよびグループのユーザに対してファイルおよびディレクトリのアクセス許可を与える方法について説明しています。

本書の構成

本書の構成は以下のとおりです。

- | | |
|-------|--|
| 第 1 章 | システムへのログインおよびログアウトの方法、コマンドの入力方法、パスワードの設定方法、オンライン・ヘルプの使用法を紹介します。 |
| 第 2 章 | ファイルとディレクトリから構成されるファイル・システムの概要を紹介します。この章では、vi テキスト・エディタについても紹介します。このプログラムを使用して、ファイルを作成し、修正することができます。 |

第 3 章	ファイルの管理方法について説明します。ファイルをリスト、表示、コピー、移動、リンク、削除する方法を理解することができます。
第 4 章	ディレクトリの管理方法について説明します。この章では、ファイルを作成、変更、表示、コピー、名称変更、削除する方法を理解することができます。
第 5 章	適切な許可を設定することによって、ファイルとディレクトリへのアクセスを制御する方法を説明します。さらにこの章では、標準パスワードとグループ・セキュリティの問題、およびその他のセキュリティ上の考慮事項の概要についても説明します。
第 6 章	Tru64 UNIX オペレーティング・システムがどのようにプロセスを作成し追跡するかを説明します。プロセスの入力、出力、およびエラー情報をリダイレクトする方法、プロセスを同時に実行する方法、プロセス情報を表示する方法、プロセスを取り消す方法を説明します。
第 7 章	Tru64 UNIX オペレーティング・システムで利用できる C, Bourne, Korn, および POSIX シェルに共通する機能について紹介します。シェルの変更、コマンド入力エイドの使用、ユーザのシェル環境の特徴 (ログイン・スクリプト、環境変数およびシェル変数) の理解、変数の設定とクリア、ログアウト・スクリプトの作成、基本的なシェル・プロシージャの作成と実行方法を理解することもできます。
第 8 章	C, Bourne, Korn, および POSIX シェルについて、それらの機能を比較しながら、詳細な参照情報を提供します。各プログラムのコマンドと環境変数について詳しく説明し、ログイン・スクリプトの設定方法を紹介します。
第 9 章	System V インタフェース定義 (SVID) に準拠するコマンド、サブルーチンおよびシステム・コールを使用するために必要な環境設定について説明します。
第 10 章	ネットワーク上の他のユーザやリモート・ホストの情報を取得する方法を紹介します。
第 11 章	他のユーザにメッセージを送信する方法を紹介します。
第 12 章	リモート・ホストへファイルをコピーする方法、または、リモート・ホスト間でファイルをコピーする方法を紹介します。
第 13 章	リモート・ホストへログインする方法、または、リモート・ホストでコマンドを実行する方法を紹介します。
第 14 章	ローカル・ホストとリモート・ホストの両方で同時に通信タスクを実行するための UNIX 間コピー・プログラム (UUCP) を紹介します。
付録 A	vi テキスト・エディタの基本的な機能の使用方法を説明します。
付録 B	ed テキスト・エディタの使用方法を説明します。ここでは ed についての詳細な情報を提供しています。このエディタはすべてのシステムに含まれており、また他のエディタが使用できないような深刻な状況においても、ed は使用できます。

- 付録 C 国際化機能について説明しています。国際化機能を使用すると、言語、慣習、および地理的位置に合った方法でデータ処理およびシステムとの対話ができます。
- 付録 D mailx セッションをカスタマイズするために .mailrc ファイルで可以使用の変数の一覧を示します。
- 付録 E mailx セッション内から特定のタスクを実行するために使用できるエスケープ・コマンドの一覧を示します。
- 付録 F mailx を使用して、メッセージの送信、読み取り、削除、保存を行うコマンドの一覧を示します。
- 付録 G アクセス制御リスト (ACL) の使用方法について説明しています。

関連資料

次の Tru64 UNIX ドキュメントがドキュメント CD-ROM に含まれています。また、別売のハードコピー版を購入することもできます。

- 『ネットワーク管理ガイド：接続編』
- 『ネットワーク管理ガイド：サービス編』
- 『ドキュメント概要』
- 『*Reference Pages Section 1*』
- 『*Reference Pages Sections 8 and 1m*』
- 『セキュリティ管理ガイド』
- 『システム管理ガイド』
- 『*Quick Reference Card*』

Tru64 UNIX のマニュアルは、次の URL から入手することもできます。

<http://tru64unix.compaq.co.jp/document/index.html>

ここにある参考文献は、個々のトピックの分かりやすい解説書として使用できるだけでなく、UNIX コマンドの理解にも役立ちます。

本書で使用する表記法

本書では次の表記法を使用しています。

% \$	パーセント記号は、C シェルのシステム・プロンプトを表します。ドル記号は、Bourne シェル、Korn シェル、および POSIX シェルの場合のシステム・プロンプトを表します。
#	番号記号は root としてログインした場合のシステム・プロンプトを表します。
% cat	対話式の例における太字(ボールド体)は、ユーザが入力する文字を示します。
<i>file</i>	イタリック体(斜体)は、変数値、プレースホルダ、および関数の引数名を示します。
[] { }	構文定義では、大カッコはオプションの項目を示し、中カッコは必須項目を示します。大カッコまたは中カッコの中の項目を縦線で区切っている場合は、そこに併記されている項目の中から1つの項目を選択することを示します。
...	構文定義では、水平の反復記号は、前の項目を1回以上繰り返して使用できることを示します。
cat(1)	リファレンス・ページの参照には、該当するセクション番号をカッコ内に示します。たとえば、cat(1) は、cat コマンドについての情報が、リファレンス・ページのセクション1に記載されていることを示します。
Ctrl/x	この記号は、スラッシュの前に指定されているキーを押しながら、スラッシュの後のキーまたはマウス・ボタンを押すことを示します。例中では、このようなキーの組み合わせは、四角あるいは大カッコで囲まれて示されます(たとえば、 <code>Ctrl/C</code>)。

基本操作

この章では、Tru64 UNIX オペレーティング・システムを使用するための基本的な操作について説明します。この章を読む前に、お使いのシステムのハードウェア構成を理解しておいてください。

UNIX オペレーティング・システムあるいはその他のオペレーティング・システムの操作についてよくご存知のユーザは、この章には簡単に目を通すだけでもよいでしょう。本書では、コマンド行からオペレーティング・システムを使用する方法について説明します。本オペレーティング・システムでは、多数のグラフィカル・ユーザ・インタフェース (GUI) 機能が提供されています。これらの機能については、別のマニュアルで説明しています。

この章では次の操作について説明します。

- オペレーティング・システムへのログイン (1.1 節)
- オペレーティング・システムからのログアウト (1.2 節)
- パスワードの設定と変更 (1.5 節)
- コマンドの実行 (1.3 節)
- コマンド実行の停止 (1.4 節)
- リファレンス・ページ (manpage) の参照と表示 (1.6.1 項)

Tru64 UNIX オペレーティング・システムの性能を十分に活用するためには、テキスト編集プログラムを使用した、ファイルの作成および修正方法を理解しておく必要があります。テキスト・エディタの概要については第 2 章を参照してください。また、vi および ed テキスト・エディタについては、それぞれ 付録 A および 付録 B を参照してください。テキスト・エディタの使用方法をマスターすれば、オペレーティング・システムを操作するために必要な基本的なスキルを身につけたことになります。

注意

システムでオプションのエンハンスド・セキュリティを実行している場合は、ログインおよびパスワードの手順が本書で説明している手順と異なることがあります。エンハンスド・セキュリティについての詳細は、システム管理者にお問い合わせいただくかあるいは『セキュリティ管理ガイド』を参照してください。

1.1 ログイン

Tru64 UNIX オペレーティング・システムを使用するためには、オペレーティング・システムがインストールされ起動されている状態で、まずシステムにログインしなければなりません。システムにログインすることにより、そのユーザは有効なシステム・ユーザとして識別され、そのユーザのための作業環境が生成されます。

ログインするためには、システム管理者からユーザ名とパスワードを取得しておく必要があります。通常、ユーザ名にはユーザの姓またはイニシャルを使用します。システムはユーザ名によって、そのユーザが権限を与えられたユーザであると識別します。パスワードには、ユーザには覚えやすく他の人には推測しにくい文字列を使用します。システムはパスワードによってユーザの身元を確認します。

ユーザ名とパスワードは、システムへのアクセスを可能にする電子的なキーと考えることもできます。ログイン・プロセスでユーザ名とパスワードを入力すると、そのユーザは権限を与えられたユーザであると認識されます。

パスワードは、他のユーザによるデータの無断使用を防止するために用意されており、システム・セキュリティ上非常に重要なものです。パスワードについての詳細は、1.5.2 項を参照してください。

ログイン・プロセスにおける最初の処理は、ログイン・プロンプトを表示することです。システムが起動していて、ワークステーションがオンになっている場合は、次のログイン・プロンプトが画面に表示されます。

login:

システムによっては、Return キーを 2, 3 回押さなければログイン・プロンプトが表示されないものもあります。

ログイン・プロンプトの画面は使用しているシステムによって多少異なるかもしれませんが。たとえば、ログイン・プロンプトのほかに、システム名とオペレーティング・システムのバージョン番号が表示される場合もあります。

システムにログインするには、次の操作を実行します。

1. ログイン・プロンプトに対してユーザ名を入力する。

入力を間違えた場合は、削除キーまたはバックスペース・キーを使用して、誤りを訂正してください。

たとえば、ユーザ名が `larry` の場合は、次のように入力します。

```
login: larry
```

次にパスワード・プロンプトが表示されます。

```
login: larry  
Password:
```

2. パスワード・プロンプトに対してパスワードを入力する。

セキュリティ上の理由から、入力したパスワードは画面には表示されません。

パスワードを間違えた場合は、その場で Return キーを押してください。パスワードが間違っている場合、システムは警告メッセージを表示して、もう一度ユーザ名とパスワードの入力を求めます。システムによっては、パスワードの入力中に削除キーやバックスペース・キーを使用して、間違いを修正することができます。

ユーザ名とパスワードを正しく入力すると、システムはシェル・プロンプトを表示します。通常、シェル・プロンプトにはドル記号 (\$) またはパーセント記号 (%) が使用されます。使用するシステムによっては、上記以外のシェル・プロンプトが使用される場合もあります。

ユーザ名およびパスワードを正しく入力したあと、システムは次のような情報を出力します。

- 最後に成功および失敗したログインの日時:

```
Last successful login for juanita: date and time on tty03  
Last unsuccessful login for juanita: date and time on tty03
```

この情報は常にチェックするようにしてください。何か不審な点がある場合、誰かがあなたのアカウントにログインしようとした (あるい

はログインした) 可能性があります。この場合はすぐにシステム管理者に連絡してください。

- パスワードが期限切れになっている場合は次のようなメッセージが表示されます：

`Your password will expire on date and time`

パスワードの設定については 1.5 節 を参照してください。

- シェル・プロンプトは、通常、\$ あるいは % ですが、ご使用のシステムによっては異なる場合があります。

注意

本書では、シェル・プロンプトとしてドル記号 (\$) を使用します。

ログインが成功するとシェル・プロンプトが表示され、ユーザはシステムに処理を実行させることができます。シェル・プロンプトは、シェルが実行状態にあるというユーザへの合図です。シェルは、ユーザが入力するコマンドを解釈し、ユーザが要求したプログラムを実行して、その結果を画面に送るためのプログラムです。コマンドとシェル・プロンプトについての詳細は、1.3 節および第 7 章を参照してください。

ログインすると、ユーザは自動的にログイン・ディレクトリに入ります。このディレクトリは、ユーザのホーム・ディレクトリとも呼ばれます。ログイン・ディレクトリについては第 2 章を参照してください。

注意

ログインした状態でシステムの前を離れないようにしてください。誰かがターミナルを使用してあなたの ID でプログラムを実行し、被害を受ける可能性があります。

シェル・プロンプトが表示されない場合、ユーザはログインに成功していません。たとえば、ユーザ名かパスワードが正しく入力されていない可能性があります。再度ログインを試みてください。ユーザ名とパスワードを正しく入力してもログインできない場合は、システム管理者に問い合わせ

てください。システムによっては、セキュリティ上の理由から、何回か連続して入力間違いをすると、システムがすべてのログイン試行を拒否します。ログインしようとして拒否された場合は、セキュリティ上の理由から、次のようなメッセージが表示されるだけです。

Login incorrect

注意

システムの設定によっては、ユーザにパスワードの入力を要求しない場合もあれば、システム管理者がすべての新規ユーザに対して共通のパスワードを設定している場合もあります。このような場合には、セキュリティを確保するために、ユーザ自身のパスワードを設定してください。パスワードの設定および変更方法については、1.5 節を参照してください。

多くのシステムでは、ユーザがログインするとウェルカム・メッセージを表示します。典型的なログイン画面の例を次に示します。例 1-1 は、実際に表示される画面ですが、システムによって異なる場合があります。

例 1-1: 典型的なログイン画面

```
Welcome to the Operating System 1
Fri Dec 7 09:48:25 EDT 19nn      2

Messages from the administrator 3

You have mail                     4
$
上記のウェルカム・メッセージには次の情報が含まれています。
```

1 挨拶の言葉

2 前回のログイン日時

ログインするときにはいつもこの情報に注意し、表示されている時間にログインした覚えがない場合はシステム管理者に報告してください。この場合、誰かが不正にシステムに入り込んだ可能性があります。

3 管理者からのメッセージ

システム管理者は、各ユーザがログインするたびに受け取るメッセージを設定していることがあります。これらのメッセージには、システムのアップデート計画や、オペレーションのスケジュール、あるいはその他の一般的な情報が含まれています。これらのメッセージは今日のメッセージ (message of the day) と

例 1-1: 典型的なログイン画面 (続き)

呼ばれ、`/etc/motd` ファイルに格納されています。このファイルを表示することにより、いつでもこれらのメッセージを参照することができます。

4 未読のメール・メッセージの有無

メールは、電子メールを送受信するためのプログラムです。未読のメール・メッセージがある場合、システムは "You have mail." というメッセージを表示します。メール・メッセージがなければ、このメッセージは表示されません。

1.1.1 ログインの制限

特定の端末に対して認証ユーザ・リストが作成されていることがあります。このリストにユーザ名が記述されていると、その端末ではログインすることができません。この場合、システムは次のようなメッセージを表示します。

```
Not authorized for terminal access--see System Administrator
```

所定の回数ログインに失敗すると、その端末は使用できなくなります。このセキュリティ機能は、特定の端末からのログインの失敗の回数に上限を設けることにより、不正なログインからシステムを保護します。

システム管理者が端末を明示的にロックすることもできます。端末が無効あるいはロックされている場合、次のようなメッセージが表示されます。

```
Terminal is disabled -- see Account Administrator
```

ログインの失敗が所定の回数繰り返された場合、アカウントを無効にすることもできます。端を無効にする機能と同じように、このセキュリティ機能は、ログインの失敗の回数に上限を設けることにより、不正なログインからシステムを保護します。システム管理者が設定した有効期限を過ぎた場合にパスワードが自動的に無効になるように設定されている場合もあります。

アカウントがシステム管理者により明示的に無効にされている場合もあります。この場合、次のようなメッセージが表示されます。

```
Account is disabled -- see Account Administrator
```

ログイン時にこれらのいずれかのメッセージが表示された場合は、システム管理者にお問い合わせください。端末あるいはアカウントが無効にされている場合、再度ログインできるようにするためには、システム管理者による設定が必要になります。

1.2 ログアウト

必要な作業を終了したら、システムからログアウトしてください。システムからログアウトすることによって、それまで使用していた作業環境は誰も使用できなくなります。オペレーティング・システムは他のユーザがログインするまで、実行可能状態で待機します。

ログアウトするには、次の操作を実行します。

1. シェル・プロンプトが表示されていることを確認する。
2. Ctrl/D を押す。

Ctrl/D が機能しない場合は、`exit` コマンドを入力します。

上記の操作を行うとシステムはログイン・プロンプトまたはログイン画面を表示します。システムによっては、ログアウトしたことを示すメッセージを表示する場合があります。

この時点で、別のユーザがログインすることができます。

1.3 コマンドの使用

オペレーティング・システム・コマンドは、Tru64 UNIX オペレーティング・システムでタスクを実行するためのプログラムです。Tru64 UNIX オペレーティング・システムでは多くのコマンドが用意されており、以降の章および関連するリファレンス・ページに説明があります。

コマンドの入力是对話処理です。ユーザがコマンドを入力すると、シェルがそのコマンドを解釈して、適切な応答をします。つまり、システムはプログラムを実行するか、またはエラー・メッセージを表示します。

シェルはユーザが入力したコマンドをすべて読み取り、ユーザが何を要求しているかをオペレーティング・システムに伝えます。つまり、シェルはコマンド・インタプリタです。

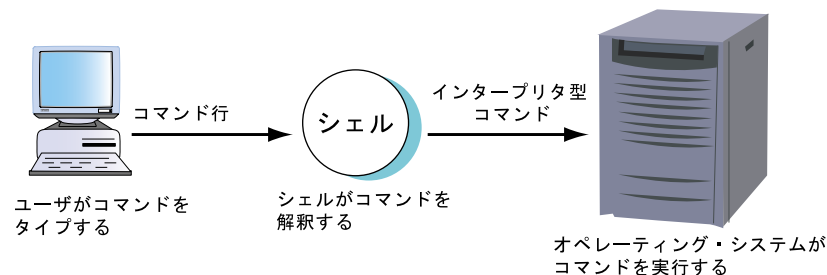
シェルは次の手順でコマンド・インタプリタとして機能します。

1. シェルはシェル・プロンプトを表示し、コマンドが入力されるまで待機します。
2. ユーザがコマンドを入力すると、シェルはそれを分析して、要求されたプログラムを検索します。

3. シェルはそのプログラムを実行するようにシステムに要求するか、あるいはエラー・メッセージを返します。
4. プログラムの実行が完了するとシェルに制御が戻り、シェルは再びシェル・プロンプトを表示します。

図 1-1 は、ユーザ、シェル、およびオペレーティング・システムの関係を示しています。シェルはコマンドを解釈するためにユーザと対話し、コマンドの実行を要求するためにオペレーティング・システムと対話します。

図 1-1: シェルによるユーザおよびオペレーティング・システムとの対話



ZK-0530U-AIJ

Tru64 UNIX オペレーティング・システムは C シェル、および Bourne、Korn、および POSIX シェルの 4 種類のシェルをサポートします。ユーザが最初にログインした際に使用するシェルは、システム管理者が設定します。シェルについての詳細は、第 7 章を参照してください。

オペレーティング・システムを使用する場合、通常は、コマンド行のシェル・プロンプトの後にコマンドを入力します。たとえば、現在の日付と時刻を表示するには、次のように入力します。

```
$ date
```

コマンドの入力中にタイプ・ミスをした場合は、削除キーまたはバックスペース・キーを使用して、間違ってタイプした文字を消し、再度タイプしてください。

コマンド名の後に入力する文字列を引数と呼びます。引数は、コマンドがその処理を完了するために必要なデータを指定します。たとえば `man` コマンドはオペレーティング・システムで使えるコマンドについての情報を提供します。`man` コマンドを使用して `date` コマンドについての詳細な情報を表示したい場合は、次のように入力します。

```
$ man date
```


コマンドには、コマンドの動作を明示的に指定するためのオプションが用意されています。オプションはフラグとも呼ばれ、コマンド名のすぐ後にタイプします。大部分のコマンドには数個のフラグが用意されています。コマンドにフラグを指定する場合、引数はコマンド行のフラグの後にタイプします。

たとえば、`man` コマンドに `-f` フラグを付けると、指定したコマンドに関する説明を 1 行で表示します。`date` コマンドの説明を 1 行で表示するためには、次のように入力します。

```
$ man -f date
```

コマンド実行中は実行しているコマンド (プログラム) に制御が移っているため、システムはシェル・プロンプトを表示しません。コマンドが処理を完了すると、システムはシェル・プロンプトを表示します。この時点でユーザは次のコマンドを入力することができます。

システムで用意されているコマンド以外に、自分専用のコマンドを作成することもできます。新たなコマンドの作成方法については、7.10.1 項を参照してください。

1.4 コマンド実行の中止

コマンドを入力した後、そのコマンドの実行を中断したい場合は、`Ctrl/C` を入力してください。`Ctrl/C` を入力すると、コマンドは処理を中止し、システムはシェル・プロンプトを表示します。この時点で、別のコマンドを入力することができます。

コマンドの実行を中止した場合の結果はコマンドによって異なります。実行中のコマンドを中止した場合の結果を見たい場合は、たとえばディレクトリ内のファイルをリストする `ls -l` コマンドや、画面にファイルを表示する `cat filename` コマンドを実行した後、`Ctrl/C` を入力してみてください。

1.5 パスワードの設定

ユーザ名は公開情報であり一般に変更しません。それに対してパスワードは非公開の情報です。

多くの場合、システム・アカウントを作成する際に、システム管理者が新規ユーザに共通のパスワードを設定します。システムによっては、この新規ユーザのパスワードは、1 回のログインに対してのみ有効で、システムにアクセスして独自のパスワードを設定するために使用します。システムに慣れたら、パスワードを変更して自分のアカウントを権限のないアクセスが

ら保護してください。また、権限のないアクセスからデータを保護するため、パスワードは定期的に変更してください。

パスワードの設定には `passwd` コマンドを使用します。アカウントにパスワードが設定されていない場合には、`passwd` コマンドを使用してパスワードを設定します。`passwd` コマンドについては、1.5.4 項を参照してください。システムがネットワークに接続されている場合には、`yppasswd` コマンドを使用してパスワードの設定または変更を行います。使用しているシステムにおける個々の手順については、システム管理者に問い合わせてください。

パスワードがどのように作成、発行、変更、無効化されるかを決定するために、システム管理者は次のようないくつかのオプションを選択できます。

- どのような状況でもユーザがパスワードを変更できるかどうか
- 特定のパスワードを以前に使用していたかどうか
- ユーザ自身で選択したパスワードを使用できるかどうか
- ユーザ選択パスワードを使用できない場合、システムはどのようなタイプのパスワードを生成するか
- いつパスワードを変更できるか、および、いつパスワードを変更しなければならないか

ユーザによるパスワードの変更が認められていない場合、`passwd` コマンドを実行すると次のようなメッセージが表示されます。

```
Password request denied.  
Reason: you do not have any password changing options.
```

この場合、システム管理者に連絡してパスワードの変更を依頼してください。

1.5.1 パスワードの期限

トラステッド・システムでは各パスワードに対して最短変更時間、有効期限、および存続期間を設定します。最短変更時間が経過するまでパスワードは変更できません。これにより、新しいパスワードを覚えなくていいように、パスワードの変更直後にパスワードを元に戻すことはできなくなります。パスワードをあまりにも早く変更しようとすると、システムは次のメッセージで応答します。

```
Password cannot be changed.  
Reason: minimum time between changes has not elapsed.
```

パスワードは有効期限になるまで有効です。パスワードの有効期限が切れると、システムに再度ログインするためにはパスワードを変更しなければなりません。パスワードの有効期限が近づくと、ログイン時にメッセージが表示されます。そのメッセージが表示されたら、パスワードを変更する必要があります。ログインしていないときにパスワードの有効期限が切れると、次のログイン時にログイン処理の一環としてパスワードを変更できます。

存続期間を超過した場合、アカウントは無効になります。無効状態のアカウントにログインしようとすると、システムはメッセージを表示します。この場合、システム管理者にアカウントを再度有効にするように頼まなければなりません。また、次のログイン時にパスワードを変更しなければなりません。

1.5.2 パスワードに関するガイドライン

識別 (ユーザ名) および認証 (パスワード) の手順は、不正なアクセスからシステムを守るための最も重要なセキュリティ・ツールの 1 つです。ログインが認められていないユーザがシステムにアクセスするためには、パスワードを不正に知って、端末に直接アクセスするか、ネットワークを介してリモート・アクセスする必要があるからです。

そのようなユーザにログインを許すと、そのユーザは密かにデータを盗みシステムを破壊できます。アクセスを受けたアカウントのシステム上での権限が高いほど、侵入者がシステムに与えるダメージが大きくなります。

侵入者の動作はあなたのアカウントに対して反映され、責任は自分で負わなければならないことを覚えておいてください。アカウントが危険にさらされないように保護するのは、ユーザの義務です。以下に示すガイドラインに従って、自分のパスワードを保護してください。

- パスワードは共有しないでください。他の人に自分のパスワードを伝えて、そのアカウントにログインさせると、システムはユーザごとに動作を把握することができなくなります。
- パスワードを書き留めてはなりません。多くのシステム侵入は、ユーザが自分のパスワードを端末に記述したために発生しています。パスワードを記録しておかなければならない場合には、その記録を鍵のかかるところに保管してください。
- 他人が見ているところでパスワードを入力しないでください。パスワードを入力しているところを見られただけで、パスワードが盗まれること

があります。パブリックな場所でワークステーションを使用している場合には特に気を付けてください。

- 特に気になる場合はパスワードを頻繁に変更してください。
- e-mail にパスワードを記述しないでください。

1.5.3 ユーザによるパスワードの決定

パスワードの変更が認められている場合、ユーザがパスワードを自由に選択できるか、あるいはシステムがパスワードを生成するかがアカウントに設定されています。どちらに設定されているかによって、passwd を実行したときにシステムが起動するダイアログが決まります。まず、システムは現在のパスワードを入力するように要求します。

Old password:

古いパスワードを入力します。正しく入力すると、システムはパスワード変更の日時を表示します。

Last successful password change for user: date and time

Last unsuccessful password change for user: date and time

これらの日時を必ず確認してください。最後にパスワードを変更した日時を正確に記憶していない場合には、少なくとも時間が妥当かどうかだけでも確認してください。

システム管理者は、アカウントのパスワード・タイプを、以下の中からユーザに選ばせることができます。

- システムが生成するランダムな発音可能な音節
- システムが生成するランダムな文字 (区切り記号および数字を含む)
- システムが生成するランダムな英字
- ユーザが決めたパスワード

次の例は、可能なオプションがすべて認められているときの入力要求を示します。

Do you want (choose one option only):

- 1 pronounceable passwords generated for you
- 2 a string of characters generated for you
- 3 a string of letters generated for you
- 4 to pick your password

Select ONE item by number:

「ユーザが決めたパスワード」を選択した場合，入力ミスを防ぐために，システムは新しいパスワードを 2 回入力するように要求します。

1.5.3.1 システムが生成するパスワードの選択

次の例は，システムが生成する発音可能なパスワードに関するダイアログを示します。

Generating random pronounceable password for user.

The password, along with the hyphenated version, is shown.

Hit <RETURN> or <ENTER> until you like the choice.

When you have chosen the password you want, type it in.

Note: Type your interrupt character or "quit" to abort at any time.

Password: saglemot Hyphenation: sag-le-mot

Enter password:

ハイフンで区切られたパスワードはパスワードをより容易に記憶できるように，パスワードの発音を手助けするために表示されています。このハイフンは入力しません。最初のパスワードが気に入らない場合，Return を押して別のパスワードを表示します。気に入ったパスワードが生成されたら，それを入力します。

パスワードを変更しないことにした場合，quitを入力するか，または中断文字 (通常は Ctrl/C) を使用します。システムは次のメッセージを表示します。

Password cannot be changed. Reason: user stopped program.

また，システムは，最後にパスワードの変更に失敗した日時もアップデートします。

他のシステム生成パスワード・タイプを選択したときのダイアログも同様です。

1.5.3.2 ユーザ選択パスワード

パスワードの選定の際には，次のガイドラインを参考にしてください。

- 辞書に収録されている単語は選択しないでください。

- ユーザ名，氏名やニックネーム (自分自身のだけでなく，家族やペットのもの)，会社名，イニシャル，自動車の車種やモデル名などの個人的な情報を，パスワードまたはパスワードの一部として使用しないでください。
- 誕生日，年金番号や口座番号，社員番号，電話番号，その他同様の情報をパスワードまたはパスワードの一部として使用しないでください。
- アカウントと一緒にシステム管理者から受け取った省略時のパスワードをそのまま使用しないでください。
- 以前に使用したパスワード，あるいは以前のパスワードに類似したパスワードは使用しないでください。
- 複数の異なるシステムにアクセスする場合，すべてのシステムで同じパスワードを使用しないでください。
- 容易に推測できるパスワードは選択しないでください。たとえばスペルを逆にしても望ましくありません。推測しにくく，覚えやすいパスワードを選択してください。
- 6文字未満の文字列をパスワードとして選択しないでください。パスワードの最大長はシステムで採用しているセキュリティ規則によって異なります。パスワードの長さは文字数ではなくバイト数で測定しますが，現在のところは文字数とバイト数は同じと考えることができます。
- 可能であれば，大文字と小文字を組み合わせたパスワードを使用してください。数字，句読点，あるいは下線 (_) を含めるのも効果的です。

多くのシステムでは，パスワードは，頻繁にであれ稀にであれ，いつでも変更することができます。ただし，システム・セキュリティを保護するため，パスワードを変更できる頻度，設定したパスワードの有効期間，あるいは変更の内容についてシステム管理者が制限を設けることがあります。パスワードに対する代表的な制限としては，次のようなものがあります。

- 文字制限
 - 英字の最小文字数
 - 句読点や数字などの最小文字数
 - 旧パスワードと新しいパスワードの差異文字の最小数
 - パスワードで許容できる連続重複文字の最小数
- 時間制限

- 設定したパスワードが期限切れになるまでの最大週数
- パスワードの変更が可能になるまでの週数

パスワードの制限についての詳細は、システム管理者に問い合わせてください。システムにインストールされてるか有効になっているセキュリティおよびアクセス制御にはいくつかのレベルがあります。アクセス制御についての詳しい説明は『セキュリティ管理ガイド』を参照してください。

1.5.4 パスワード設定の手順

パスワードを設定または変更するための手順は次のとおりです。

1. `passwd` コマンドを入力する。

```
$ passwd
```

`passwd` コマンドを入力すると、システムは次のようなメッセージを表示して、旧パスワードを入力するよう求めます。

```
Changing password for username  
Old password:
```

旧パスワードが設定されていない場合、システムはこのプロンプトを表示しません。その場合は手順 3 に進んでください。

2. 旧パスワードを入力する。

セキュリティ上の理由から、パスワードをタイプしても画面には表示されません。

旧パスワードを確認すると、システムは新しいパスワードの入力を促すプロンプトを表示します。

```
New password:
```

3. このプロンプトの後に新しいパスワードを入力する。

入力した新しいパスワードは画面には表示されません。

最後に、新しいパスワードの確認のために、システムは新しいパスワードの再入力を促すプロンプトを表示します。

```
Retype new password:
```

4. 再度新しいパスワードを入力する。

このとき、新しいパスワードは画面には表示されません。シェル・プロンプトが再び画面に表示されると、新しいパスワードが有効になります。

パスワードを変更しようとしたときに新しいパスワードが適切でない場合、そのパスワードの具体的な問題点およびそのシステムに設定されている制限に関するメッセージが表示されます。表示されるメッセージおよび詳しさのレベルは、システムで有効になっているセキュリティおよびアクセス制御メカニズムによって決まります。

注意

パスワードを設定すると、ログイン時には常にそのパスワードが必要になりますので、設定したパスワードは正しく記憶するようにしてください。万一設定したパスワードを忘れた場合は、システム管理者にご相談ください。

1.6 ヘルプの利用

本書では、コマンド行からのコマンドの入力および実行方法について説明します。グラフィカル・ユーザ・インタフェース (GUI) を使用している場合は、ウィンドウ・マネージャのヘルプを参照するか、またはシステム管理者に問い合わせてください。

作業に必要な基本的なオペレーティング・システム・コマンドのほとんどについては、本書で説明しています。本書で説明しているコマンドおよびその他のコマンドの詳細については、リファレンス・ページを参照してください。リファレンス・ページは次の3種類の形態で提供されています。

- オンライン (1.6.1 項を参照)
- ハードコピー (関連資料を参照)
- HTML 形式

注意

日本語リファレンス・ページについてはオンラインでのみ提供しています。

どの形式のリファレンス・ページがシステムにインストールされているかは、システム管理者に問い合わせてください。ハードコピーあるいはHTML形式

のリファレンス・ページを利用できない場合は、次のいずれかのコマンドを使用して、コマンドに関する情報をオンラインで入手することができます。

- `man` コマンド
オンライン・リファレンス・ページを表示します。
- `apropos` コマンド
指定したキーワードに関係する各コマンドの概要を 1 行で表示します。

次の各項でこれらの機能について説明します。

1.6.1 オンライン・リファレンス・ページの表示と印刷 (`man` コマンド)

オンライン・リファレンス・ページは、オペレーティング・システムで利用できるコマンドについての詳細な情報を提供します。リファレンス・ページを参照するためには、`man` コマンドを使用します。例 1-2 では、`date` コマンドのリファレンス・ページを参照しています。実際の表示内容はご使用のシステムによって異なる場合があります。

例 1-2: `date` コマンドのリファレンス・ページ

```
$ man date
date(1)                                date(1)

NAME

    date - Displays or sets the date

SYNOPSIS

    Without Superuser Authority - Displays the Date

    date [-u] [+field_descriptor ...]

    With Superuser Authority - Sets the Date

    date [-nu] [MMddhhmm.ssyy | alternate_date_format] [+field_descriptor ...]

    Using XPG4-UNIX - Sets or Displays the Date

    date [-u] mmddHHMM[yy]

    date [-u] [+field_descriptor ...]

    Using the Century Field Provided by HP - Sets the Date

    date mmddHHMM[[cc]yy][.ss]
    date [[cc]yy]mmddHHMM[.ss]
    date mmddHHMM[.ss][[cc]yy]]

STANDARDS

    Interfaces documented on this reference page conform to industry standards
```

例 1-2: date コマンドのリファレンス・ページ (続き)

```
as follows:

date:  XPG4, XPG4-UNIX

Refer to the standards(5) reference page for more information about indus-
try standards and associated tags.
--More-- (6%)
```

ページの最下行に表示される `--More-- (6%)` は、リファレンス・ページの 6% が現在表示されていることを示しています。次の画面を表示させるには、スペース・バーを押してください。次の 1 行を表示させるには Return キーを押します。また、リファレンス・ページの表示を終了してシェル・プロンプトに戻るためには `q` と入力してください。

リファレンス・ページを印刷する場合は、次のコマンド構文を使用します。

man *manpage* | **lpr** -P *printer_name*

たとえば、`date` コマンドのリファレンス・ページを特定のプリンタに出力する場合は、次のように入力します。

```
$ man date | lpr -Pprinter_name
```

このコマンドを実行すると、`date` コマンドのリファレンス・ページがプリント・キュー *printer_name* に登録されます。 `lpr` コマンドについての詳細は、3.3 節を参照してください。

コマンドについての説明を 1 行で表示させるには、`man -f` コマンドを使用します。たとえば `who` コマンドの簡単な説明を表示させるには、次のように入力します。

```
$ man -f who
who (1)      - Identifies users currently logged in
$
```

`man` コマンドとそのオプションに関する詳細な情報が必要な場合は、次のように入力すると、完全なリファレンス・ページを表示させることができます。

```
$ man man
```

1.6.2 キーワードを使用したコマンドの検索

`apropos` コマンドと `man -k` コマンドは、コマンド名を忘れた場合に役立つツールです。

注意

`apropos` コマンドおよび `man -k` コマンドは、`whatis` データベースにアクセスします。このデータベースは、システム管理者がオペレーティング・システムをインストールしたときに省略時の `whatis` データベースをロードしている場合、または、`catman` コマンドを使用して `whatis` データベースを作成している場合にのみ利用可能です。

`apropos` コマンドと `man -k` コマンドはまったく同じように機能します。いずれのコマンドも、コマンドに関する記述をキーワードとして指定すると、それらのキーワードを含むすべてのリファレンス・ページをリストします。

次の例に示すように、キーワードが2ワード以上の場合は、キーワードを一重引用符 (') または二重引用符 (") で囲みます。キーワードが1ワードの場合には、引用符で囲む必要はありません。

たとえば、システムにログインしているユーザを表示するためのコマンド名を思い出せない場合には、次のいずれかのコマンドを入力すると、ログインしているユーザ名の表示に関するすべてのリファレンス・ページの名前と説明が表示されます。

```
$ apropos "logged in"
```

あるいは

```
$ man -k 'logged in'
```

上記のコマンドを入力するとシステムは次のような表示を出力します。

```
rusers (1) - Displays a list of users who are logged in to a remote machine
rwho (1)   - Shows which users are logged in to hosts on the local network.
who (1)    - Identifies users currently logged in
```

注意

上記の例でカッコで囲まれた番号は、リファレンス・ページのセクション番号を示します。リファレンス・ページ・ファイルの構造についての説明は、`man(1)` リファレンス・ページを参照してください。

上記の出力から、`rwwho`、`rusers` および `who` コマンドを使用することによりシステムにログインしているユーザを表示できることがわかります。`man` コマンドを使用すると、これらのコマンドの使用方法を調べることができます。

ファイルとディレクトリの概要

この章では、ファイル、ファイル・システム、テキスト・エディタについて説明します。

ファイルとは、コンピュータに格納されているデータの集まりです。代表的なファイルとしてはメモ、レポート、通信文、プログラムなどがあります。ファイル・システムとは、使いやすい形態でファイルをディレクトリに配置したものです。

テキスト・エディタとは、新しいファイルを作成したり、既存のファイルを修正したりするためのプログラムのことです。

この章を読むことにより、次の項目について理解することができます。

- vi テキスト・エディタによるファイルの作成方法
この章で作成するファイルは、本書の各章の例を通して作業する際に利用できます (2.2 節)。
- ファイル・システムの構成要素と概念 (2.3 節)。

以上の項目を理解することにより、情報の種類および作業方法に適したファイル・システムの設計が可能になります。

2.1 テキスト・エディタの概要

エディタとは、テキスト、プログラム、データなどを含むファイルを、作成あるいは変更するためのプログラムです。エディタには、ワード・プロセッサやパブリッシング・ソフトウェアのようなフォーマット機能およびプリント機能はありません。

テキスト・エディタを使用すると、次の操作が可能になります。

- ファイルの作成、読み取り、書き込み
- データの表示、検索
- データの追加、置換、削除

- データの移動およびコピー
- オペレーティング・システム・コマンドの実行

編集は編集バッファで行われ、ユーザはそのデータを保存または破棄することができます。

Tru64 UNIX オペレーティング・システムで利用できるテキスト編集プログラムは、`vi` および `ed` です。それぞれのエディタには、それぞれのテキスト表示方法、サブコマンドおよび規則があります。

`vi` についての詳細は、2.2 節および付録 A を参照してください。 `ed` については、付録 B を参照してください。

システムによっては、これ以外のエディタを使用できる場合もあります。詳細については、システム管理者に問い合わせてください。

2.2 vi テキスト・エディタによるサンプル・ファイルの作成

この節では、`vi` テキスト・エディタで 3 種類のファイルを作成する方法を紹介します。

この節の主な目的は、最小限のコマンドを使用してファイルを作成することです。これらのファイルは、本書の以降の章で紹介する例を通して実際に作業する際に利用します。`vi` についての詳細は、付録 A および `vi(1)` リファレンス・ページを参照してください。

注意

その他の編集プログラムに慣れている場合は、そのプログラムを使用して、次に説明する 3 つのサンプル・ファイルを作成してもかまいません。また、すでに編集プログラムで 3 つのサンプル・ファイルを作成している場合は、例で使用しているファイル名の代わりに、それらのファイル名を使用してもかまいません。

次の手順に従ってサンプル・ファイルを作成する際は、**ボールド (boldface: 太字)** 体で印刷されている文字列を入力してください。システムが表示するプロンプトおよび出力は、別の (like this のような) 書体で示してあります。

次の手順に従って、3つのサンプル・ファイルを作成してください。

1. シェル・プロンプトで `vi` と新しいファイル名をタイプして、Return キーを押し、`vi` プログラムを開始します。

```
$ vi file1 Return
```

`file1` ファイルは新しいファイルなので、カーソルは画面の最上部に表示されます。

```
~  
~  
~  
~  
~  
~
```

```
"file1" [New file]
```

画面上のチルダ (~) で始まる空白行は、その行にテキストを含んでいないことを示します。まだ何もテキストを入力していないので、全行の先頭にチルダが表示されます。

2. 新しいファイルにテキストを書き込むことをシステムに指示するため、英小字の `i` (insert text) をタイプしてください。ここでタイプした英字 `i` は、画面には表示されません。

次のテキストをタイプして、各行の終わりで Return キーを押してください。間違ってタイプした場合、次のテキスト行に移る前にそれを訂正するには、`'x'` キー、または他の `'vi'` 削除コマンドを使用します。

```
You start the vi program by entering Return  
the vi command optionally followed by the name Return  
of a new or existing file. Escape
```

```
~  
~  
~  
~  
~  
~
```

```
"file1" [New file]
```

3. 現在の作業が終了したことをシステムに知らせるには、Escape キーを押してください。最下行モードに移るには、コロン (:) をタイプします。

注意

端末あるいはワークステーションの設定によっては Escape キーは別の機能を実行するようにプログラムされている場合が

あります。このような場合、キーボード上のいずれかのファンクション・キーが "escape" の機能を実行するように設定されている可能性があります。この機能は `F11` キーに割り当てられていることがよくあります。Escape キーが正しく動作しない場合には、システム管理者に問い合わせてください。

コロンは次のように、画面の最下行にプロンプトとして表示されます。

```
You start the vi program by entering
the vi command optionally followed by the name
of a new or existing file.
~
~
~
~
~
~
:
```

- 次に、英小字の `w` を入力してください。ここで英字 `w` を入力すると、新しいファイルのコピーを現在のユーザ・ディレクトリに書き込む、つまり保存することをシステムに指示します。現在のディレクトリについての説明は、第 4 章を参照してください。

画面は次のようになります。

```
You start the vi program by entering
the vi command optionally followed by the name
of a new or existing file.
~
~
~
~
~
~
"file1" [New file] 3 lines, 111 characters
```

システムは新しいファイル名と、そのファイルに含まれるテキストの行数および文字数を表示します。

この時点で `vi` テキスト・エディタはまだ実行中です。続けて別の2つのサンプル・ファイルを作成します。手順は `file1` の作成方法と同じですが、入力するテキストが異なります。

5. コロン (:) をタイプします。コロンは画面の最下行にプロンプトとして表示されます。次に `vi file2` を入力して、2 番目のサンプル・ファイルを作成します。

システムは次のような画面を表示します。

```
~
~
~
~
~
~
~
"file2" No such file or directory

file2 No such file or directory というメッセージは、file2
が新しいファイルであることを示しています。
```

6. 英小字の `i` をタイプして、新しいファイルにテキストを書き込むことをシステムに指示してください。

次のテキストをタイプします。

```
If you have created a new file, you will find Return
that it is easy to add text. Escape
```

7. コロン (:) をタイプしてから英小字の `w` を入力して、ファイルを現在のディレクトリに書き込む、つまり保存してください。

画面は次のようになります。

```
If you have created a new file, you will find
that it is easy to add text.
~
~
~
~
~
~
~
"file2" [New file] 2 lines, 75 characters
```

8. 手順 5 に従って、3 番目のファイルを作成してください。ただし、ファイル名は `file3` とし、次のテキストをタイプしてください。

```
You will find that vi is a useful Return
editor that has many features. Escape
```

9. 次に、コロン (:) を入力してから `wq` コマンドを入力してください。

wq コマンドを実行するとファイルを書き込み、エディタを終了して、シェル・プロンプトへ戻ります。

2.3 ファイル，ディレクトリ，およびパス名

ファイルはコンピュータに格納されているデータの集まりです。コンピュータに格納されているファイルは，ファイリング・キャビネットに保管されている文書にたとえることができます。ユーザはそれを検索，オープン，処理，クローズしたり，ユニットとして格納することができます。いずれのコンピュータ・ファイルも，ユーザおよびシステムが参照するためのファイル名を持っています。

ファイル・システムは，使いやすい形態にファイルを配置したものです。一般に情報を整理する際には，コンピュータ・ファイル・システムのようなものを作成します。たとえば，マニュアルのファイル・システム (キャビネット，引出し，フォルダ，および文書) の構造は，コンピュータ・ファイル・システムの構造に類似しています。(ファイル・ストレージを管理しているソフトウェアもファイル・システムと呼ばれますが，この章ではこの意味で使用することはありません。システムによっては，このソフトウェアをファイル・マネージャと呼ぶ場合もあります。)

マニュアルであれコンピュータであれ，ファイル・システムを構成したユーザは，そのシステムの構造を理解しているので，迅速に特定の情報を見つけることができます。

ファイル・システムを理解するためには，まず，次の3つの概念を理解する必要があります。

- ファイルおよびファイル名
- ディレクトリおよびサブディレクトリ
- ツリー構造およびパス名

2.3.1 ファイルおよびファイル名

ファイルには，ドキュメントのテキスト，コンピュータ・プログラム，総勘定元帳のための記録，コンピュータ・プログラムの数値データあるいは統計的出力などのデータを含むことができます。

ファイル名には次に示す文字を除いて、どのような文字でも使用することができます。これらの文字は、シェルにとって特別な意味を表すため、ファイル名には使用しないでください。

- スラッシュ (/)
- バックスラッシュ (\)
- アンパサンド (&)
- 左右の山カッコ (< と >)
- 疑問符 (?)
- ドル記号 (\$)
- 左大カッコ ([)
- アスタリスク (*)
- チルダ (~)
- 縦線つまりパイプ記号 (|)
- 番号記号 (#)

ピリオドまたはドット (.) はファイル名の途中に使用できますが、通常、ファイル名の先頭には使用しません。ファイル名の先頭にドットを使用すると、そのファイルは隠しファイル (hidden file) となります。隠しファイルは、オプションを指定しないで `ls` コマンドを実行した際に、ファイル・リストに含まれません。シェルにとって特別な意味を持つ文字については、8.2.2 項および 8.3.2 項を参照してください。隠しファイルのリストについては、3.1.3 項を参照してください。

注意

Tru64 UNIX オペレーティング・システムはファイル名の太文字と小文字とを区別します。たとえば、次の 3 つのファイル名 `filea`、`Filea`、`FILEA` は、それぞれ異なるファイルを示します。

ファイル名には、そのファイルの内容を表す名前を使用することをお勧めします。たとえば、広告 (advertising) に関するメモを含むファイルの名前として `memo.advt` を使用すると、このファイルの内容が即座に推測できます。

これに対し、filea、fileb、filec などのファイル名を使用すると、そのファイルの内容を推測することは困難になります。

また、一貫したパターンに従って、関連するファイルを命名するのもよいでしょう。たとえば、複数の章で構成される広告に関するレポートがあり、各章がそれぞれ別のファイルとして存在する場合は、次の例のように命名してもよいでしょう。

```
chap1.advt  
chap2.advt  
chap3.advt
```

注意

ユーザが起動するプログラムの多くは、ドット(.)に続くファイル名の一部を使用します。これは拡張子と呼ばれ、ファイルの目的を示すインディケータとして使用されます。

ファイル名の最大長は、オペレーティング・システムで使用しているファイル・システムによって異なります。たとえば、ファイル・システムによっては最大 255 文字 (省略時の値) のファイル名が許されるものもあれば、最大 14 文字のファイル名しか許されないものもあります。

ファイル名の最大長は、意味のあるファイル名を付けるうえで重要な要素となります。詳細についてはシステム管理者に問い合わせてください。

2.3.2 ディレクトリおよびサブディレクトリ

ファイルはグループおよびサブグループに編成することができます。これは、マニュアルのファイル・システムにおけるキャビネット、引出し、フォルダなどに該当するものです。

このグループおよびサブグループは、それぞれディレクトリおよびサブディレクトリと呼ばれます。ディレクトリとサブディレクトリが適切に構成されていれば、ファイル内のデータの検索や操作が迅速に行えます。

ディレクトリは次の点でファイルと異なります。

- ディレクトリは構成のためのツールです。一方、ファイルはデータの格納場所です。

- ディレクトリにはファイルあるいは他のディレクトリ、またはその両方が含まれます。

ユーザがログインすると、システムはユーザを自動的に各ユーザのログイン・ディレクトリに入れます。このディレクトリは、ホーム・ディレクトリとも呼ばれます。また、システムは、ユーザの HOME 環境変数をログイン・ディレクトリの絶対パス名に設定します。このディレクトリは、各ユーザ・アカウントを作成する際に、システム管理者によって作成されます。ただし、ユーザのログイン・ディレクトリの下にすべてのファイルを格納するのが、ファイルを構成する最も有効な方法であるとは限りません。

システムで作業していると、新たにディレクトリやサブディレクトリを作成して、ファイルを新しいグループに編成する必要がでてくるかもしれません。たとえば、自動車会社の営業部に所属するユーザが4種類の車種を担当している場合には、ログイン・ディレクトリの下に各車種に対応するサブディレクトリを作成し、それぞれのサブディレクトリに、各車種に関するメモ、レポート、および売上データを格納すると便利です。

実用的なディレクトリ構造でファイルを編成すると、ディレクトリ間を容易に移動して作業することができます。ディレクトリの作成とディレクトリ間の移動については、第4章を参照してください。

2.3.3 現在のディレクトリ (作業ディレクトリ) の名前の表示 (pwd コマンド)

現在ユーザが作業しているディレクトリを、現在のディレクトリまたは作業ディレクトリと呼びます。

現在どのディレクトリで作業しているのか、あるいはそのディレクトリがファイル・システムのどこに位置するのかわからない場合は、次のように pwd (print working directory) コマンドを入力してください。

```
$ pwd
```

このコマンドを入力すると、システムは次のような形式で現在のディレクトリ名を表示します。

```
/usr/msg
```

この例では、ユーザが現在 `usr` ディレクトリの下にある `msg` ディレクトリで作業していることを示しています。

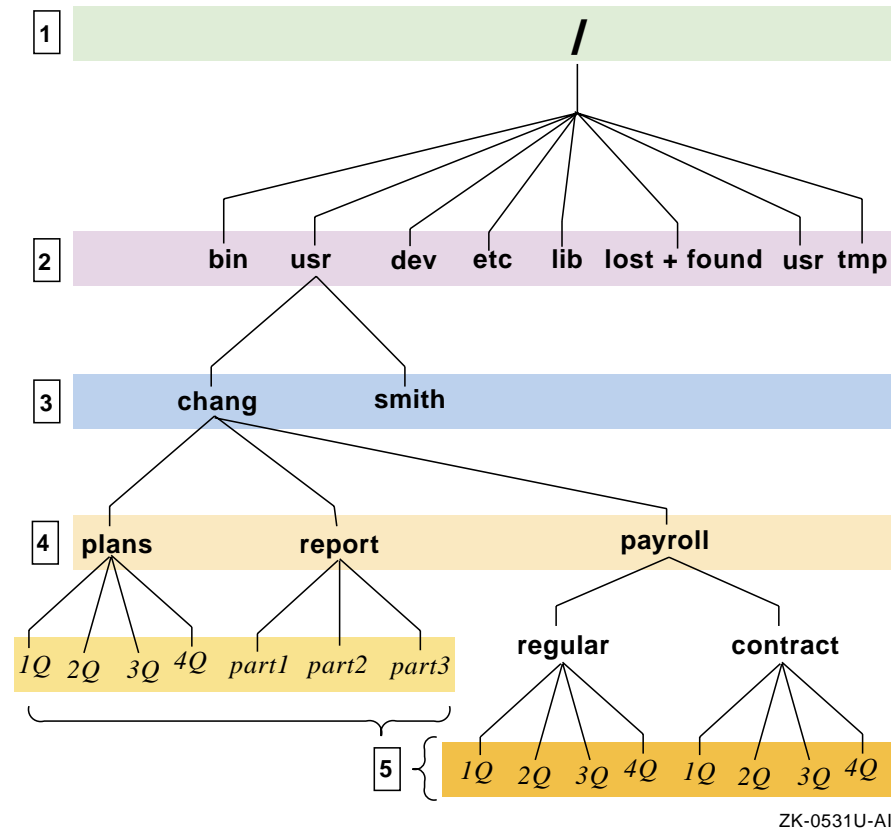
/usr/msg という表記は、作業ディレクトリのパス名と呼ばれます。パス名については 2.3.4 項を参照してください。pwd コマンドについての詳細は、pwd(1) リファレンス・ページを参照してください。

2.3.4 ツリー構造ファイル・システムおよびパス名

ファイル・システムにおけるファイルとディレクトリは、樹木を逆にしたような構造で階層的に配置されています。この配置をツリー構造と呼びます。ディレクトリ構造についての詳細は、hier(5) リファレンス・ページを参照してください。

図 2-1 は、典型的なファイル・システムのツリー構造の例です。この例では、ディレクトリ名は太字体で、ファイル名はイタリック体で示されています。

図 2-1: 典型的なファイル・システム



- ① 図 2-1 に示されているファイル・システムの最上部にあるディレクトリは、ルート・ディレクトリと呼ばれます。ルート・ディレクトリはスラッシュ (/) で表します。
- ② ルート・ディレクトリの次のレベルには 8 つのディレクトリがあり、それぞれにサブディレクトリおよびファイルが存在します。図 2-1 では、usr ディレクトリの下にあるサブディレクトリのみを示しています。このシステムでは、usr ディレクトリの下に各ユーザのログイン・ディレクトリが存在します。
- ③ ツリー構造の 3 番目のレベルには、2 人のシステム・ユーザ smith および chang のログイン・ディレクトリがあります。smith と chang は、ログインするとこのディレクトリに入ります。
- ④ 図 2-1 の 4 番目のレベルには、chang のログイン・ディレクトリの下に、plans、report、payroll の 3 つのディレクトリが存在します。
- ⑤ ツリー構造の 5 番目のレベルには、ファイルとディレクトリの両方が含まれています。plans ディレクトリには、各四半期のプランに関する 4 つのファイルが含まれ、report ディレクトリには、レポートを構成する 3 つのファイルが含まれています。また、5 番目のレベルには、payroll ディレクトリの 2 つのサブディレクトリ regular および contract が含まれています。

上位のディレクトリは親ディレクトリと呼ばれます。たとえば、図 2-1 では、ディレクトリ plans、report、および payroll はすべて chang を親ディレクトリとしています。

パス名はファイル・システム内におけるディレクトリやファイルの位置を指定します。たとえば、ディレクトリ X にあるファイル A での作業を終了して、ディレクトリ Y にあるファイル B へ移る場合に、ファイル B のパス名を指定すると、オペレーティング・システムはこのパス名を使用してシステム・ファイル内を探索し、ファイル B を探します。

パス名は、ディレクトリ名をスラッシュ (/) で区切って表し、パス名の最後はディレクトリ名またはファイル名になります。パス名の最初の要素は、システムがどこから探索を始めるかを指定し、最後の要素は探索のターゲットを指定しています。次のパス名の例は、図 2-1 のディレクトリ構造のものです。

```
/usr/chang/report/part3
```

最初のスラッシュ (/) はルート・ディレクトリを表し、探索の始点を示しています。このパス名は、ルート・ディレクトリの下 `user` ディレクトリの下 `chang` ディレクトリの下 `report` ディレクトリにある `part3` ファイルを検索することを示しています。

次のような操作を行う場合には、パス名を使用して操作する対象を指定します。

- 現在のディレクトリの変更
- ファイルへのデータの送信
- ファイルのコピーおよび移動

スラッシュ (/) (ルート・ディレクトリを表す記号) で始まるパス名は、完全パス名または絶対パス名と呼ばれます。完全パス名はファイルまたはディレクトリの完全な名前と考えることもできます。完全パス名を指定すれば、ファイル・システムのどこで作業していても、間違いなくファイルまたはディレクトリを見つけ出すことができます。

ファイル・システムでは、相対パス名を使用することもできます。相対パス名は現在のディレクトリからの相対的な位置を示します。したがって、相対パス名はルート・ディレクトリを表す / では始まりません。

相対パス名は次のいずれかの方法で指定することができます。

- 現在のディレクトリ内のファイル名
- 現在のディレクトリより 1 レベル下のディレクトリ名で始まるパス名
- .. で始まるパス名 (.. は親ディレクトリを示す相対パス名)
- . で始まるパス名 (. は現在のディレクトリを示す相対パス名)

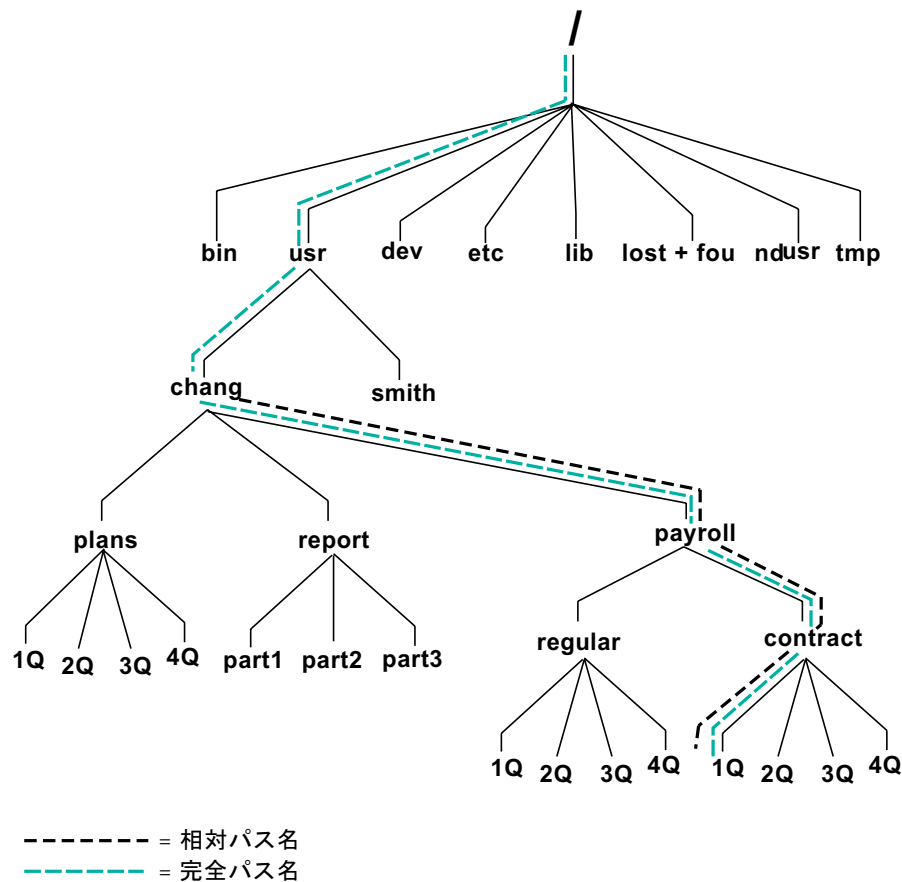
この相対パス名表記法は、現在のディレクトリで、ユーザ自身のバージョンのオペレーティング・システム・コマンド (たとえば `./ls`) を実行する場合などに便利です。

各ディレクトリには .. および . が必ず含まれています。

たとえば、図 2-2 のディレクトリ構造において、現在のディレクトリが `chang` である場合、`contract` ディレクトリにある `1Q` ファイルの相対パス名は `payroll/contract/1Q` です。この相対パス名を、同じファイルの完

全パス名 `/usr/chang/payroll/contract/1Q` と比較すると、相対パス名を使用の方が、入力の手間が少なく便利であることがわかります。

図 2-2: 相対パス名と完全パス名



ZK-0532U-AIJ

C シェル および Korn または POSIX シェルでは、相対パス名の先頭にチルダ (~) を使用することもできます。チルダは、単独で使った場合、ユーザのログイン・ディレクトリ (ホーム・ディレクトリ) を示します。チルダの後にユーザ名を指定すると、同じシステム上の別のユーザのログイン・ディレクトリ (ホーム・ディレクトリ) を示します。

たとえば、ユーザ自身のログイン・ディレクトリを指定する場合は、チルダのみを使用します。ユーザ `chang` のログイン・ディレクトリを指定する場合は、`~chang` と指定します。

相対パス名の使用方法についての詳細は、第 4 章を参照してください。

注意

システム内の他のユーザが所有するファイルあるいはディレクトリに対するアクセスは、それらのファイルあるいはディレクトリに設定されているファイル許可に左右されます。ファイルおよびディレクトリの許可についての詳細は、第 5 章を参照してください。

また、システムにエンハンスド・セキュリティ機能が設定されている場合は、ファイルおよびディレクトリへのアクセスに影響することがあります。その場合には、システム管理者に問い合わせてください。

2.4 パターン照合によるファイル指定

コマンドには引数としてファイル名を指定するものがあります。複数のファイル名をコマンドの引数として使用する場合は、たとえば次の例のように、各ファイルの完全な名前を入力する方法があります。

```
$ ls file1 file2 file3
```

ただし、ファイル名に共通なパターンがある場合 (この例では `file` というプレフィックス) には、シェルにそのパターンを照合させて、ファイル名のリストを生成し、自動的にそのファイル名リストを引数としてコマンドに渡すことができます。

たとえばアスタリスク (*) はワイルドカードと呼ばれ、どんな文字列にも対応します。次の例では、`ls` コマンドは、現在のディレクトリにあるファイルのうち、プレフィックス `file` を含むすべてのテキスト・ファイル名を検索します。

```
$ ls file*
```

`file*` は、`file` で始まり任意の文字列で終わるすべてのファイル名と一致します。シェルはこのパターンに一致するすべてのファイル名を `ls` コマンドの引数として渡します。

このように、引数としてファイル名を指定する場合は、必ずしも各ファイルの完全名を入力する必要はありません。`ls` コマンドの引数として完全なファイル名を指定した場合も `file*` を指定した場合も、コマンド実行の結果

は同じです。つまり、そのディレクトリにある file というプレフィックスの付いたすべてのファイルを ls の引数として渡します。

パターン照合の一般規則には 1 つだけ例外があります。ドット (.) で始まるファイル名を指定する場合は、ドットを明示的に照合する必要があります。ls * コマンドを実行した場合、現在のディレクトリにあるファイルのうちドットで始まるファイル名は表示しません。ls -a コマンドを実行すると、ファイル名がドットで始まるものを含めてすべてのファイル名を表示します。

この制限は、シェルが自動的に相対ディレクトリ名を照合するのを防ぐために設定されています。つまり、現在のディレクトリを表すドット (.) および親ディレクトリを表すドット・ドット (..) を照合するのを防ぐのが目的です。相対ディレクトリ名についての詳細は、第 4 章を参照してください。

パターンがどのファイル名とも一致しない場合、シェルはそのことを知らせるメッセージを表示します。

Tru64 UNIX オペレーティング・システムのシェルでは、アスタリスク (*) 以外にもパターン照合文字が用意されています。Tru64 UNIX オペレーティング・システムで利用できるすべてのパターン照合文字を次の表 2-1 で示します。

表 2-1: パターン照合文字

文字	機能
*	空文字列も含めてすべての文字列と一致する。 たとえば、th*はth, theodore, theresaなどと一致する。
?	すべての単一の文字と一致する。 たとえば、304?bは304Tb, 3045b, 304Bbなどと一致する。つまり、304で始まって b で終わり、その間に文字が 1 つあるすべての文字列と一致する。
[...]	大カッコで囲まれた文字のいずれか 1 つと一致する。 たとえば、[AGX]*は、現在のディレクトリ内の A, G, または X で始まるすべてのファイル名と一致する。

表 2-1: パターン照合文字 (続き)

文字	機能
[.-]	指定した範囲に含まれる文字と一致する。 範囲の指定は現在のロケールで定義されている方法に依存する。ロケールについての詳細は、付録 C を参照。 たとえば、[T-W]* は、現在のディレクトリ内の T, U, V, または W で始まるすべてのファイル名と一致する。
[!...]	指定した文字を除くすべての単一の文字と一致する。 たとえば、[!abyz]* は、現在のディレクトリ内にある、a, b, y および z 以外の文字で始まるすべてのファイル名と一致する。 このパターン照合は Bourne, Korn, および POSIX シェルでのみ利用可能。

国際化されたオペレーティング・システムである Tru64 UNIX オペレーティング・システムは、表 2-2に示すパターン照合機能も備えています。

表 2-2: 国際化パターン照合文字

文字	機能
[:class:]	文字クラス名をデリミタ [: および :] で囲んで指定すると、指定したクラスの文字セットのいずれとも一致する。 サポートされるクラスは alpha, upper, lower, digit, alnum, xdigit, space, print, punct, graph, and cntrl。 たとえば、alpha という文字クラス名は、現在のロケールで定義されているすべての英字 (大文字および小文字) と一致する。アメリカ英語をベースとしたロケールを実行している場合、alpha はアルファベット (A-Z, a-z) のどの文字とも一致する。
[=char=]	デリミタ [= および =] で囲まれた文字は、すべての等価クラス文字と一致する。 等価クラスとは、すべて同じ一次ロケーションにソートされる照合要素の集まりである。等価クラスは一般に、一次-二次ソートに対処するように設計されている。つまり、フランス語のように、文字グループを定義して、同じ一次ロケーションにソートした後、発音記号を含めて二次ソートする言語のためのクラスである。

国際化パターン照合文字についての詳細は、`grep(1)` リファレンス・ページを参照してください。Tru64 UNIX オペレーティング・システムの国際化機能についての詳細は、付録 C を参照してください。



3

ファイルの管理

この章では、システムでファイルを管理する方法について説明します。この章の説明を理解すると、次の操作が可能になります。

- ファイルのリスト (3.1 節)
- ファイルの表示 (3.2 節)
- ファイルの印刷 (3.3 節)
- ファイルのリンク (3.4 節)
- ファイルのコピー (3.5 節)
- ファイルの名前変更、移動 (3.6 節)
- ファイルの比較 (3.7 節)
- ファイルのソート (3.8 節)
- ファイルの削除 (3.9 節)
- ファイル・タイプの決定 (3.10 節)

この章で説明する例に従って (同じ順序で) 実際に操作することにより、ファイルの管理について理解することができます。画面に表示される情報が本書の説明と一致するように、必ずすべての例を実行してください。

例を実行するためにはまずログインしなければなりません。また、第 2 章で作成した次の 3 つのファイルが、ログイン・ディレクトリに存在する必要があります。

- file1
- file2
- file3

ログイン・ディレクトリに存在するファイルのリストを作成するためには、ls コマンドを入力してください。ls コマンドについては、3.1 節で説明

します。例とは異なるファイル名を使用している場合は、適当に置き換えてコマンドを実行してください。

ログイン・ディレクトリに戻る場合は、次のように `cd` (change directory) コマンドを入力してください。

```
$ cd
```

なお、上記の例でドル記号 (\$) はシェル・プロンプトを表しています。システムによっては表示されるシェル・プロンプトが異なる場合もあります。

また、この章の例に取り組む前に、ログイン・ディレクトリの下にサブディレクトリ `project` を作成しておいてください。ディレクトリを作成するコマンドは `mkdir` (make directory) です。ログイン・ディレクトリから次のように `mkdir` コマンドを入力します。

```
$ mkdir project
```

`cd` コマンドおよび `mkdir` コマンドについての詳細は、それぞれ 4.2 節または `cd(1)` リファレンス・ページ、および 4.1 節または `mkdir(1)` リファレンス・ページを参照してください。

3.1 ファイルのリスト (`ls` コマンド)

`ls` コマンドは、ディレクトリの内容を表示します。このコマンドは、現在のディレクトリにあるファイルおよびサブディレクトリのリストを作成するためのものです。現在のディレクトリ以外のディレクトリの内容など、他の種類の情報を表示することもできます。

`ls` コマンドの構文は次のとおりです。

ls

`ls` コマンドには、フラグと呼ばれる多数のオプションが用意されています。フラグを使用すると、ディレクトリの内容についての情報をいくつかの形式で表示することができます。`ls` コマンドのフラグについての詳細は、3.1.3 項を参照してください。

3.1.1 現在のディレクトリの内容のリスト

現在のディレクトリの内容をリストする場合は、次のように入力します。

```
$ ls
```


ls コマンドをフラグなしで使用すると、現在のディレクトリに存在するファイル名およびディレクトリ名がリストされます。

```
$ ls
file1      file2      file3      project
$
```

次のコマンド形式を使用して、現在のディレクトリの内容の一部をリストすることもできます。

ls filename

filename 変数にはファイル名あるいはスペースで区切った複数のファイル名のリストを指定します。パターン照合文字を使用してファイルを指定することもできます。パターン照合文字については第 2 章を参照してください。

たとえば、文字列 *file* で始まるファイル名をリストする場合は、次のコマンドを入力します。

```
$ ls file*
file1 file2 file3
$
```

3.1.2 他のディレクトリの内容のリスト

現在のディレクトリ以外のディレクトリの内容のリストをする場合は、次のコマンドを入力します。

ls dirname

dirname 変数にはディレクトリのパス名を指定します。

次に示すのは、現在のディレクトリがユーザのログイン・ディレクトリである場合に */users* ディレクトリの内容を表示する例です。システムには、*/users* ディレクトリと同じ名前の別のディレクトリが存在する可能性があります。*/users* のスラッシュ (/) は、ルート・ディレクトリから探索を始めることをシステムに指示しています。

```
$ ls /users
amy      beth      chang     george    jerry     larry
mark     monique  ron
$
```

ls コマンドは、現在のロケールで定義されている照合順で、ディレクトリ名およびファイル名をリストします。ロケールについての詳細は、付録 C を参照してください。

3.1.3 ls コマンドのフラグ

通常、ls コマンドは指定したディレクトリに含まれているファイルとディレクトリの名前を表示します。ただし ls には、リストする項目についての追加情報を表示したり、システムによるリスト表示の方法を変えるためのいくつかのフラグが用意されています。

ls コマンドにフラグを指定する場合は、次の構文を使用してください。

ls *-flagname*

-flagname 変数には 1 つ以上のフラグ (オプション) を指定します。たとえば、*-l* フラグはディレクトリの内容についての詳細なリストを作成します。

複数のフラグを使用する場合は、フラグ名を 1 つの文字列として一緒に入力してください。次に例を示します。

```
$ ls -lta
```

表 3-1 に、ls コマンドでよく使用する便利なフラグを示します。

表 3-1: ls コマンドのフラグ

フラグ	機能
-a	隠しファイルも含め、すべてのエントリをリストする。このフラグを指定しない場合、ls コマンドは .profile、.login、および相対パス名のようにドット (.) で始まるエントリの名前はリストしない。
-l	詳細なリストを表示する。具体的には、リストされる各ファイルおよびディレクトリのタイプ、許可、リンク数、所有者、グループ、サイズ、および最後の修正日時を示す。
-r	ソート順を逆にしてリストする。ls -r は ls の逆順でリストし、ls -tr は ls -t の逆順でリストする。
-t	ファイルおよびディレクトリを、名前による照合順序ではなく、最後に修正された日時に従って、最新のものから順にソートする。
-F	ファイルがディレクトリの場合は、各ファイル名の後ろに / (スラッシュ) を付け、ファイルが実行できる場合は、各ファイル名の後ろに * (アスタリスク) を付ける。
-R	すべてのサブディレクトリを再帰的にリストする。各ディレクトリおよびサブディレクトリを下って、ディレクトリ・ツリー全体のリストを提供する。

例 3-1 に示すのは、*-l* フラグを使用して現在のディレクトリの詳細なリストを表示する例です。

例 3-1: ディレクトリの詳細なリスト (ls -l)

```
$ ls -l
total 4
-rw-r--r-- 1 larry system 101 Jun 5 10:03 file1
-rw-r--r-- 1 larry system 75 Jun 5 10:03 file2
-rw-r--r-- 1 larry system 65 Jun 5 10:06 file3
drwxr-xr-x 2 larry system 32 Jun 5 10:07 project
```

1

このディレクトリに含まれるファイルが占める 512 バイト・ブロック数。

2

1 — 各ファイルへのリンク数。 ファイル・リンクについての詳細は 3.4 節を参照。

3

101 — ファイルのバイト数。

4

larry — ファイルの所有者のユーザ名。 実際には, larry の代りにユーザのユーザ名が表示されます。

5

system — ファイルが属するグループ。 実際には, system の代りにユーザが所属するグループ名が表示されます。

6

file3 — ファイルまたはディレクトリの名前。

7

Jun 5 10:06 — ファイルが作成された日時。 あるいは最後に修正された日時。

8

drwxr-xr-x — 各ファイルまたはディレクトリに設定されているファイル・タイプと許可。 このフィールドの最初の文字はファイル・タイプを示します。:

- (ハイフン) 通常ファイル

b ブロック型特殊ファイル

c 文字型特殊ファイル

d ディレクトリ

l シンボリック・リンク

p パイプ型特殊ファイル (先入れ先出し)

s ローカル・ソケット

残りの文字列は, 所有者, グループ, およびその他のユーザに対して, 読み取り許可 (r), 書き込み許可 (w), 実行許可 (x) がそれぞれ設定されているかどうかを示します。 (-) が表示された場合, 対応する許可が設定されていません。

さらに, その他の許可情報を表示することもできます。 許可についての詳細は第 5 章を参照。

Tru64 UNIX オペレーティング・システムでは, ls コマンドには上記以外にもフラグが用意されています。ls コマンド・フラグについての詳細は, ls(1) リファレンス・ページを参照してください。

3.2 ファイル内容の表示

テキスト・エディタを使用して、システムに格納されているテキスト・ファイルの内容を参照することができます。ファイル内容を参照するだけで変更を行わない場合は、いくつかのオペレーティング・システムのコマンドを使用することができます。以降の各項で、これらのコマンドについて説明します。

3.2.1 フォーマットिंगを伴わないファイル表示 (**pg**, **more**, **cat** コマンド)

次のコマンドは、内容の体裁を制御する特殊文字を一切付け加えずに、ファイルをそのまま表示します。

- `pg`
- `cat`
- `more`
- `page`

フォーマット処理を行ってファイルを表示するコマンドについては、3.2.2 項を参照してください。

フォーマット処理を伴わないファイル表示コマンドの構文は次のとおりです。

command filename

変数 *command* には `pg`、`more`、`page`、または `cat` のいずれかのコマンド名を指定します。 *filename* 変数にはファイル名を指定します。複数のファイル名を指定する場合は、スペースで区切ります。パターン照合文字を使用してファイルを指定することもできます。パターン照合文字については第 2 章を参照してください。

`pg` コマンドを使用すると、1 つまたは複数のファイルの内容を参照することができます。例 3-2 に示すのは、`pg` コマンドを使用して、ログイン・ディレクトリにある `file1` ファイルの内容を表示する例です。

例 3-2: `pg` コマンドの出力 (1 つのファイル)

```
$ pg file1
You start the vi program by entering
the command vi, optionally followed by the name
of a new or existing file.
```

例 3-2: pg コマンドの出力 (1 つのファイル) (続き)

\$

file1 および file2 の両方のファイルの内容を表示するには、両方のファイル名をコマンド行に指定します。表示するファイルの行数が多くて一画面におさまらない場合、pg コマンドは各画面を表示するたびにポーズを置きます。次の画面を表示するには Return キーを押します。現在のファイルの終わりに到達すると、次のファイルの名前がプロンプトとして表示されます。現在のファイルの終わりで Return キーを押すと、次のファイルの始まりが表示されます。pg コマンドでは、コマンド行に指定した順にファイルが表示されます。例 3-3 の (EOF) : (end of file) は、現在のファイルの終端を意味しています。

例 3-3: pg コマンドの出力 (複数のファイル)

```
$ pg file1 file2
You start the vi program by entering
the command vi, optionally followed by the name
of a new or existing file.
(EOF): Return
(Next file: file2) Return
If you have created a new file, you will find
that it is easy to add text.
(EOF): Return
$
```

Next file: *filename* プロンプトに対して -n オプションを入力すると、次のファイルではなく、以前のファイルが表示されます。

more コマンドもコマンド行に複数のファイルを指定することができます。more コマンドは pg コマンドと同じような方法で長いファイルを扱います。ファイルの行数が多く、1 画面におさまらない場合、more コマンドはポーズを置き、それまでにファイルの何パーセントを表示したかを知らせるメッセージを表示します。この時点で、次のいずれかの操作を行うことができます。

- ファイルの残りを 1 ページずつ表示させる場合は、スペース・バーを押す。
- 1 行ずつ表示させる場合は、Return キーを押す。

- ファイルの表示を中止する場合は、q をタイプする。

page コマンドは more コマンドとほぼ同様ですが、ファイルの行数が多くて 1 ページに収まらない場合、page コマンドは画面をクリアして画面の最上部から表示を始めます。ご使用のオペレーティング・システム環境や表示デバイスによっては、この違いはほとんど認識されない場合もあります。

cat コマンドを使用してテキストを表示することもできます。ただし、このコマンドはファイルのページ付けを行わないため、1 画面よりも長いファイルを表示すると一挙にファイルの最後まで表示され、内容を読むことができません。

このような場合は、Ctrl/S を押して表示を停止させてください。こうすることにより、テキストを読むことができます。ファイルの残りの部分を表示させたいときは、Ctrl/Q を押してください。長いファイルを表示させる場合は、cat コマンドではなく pg コマンド、more コマンド、または page コマンドを使用するのがよいでしょう。

pg、more、page、および cat コマンドには、それぞれオプションが用意されています。詳細については、リファレンス・ページの cat(1)、more(1)、page(1)、および pg(1) を参照してください。

3.2.2 フォーマットを伴うファイル表示 (pr コマンド)

フォーマットとは、ファイルの内容を表示または印刷する際に、見た目を制御するための処理です。pr コマンドは、簡潔で便利なスタイルにファイルをフォーマットします。

注意

pr コマンドはテキスト・フォーマット情報は解釈しないため、ファイル内にこれらのマクロがある場合には注意してください。このコマンドは nroff や troff のようにファイルをフォーマットするわけではありません。ワード・プロセッサやデスクトップ・パブリッシング・ソフトウェアで作成されたファイルは、pr コマンドでは認識されません。

簡単なフォーマット処理を行ってファイルを表示する場合は、次の構文で pr コマンドを実行します。

pr *filename*

filename 変数には、ファイル名、ファイルの相対パス名、ファイルの完全パス名、あるいはスペースで区切ったファイル名リストを指定します。使用するコマンド形式は、ファイルが現在のディレクトリとの関係でどこに位置しているかによって変わります。パターン照合文字を使用してファイルを指定することもできます。パターン照合については第 2 章を参照してください。*filename* をダッシュ (-) として指定することができます。この場合、**pr** コマンドは、ファイルの終了を示すマーク (通常は Ctrl/D) で入力を終了するまで、端末からの入力を読み取ります。

pr コマンドをオプションなしで実行すると、次の処理を行います。

- ファイルの内容を各ページに分割する。
- 各ページの上部に見出しとして、日時、ページ番号、およびファイル名を表示する。
- ページの終わりに 5 行のブランク行を追加する。

pr コマンドを使用してファイルを表示すると、内容が一挙にスクロールして読むことができないかもしれません。その場合には、**pr** コマンドと **more** コマンドを併用することによって、フォーマットされたファイルを読むことができます。**more** コマンドは、1 画面分のテキストを表示した時点でポーズを置くようシステムに指示します。

たとえば、長いファイル `report` を表示し、画面が一杯になった時点で停止するように指示する場合は、次のコマンドを入力します。

```
$ pr report | more
$
```

システムが最初の 1 画面分のテキストを表示して停止したら、スペース・バーを押して次の画面を表示します。上記のコマンドは、パイプ記号 (|) を使用して、**pr** コマンドからの出力を **more** コマンドへの入力として使用しています。パイプについての詳細は、7.4.2 項を参照してください。

pr コマンドには、より美しいフォーマットでファイルを表示するためのフラグが用意されています。表 3-2 に、**pr** で使用できるいくつかのフラグを示します。

表 3-2: pr コマンドのフラグ

フラグ	機能
<code>+page</code>	<p><code>page</code> に指定したページからフォーマットを開始する。このフラグを省略した場合、1 ページ目からフォーマットを開始する。</p> <p>たとえば、<code>pr +2 file1</code> というコマンドは、<code>file1</code> の 2 ページ目からフォーマットを開始する。</p>
<code>-column</code>	<p>各ページを <code>column</code> に指定した数のカラムでフォーマットする。このフラグを省略した場合、<code>pr</code> コマンドは各ページを 1 つのカラムでフォーマットする。</p> <p>たとえば、<code>pr -2 file1</code> というコマンドは、<code>file1</code> を 2 つのカラムでフォーマットする。</p>
<code>-m</code>	<p>指定されたすべてのファイルを、各ファイル 1 カラムで同時に横に並べてフォーマットする。</p> <p>たとえば、<code>pr -m file1 file2</code> というコマンドは、<code>file1</code> の内容を左のカラムに、<code>file2</code> の内容を右のカラムに表示する。</p>
<code>-d</code>	<p>出力をダブル・スペースでフォーマットする。このフラグを省略した場合、出力はシングル・スペースでフォーマットされる。</p> <p>たとえば、<code>pr -d file1</code> というコマンドは、<code>file1</code> をダブル・スペースでフォーマットして表示する。</p>
<code>-f</code>	<p>新しいページに進む場合に改ページ文字を使用する。このフラグを省略した場合、<code>pr</code> コマンドは行送り文字のシーケンスを発行する。標準出力が端末の場合、最初のページを開始する前にポーズを置く。</p>
<code>-F</code>	<p>新しいページに進む場合に改ページ文字を使用する。このフラグを省略した場合、<code>pr</code> コマンドは行送り文字のシーケンスを発行する。標準出力が端末の場合、最初のページを開始する前にポーズは置かない。</p>
<code>-w num</code>	<p>行の長さを <code>num</code> に指定したカラム数に設定する。このフラグを省略した場合、行の長さは 72 カラムになる。</p> <p>たとえば、<code>pr -w 40 file1</code> というコマンドは、<code>file1</code> ファイルの行の長さを 40 カラムに設定する。</p>

表 3-2: pr コマンドのフラグ (続き)

フラグ	機能
<code>-o num</code>	<code>num</code> に指定したカラム位置だけ各行をオフセット (インデント) する。このフラグを省略した場合、オフセットは 0 カラム位置になる。 たとえば、 <code>pr -o 5 file1</code> というコマンドは、 <code>file1</code> ファイルの各行を 5 スペースだけインデントする。
<code>-l num</code>	ページ長を <code>num</code> に指定した値に設定する。このフラグを省略した場合、ページ長は 66 行になる。 たとえば、 <code>pr -l 30 file1</code> というコマンドは、 <code>file1</code> ファイルのページ長を 30 行に設定する。
<code>-h string</code>	各ページの上部に表示されるヘッダ (タイトル) として、ファイル名ではなくこのフラグの後に指定した文字列を使用する。 <code>string</code> がブランクまたは特殊文字を含んでいる場合は、文字列を一重引用符 (<code>' '</code>) で囲む。 たとえば、 <code>pr -h 'My Novel' file1</code> というコマンドは、タイトルとして "My Novel" を指定する。
<code>-t</code>	ヘッダおよび各ページ末のブランク行をフォーマットしないことを指定する。 たとえば、 <code>pr -t file1</code> というコマンドは、ヘッダおよび各ページ末のブランク行を省略して、 <code>file1</code> ファイルをフォーマットするよう指定する。
<code>-schar</code>	各カラムをブランク・スペースではなく文字 <code>char</code> で区切る。特殊文字を使用する場合は一重引用符で囲む。 たとえば、 <code>pr -s '*' file1</code> というコマンドは、アスタリスクでカラムを区切るよう指定する。

`pr` コマンドには、同時に複数のフラグを指定することができます。次の例では、下記のような形式で `file1` をフォーマットするように `pr` コマンドに指示しています。

- ダブル・スペース (`-d`)
- タイトルをファイル名ではなく "My Novel" とする (`-h`)。

```
$ pr -dh 'My Novel' file1
$
```

`pr` コマンドおよびそのフラグについての詳細は、`pr(1)` リファレンス・ページを参照してください。

3.3 ファイルの印刷 (`lpr` , `lpq` , `lprm` コマンド)

ファイルをシステム・プリンタに送る場合は、`lpr` コマンドを使用します。`lpr` コマンドはプリント・キューにファイルを登録します。プリント・キューとは、印刷されるのを待っているファイルのリストです。`lpr` コマンドでファイルをキューに登録しておけば、そのファイルが印刷されるのを待ちながら、システムで他の作業を続けることができます。

`lpr` コマンドの一般的な構文は次のとおりです。

`lpr filename`

filename 変数には、ファイル名、ファイルの相対パス名、ファイルの完全パス名、あるいはスペースで区切ったファイル名リストを指定します。使用するフォーマットは、現在のディレクトリとの関係でファイルがどこに位置しているかによって変わります。パターン照合文字を使用してファイルを指定することもできます。パターン照合については第 2 章を参照してください。

システムに複数のプリンタが接続されている場合は、次の形式で、ファイルをどのプリンタで印刷するかを指定します。

`lpr -P printername filename`

`-P` フラグは、ファイルを印刷するプリンタを指定するためのフラグです。*printername* 変数にはプリンタ名を指定します。プリンタ名は、プリンタの場所を示すものであったり (`southmailroom` など)、管理者の名前であったり、あるいはその他の説明的な名称であったりします。システムで利用可能なプリンタの種類が複数ある場合は、それらの機能を示す `slide` や `color` などの名前を割り当てる場合もあります。ご使用のシステムで利用できるプリンタの構成については、システム管理者にお問い合わせください。

システムに複数のプリンタが接続されている場合は、そのうちの 1 つが省略時のプリンタとして設定されています。特定の *printername* が入力されない場合、プリント要求はその省略時のプリンタに送られます。システムで利用できるプリンタの名前を調べるには、`lpstat` コマンドを使用します。

例 3-4 に示すのは、1 つあるいは複数のファイルを `lp0` というプリンタで印刷する場合の例です。

例 3-4: lpr コマンドの使用

```
$ lpr -Plp0 file1 1
$ lpr -Plp0 file2 file3 2
$
```

- 1 最初の lpr コマンドを実行すると、file1 を lp0 プリンタに送り、\$ プロンプトを表示します。
- 2 2 番目の lpr コマンドを実行すると、file2 ファイルおよび file3 ファイルを同じプリント・キューに送り、ファイルの印刷が終わる前にシェル・プロンプトを表示します。

lpr コマンドにフラグを指定することによって、ファイルの印刷方法を制御することができます。lpr コマンドのフラグは次の形式で使します。

lpr *flag filename*

lpr コマンドで利用できるフラグのいくつかを表 3-3 に示します。lpr コマンド・フラグについての完全な説明は、lpr(1) リファレンス・ページを参照してください。

表 3-3: lpr コマンドのフラグ

フラグ	機能
-#num	ファイルを num 部印刷する。このフラグを省略した場合、lpr はファイルを 1 部印刷する。たとえば、lpr -#2 file1 というコマンドは、file1 ファイルを 2 部印刷する。
-wnum	行の長さを num カラムに設定する。このフラグを省略した場合、行の長さは 72 カラムになる。たとえば、lpr -w40 file1 というコマンドは、file1 ファイルを行の長さ 40 カラムで印刷する。
-inum	各行を num スペース位置だけオフセット (インデント) する。このフラグを省略した場合、オフセットは 8 スペースになる。たとえば、lpr -i5 file1 というコマンドは、各行を 5 スペースずつインデントして file1 ファイルを印刷する。
-P	pr をフィルタとして使用して、ファイルをフォーマットする。

表 3-3: lpr コマンドのフラグ (続き)

フラグ	機能
-T 'string'	pr が指定するヘッダとして、ファイル名ではなく -T の後の文字列を使用する。このフラグを指定する場合は、-p フラグを同時に使用する。文字列に空白または特殊文字が含まれる場合は、一重引用符(')で囲む。たとえば、lpr -p -T 'My Novel' file1 というコマンドは、タイトルとして "My Novel" を指定する。
-m	ファイルの印刷が完了したらメールを送る。たとえば、lpr -m file1 コマンドを実行すると、file1 ファイルの印刷が完了したらメールが送信される。

lpr コマンドを入力すると、プリント要求はプリント・キューに登録されます。

登録されたプリント要求のプリント・キューにおける位置を知りたい場合は、lpq コマンドを使用します。プリント・キューの内容を表示するには、次のように入力します。

\$ lpq

すべての印刷がすでに完了している場合、あるいはプリント・キューにプリント要求がない場合、システムは次のメッセージを表示します。

no entries

プリント・キューにエントリが存在する場合、システムはそれらのエントリをリストして、現在どの要求がプリントされているかを示します。次に示すのは、プリント・キュー・エントリ・リストの例です。

Rank	Owner	Job	Files	Total Size
active	marilyn	489	report	8470 bytes
1st	sue	135	letter	5444 bytes
2nd	juan	360	(standard input)	969 bytes
3rd	larry	490	travel	1492 bytes

lpq コマンドは、各プリント・キュー・エントリについて次の情報を表示します。

- キュー内の優先順位
- 所有者
- ジョブ番号
- ファイル名

- ファイルのバイト数

前のプリント・キュー・エントリ・リストの例では、Marilyn のレポート (ジョブ番号 489) は現在印刷中であり、Sue、Juan、および Larry の要求は待機中です。

ファイルを印刷する際、プリント・キュー内のジョブの位置およびファイル・サイズによって、ユーザが要求したジョブがいつ完了するかを予想することができます。一般的には、キュー内の優先番号が低くファイル・サイズが大きければ、それだけ余計に時間がかかります。

システムに複数のプリンタが接続されている場合は、次のコマンド構文を使用して、どのプリント・キューを参照するかを指定します。

lpq -P *printername filename*

プリント・キューを指定する場合は -P フラグを使用します。 *printername* 変数にはプリンタ名を指定します。 *printername* 変数は、プリント要求の開始時に使用したものと同じでなければなりません。すべてのプリンタの名前を参照する場合は、lpstat -s コマンドを使用します。lpq コマンドの完全な説明については、lpq(1) リファレンス・ページを参照してください。

キューに登録したプリント要求を取り消す場合は、lprm コマンドを使用します。lprm コマンドの構文は次のとおりです。

lprm -P *printername jobnumber*

jobnumber 変数には、システムがプリント要求時に割り当てたジョブ番号を指定します。 *printername* 変数は、プリント要求を開始したときに使用した名前と同じでなければなりません。ジョブ番号はlpq コマンドで参照できます。

たとえば、ユーザ Larry が自分のプリント要求を取り消したい場合は、次のように入力します。

```
$ lprm 490
$
```

これで travel ファイルのプリント・ジョブはプリント・キューから削除されます。

lprm コマンドについての詳しい説明は、lprm(1) リファレンス・ページを参照してください。

ここでは、ファイルを印刷するコマンドについて基本的な説明をしています。システムの印刷機能および利用可能なコマンドについての詳細な説明は、`lp(1)`、`cancel(1)`、および `lpstat(1)` リファレンス・ページを参照してください。

3.4 ファイルのリンク (ln コマンド)

リンクとは、ファイル名とファイルそのものとの結びつきのことです。通常、各ファイルには 1 つのリンクが存在します。つまり、各ファイルはそれぞれ 1 つのファイル名と結びつけられています。しかし、`ln (link)` コマンドを使用すると、1 つのファイルに対して同時に複数のファイル名を結びつけることができます。

リンクは、複数のディレクトリで同じデータを使用する場合に便利です。たとえば、組立ラインの生産統計を格納するファイルがあるとします。このファイルのデータを 2 つの異なる文書 (たとえば、マネージメントのために作成する月報およびライン作業員のために作成する月別概要) で使用する場合を考えてみます。

たとえば、この統計ファイルを 2 つの異なるファイル名 `mgmt.stat` および `line.stat` とリンクして、それぞれのファイル名を 2 つの異なるディレクトリに置くことができます。こうすると、ファイルのコピーは 1 つだけで済み、記憶領域を節約することができます。また、ファイルをリンクしておけば、統計ファイルが更新された際に複数のファイルを更新する必要がありません。`mgmt.stat` ファイルと `line.stat` ファイルは互いにリンクしているので、一方を編集すれば他方も自動的に更新されることになり、どちらのファイル名も常に同じデータを参照することになります。

3.4.1 ハード・リンクとソフト・リンク

リンクにはハード・リンクとソフト・リンク (シンボリック・リンク) の 2 種類があります。

- ハード・リンク

同じファイル・システム内のファイルのリンクのみ可能です。ハード・リンクを作成する場合は、1 つのファイルに対して別のファイル名を付与します。オリジナルのファイル名を含めて、ファイルのハード・リンク名はすべて対等です。一方のファイル名を「本当の名前」、他方を「単なるリンク」と考えるのは正しくありません。

- ソフト・リンク (シンボリック・リンク)

ファイルとディレクトリのどちらでもリンクすることができます。さらに、異なるファイル・システム間でもファイルあるいはディレクトリのどちらでもリンクすることができます。シンボリック・リンクは、別のファイルまたは別のディレクトリへのポインタを含む別個のファイルです。このポインタはデスティネーション・ファイルまたはディレクトリへのパス名です。オリジナルのファイル名のみがそのファイルまたはディレクトリの本当の名前です。ハード・リンクとは異なり、ソフト・リンクは「単なるリンク」です。

ハード・リンクでもソフト・リンクでも、ある名前呼び出したファイルに対して変更を加えた後、別の名前と同じファイルを呼び出すと、変更した内容が反映されています。

ハード・リンクとソフト・リンクとの大きな違いは、ファイルを削除する際に明らかになります。複数の名前がハード・リンクされているファイルは、それらの名前がすべて削除されるまで残ります。複数の名前がソフト・リンクされているファイルは、オリジナルの名前が削除されると消滅します。その場合、残っているソフト・リンクは実在しないファイルを指すこととなります。リンクの削除については 3.4.5 項を参照してください。

3.4.2 リンクとファイル・システム

ここでリンクについて説明するときに使用するファイル・システムという用語は、前章での説明とは意味が異なります。一般にファイル・システムとは、ファイルを使用しやすい状態に配置したディレクトリ構造と定義されます。ただし、ここで使用するファイル・システムという用語は、より厳密な意味をもっています。つまり、単一のディスク・パーティション内に格納されているファイルおよびディレクトリを意味します。ディスク・パーティションとは、ファイル・ディレクトリを格納するために作成された物理ディスク、またはその一部のことです。

Tru64 UNIX システムで、あるディレクトリがどのディスク・パーティションに含まれているかを調べる場合は、`df` コマンドを使用します。次に示すのは、`df` コマンドを使用して、ディレクトリ `/u1/info`、`/etc`、`/tmp` がどのファイル・システムに含まれているかを調べる例です。この例では、ディレクトリ `/u1/info` と `/etc` は異なるファイル・システムに存在し、`/etc` と `/tmp` は同じファイル・システムに存在しています。

```

$ df /u1/info
Filesystem 512-blks      used avail capacity Mounted on
/dev/disk/dsk2c    196990  163124  14166    92%    /u1
$ df /etc
Filesystem 512-blks      used avail capacity Mounted on
/dev/rz3a      30686   19252   8364    70%    /
$ df /tmp
Filesystem 512-blks      used avail capacity Mounted on
/dev/rz3a      30686   19252   8364    70%    /
$

```

df コマンドについての詳細は、df(1) リファレンス・ページを参照してください。

3.4.3 リンクの使用

同じファイル・システム内のファイルをリンクする場合は、次のコマンド構文を使用します。

```
ln /dirname1/filename1 /dirname2/filename2
```

/dirname1/filename1 変数は既存ファイルのパス名です。
 /dirname2/filename2 変数は、既存のファイル /dirname1/filename1 にリンクする同じファイル・システム内の新しいファイルのパス名です。
 同じディレクトリにあるファイルをリンクする場合は、dirname1 および dirname2 は省略できます。

異なるファイル・システム間でファイルあるいはディレクトリをリンクする場合は、シンボリック・リンクを作成します。シンボリック・リンクを作成する場合は、ln コマンドに -s フラグを指定し、両方のファイルの完全パス名を指定します。シンボリック・リンクを作成する場合の ln コマンドの構文は次のとおりです。

```
ln -s /dirname1/filename1 /dirname2/filename2
```

/dirname1/filename1 変数は既存ファイルのパス名です。
 /dirname2/filename2 変数は別のファイル・システムまたは同じファイル・システム内の新しいファイルのパス名です。

例 3-5 では、新しいファイル名 checkfile を既存ファイル file3 にリンクしています。その後、more コマンドを使用して file3 ファイルと checkfile ファイルの内容が同じであることを確認しています。

例 3-5: ファイルのリンク

```
$ ln file3 checkfile [1]
$ more file3 [2]
You will find that vi is a useful [3]
editor that has many features. [3]
$ more checkfile [4]
You will find that vi is a useful [3]
editor that has many features. [3]
$
```

[1] 2つのファイルの間にハード・リンクを作成します。

[2] file3 のテキストを表示します。

[3] 次に checkfile のテキストを表示します。

[4] file3 ファイルの内容と checkfile ファイルの内容がともに同じであることに注意してください。一方のファイル名でアクセスしてファイルの内容を変更した後、もう一方のファイルにアクセスすると変更した内容が表示されます。たとえば、file3 ファイルの内容を更新すれば、checkfile ファイルの内容も更新されます。

2つのファイルがそれぞれ2つの異なるファイル・システムのディレクトリにある場合は、それらをリンクするためにはシンボリック・リンクを使用する必要があります。たとえば、/reports ディレクトリにある newfile ファイルを、/summary ディレクトリにある mtgfile ファイルにリンクする場合は、次のコマンドを使用して、シンボリック・リンクを作成します。

```
$ ln -s /reports/newfile /summary/mtgfile
$
```

これらの各ファイルの内容は、ハード・リンクの場合と同じように更新されます。

ln コマンドおよびファイルのリンクについての詳細は、ln(1) リファレンス・ページを参照してください。

3.4.4 リンクの仕組み (ファイル名とファイル通し番号)

各ファイルには、ファイル通し番号と呼ばれるファイル固有の識別番号が付与されます。ファイル通し番号はファイル名ではなく、ファイルそのもの、すなわち特定の場所に格納されているデータを参照します。システムは、ファイル通し番号を使用して、各ファイルを同じファイル・システム内の他のファイルと区別します。

各ディレクトリが保持している情報は、物理ファイルを表すファイル通し番号とファイル名との間のリンク情報です。複数のファイル名を同一の物理ファイルすなわち同一のファイル通し番号にリンクすることにより、ファイルを複数のファイル名とリンクすることができます。

現在のディレクトリにあるファイルのファイル通し番号を表示するためには、`-i` フラグを指定して `ls` コマンドを実行します。

ls -i

ログイン・ディレクトリにあるファイルのファイル通し番号を調べてみましょう。各ファイル名の前に表示される番号が、そのファイルのファイル通し番号です。

```
$ ls -i
1079 checkfile  1077 file1   1078 file2   1079 file3
$
```

この例では、`file3` ファイルと `checkfile` ファイルのファイル通し番号が同じ 1079 です。これは、この 2 つのファイルがリンクしていることを示します。

ファイル通し番号は特定のファイル・システム内のファイルを表すため、異なるファイル・システム間ではハード・リンクは存在しません。

シンボリック・リンクの場合、リンクしたファイルにはオリジナルのファイルとは異なる新しいファイル通し番号が付与されます。つまりシンボリック・リンクでは、オリジナル・ファイルのファイル通し番号に対して別のファイル名を結び付けるのではなく、独自のファイル通し番号を持つ独立したファイルを作成します。シンボリック・リンクはファイル通し番号によってではなく、ファイル名によってオリジナルのファイルを参照するため、シンボリック・リンクは異なるファイル・システム間でも正しく機能します。

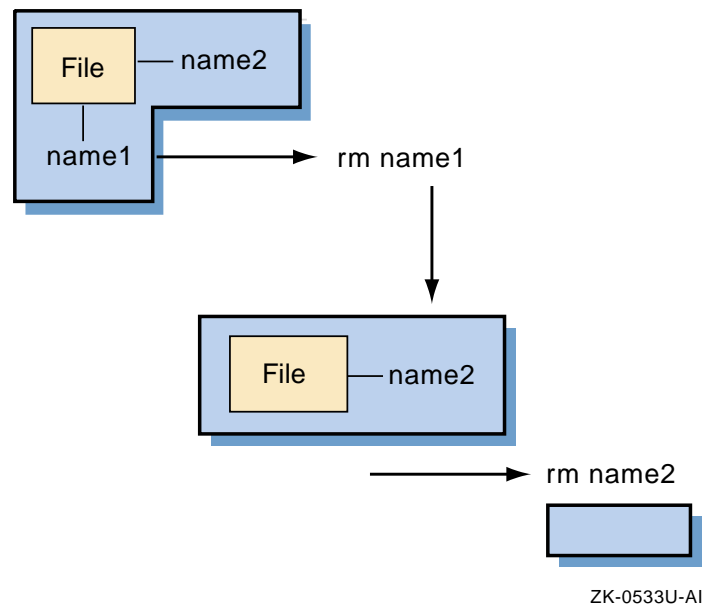
3.4.5 リンクの削除

`rm` (remove file) コマンドはいつもファイルを削除するとは限りません。たとえば、あるファイルが複数のファイル名にリンクされている場合、すなわち複数のファイル名が同じファイル通し番号を参照している場合、`rm` コマンドはファイル通し番号とそのファイル名とのリンクは削除しますが、物理ファイルはそのまま残します。図 3-1 に示すように、`rm` コマンドは、ファイルとファイル名との最後のリンクを削除した場合に初めて物理ファイル

を削除します。シンボリック・リンクを削除すると、リンク・ファイルまたはディレクトリへのポインタを指定するファイルが削除されます。

`rm` コマンドについての詳細は、3.9 節または `rm(1)` リファレンス・ページを参照してください。

図 3-1: リンクとファイルの削除



各ファイルのファイル通し番号および特定のファイル通し番号にリンクされているファイル名の数を表示する場合は、`-i` (print i-number) および `-l` (long listing) フラグを指定して、`ls` コマンドを使用します。

ls -il

ログイン・ディレクトリのリンクを調べてみましょう。実際に表示される画面は、ここで示す例とは異なります。

```
$ ls -il
total 3
1079 -rw-r--r-- 2 larry system 65 Jun 5 10:06 checkfile
1077 -rw-r--r-- 1 larry system 101 Jun 5 10:03 file1
1078 -rw-r--r-- 1 larry system 75 Jun 5 10:03 file2
1079 -rw-r--r-- 2 larry system 65 Jun 5 10:06 file3
1080 drwxr-xr-x 2 larry system 32 Jun 5 10:07 project
$
```

各エントリの最初の数字はそのファイル名のファイル通し番号を示しています。各行の 2 番目の要素はファイル許可を示しています。ファイル許可については、第 5 章を参照してください。

各エントリの 3 番目のフィールド、つまりユーザ名の左に表示される数字は、そのファイル通し番号へのリンク数を表します。file3 と checkfile が 1079 という同じファイル通し番号を持ち、これらのファイルのリンク数とともに 2 であることに注意してください。rm コマンドを使用してファイル名を削除するたびに、そのファイル通し番号へのリンク数が 1 ずつ減少します。

次に示すのは、rm コマンドを使用して checkfile ファイルを削除する例です。

```
$ rm checkfile
$
```

このコマンドを実行した後、ls -il コマンドでディレクトリの内容をリストすると、ファイル通し番号 1079 (file3) のリンク数が 1 つだけ減少しているのがわかります。

```
$ ls -il
total 3
1077 -rw-r--r-- 1 larry system 101 Jun 5 10:03 file1
1078 -rw-r--r-- 1 larry system 75 Jun 5 10:03 file2
1079 -rw-r--r-- 1 larry system 65 Jun 5 10:06 file3
1080 drwxr-xr-x 2 larry system 32 Jun 5 10:07 project
$
```

3.5 ファイルのコピー (cp コマンド)

この節では、ローカル・システムでファイルをコピーする方法について説明します。リモート・システムとの間でファイルをコピーする方法については、第 12 章および第 14 章を参照してください。

cp (copy) コマンドは、現在のディレクトリ内で、あるいは 1 つのディレクトリから別のディレクトリへファイルをコピーします。

cp コマンドは、重要なファイルのバックアップ・コピーを作成する際に特に有用です。バックアップ・ファイルとオリジナル・ファイルは 2 つの別のファイルなので、バックアップ・ファイルを作成しておけば、オリジナル・ファイルに問題が発生した場合に回復することができます。また、編集中のオリジナル・ファイルに対する変更を取り消したい場合は、バックアップ・ファイルを使用して、改めて作業を始めることができます。

実際にファイルのコピーを作成する `cp` コマンドと、1つのファイルに対して複数のファイル名を作成する `ln` コマンドとを比較してみてください。 `ln` コマンドについては 3.4 節で詳しく説明しています。 `cp(1)` および `ln(1)` リファレンス・ページも参照してください。

`cp` コマンドの構文は次のとおりです。

cp *source destination*

source 変数にはコピー元のファイル名を指定します。 *destination* 変数には、*source* ファイルをコピーする先のファイル名を指定します。 *source* および *destination* 変数には、現在のディレクトリ内のファイル名、あるいは異なるディレクトリのパス名のいずれも指定することができます。ただし、この節では、異なるディレクトリ間でファイルをコピーする場合についてのみ説明します。ディレクトリ全体の内容を別のディレクトリにコピーする (`-r` オプションの使用) 方法については、4.4 節を参照してください。

あるファイルを別のディレクトリに同じファイル名でコピーする場合、*source* にはファイル名を指定し、*destination* にはコピー先のディレクトリ・パス名を指定します。 *source* 変数には、パターン照合文字を使用することもできます。

3.5.1 現在のディレクトリ内でのファイルのコピー

`cp` コマンドはファイルをコピーします。コピー先ファイル名が存在していなければ、新規にそのファイルを作成します。しかし、コピー先ファイル名と同じ名前のファイルがすでに存在している場合、`cp` コマンドはコピー元ファイルを既存のコピー先ファイルの上にコピーします。

注意

コピー先ファイル名が現在のディレクトリ内に存在する場合、シェルは `cp` コマンドでコピー元ファイルをコピーする前に、コピー先ファイルの内容を消去します。したがって `cp` コマンドを実行する前に、コピー先ファイルの内容が必要でないか、あるいはそのファイルのバックアップ・コピーが存在するかを確認してください。

C シェルを使用している場合は、コピー先ファイルの消去を防ぐための `noclobber` 変数を使用できます。 `noclobber` 変数については表 8-6 を参照してください。

次の例では、現在のディレクトリにコピー先ファイル名が存在しないので、`cp` コマンドは新規にファイルを作成します。

まず、ログイン・ディレクトリの内容をリストします。

```
$ ls
file1    file2    file3    project
$
```

次に `file2` ファイルをコピーして `file2x` ファイルを作成します。

```
$ cp file2 file2x
$
```

ディレクトリの内容をリストして、ファイルのコピーが成功したことを確認します。

```
$ ls
file1    file2    file2x   file3    project
$
```

3.5.2 別のディレクトリへのファイルのコピー

別のディレクトリへファイルをコピーする場合は、`cp` コマンドを実行する前にコピー先のディレクトリを作成しておきます。すでにコピー先ディレクトリが存在する場合、この操作は必要ありません。ディレクトリの作成には `mkdir` コマンドを使用します。

次のように、`reports` というサブディレクトリを作成します。

```
$ mkdir reports
$
```

`file2` ファイルを `reports` ディレクトリにコピーするには、次のように入力します。

```
$ cp file2 reports
$
```

ここで `reports` ディレクトリの内容をリストして、`file2` ファイルがコピーされていることを確認します。

```
$ ls reports
file2
$
```

`cp` コマンドを使用して、複数のファイルを別のディレクトリにコピーすることもできます。この場合は次の構文を使用します。

```
cp filename1 filename2 dirname
```

次の例では、`cp` コマンドを使用して `file2` および `file3` ファイルを `reports` ディレクトリにコピーした後、コピーが成功していることを確認するために、`ls` コマンドを使用して `reports` ディレクトリの内容をリストしています。

```
$ cp file2 file3 reports
$ ls reports
file2  file3
$
```

この例では、`dirname` ディレクトリのコピー先ファイル名 `file2` および `file3` を指定していません。コピー先ファイル名を省略すると、元のファイル名でファイルがコピーされます。

パターン照合文字を使用してファイルをコピーすることもできます。たとえば、`file1`、`file2` および `file3` ファイルを `reports` ディレクトリへコピーする場合は、次のように入力します。

```
$ cp file* reports
$
```

ファイルを別のディレクトリにコピーする際に、コピー先ファイル名を変更したい場合は、次の構文を使用します。

`cp source destination_directory/ filename`

`source` 変数にはコピー元ファイル名を、`destination_directory` 変数にはコピー先ディレクトリ名を、`filename` には新ファイル名を指定します。

次の例では、`file3` ファイルを新しいファイル名 `notes` で `reports` ディレクトリにコピーし、`ls` コマンドを使用して `reports` ディレクトリの内容をリストしています。

```
$ cp file3 reports/notes
$ ls reports
file1  file2  file3  notes
$
```

3.6 ファイル名の変更およびファイルの移動 (`mv` コマンド)

`mv` (move) コマンドを使用して、次の操作を行うことができます。

- 別のディレクトリへのファイルの移動
- ファイル名の変更
- ディレクトリ名の変更

mv コマンドの構文は次のとおりです。

mv *oldfilename newfilename*

oldfilename 変数には、移動または名前を変更したいファイルの名前を指定します。*newfilename* 変数には新しいファイル名を指定します。これらの変数には、現在のディレクトリにあるファイル名または別のディレクトリにあるファイルのパス名のどちらでも指定できます。また、パターン照合文字を使用することもできます。

mv コマンドは新しいファイル名を既存のファイル通し番号にリンクし、旧ファイル名とそのファイル通し番号とのリンクを切断します。ln コマンドと cp コマンドを使用する (3.4 節および 3.5 節を参照) 代わりに、mv コマンドを使用すると便利です。mv(1)、ln(1)、および cp(1) リファレンス・ページも参照してください。

3.6.1 ファイル名の変更

次の例では、まず `ls -i` コマンドを使用して、現在のディレクトリ内の各ファイルのファイル通し番号をリストします。次に、mv コマンドを実行して、file2x ファイルの名前を newfile に変更します。実際に表示されるファイル通し番号は異なります。

```
$ ls -i
1077 file1      1088 file2x    1080 project
1078 file2      1079 file3     1085 reports
$ mv file2x newfile
$
```

再びディレクトリの内容をリストします。

```
$ ls -i
1077 file1      1079 file3     1080 project
1078 file2      1088 newfile  1085 reports
$
```

ここでは次のことに注意してください。

- mv コマンドにより file2x のファイル名が newfile に変更されている。
- 元のファイル (file2x) のファイル通し番号と newfile ファイルのファイル通し番号は同じ 1088 である。

mv コマンドはファイル通し番号 1088 とファイル名 file2x との関係を切り離し、代わりにファイル通し番号 1088 とファイル名 newfile とを結び

つけています。ただし、このコマンドはファイル自体を変更することはありません。

3.6.2 別のディレクトリへのファイルの移動

`mv` コマンドを使用して、ファイルを現在のディレクトリから別のディレクトリに移動することもできます。

注意

移動先のディレクトリ名は注意して入力してください。`mv` コマンドはファイル名とディレクトリ名とを区別しないため、存在しないディレクトリ名を入力すると、`mv` コマンドはそのディレクトリ名を新しいファイル名と解釈します。この結果、別のディレクトリへの移動ではなく、ファイル名の変更が行われることになります。

次の例では、`ls` コマンドでログイン・ディレクトリの内容をリストし、次に `mv` コマンドで、`file2` ファイルを現在のディレクトリから `reports` ディレクトリに移動しています。その後、再度 `ls` コマンドを使用して、ログイン・ディレクトリの `file2` ファイルが削除されていることを確認しています。

```
$ ls
file1  file2  file3  newfile  project  reports
$ mv file2 reports
$ ls
file1  file3  newfile  project  reports
$
```

`reports` ディレクトリの内容をリストすると、`file2` ファイルが移動していることがわかります。

```
$ ls reports
file2  file3  notes
$
```

パターン照合文字を使用してファイルを移動することもできます。たとえば、`file1` および `file3` ファイルを `reports` ディレクトリに移動する場合は、次のコマンドを入力します。

```
$ mv file* reports
$
```

ログイン・ディレクトリの内容をリストすると、`file1` および `file3` ファイルが移動していることがわかります。

```
$ ls
newfile project reports
$
```

file1, file2, file3 の各ファイルをログイン・ディレクトリにコピーする場合は、次のように入力します。次の例で使用しているドット (.) は現在のディレクトリを指定します。この例では、現在のディレクトリはログイン・ディレクトリです。

```
$ cp reports/file* .
$
```

ls コマンドを実行すると、ファイルがログイン・ディレクトリに戻っていることが確認できます。

```
$ ls
file1    file2    file3    newfile project reports
$
```

ここで reports ディレクトリの内容をリストすると、file1, file2, および file3 の各ファイルが reports ディレクトリにあることが確認できます。

```
$ ls reports
file1    file2    file3    newfile project reports
$
```

3.7 ファイルの比較 (diff コマンド)

diff コマンドを使用すると、2 つのテキスト・ファイルの内容を比較することができます。diff コマンドは、多少違いがあると予想される 2 つのファイルの内容の違いを正確に示したい場合に使用します。

diff コマンドの構文は次のとおりです。

diff *file1 file2*

diff コマンドは、2 つのファイルを 1 行ずつ走査します。異なる行が見つかると、次の情報を出力します。

- 違いのある行番号
- 違いの種類

記述の追加であるか、削除であるか、それとも行の変更であることを明示します。

違いが追加である場合は、次の形式で情報を表示をします。

```
l[,l] a r[,r]
```

`l` は *file1* の行番号であり、`r` は *file2* の行番号です。`a` は追加を示しています。違いが削除によるものである場合、`diff` コマンドの出力では `d` で示され、行の変更である場合は `c` で示されます。

これらの情報の次に、実際に異なる行が表示されます。左端のカラムの左山カッコ (<) は *file1* 中の実際の行を示し、右山カッコ (>) は *file2* 中の実際の行を示します。

たとえば、ファイル *jan15mtg* および *jan22mtg* の内容が次のような出席者名簿であるとしします。

jan15mtg	jan22mtg
alice	alice
colleen	brent
daniel	carol
david	colleen
emily	daniel
frank	david
grace	emily
helmut	frank
howard	grace
jack	helmut
jane	jack
juan	jane
lawrence	juan
rusty	lawrence
soshanna	rusty
sue	soshanna
tom	sue
	tom

これらのファイルの内容を表示して 1 行ずつ比較する代わりに、`diff` コマンドを使用して、*jan15mtg* と *jan22mtg* の内容を比較することができます。この場合、`diff` コマンドの出力は次のようになります。

```
$ diff jan15mtg jan22mtg
2a3,4
> brent
```

```
> carol
10d11
< howard
$
```

この出力から、Brent と Carol が 1 月 15 日の会議に欠席し、Howard が 1 月 22 日の会議に欠席していたことがわかります。

2 つのファイルに違いがない場合は、システムは単にプロンプトを返します。詳細は、`diff(1)` リファレンス・ページを参照してください。

3.8 ファイル内容のソート (sort コマンド)

`sort` コマンドを使用すると、テキスト・ファイルの内容をソートすることができます。このコマンドは、1 つのファイルに対しても、複数のファイルに対しても使用できます。

`sort` コマンドの構文は次のとおりです。

sort *filename*

filename 変数には、ファイル名、ファイルの相対パス名、ファイルの完全パス名、あるいはスペースで区切ったファイル名リストを指定します。また、パターン照合文字を使用してファイルを指定することもできます。パターン照合文字については第 2 章を参照してください。

`sort` コマンドの使用例としては、名前のリストをソートして、現在のロケールで定義されている照合順に並べることなどがあります。たとえば、`list1`、`list2`、および `list3` の 3 つのファイルに、それぞれ名前のリストが含まれているとします。

list1	list2	list3
Zenith, andre	Rocca, Carol	Hamilton, Abe
Dikson, Barry	Shepard, Louis	Anastio, William
D'Ambrose, Jeanette	Hillary, Mimi	Saluccio, William
Julio, Annette	Chung, Jean	Hsaio, Peter

3 つのファイルに含まれているすべての名前をソートするには、次のように入力します。

```
$ sort list*
Anastio, William
Chung, Jean
D'Ambrose, Jeanette
```

```
Dickson, Barry  
Hamilton, Abe  
Hillary, Mimi  
Hsaio, Peter  
Julio, Annette  
Rocca, Carol  
Saluccio, Julius  
Shepard, Louis  
Zenith, andrew  
$
```

次のように入力して、指定するファイルに画面出力をリダイレクトすると、新しいソート済みのリストを含むファイルを作成することができます。

```
$ sort list* > newlist  
$
```

出力のリダイレクトについての詳細は、第 6 章を参照してください。sort コマンドとそのオプションについての詳細は、sort(1) リファレンス・ページを参照してください。

3.9 ファイルの削除 (rm コマンド)

ファイルが必要ではなくなった場合には、rm (remove file) コマンドを使用して、ファイルを削除することができます。1 つのファイルを削除することも、複数のファイルを同時に削除することもできます。

rm コマンドの構文は次のとおりです。

rm *filename*

filename 変数には、ファイル名、ファイルの相対パス名、ファイルの完全パス名、あるいはファイル名リストを指定します。使用するコマンド形式は、ファイルが現在のディレクトリとの関係でどこに位置しているかによって変わります。このコマンドについての詳細は、rm(1) リファレンス・ページを参照してください。

3.9.1 単一のファイルの削除

次の例では、file1 ファイルをログイン・ディレクトリから削除しています。

まず、cd (change directory) コマンドでログイン・ディレクトリに移ります。次に、pwd (print working directory) コマンドを入力して、現在のディレクトリがログイン・ディレクトリであることを確認し、ディレクトリの

内容をリストします。実際には、`/u/uname` にはログイン・ディレクトリの名前が表示されます。

```
$ cd
$ pwd
/u/uname
$ ls
file1    file2    file3    newfile  project  reports
$
```

`rm` コマンドを使用して `newfile` を削除した後、ディレクトリの内容をリストして、システムがそのファイルが削除されたことを確認します。

```
$ rm newfile
$ ls
file1    file2    file3    project  reports
$
```

ディレクトリからファイルを削除するには、そのディレクトリにアクセスする許可が必要です。ディレクトリ許可については、第 5 章を参照してください。

注意

`rm` コマンドは、1 つまたは複数のファイルを削除するだけでなく、ファイルとファイル名とのリンクも削除します。`rm` コマンドは、そのファイルへの最後のリンクを削除する際に、ファイルそのものも削除します。`rm` コマンドによるリンクの削除については、3.4.5 項を参照してください。

3.9.2 複数のファイルの削除 (パターン照合)

`rm` コマンドとともにパターン照合文字を使用して、複数のファイルを同時に削除することができます。パターン照合文字については第 2 章を参照してください。

たとえば、現在のディレクトリに次のファイルが含まれているとします。

- `receivable.jun`
- `payable.jun`
- `payroll.jun`
- `expenses.jun`

`rm *.jun` コマンドを使用すると、これらの 4 つのファイルをすべて削除することができます。

注意

パターン照合文字 `*` を使用する場合は十分に注意してください。たとえば、通常のユーザの場合、`rm *` コマンドを入力すると、ドット (`.`) で始まるファイル名を持つファイルを除いて、現在のディレクトリにあるすべてのファイルが削除されます。

ファイル名の始めまたは終わりに使用するパターン照合文字 `*` には、特に注意してください。 `rm *name` の代わりに誤って `rm *name` と入力すると、`name` で終わるファイルだけでなく、現在のディレクトリにあるすべてのファイルを削除します。

`rm` コマンドに `-i` フラグを指定すると、ファイルを削除する前に、確認のためのプロンプトを表示します。

パターン照合の例を実行するには、ディレクトリにファイル `record1`、`record2`、`record3`、`record4`、`record5`、および `record6` が入っていない必要があります。次のように `touch` コマンドを使用して、ログイン・ディレクトリにこれらのファイルを作成します。

```
$ touch record1 record2 record3 record4 record5 record6
$
```

`touch` コマンドは、上の例のように空のファイルを作成したい場合に便利です。 `touch` コマンドについての詳細は、`touch(1)` リファレンス・ページを参照してください。

ファイル名が 1 文字だけ異なるファイルを削除する場合は、`rm` コマンドでパターン照合文字 `?` を使用することもできます。たとえば、現在のディレクトリに `record1`、`record2`、`record3`、および `record4` の各ファイルが含まれている場合、`rm record?` コマンドを使用すると、4 つのファイルすべてを削除することができます。

パターン照合文字についての詳細は、第 2 章を参照してください。

パターン照合文字を使用する場合は、`rm` コマンドの `-i` (interactive) フラグが便利です。 `rm -i` コマンドを使用すると、ファイルを選択的に削除することができます。 コマンドによって選択された各ファイルについてプロ

ンプトが表示されるので、ファイルを削除するか、あるいはそのまま残すかを選択することができます。

たとえば、文字列 `record` で始まる 6 つのファイルのうち 4 つを削除したい場合は、次のように入力します。

```
$ rm -i record?
rm: remove record1?n
rm: remove record2?y
rm: remove record3?y
rm: remove record4?y
rm: remove record5?y
rm: remove record6?n
$
```

注意

`rm` コマンドは、ファイルを削除するだけでなく、`-r` フラグを指定すると、ファイルとディレクトリを同時に削除します。詳細は、第 4 章を参照してください。

3.10 ファイル・タイプの参照 (file コマンド)

ファイルに含まれている情報がどのような種類のデータであるかをファイルの内容を表示しないで調べる場合は、`file` コマンドを使用します。`file` コマンドは、ファイルが次のどのタイプであるかを表示します。

- テキスト・ファイル
- ディレクトリ
- FIFO (パイプ) 型特殊ファイル
- ブロック型特殊ファイル
- 文字型特殊ファイル
- C または FORTRAN プログラミング言語のソース・コード
- 実行可能 (バイナリ) ファイル
- `ar` フォーマットのアーカイブ・ファイル
- `cpio` または拡張 `tar` フォーマットのアーカイブ・ファイル
- `zip` フォーマットのアーカイブ・ファイル

- gzip フォーマットの圧縮データ・ファイル
- コマンド・テキストのファイル (シェル・スクリプト)
- .voc , .iff , または .wav フォーマットのオーディオ・ファイル
- TIFF , GIF , MPEG , または JPEG フォーマットのイメージ・ファイル

`file` コマンドは、コンパイル済みのプログラムや、オーディオ・データ、またはイメージ・データがファイルに含まれているかどうかを確認する際に有用です。これらのタイプのファイルの内容を表示すると画面が乱れます。これらのタイプのファイルの目的はまだわからないかも知れませんが、UNIX コマンド使用の経験が豊富になるにつれ、目的がよく理解できるようになります。

`file` コマンドの構文は次のとおりです。

file *filename*

filename 変数には、ファイル名、ファイルの相対パス名、ファイルの完全パス名、あるいはスペースで区切ったファイル名リストを指定します。使用するコマンド形式は、ファイルが現在のディレクトリとの関係でどこに位置しているかによって変わります。また、パターン照合文字を使用してファイルを指定することもできます。パターン照合文字については第 2 章を参照してください。

たとえば、ログイン・ディレクトリにあるファイルのファイル・タイプを参照する場合は、次のように入力します。

```
$ cd
$ pwd
/u/uname
$ file *
file1:  ascii text
file2:  English text
file3:  English text
project: directory
record1: empty
record6: empty
reports: directory
$
```

この例では、`file1`、`file2`、および `file3` を英語のテキスト・ファイル、`project` と `reports` をディレクトリ、`record1` と `record6` を空ファイルと識別しています。

`file` コマンドについての詳細は、`file(1)` リファレンス・ページを参照してください。

ディレクトリの管理

この章では、システムでディレクトリを管理する方法を説明します。この章を理解すると、次の操作が可能になります。

- ディレクトリの作成 (4.1 節)
- ディレクトリの変更 (4.2 節)
- ディレクトリの表示 (4.3 節)
- ディレクトリのコピー (4.4 節)
- ディレクトリ名の変更 (4.5 節)
- ディレクトリの削除 (4.6 節)

この章の例に従って実際に操作してみることで、ディレクトリの管理について理解することができます。

例を実行するためには、まずログインする必要があります。また、ログイン・ディレクトリに次のファイルおよびディレクトリが存在する必要があります。

- file1, file2, file3, record1, および record6 の各ファイル
- file1, file2, file3, および notes ファイルを含む reports サブディレクトリ
- ファイルを含まない project サブディレクトリ

上記の名前とは異なるファイル名を使用している場合には、この章の例を実行する際に適当に置き換えてください。現在のディレクトリ内にあるファイルのリストを表示する場合は、第 3 章で説明した `ls` コマンドに、表 3-1 にある `-R` および `-F` フラグを指定して使用します。画面は次のようになります。

```
$ ls -RF
file1      file2      file3      project/   record1    record6
reports/

./project:
```

```
./reports:
file1  file2  file3  notes
$
```

4.1 ディレクトリの作成 (mkdir コマンド)

ディレクトリを使用すると、個々のファイルを利用しやすいグループに編成することができます。たとえば、レポートのすべての項目を `reports` ディレクトリに保管したり、費用見積もりに関するデータおよびプログラムを `estimate` ディレクトリに保管することができます。各ディレクトリにはファイルおよびディレクトリを含むことができます。

システム管理者がユーザのアカウントを設定する際に、ログイン・ディレクトリは必ず作成します。しかし、システムで作業するうちに、作成あるいは編集したファイルを編成するための新たなディレクトリが必要になります。新しいディレクトリは `mkdir` (make directory) コマンドを使用して作成します。

`mkdir` コマンドの構文は次のとおりです。

mkdir *dirname*

dirname 変数には新しいディレクトリ名を指定します。

システムはユーザの作業ディレクトリのサブディレクトリとして *dirname* ディレクトリを作成します。つまり、新しいディレクトリを現在のディレクトリの 1 レベル下に作成します。

次の例では、`cd` コマンドを入力してログイン・ディレクトリに戻り、`project2` ディレクトリを作成します。

```
$ cd
$ mkdir project2
$
```

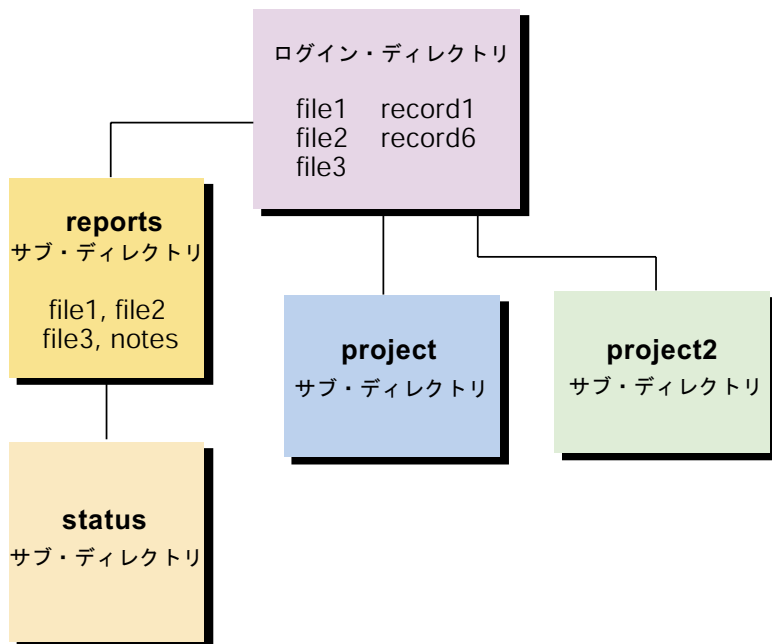
ここで、相対パス名を入力して、`reports` ディレクトリ内にサブディレクトリを作成します。

```
$ mkdir reports/status
$
```

図 4-1 に現在のファイル・システムのツリー構造を示します。 `project`、`project2`、および `reports` ディレクトリは、ログイン・ディレクトリより 1 レベル下位に位置しており、`status` ディレクトリは `reports` ディレクトリより 1 レベル下位に位置しています。

4-2 ディレクトリの管理

図 4-1: ディレクトリとサブディレクトリとの関係



ZK-0534U-AIJ

ファイル名と同じように、ディレクトリ名の最大長はシステムで使用しているファイル・システムによって異なります。たとえば、255 バイト (省略時の値) の最大長が許されるファイル・システムもあれば、14 バイトの最大長しか許されないファイル・システムもあります。

ディレクトリに意味のある名前を付けるためには、ディレクトリ名の最大長を知っておく必要があります。詳細はシステム管理者に問い合わせてください。

オペレーティング・システムにはファイル名とディレクトリ名とを区別するための記号や表記はありませんので、ファイルとディレクトリを区別するためにユーザ独自の命名規則を定めるのも 1 つの方法です。

ただし、`ls -F` コマンドを使用すれば、ディレクトリの内容を表示するときに、ファイルとディレクトリとを見分けることができます。このコマンドについての詳細は、4.3 節を参照してください。

4.2 ディレクトリの変更 (cd コマンド)

cd (change directory) コマンドは、現在のディレクトリ (作業ディレクトリ) を変更するためのコマンドです。適切なパス名を指定して cd を実行することにより、同一ファイル・システム内であれば、自由にディレクトリを移動することができます。

注意

cd コマンドで現在のディレクトリを変更するためには、そのディレクトリにアクセスする実行許可が必要です。ディレクトリ許可については、第 5 章を参照してください。

cd コマンドの構文は次のとおりです。

cd *pathname*

pathname 変数には、現在のディレクトリとして設定するディレクトリの完全パス名あるいは相対パス名を指定します。

パス名を省略して cd を入力すると、ログイン・ディレクトリに移ります。

現在のディレクトリ名を確認するためには、pwd (print working directory) コマンドを入力します。pwd コマンドについては第 2 章を参照してください。

4.2.1 現在のディレクトリの変更

次の例では、pwd コマンドを入力して作業ディレクトリの名前を表示した後、cd コマンドを使用して現在のディレクトリを変更します。

まず、パス名を指定しないで cd コマンドを入力し、ログイン・ディレクトリに戻ります。次に、pwd コマンドを入力して、現在のディレクトリがログイン・ディレクトリであることを確認します。/u/username の表示は、使用しているシステムによって異なります。

```
$ cd
$ pwd
/u/username
$
```

ここで、相対パス名 `project2` を指定して cd コマンドを入力し、`project2` ディレクトリに移ります。

```
$ cd project2
$
```

再び `pwd` コマンドを入力して、`project2` が現在のディレクトリであることを確認します。その後、`cd` コマンドを入力してログイン・ディレクトリに戻ります。

```
$ pwd
/u/uname/project2
$ cd
$
```

ファイル・システムのツリー構造において、別の分岐である `status` ディレクトリに現在のディレクトリを変更するためには、完全パス名を指定して `cd` コマンドを入力します。

```
$ cd reports/status
$ pwd
/u/uname/reports/status
$
```

4.2.2 相対パス名の使用

相対パス名を使用すると、少ないキーストロークでディレクトリを変更することができます。相対パス名には次のような記号を使用します。

- ドット (`.` および `..`)
- チルダ (`~`)

すべてのディレクトリには、ドット (`.`) とドット・ドット (`..`) で表現される 2 つエントリが必ず含まれています。これらのエントリは現在のディレクトリに対して相対的なディレクトリを参照します。

ドット (`.`) 現在のディレクトリを参照する。

ドット・ドット (`..`) 現在のディレクトリの親ディレクトリを参照する。
親ディレクトリとは、ファイル・システムのツリー構造において、現在のディレクトリのすぐ上位に位置するディレクトリのことです。

`.` および `..` エントリを含め、ピリオドで始まるファイルをリストする場合は、`-a` フラグを指定して `ls` コマンドを実行します。

次の例では、まずログイン・ディレクトリに移り、その後 reports ディレクトリに移っています。

```
$ cd
$ cd reports
$
```

ここで ls コマンドを実行すると、次のように reports ディレクトリに含まれるファイルおよび先程作成した status サブディレクトリが表示されます。

```
$ ls
file1  file2  file3  notes  status
$
```

ここで ls -a コマンドを実行すると、上記のファイルおよびディレクトリに加えて、ドット(.)で始まる相対ディレクトリ名がリストされます。

```
$ ls -a
./  ../  file1  file2  file3  notes  status
$
```

相対ディレクトリ名 .. を使用すると、ファイル・システムのツリー構造において、現在のディレクトリより上位にあるファイルおよびディレクトリを参照することができます。つまり親ディレクトリに移動したい場合は、親ディレクトリの完全パス名ではなく、相対ディレクトリ名を使用することができます。

次の例では、cd .. コマンドを使用して、現在のディレクトリを reports ディレクトリから、その親ディレクトリであるログイン・ディレクトリに変更しています。

```
$ pwd
/u/username/reports
$ cd ..
$ pwd
/u/username
$
```

ディレクトリ構造を上位に2レベル以上移動する場合は、相対ディレクトリ名を次の例のように使用することができます。次の pwd コマンドに対する応答として表示されるスラッシュ(/)は、ルート・ディレクトリを表しています。

```
$ cd ../../
$ pwd
/
$
```

Korn または POSIX シェルおよび C シェルでは、ユーザのログイン・ディレクトリを指定するためにチルダ(~)を使用することができます。たとえ

ば、ユーザ自身のログイン・ディレクトリを指定する場合は、次のようにチルダを使用します。

```
$ cd ~
```

この例では、チルダ記号を使用することはキーストロークの節約とはなりません。Tru64 UNIX オペレーティング・システムでは、`cd` を入力することにより、ファイル・システム内のどの場所からでもログイン・ディレクトリに移ることができるためです。この機能はすべてのシェルで有効です。

ログイン・ディレクトリより下位のディレクトリにアクセスする場合は、チルダ記号を使用するとキーストロークの節約になります。たとえば、ファイル・システム内の任意のディレクトリから `reports` ディレクトリへアクセスする場合は、次のように入力します。

```
$ cd ~/reports
```

別のユーザのログイン・ディレクトリ、あるいはその下位ディレクトリまたはファイルにアクセスする場合も、チルダ記号は大変便利です。他のユーザのログイン・ディレクトリの正確な位置を知らなくても、適切な許可を得ていれば、チルダ記号を使用することにより、最小限のキーストロークで移動することができます。

たとえば、次のように入力することにより、ファイル・システム内のどこからでも、ユーザ `jones` のログイン・ディレクトリに移動することができます。

```
$ cd ~jones
```

また、ユーザ `jones` のログイン・ディレクトリのすぐ下位にある `status` ディレクトリ内のファイルにアクセスする場合は、次のように入力します。

```
$ cd ~jones/status
```

4.2.3 シンボリック・リンクを介したディレクトリへのアクセス

ディレクトリがシンボリック・リンクで結びつけられているとき、`cd` コマンドで親ディレクトリにアクセスする場合は、`cd` コマンドに完全ディレクトリ名を指定するかあるいは相対ディレクトリ名を指定するかによって結果が異なります。シンボリック・リンクしている親ディレクトリにアクセスする場合は、完全パス名を使用します。

たとえば、ユーザ `user2` が、`/u/user1/project` へのシンボリック・リンクである `/u/user2/project` ディレクトリ上のファイルで作業している場合、実際の親ディレクトリ (`/u/user2`) へ移動するためには、次のように入力します。

```
$ cd /u/user2
$ pwd
/u/user2
$
```

一方、相対ディレクトリ名 (..) を指定した場合は、シンボリック・リンクの親ディレクトリにアクセスすることになります。たとえば、ユーザ user2 が、/u/user1/project へのシンボリック・リンクである /u/user2/project ディレクトリ上のファイルで作業をしている場合、cd .. コマンドを実行すると、次のような結果になります。

```
$ cd ..
$ pwd
/u/user1
$
```

ユーザ user2 は、/u/user2 ディレクトリではなく、/u/user1 ディレクトリに移動します。

シンボリック・リンクについての詳細は、3.4 節を参照してください。

4.3 ディレクトリの表示 (ls -F コマンド)

各ディレクトリにはサブディレクトリおよびファイルを含むことができます。サブディレクトリを表示する場合は、ls -F コマンドを使用します。このコマンドは現在のディレクトリの内容を表示するとき、ディレクトリに対しては後ろにスラッシュ (/) を付けて、ファイルと容易に識別できるようにします。

ls -F コマンドの構文は次のとおりです。

ls -F

次の例では、まずログイン・ディレクトリに戻り、ls -F コマンドを実行して、ディレクトリの内容を表示しています。project、project2、reports の各ディレクトリにはスラッシュが付いています。

```
$ cd
$ ls -F
file1      file3      project2/  record6
file2      project/   record1    reports/
$
```

Korn または POSIX シェルおよび C シェルのユーザのなかには、ls コマンドの別名を定義して、ls を入力すると ls -F コマンドが実行されるように設定する人もいます。別名の定義についての詳細は、第 8 章を参照してください。

4.4 ディレクトリのコピー (cp コマンド)

-r フラグを指定して cp コマンドを使用すると、ディレクトリおよびディレクトリ・ツリーをファイル・システムの別の場所に再帰的にコピーすることができます。

cp -r コマンドの構文は次のとおりです。

cp -r *source destination*

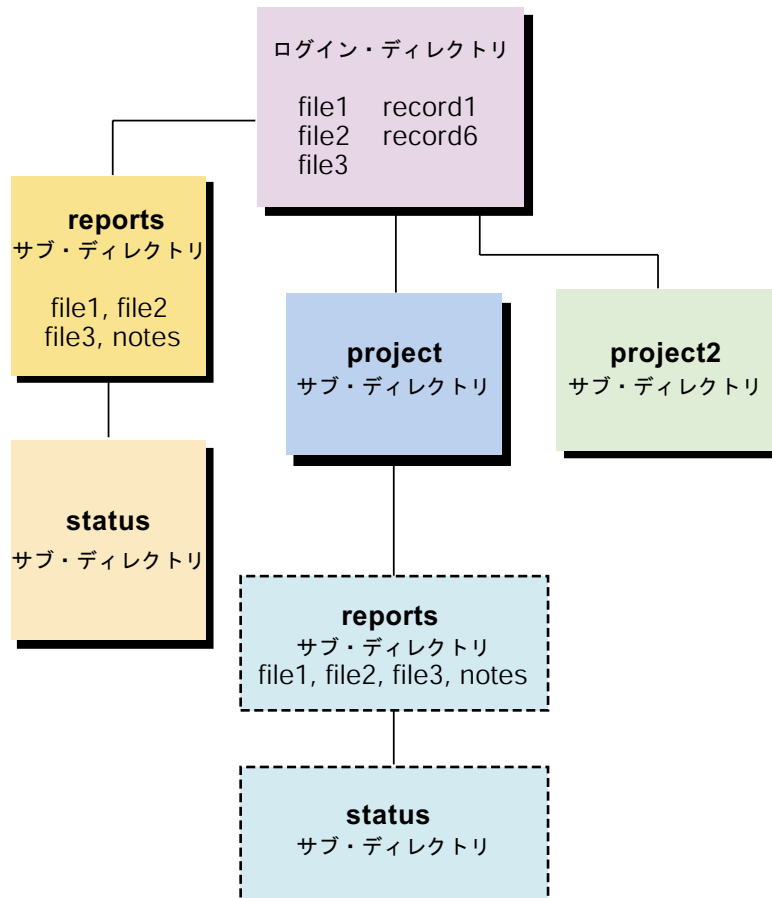
source 変数にはコピーする元のディレクトリ名を指定します。

destination 変数にはコピー先ディレクトリを指定します。

図 4-2 は、次の例で cp -r コマンドがディレクトリ・ツリー reports をディレクトリ project にコピーする手順を示しています。この例では、ログイン・ディレクトリからコマンドを入力すると仮定します。

```
$ cp -r reports project
$
```

図 4-2: ディレクトリ・ツリーのコピー



ZK-0535U-AIJ

reports サブディレクトリ内のファイル, file1, file2, file3, notes
に加えて, サブディレクトリ status が project ディレクトリにコピー
されます。

4.5 ディレクトリ名の変更 (mv コマンド)

ディレクトリが同じディスク・パーティションに含まれている場合に限り,
mv コマンドを使用して, ディレクトリ名を変更することができます。

mv コマンドの構文は次のとおりです。

mv *olddirectoryname newdirectoryname*

`olddirectoryname` 変数には、移動または名前の変更を行いたいディレクトリの名前を指定します。`newdirectoryname` 変数には、新しいディレクトリ名を指定します。

次の例では、まず `reports` ディレクトリに移り、`ls -i -d status` コマンドを入力して、`status` ディレクトリのファイル通し番号をリストします。

```
$ cd reports
$ ls -i -d status
1091 status
$
```

ここで、`mv` コマンドを入力して、ディレクトリ名 `status` を `newstatus` に変更します。次に、`newstatus` ディレクトリのファイル通し番号をリストします。

```
$ mv status newstatus
$ ls -i -d newstatus
1091 newstatus
$
```

2 番目の `ls -i -d` コマンドでは、元のディレクトリ名 `status` がリストされません。新しいディレクトリ `newstatus` がリストされ、`status` ディレクトリと同じファイル通し番号 (1091) が表示されています。

4.6 ディレクトリの削除 (`rmdir` コマンド)

ディレクトリが不要になった場合には、`rmdir` (remove directory) コマンドを使用して、不要なディレクトリをファイル・システムから削除することができます。このコマンドで削除できるのは、ファイルおよびサブディレクトリを含まない空ディレクトリのみです。ディレクトリからのファイルの削除については、4.6.4 項および 3.9 節を参照してください。

`rmdir` コマンドの構文は次のとおりです。

`rmdir` *dirname*

dirname 変数には削除したいディレクトリの名前またはパス名を指定します。

次の各項の例を実行する前に、`project2` ディレクトリに 3 つのサブディレクトリを作成しておきます。

まず、`cd project2` コマンドを使用して、`project2` を現在のディレクトリとして設定します。次に、`mkdir` コマンドを使用して、`schedule`、

tasks および costs の各ディレクトリを作成します。その後、project2 ディレクトリの内容をリストします。

```
$ cd project2
$ mkdir costs schedule tasks
$ ls -F
costs/  schedule/  tasks/
$
```

最後に、cd コマンドを使用してログイン・ディレクトリに戻ります。

```
$ cd
$ pwd
/u/username
$
```

4.6.1 空ディレクトリの削除

rmdir コマンドは空ディレクトリだけを削除します。ファイルやサブディレクトリを含むディレクトリを削除しようとすると、rmdir コマンドは次のようなエラー・メッセージを表示します。

```
$ rmdir project2
rmdir: project2 not empty
$
```

注意

あるディレクトリでユーザが作業しているとき、そのディレクトリを削除することはできません。ディレクトリを削除するには、ディレクトリ・ツリーの他の位置に移動する必要があります。詳細については 4.6 節を参照してください。

ディレクトリ project2 を削除するためには、あらかじめそのディレクトリに含まれるファイルおよびディレクトリを削除しておかなければなりません。次の例では、cd コマンドで project2 を現在のディレクトリに設定した後、ls -F コマンドで project2 の内容をリストします。

```
$ cd project2
$ ls -F
costs/  schedule/  tasks/
```

ここで、現在のディレクトリから schedule サブディレクトリを削除して、project2 ディレクトリの残りの内容をリストします。

```
$ rmdir schedule
$ ls -F
costs/  tasks/
$
```

project2 ディレクトリにはまだ 2 つのサブディレクトリ、costs および tasks が含まれています。これらのディレクトリは、次の項で説明するパターン照合文字を使用して削除することができます。これらのサブディレクトリが削除されれば、4.6 節で説明するように、project2 ディレクトリを削除することができます。

4.6.2 複数のディレクトリの削除

rmdir コマンドでパターン照合文字を使用すると、複数のディレクトリを一度に削除することができます。パターン照合文字についての詳細は、第 2 章を参照してください。

ユーザが project2 ディレクトリで作業している場合、2 つのサブディレクトリ costs および tasks を削除するためには、rmdir *s?s コマンドを入力します。

```
$ rmdir *s?s
$ ls
$
```

ls コマンドを入力すると、project2 ディレクトリが空になったことを確認できます。

注意

rmdir コマンドにアスタリスク (*) だけを指定すると (rmdir *)、現在のディレクトリからすべての空ディレクトリが削除されます。パターン照合文字 * の使用に際しては、十分に注意してください。

4.6.3 現在のディレクトリの削除

現在のディレクトリで作業している間は、そのディレクトリを削除することはできません。他のディレクトリに移ってから削除する必要があります。通常は、cd .. コマンドを入力して、現在のディレクトリの親ディレクトリに移動してから、rmdir に削除したいディレクトリのパス名を指定して入力します。

project2 ディレクトリは空になっています。project2 ディレクトリを削除するには、project2 の親ディレクトリであるログイン・ディレクトリに移動してから、`rmdir dirname` コマンドを実行します。その後、`ls` を入力して、project2 ディレクトリが削除されていることを確認します。

```
$ cd
$ rmdir project2
$ ls
file1  file2  file3  project/  record1  record6  reports/
$
```

ログイン・ディレクトリから project2 ディレクトリが削除されています。

4.6.4 ファイルとディレクトリの同時削除 (`rm -r` コマンド)

`rmdir` コマンドで削除できるのはディレクトリだけで、ファイルを削除することはできません。ただし、`-r` (recursive) フラグを指定して `rm` コマンドを使用すると、ファイルとディレクトリを同時に削除することができます。

`rm -r` コマンドは指定したディレクトリからファイルを削除した後に、ディレクトリ自体を削除します。このコマンドでは、指定したディレクトリおよびその下のサブディレクトリが(中に入っているファイルを含めて)すべて削除されます。このコマンドの使用に際しては注意が必要です。

`rm -r` コマンドの構文は次のとおりです。

`rm -r pathname`

pathname 変数には、削除したいディレクトリの完全パス名あるいは相対パス名を指定します。パターン照合文字を使用して、ファイルを指定することもできます。

注意

`-r` フラグを使用する場合は、必ずその機能を理解してから使用してください。たとえば、ログイン・ディレクトリから `rm -r *` コマンドを実行すると、ユーザがアクセスできるすべてのファイルおよびディレクトリが削除されます。ユーザがスーパーユーザ権限を持ち、ルート・ディレクトリにいる場合には、このコマンドはすべてのシステム・ファイルを削除します。スーパーユーザ権限についての詳細は、5.7 節を参照してください。

ファイルまたはディレクトリを削除するために `rm -r` コマンドを使用する場合は、次のように `-i` フラグを指定するとよいでしょう。

`rm -ri` *pathname*

この形式でコマンドを入力すると、システムは指定された要素を削除する前に、確認のためのプロンプトを表示します。プロンプトに対して `y` (yes) あるいは `n` (no) と入力することによって、ファイルまたはディレクトリを実際に削除するかどうかを制御できます。ただし、`-ri` オプションを使用すると、多くのプロンプトに対して応答しなければならない場合があります。



5

ファイルとディレクトリへのアクセス制御

この章では、次のような項目について説明することにより、システムあるいはファイルおよびディレクトリへのアクセスを制御する方法を説明します。

- パスワード、グループ、およびシステム・セキュリティ (5.1 節)
- ファイルおよびディレクトリ許可 (5.2 節)
- ファイルおよびディレクトリ許可の表示および設定 (5.3 節)
- 所有者とグループの変更 (5.8 節)
- ファイル・アクセス ID の変更 (5.6 節)
- スーパユーザ (5.7 節)
- セキュリティに関する追加情報 (5.9 節)

この章で説明する例に従って実際に操作することにより、上記のトピックについて理解することができます。

例を実行するためには、まずシステムにログインする必要があります。また、ログイン・ディレクトリには次のファイルおよびディレクトリが必要です。

- `file1` , `file2` , `file3` , `record1` , `record6` の各ファイル
- `file1` , `file2` , `file3` , `notes` の各ファイルおよび `newstatus` サブディレクトリを含む `reports` サブディレクトリ
- `file1` , `file2` , `file3` , `notes` の各ファイルおよび `status` サブディレクトリを含む `project` サブディレクトリ

上記の名前とは異なるファイル名を使用している場合は、適当に置き換えて例を実行してください。

5.1 パスワード・セキュリティ・ファイルとグループ・セキュリティ・ファイル

ユーザがシステムにログインするためには、ユーザ・アカウントを作成しておく必要があります。通常、ユーザ・アカウントの追加はシステム管理者が行います。ユーザ・アカウントの追加は日常的な作業ですが非常に重要です。

ユーザ・アカウントが作成されると、新しいユーザに関する情報が次の2つのファイルに追加されます。

`/etc/passwd` このファイルには、システム的全ユーザについてのユーザ情報が含まれている。

`/etc/group` このファイルには、システム上の全グループについてのグループ情報が含まれている。

これらのファイルはシステムを使用できるユーザ名と各ユーザのアクセス権を定義します。さらに、他のすべてのシステム・セキュリティ制御も、パスワード・セキュリティおよびグループ・セキュリティに依存しています。以降の各項で、`/etc/passwd` ファイルと `/etc/group` ファイルについて説明します。

5.1.1 `/etc/passwd` ファイル

`/etc/passwd` ファイルには、システム的全ユーザのログイン・アカウントとその属性の定義が含まれています。このファイルを変更するためには、スーパーユーザ特権が必要です。詳しくは 5.7 節を参照してください。

`/etc/passwd` ファイル内の各記述は、個々のユーザのログイン・アカウントを定義しています。各フィールドはコロンで区切られ、最後のフィールドは改行復帰文字 (リターン) で終わります。`/etc/passwd` ファイルの情報は、次のような形式で記述します。

```
username:password:UID:GID:gecos:login_directory:login_shell
```

各フィールドの意味は次のとおりです。

`username` ユーザのログイン名

`password` 暗号化されたユーザのパスワード

5-2 ファイルとディレクトリへのアクセス制御

暗号化することによって、権限のないユーザまたはプログラムがパスワードを不正に使用するのを防止する。パスワードが設定されていないユーザの場合、このフィールドは空白になる。

UID

ユーザ ID

システムがユーザを識別するための固有の番号。

GID

グループ ID

システムがユーザの省略時のグループを識別するための番号。ユーザは 1 つまたは複数のグループに所属できる。

gecos

通常、このフィールドにはユーザ自身に関する一般情報が含まれ、インストレーション固有のフォーマットで保存される。通常は、以下の情報をコンマで区切ったリストである。

name フルネーム

office オフィスの番号

wphone オフィスの電話番号

hphone 自宅の電話番号

login_directory

ログイン・ディレクトリ

システムにログインしたときの現在のディレクトリ。ユーザが自分専用のファイルを格納するために使用する。

login_shell

ログイン・シェル

システムにログインした時に login プログラムによって実行されるプログラム。通常は、コマンドを解釈するために使用されるシェル・プログラム。シェルについての詳細は、第 7 章および第 8 章を参照。

次に示すのは `/etc/passwd` ファイルの記述例です。

```
lee:NebPsa9qxMkbD:201:20:Lee Voy,sales,x1234:/users/lee: \
                                     /usr/bin/posix/sh
```

ユーザ・アカウント `lee` はユーザ ID が 201 で、グループ ID が 20 です。Lee のフルネームは Lee Voy で、所属は営業、内線番号 1234 です。ログイン・ディレクトリは `/users/lee` で、POSIX シェル (`/usr/bin/posix/sh`) がコマンド・インタプリタとして定義されます。パスワード・フィールドには暗号化された Lee のパスワードが記述されています。

5.1.2 `/etc/group` ファイル

`/etc/group` ファイルには、システムを使用するすべてのグループのログイン・アカウントの定義が含まれています。このファイルを変更するためには、スーパーユーザ特権を必要とし、`root` としてログインする必要があります。ユーザ名は '`etc/group`' ファイルに追加します。詳細は 5.7 節を参照してください。

グループ・データベース内の各記述は、それぞれ 1 つのグループのログイン・アカウントを定義しています。グループを定義することにより、共通の関心事のあるユーザや、同じプロジェクトにかかわっているユーザの間でファイルを共用することができます。

`/etc/group` ファイルの各行は、4 つのフィールドで構成されています。各フィールドはコロンの区切られ、最後のフィールドは改行復帰文字で終わります。`/etc/group` ファイルの情報は、次のような形式で記述します。

```
groupname:password:GID:user1[, user2, ..., userN ]
```

各フィールドの意味は次のとおりです。

<i>groupname</i>	システムがグループを識別するための固有な文字列。
<i>password</i>	このフィールドは常に空。このフィールドのエントリは無視される。
<i>GID</i>	グループ ID システムがグループを識別するための固有な番号。
<i>usernames</i>	グループに所属しているユーザのリスト。

5.2 ファイルとディレクトリの保護

オペレーティング・システムには、自分のファイルおよびディレクトリへのアクセスを制御するための多数のコマンドが用意されています。ファイルやディレクトリは、許可を設定または変更することによって保護することができます。許可とは、システムで作業しているユーザが格納データをどのように使用できるかを決定するコードです。

許可の設定または変更は、ファイルやディレクトリに対する保護の設定または変更と同じことです。一般に次のような場合はデータを保護します。

- ファイルあるいはディレクトリに機密情報が含まれている場合
- 他のユーザがファイルあるいはディレクトリを勝手に変更するのは問題がある場合

注意

多くの場合、システムでは個々の動作をお互いに通知することなく複数のユーザが同時に同じファイルの変更を行うことが可能です。この場合、システムは最後のユーザが行った変更を保管してファイルをクローズします。そのため、それ以前に他のユーザが行った変更は失われます(テキスト・エディタのなかには、このことをユーザに警告するものもあります)。このような問題を避けるために、許可されたユーザだけがファイルを修正できるようにファイル許可を設定すると良いでしょう。また、各ユーザは、ファイルを使用した時期および理由を通知し合うのが望ましいでしょう。

各ファイルおよびディレクトリには、関連する 9 つの許可を設定することができます。

ファイルとディレクトリに対しては次の 3 種類の許可を設定できます。

- `r` (読み取り)
- `w` (書き込み)
- `x` (実行)

また、これら 3 種類の許可は、次の 3 つのクラスのユーザに対して設定できます。

- **u (ユーザ/所有者)**
ファイルまたはディレクトリのユーザ/所有者は、通常それらを作成したユーザです。
- **g (グループ)**
グループは、ファイルが所属するグループを指定します。
所定のグループに属するすべてのユーザは、そのグループのアクセス許可を取得します。
- **o (他のすべてのユーザ; ワールドとも呼ばれる)**
すべてのユーザはその他のアクセス許可を取得します。

注意

グループ・アクセス許可はそのグループに属するすべてメンバーに与えられます。その他のアクセス許可は、すべてのユーザーに与えられます。特定のユーザあるいはグループ・メンバーに対してアクセス許可を設定するには、アクセス制御リスト (ACL) を使用します。ACL についての詳細は付録 G を参照してください。

表 5-1 に示すように、3 種類の許可は、ファイルとディレクトリとで多少意味が異なります。

表 5-1: ファイル許可とディレクトリ許可の違い

許可	ファイル	ディレクトリ
r (読み取り)	内容の表示および印刷が可能。	内容の参照は可能だが探索はできない。通常、r と x を一緒に指定する。
w (書き込み)	内容の変更および削除が可能。	ファイルあるいはサブディレクトリの追加および削除が可能。
x (実行)	ファイルをプログラムとして使用できる。	ディレクトリの探索が可能。

5.3 ファイル許可およびディレクトリ許可の表示 (ls コマンド)

現在のファイル許可を表示するには、ls コマンドに -l フラグを指定して使用します。1つのファイルあるいは指定したファイルの許可を表示する場合は、次の構文を使用します。

```
$ ls -l filename
```

filename 変数には、ファイル名あるいはスペースで区切ったファイル名リストを指定します。パターン照合文字を使用してファイルを指定することもできます。詳細については 5.4.1.3 項を参照してください。

現在のディレクトリにあるすべてのファイルの許可を表示するには、*filename* 変数を省略して ls -l コマンドを入力します。

```
$ ls -l
total 7
-rw-r--r-- 1 larry system 101 Jun 5 10:03 file1
-rw-r--r-- 1 larry system 171 Jun 5 10:03 file2
-rw-r--r-- 1 larry system 130 Jun 5 10:06 file3
drwxr-xr-x 2 larry system 32 Jun 5 10:07 project
-rw-r--r-- 1 larry system 0 Jun 5 11:03 record1
-rw-r--r-- 1 larry system 0 Jun 5 11:03 record6
drwxr-xr-x 2 larry system 32 Jun 5 10:31 reports
$
```

上記の表示の先頭の文字列は、そのファイルまたはディレクトリの許可を示しています。たとえば、4 番目のエントリの drwxr-xr-x は、次のことを示しています。

- 最初の d -- ディレクトリ。
- 次の rwx -- 所有者によるディレクトリ内容の参照、ディレクトリへの書き込み、ディレクトリの検索が可能。
- 次の r-x -- 所有者と同じグループに属するユーザによるディレクトリ内容の参照および検索は可能だが、書き込みはできない。
- 次の r-x -- 他のすべてのユーザによる参照および検索は可能だが、書き込みはできない。

上記の表示の 3 番目のフィールド (larry) はファイルの所有者を示し、4 番目のフィールド (system) はファイルが所属するグループを示しています。

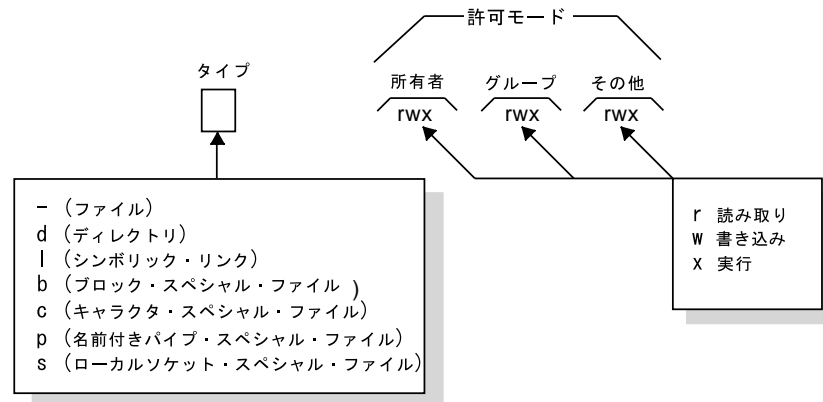
特定のディレクトリの許可を表示するには、次のように ls -ld コマンドを使用します。

```
$ ls -ld reports
drwxr-xr-x  2 larry    system      32 Jun  5 10:31 reports
$
```

ファイルまたはディレクトリに関する許可を一括して、許可コードと呼びます。図 5-1 に示すように、許可コードは 4 つの部分から構成されています。

- 最初の 1 文字はファイル・タイプを示します。ダッシュ (-) は通常ファイル, d はディレクトリ, l はシンボリック・リンクを示します。それ以外の文字はいずれも入出力装置を示します。
- 次の 3 文字の許可フィールドはユーザ (所有者) 許可を示します。この許可は読み取り, 書き込み, 実行のいずれの組み合わせも可能です。
- 次の 3 文字の許可フィールドはグループ許可を示します。
- 次の 3 文字の許可フィールドは他のすべてのユーザについての許可を示します。

図 5-1: ファイルおよびディレクトリの許可フィールド



ZK0536URJ

ファイルまたはディレクトリを作成する際に、システムはあらかじめ決められた許可コードを自動的に設定します。ファイル許可コードの例を次に示します。

```
-rw-r--r--
```

このファイル許可コードは、所有者に対して読み取りおよび書き込み許可を与えているのに対し、グループと他のすべてのユーザに対しては、読み取り許可のみを与えています。許可フィールドに表示されるダッシュ (-) は、指定したユーザ・クラスではその操作が許可されていないことを示します。

5-8 ファイルとディレクトリへのアクセス制御

ディレクトリ許可コードの例を次に示します。

```
drwxr-xr-x
```

このディレクトリ許可コードは、所有者に対して読み取り、書き込み、および探索許可を与えているのに対し、グループと他のすべてのユーザに対しては、読み取りおよび探索許可を与えています。

システムが提供する省略時の許可コードを使用すれば、ファイルやディレクトリを作成するたびに明示的にコードを指定する必要はありません。ユーザ固有の省略時の許可コードを作成したい場合は、`umask` コマンドを使用してユーザ・マスクを変更する必要があります。`umask` コマンドについては、5.5 節を参照してください。

注意

`ls` コマンドで表示されるファイルの所有者、グループ、およびその他のアクセス許可により、プロセスによるファイルへのアクセスが許可されていたとしても、そのファイルの ACL がそのプロセスによるアクセスを認めていない場合があります。この現象は、そのファイルの属するグループと同じグループへのアクセス許可をそのプロセスが持っている場合であっても発生することがあります。ACL についての詳細は付録 G を参照してください。

5.4 ファイルおよびディレクトリ許可の設定 (`chmod` コマンド)

許可を変更できると、自分のデータの使用方法を自由に制御することができます。ファイルあるいはディレクトリの許可を設定または変更する場合は、`chmod` (`change mode`) コマンドを使用します。

たとえば、ユーザはファイルを読み取ったり、修正したり、実行したりすることを自分に許可しています。通常は、同じグループの他メンバにも、ファイルの読み取りを許可しています。作業の性質とグループの構成によっては、他のメンバにファイルの修正または実行を許可する場合もあるでしょう。ただし、他のすべてのシステム・ユーザには、ファイルにアクセスすることを禁止するのが一般的です。

注意

ファイルまたはディレクトリの許可を変更するためには、ファイルまたはディレクトリの所有者である (またはスーパーユーザの権限を持っている) 必要があります。つまり、`ls -l` コマンドを実行したときに、ユーザ名がそのファイルのリストの 3 番目のフィールドに表示されなければなりません。

重要なことですが、ファイルやディレクトリへのアクセスをどんなに制限しても、スーパーユーザはいつでもその制限を無効にできるということを認識しておいてください。たとえば、`chmod` コマンドを使用して、自分しかファイル `report20` にアクセスできないように指定したとします。しかし、そのように設定しても、スーパーユーザはそのファイルにアクセスできます。詳細は、5.7 節を参照してください。

`chmod` コマンドを使用して許可を設定する場合には、次の 2 通りの指定方法があります。

- 許可を英字と演算記号で指定する。
- 許可を 8 進数で指定する。

以降の各項で、許可を英字と演算記号で指定する方法と、8 進数で指定する方法について説明します。

5.4.1 英字と演算記号による許可の指定

英字と演算記号を使用して、ファイルおよびディレクトリの許可を変更することができます。

英字と演算記号を使用するときの `chmod` コマンドの形式は次のとおりです。

`chmod` *userclass-operation-permission filename*

userclass-operation-permission 変数は、有効にしたいユーザ・クラス/グループ/操作/許可コードを指定する 3 つのコードを表します。*filename* 変数には、許可を変更したいファイルの名前を指定します。パターン照合文字を使用してファイルを指定することもできます。詳しくは 5.4.1.3 項を参照してください。

ユーザ・クラス、操作、および許可は次のように定義されます。

- *userclass* を表すための英字

- u ユーザ (所有者)
- g グループ
- o 他のすべてのユーザ (所有者とグループを除く)
- a 全ユーザ (所有者, グループ, 他のすべてのユーザ)
- *operation* を表すためのシンボル
 - + 許可の追加
 - 許可の削除
 - = 以前の設定を無視して許可の割り当て
- *permission* のタイプを表す文字
 - r 読み取り
 - s ユーザまたはグループ ID の設定。この許可ビットでは、ファイルが起動されたときに、ファイルの所有者またはグループ所有者を実効ユーザ ID または実効グループ ID として設定します。この許可設定を u または g オプションと組み合わせて使用すると、通常はその他のユーザにはアクセスできないファイルに対して、一時的または制限付きのアクセスを許可できるようになります。詳細リスト (ls を参照) では、ユーザまたはグループの実行許可の位置に s が表示されて、ファイルが set-user-ID または set-group-ID の許可モードで実行されることを示します。
 - w 書き込み
 - x 実行

5.4.1.1 ファイル許可の変更

次の例では、まず `ls -l` コマンドを入力して、ファイル `file1` の許可を表示します。

```
$ ls -l file1
-rw-r--r--  1 larry   system    101 Jun  5 10:03 file1
$
```

所有者 (larry) に読み取りと書き込み許可があるのに対し、グループおよび他のユーザには読み取り許可しかないことに注意してください。

ここで、フラグ `go+w` を指定して `chmod` コマンドを入力します。このコマンドは、グループ (g) と他のユーザ (o) の両方に、すでに認められている読み取りアクセスに加えて、`file1` に対する書き込み (+w) アクセスを与えることによって、許可を拡張するものです。

```
$ chmod go+w file1
$
```

次に、そのファイルの新しい許可を表示します。

```
$ ls -l file1
-rw-rw-rw- 1 larry  system    101 Jun  5 10:03 file1
$
```

これで、グループと他のすべてのシステム・ユーザに、file1 に対する書き込み許可が与えられました。

5.4.1.2 ディレクトリ許可の変更

ディレクトリ許可の変更手順は、ファイル許可の変更手順と同じです。ただし、ディレクトリについての情報をリストするときは、ls -ld コマンドを使用します。

```
$ ls -ld project
drwxr-xr-x 2 larry  system    32 Jun  5 10:07 project
$
```

ここで、chmod g+w コマンドで許可を変更し、グループ(g)にディレクトリ project に対する書き込み許可(+w)を認めるようにします。

```
$ chmod g+w project
$ ls -ld project
drwxrwxr-x 2 larry  system    32 Jun  5 10:07 project
$
```

5.4.1.3 パターン照合文字の使用

ディレクトリ内のすべてのエントリの許可に対して同じ変更を行いたい場合は、chmod コマンドでパターン照合文字のアスタリスク(*)を使用することができます。パターン照合文字については第2章を参照してください。

次の例では、chmod g+x * コマンドは、グループ(g)に対し、現在のディレクトリ内のすべてのファイル(*)について実行(x)許可を与えます。

```
$ chmod g+x *
$
```

ここで、ls -l コマンドを入力し、グループに対して現在のディレクトリ内のすべてのファイルについての実行(x)許可が与えられたことを確認してください。

```
$ ls -l
total 7
-rw-rwxrw- 1 larry  system    101 Jun  5 10:03 file1
```

5-12 ファイルとディレクトリへのアクセス制御

```
-rw-r-xr-- 1 larry system 171 Jun 5 10:03 file2
-rw-r-xr-- 1 larry system 130 Jun 5 10:06 file3
drwxrwxr-x 2 larry system 32 Jun 5 10:07 project
-rw-r-xr-- 1 larry system 0 Jun 5 11:03 record1
-rw-r-xr-- 1 larry system 0 Jun 5 11:03 record6
drwxr-xr-x 2 larry system 32 Jun 5 10:31 reports
$
```

5.4.1.4 絶対許可の設定

絶対許可割り当て(=)は、許可が以前どのように設定されていたかに関係なく、ファイルについてのすべての許可を再設定します。

例 5-1 では、`ls -l` コマンドで `file3` ファイルの許可を表示した後、`chmod a=rwx` コマンドで、すべてのユーザ(a) に対し 3 つの許可(rwx) すべてを与えます。

例 5-1: 絶対許可の設定

```
$ ls -l file3
-rw-r-xr-- 1 larry system 130 Jun 5 10:06 file3
$ chmod a=rwx file3
$ ls -l file3
-rwxrwxrwx 1 larry system 130 Jun 5 10:06 file3
$
```

許可を削除するために絶対割り当てを使用することもできます。例 5-2 では、`chmod a=rw newfile` コマンドは、ファイル `file3` からすべてのユーザ(a) の実行許可(x) を削除します。

例 5-2: 絶対許可の削除

```
$ chmod a=rw file3
$ ls -l file3
-rw-rw-rw- 1 larry system 130 Jun 5 10:06 file3
$
```

5.4.2 8 進数による許可の指定

8 進数を使用して、ファイルおよびディレクトリの許可を変更することもできます。

8 進数許可コードを `chmod` コマンドで使用するときは、次の形式でコマンドを入力します。

chmod *octalnumber filename*

octalnumber 変数は、所有者、グループ、および他のユーザについての許可を指定する 3 桁の 8 進数です。 *filename* 変数は、許可を変更するファイルの名前です。これはファイル名でも、スペースで区切ったファイル名リストでもかまいません。パターン照合文字を使用してファイルを指定することもできます。詳しくは 5.4.1.3 項を参照してください。

8 進数は各タイプの許可に対応します。

4 = read
2 = write
1 = execute

許可のグループ (許可フィールド) を指定するには、該当する 8 進数を加算します。 *r* , *w* , および *x* はそれぞれ、読み取り、書き込み、および実行を示します。

3 = -wx (2 + 1)
6 = rw- (4 + 2)
7 = rwx (4 + 2 + 1)
0 = --- (許可なし)

表 5-2 は、8 通りの許可の組み合わせを列記したものです。

表 5-2: 許可の組み合わせ

8 進数	2 進数	許可	説明
0	000	なし	許可は一切与えられていない
1	001	--x	実行
2	010	-w-	書き込み
3	011	-wx	書き込み/実行
4	100	r--	読み取り
5	101	r-x	読み取り/実行
6	110	rw-	読み取り/書き込み
7	111	rwx	読み取り/書き込み/実行

ファイルまたはディレクトリの完全な許可コードは 3 桁の 8 進数で指定され、それぞれの数字が所有者、グループ、および他のユーザに対応します。表 5-3 に、いくつかの代表的な許可コードと、それらと許可フィールドとの関係を示します。

表 5-3: 8 進数と許可フィールドとの関係

8 進数	所有者 フィールド	グループ・ フィールド	他のユーザ・ フィールド	完全なコード
777	rwX	rwX	rwX	rwXrwxrwx
755	rwX	r-X	r-X	rwXr-Xr-X
700	rwX	---	---	rwX-----
666	rw-	rw-	rw-	rw-rw-rw-

次の例では、8 進数を使用して file3 の許可を変更しています。

```
$ ls -l file3
-rw-rw-rw- 1 larry system 130 Jun 5 10:06 file3
$ chmod 754 file3
$ ls -l file3
-rwxr-xr-- 1 larry system 130 Jun 5 10:06 file3
$
```

5.5 ユーザ・マスクを使用した省略時の許可の設定

ファイルやディレクトリを作成すると、それに対して省略時の許可が設定されます。この省略時の許可は、最初はオペレーティング・システムまたは実行しているプログラムのいずれかによって設定されます。これにより、ファイルやディレクトリを作成するたびに、許可コードを指定する手間が省けます。オペレーティング・システムは、省略時の許可として、実行ファイルに対しては 777 を、それ以外のファイルに対しては 666 を割り当てます。

ファイルまたはディレクトリの作成時にプログラムが設定する許可をさらに制限する場合は、umask コマンドを使用してユーザ・マスクを指定します。

ユーザ・マスクは、ファイルまたはディレクトリが作成されるときにアクセス許可を決定する数値です。したがって、ファイルまたはディレクトリを作成するとき、作成用プログラムが指定する許可から、umask 値が禁止するものをマイナスした結果に設定されます。

umask コマンドの形式は次のとおりです。

umask *octalnumber*

octalnumber 変数は、省略時の許可 (777 または 666) からマイナスする許可を指定する 3 桁の 8 進数です。

ユーザ・マスクの設定は、5.4.2 項で説明している許可ビットの設定に非常に似ています。ファイルやディレクトリの許可コードは 3 桁の 8 進数で指定されます。各桁が許可の種類を表します。各桁の位置 (1 桁目, 2 桁目, または 3 桁目) は、下記に対応する 3 ビットを表します。

- 1 桁目はファイルの所有者 (ユーザ) 用
- 2 桁目はファイルのグループ用
- 3 桁目はその他のユーザ用

ただし、ユーザ・マスクを設定すると、ファイルを作成するプログラムが要求する許可にかかわらず、どの許可を与えないかを実際に指定することになります。

表 5-4 は、可能な 8 通りの unmask 許可の組み合わせをわかりやすく列記したものです。umask 許可の値は、通常の許可コードで指定される値の逆です。これらの許可値は、ファイル作成プログラムによって設定される値に対して適用されます。

表 5-4: umask 許可の組み合わせ

8 進数	許可	説明
0	rwx	読み取り/書き込み/実行
1	rw-	読み取り/書き込み
2	r-x	読み取り/実行
3	r--	読み取り
4	-wx	書き込み/実行
5	-w-	書き込み
6	--x	実行
7	なし	許可は一切与えられていない

たとえば、027 というユーザ・マスクを指定すると、次のような許可が設定されます (ファイルは実行ファイルとします)。

- 所有者には、ファイル作成プログラムが要求するすべての許可が認められる。
- グループには書き込み許可は与えられない。
- その他のユーザには許可は全く与えられない。

自分のファイルおよびディレクトリに設定するユーザ・マスク値は、情報資源をシステムでどのように共用するかによって異なります。次のガイドラインを参考にしてください。

- 非常にオープンなコンピュータ環境では、ユーザ・マスク値として 000 を指定するとよいでしょう。これはファイルおよびディレクトリのアクセスに一切制限を認めないものです。この場合、プログラムがファイルを作成して許可コードを指定するとき、ユーザ・マスクは、作成プログラムが指定した許可に一切制限を加えないことになります。
- もう少し安全性が要求される環境では、ユーザ・マスク値として 066 を指定するとよいでしょう。これは所有者には全面的にアクセスを認めますが、他のすべてのユーザには、ファイルの読み取りおよび書き込みを禁じるものです。この場合、ファイルが作成されるとき、許可は、作成プログラムが指定する許可から、所有者以外のすべてのユーザに読み取り/書き込みアクセスを禁止するユーザ・マスク制限をマイナスした値に設定されます。
- 安全性が非常に重視される環境では、ユーザ・マスク値として 077 を指定するとよいでしょう。これは、ユーザだけがファイルにアクセスできるということです。この場合、ファイルを作成するとき、許可は、作成プログラムが指定する許可から、ユーザ以外のすべての人にファイルの読み取り、書き込み、実行を禁止するユーザ・マスク制限をマイナスした値に設定されます。

`umask` がどのように機能するかを理解するため、次のコマンドを入力してみましょう。

```
$ umask 037
```

このコマンドは 740 という許可コードを設定し、次のような結果が生じます (ファイルが実行ファイルである場合)。

- 所有者にはすべての許可が認められる。
- グループのメンバには書き込みおよび実行許可は認められない。
- 他のユーザには一切の許可が認められない。

ファイルを作成した直後は、省略時の設定で、エディタは常に次の省略時の許可を割り当てます。つまり、所有者にはすべての許可が認められ、他のすべてのユーザには読み取りと実行許可だけが認められます。ただし、以前に 037 というユーザ・マスクを設定しているため、ファイル許可にさらに制限を加えます。その結果、所有者には相変わらずすべての許可が認められていますが、グループはファイルを実行できず、また他のすべてのユーザには許可が一切認められないことになります。

5.5.1 umask の設定

umask コマンドは 2 通りの方法で実行することができます。

- ログイン・スクリプトに含める。

これはユーザ・マスクを指定するうえで最も一般的でかつ効率的な方法です。こうしておくと、ログインするといつも、指定された値が自動的に設定されます。ログイン・スクリプトについての説明は、第 7 章を参照してください。ログイン・スクリプトに umask コマンドを記述する例については、第 8 章を参照してください。

- ログイン・セッション中にシェル・プロンプトに対して入力する。

設定されるユーザ・マスク値はそのログイン・セッションについてのみ有効です。

テキスト・エディタによって作成されるファイルの許可を制限する際に、ユーザ・マスクがどのように機能するかを示す例として、次の操作を実行してみてください。

1. 次のコマンドを入力して、自分のユーザ・マスクの現在値を調べる。

```
$ umask
```

ユーザ・マスク値が 000 であれば、ファイルを作成するプログラムが設定する許可に対して一切制限を加えません。ステップ 3 に進んでください。

ユーザ・マスク値が設定されていれば、それを書き留めます。その後、ステップ 2 に進んでください。

2. ユーザ・マスク値を 000 に設定する。

これにより、ファイルを作成するプログラムによって設定される許可に対して一切の制限を加えません。ユーザ・マスクを設定する前に、現

在のユーザ・マスク値を必ず書き留めておいてください。再設定する必要が生じるかもしれません。

次のコマンドを入力します。

```
$ umask 000
```

3. ファイルを作成し、保管してから、エディタを終了する。
4. `ls-l` コマンドを使用して、そのファイルの許可を表示する。
読み取り/書き込み許可がすべてのユーザに与えられているとします。

```
$ ls -l
-rw-rw-rw-  1 user-name  15 Oct 27 14:42 yourfile
$
```

5. 次のコマンドを入力して、ユーザ・マスクを `022` に再設定する。

```
$ umask 022
```

ユーザ・マスク `022` は次のとおり許可制限を設定します。つまり、所有者にはすべての許可が認められ、その他のすべてのユーザには読み取りおよび実行許可だけが認められます。

6. 別のファイルを作成し、保管してから、エディタを終了する。
7. `ls -l` コマンドを使用して、ファイルの許可を表示する。

```
$ ls -l
-rw-r--r--  1 user-name  15 Oct 27 14:45 yourfile2
$
```

ここで、ユーザ・マスク値 `022` に従って、グループと他のすべてのユーザに対する書き込み許可が削除されます。

8. ユーザ・マスクを元の値または別の値 (任意) に再設定する。

注意

ユーザ・マスクでファイルやディレクトリへのアクセスにどんな制限を課しても、スーパーユーザ特権を有するユーザは、その制限を無効にできます。詳細は、5.7 節を参照してください。

ユーザ・マスクを指定して得られる結果が、意図したものと異なる場合は、システム管理者に問い合わせてください。

オペレーティング・システムでの省略時のユーザ・マスク値は 022 です。これは所有者にはすべてのアクセスを許可し、グループのメンバや他のすべてのユーザにはファイルへの書き込みを禁じるものです。ただし、ご使用のシステムの設定によってはユーザ・マスクの省略値が異なる場合もあります。

5.6 ファイルにアクセスするための ID の変更

`su` コマンドを使用すれば、ログイン・セッション中に ID を変更することができます。ID を変更する理由は、自分が所有していないファイルにアクセスできるようにするためです。システムのセキュリティを保護するため、所有者またはシステム管理者の許可がない限り、別の ID を引き継ぐべきではありません。

`su` コマンドを使用すれば、そのユーザのパスワードを知っている場合に限って、そのアカウントにログインできます。`su` コマンドはユーザを認証してから、プロセスのユーザ ID と実効ユーザ ID の両方を、新たに指定されたユーザ ID の値に再設定します。実効ユーザ ID とは、プロセスで現在有効なユーザ ID のことです。ただし、それはログインしている人のユーザ ID ではないこともあります。

`su` コマンドの形式は次のとおりです。

`su username`

`username` 変数は、引き継ぎたい ID を持っている人のユーザ名です。

ID を変更した後、自分が引き継いだ ID を確認したい場合は、`whoami` コマンドを使用してください。このコマンドは自分が引き継いだ ID のユーザ名を表示します。

新しい ID の下での作業が完了したら、自分自身のログイン ID に戻ります。これを行うには、`Ctrl/D` を押すか、または `exit` コマンドを入力します。

次の例 5-3 では、Juan が Lucy の ID を引き継ぎ、コマンドでそのことを確認し、ファイルを削除してから、コマンドで自分のログイン ID に戻る方法を示しています。

例 5-3: su コマンドの使用

```
$ whoami 1
juan
$ su lucy 2
Password: ... 3
$ whoami 4
lucy
$ rm file9 5
$ exit 6
$ whoami 1
juan
$
```

- 1 whoami コマンドで ID を確認しています。
 - 2 su コマンドで lucy の ID を使用できるようにしています。
 - 3 セキュリティ上の理由から、入力したパスワードは画面には表示されません。
 - 4 ID が lucy になっていることを確認しています。
 - 5 ファイルを削除しています。
 - 6 exit コマンドを実行すると元の ID に戻ります。
-

詳しくは、su(1) および whoami(1) リファレンス・ページを参照してください。

5.7 スーパユーザの概念

システムには、一般ユーザの許可を無効にするスーパユーザが存在します。スーパユーザはルートとも呼ばれます。

ルート・ユーザはシステムの実行に関する絶対的権限を持っています。このユーザにはすべてのファイルとすべての装置にアクセスする権利があり、システムに対して変更を行うことができます。ルート・ユーザはスーパユーザ特権を持っているユーザであるとも表現できます。

ルート・ユーザが実行する作業の例を次に示します。

- 通常では一般ユーザが変更できないファイル (たとえば、`/etc/passwd`) の編集
- すべてのファイルの所有権と許可の変更
- `mount` や `reboot` のような制限付きコマンドの実行
- システムで実行しているプロセスの強制終了

- ユーザ・アカウントの追加，削除
- システムのブートおよびシャットダウン
- システムのバックアップ

上記のタスクの多くは通常，システム管理者が実行するものです。これらの作業を実行するシステム管理者はスーパーユーザ特権を必要とします。システム管理者の任務はシステムを管理することであり，上記タスクの実行の他に，新しいソフトウェアをインストールしたり，システム性能を解析したり，ハードウェア故障を通報したりすることが含まれます。

ユーザがシステムのシステム管理者であるかどうか，つまり，ルート特権を持つか持たないかは，システム環境によります。サイト構成および担当する業務により，ユーザの特権が決定されます。

たとえば，端末またはワークステーションから集中システムにアクセスして作業を行う場合は，おそらくシステム管理者ではなくルート特権を持たないと思われます。このような場合は，システムの維持，構成，およびアップグレードに責任を負うシステム管理者が，ルート特権を持つことになります。

独立した，あるいは他のワークステーションまたはシステムとネットワークで接続されたワークステーションから自分のタスクを遂行している場合は，たしかに自分のワークステーションについてルート特権を持っているかもしれませんが，サイトのシステム管理者ではありません。このような場合には，自分のワークステーションだけを管理できます。共用マシンおよびネットワークはシステム管理者が管理することになります。

ルート・ユーザになるためには，`su` コマンドを使用しますが，ルート・ユーザのパスワードも知っていなければなりません。

su

次の例は，Juan が管理者のタスクを遂行するためにルート・ユーザとなる手順を示しています。

```
$ su
Password: ...
#
```

新しいプロンプトの番号記号 (#) は，Juan がルート・ユーザになり，新たにシェルが作成されたことを示します。ルート・ユーザのシェルは `/etc/passwd` ファイルに定義されています。これで Juan は管理者としてのタスクを遂行することができます。

注意

システムの不正使用によってデータの改ざんや破壊が発生する可能性があるため、システムに対して絶対的権限を持っているルート・ユーザは、パスワードの保護には十分に気をつけてください。

ルート・ユーザとしての作業を完了したら、自分のログイン ID に戻ります。これを行うには、Ctrl/Dを押すか、または `exit` コマンドを入力します。その後、システム・プロンプトが表示されます。

5.8 所有者とグループの変更 (`chown` および `chgrp` コマンド)

許可を設定するだけでなく、所有者やグループを変更することによっても、ファイルやディレクトリへのアクセスを制御できます。所有者を変更するときは `chown` コマンドを、また、グループを変更するときは `chgrp` コマンドを使用してください。

注意

`chown` コマンドを使用するためには、スーパーユーザ特権が必要です。詳しくは 5.7 節を参照してください。

`chown` コマンドの形式は次のとおりです。

`chown` *owner filename*

owner 変数はファイルの新しい所有者のユーザ名です。 *filename* 変数は、所有権を変更する 1 つまたは複数のファイルのリストです。パターン照合文字を使用してファイルを指定することもできます。詳しくは 5.4.1.3 項を参照してください。

`chgrp` コマンドの形式は次のとおりです。

`chgrp` *group file*

group 変数は、新しいグループのグループ ID またはグループ名です。ファイルのグループ所有権を変更するためには、ファイルの変更先のグループのメンバでなければなりません。 *file* 変数は、所有権を変更する 1 つまたは複数のファイルのリストを指定します。

詳しくは、`chown(1)` および `chgrp(1)` リファレンス・ページを参照してください。

5.9 セキュリティに関する追加情報

ユーザのサイトで実施されているセキュリティ・ガイドラインは、ファイルを無断アクセスから保護するためのものです。セキュリティ・ガイドラインについての詳細は、システム管理者に問い合わせてください。

さらに、信用の置けないソフトウェア (ソースのわからないものや、システム・セキュリティの点で妥当性が確認されていないソフトウェア) の実行は避けるのが賢明です。プログラムを実行するときには、そのプログラムはユーザの持つ全アクセス権を有しているため、そのプログラムが機密ファイルに違法にアクセスして読んだり、または改変しても、それを妨げる手段は何もありません。

セキュリティを危うくする 3 種類のプログラムに留意すべきです。

- トロイの木馬

トロイの木馬は、定義されたタスクを正しく遂行するか、または一見、遂行するように見えるプログラムです。しかし、このプログラムは悪意のある隠れた機能も実行します。トロイの木馬プログラムは、ユーザの意図するプログラムを実行するように見えますが、同時に望ましくない動作も実行します。ユーザのファイルを改変または削除することによって破壊するか、あるいは違法なコピーを作ることによってその価値を危うくしないとも限りません。

代表的なトロイの木馬は `login` というプログラムです。これはシステムのログイン・プロンプトを真似たプロンプトを画面に表示して、ユーザがユーザ名とパスワードを入力するのを待ちます。プログラムはこの情報を、トロイの木馬を組み込んだユーザにメールまたはコピーします。トロイの木馬が終了するとき、プログラムは `Login incorrect` と表示します。その後、本当の `login` プログラムが実行されます。ほとんどのユーザはパスワードを不正確に打ち込んだのだと思い、だまされたことには気付きません。

- コンピュータ・ワーム

コンピュータ・ワームは、コンピュータ・ネットワークを動きまわって、それ自体のコピーを作るプログラムです。たとえば、ログイン・コンピュータ・ワームはシステムにログインし、そのシステムに自らをコ

ピーすると実行を開始し、また別のシステムにログインします。こうして、このプロセスを無限に続けて行きます。

- コンピュータ・ウイルス

コンピュータ・ウイルス・プログラムはトロイの木馬の一種です。通常、トロイの木馬はお目当てのユーザ（一般には特権を持つユーザ）が実行してくれるのを受動的に待ちます。しかしウイルスは、他の実行可能なファイルに侵入して自らを伝播させていきます。そのため、プライバシーや完全性に対する脅威と被害の範囲が増大します。

システム管理者がインストールしたものではないプログラムには注意してください。掲示板や信頼できないソースから入手したプログラムは、特に用心が必要です。たとえプログラムのソース・コードがある場合でも、それが信頼に値するかどうかを確定することは、十分慎重にプログラムを検査したとしても常に可能なわけではありません。



プロセスの使用

この章では、オペレーティング・システムのプロセスについて説明します。
この章では次のような項目について説明します。

- プログラムおよびプロセスの理解 (6.1 節)
- プロセス入力、出力、およびエラーのリダイレクト (6.2 節)
- フォアグラウンドとバックグラウンドでのプロセスの実行 (6.3 節)
- プロセスの状態のチェック (6.4.1 項)
- プロセスの取り消し (6.4.2 項)
- ユーザとユーザのプロセスに関する情報の表示 (6.5 節)

この章で説明する例に従って実際に操作することにより、上記のトピックについて理解することができます。画面に表示される情報が本書の説明と一致するように、必ずすべての例を実行してください。

6.1 プログラムとプロセス

プログラムとは、コンピュータが解釈して実行することのできる命令の集合です。ほとんどのプログラムは、次の2つのカテゴリのどちらかに含まれます。

- アプリケーション・プログラム
テキスト・エディタ、課金パッケージ、あるいは表計算シートなどが含まれます。
- オペレーティング・システムの構成要素であるプログラム
コマンド、シェル、ログイン・プロシージャなどが含まれます。

プログラムは、実行されている間はプロセスと呼ばれます。オペレーティング・システムはすべてのプロセスに、プロセス識別子 (PID) と呼ばれる一意の番号を割り当てます。

オペレーティング・システムでは、複数の異なるプロセスを同時に実行することができます。複数のプロセスを実行しているときは、オペレーティング・システムに組み込まれているスケジューラが、設定されている優先順位に基づいて、各プロセスに対しコンピュータ時間の公正な割り当てを行います。

6.2 標準入力，出力，およびエラー

プロセスが実行を開始すると、オペレーティング・システムはそのプロセスのために 3 つのファイル、`stdin` (標準入力)、`stdout` (標準出力)、および `stderr` (標準エラー) をオープンします。プログラムはこれらのファイルを次のように使用します。

- 標準入力

プログラムが入力を読み取る場所です。省略時には、プロセスはキーボードから `stdin` を読み取ります。

- 標準出力

プログラムが出力を書き込む場所です。省略時には、プロセスは画面に `stdout` を出力します。

- 標準エラー

プログラムがエラー・メッセージを書き込む場所です。省略時には、プロセスは画面に `stderr` を出力します。

たいていの場合は、省略時の標準入力、標準出力、および標準エラーで十分ですが、標準入力、標準出力、および標準エラーをリダイレクトするほうが便利な場合があります。以降の各項で、これらの手順について説明します。

6.2.1 入力と出力のリダイレクト

コマンドは通常、キーボード (標準入力) から入力を読み取り、ディスプレイ (標準出力) に出力を書き込みます。しかし場合によっては、コマンドに、ファイルから入力を読み取らせたり、ファイルに出力を書き込ませたり、あるいはその両方を行わせたいこともあるでしょう。表 6-1 に示すシェル表記を使用すると、コマンドに対する入力ファイルと出力ファイルを選択することができます。この表記はすべてのシェルで使用できます。

表 6-1: 入力読み取りと出力リダイレクトのためのシェル表記

表記	作業	例
<	ファイルから標準入力を読み取る。	wc < file3
>	ファイルに標準出力を書き込む。	ls > file3
>>	ファイルの終わりに標準出力を追加する。	ls >> file3

以降の項で、ファイルから入力を読み取る方法と、ファイルに出力を書き込む方法について説明します。

6.2.1.1 ファイルからの入力の読み取り

すべてのシェルで、プロセスの標準入力をリダイレクトして、入力をキーボードからではなくファイルから読み取るようにすることができます。

入力リダイレクションは、stdin (ユーザのキーボード) からの入力を受け入れるコマンドであれば、どんなコマンドとも使用することができます。who などの入力を受け入れないコマンドでは、入力リダイレクションは使用できません。

入力をリダイレクトするには、左山カッコ (<) を次の例のように使用します。

```
$ wc < file3
      3      27     129
$
```

wc (word count) コマンドは指定されたファイル内の行数、ワード数、およびバイト数をカウントします。したがって、file3 には 3 行、27 ワード、129 バイトが含まれています。引数を指定しなければ、wc コマンドはキーボードからの入力を読み取ります。この例では、wc の入力は file3 という名前のファイルから読み取られています。

なお、次のように入力しても、同じ出力を表示させることができます。

```
wc file3
```

これは、ほとんどのコマンドでは、< を使用しなくても入力ファイルを指定できるからです。

ただし、mail のように、特殊機能のために < の使用を要求するコマンドも少数ながらあります。たとえば、次のコマンドに注意してください。

```
$ mail juan < report
```

このコマンドは、ユーザ `juan` にファイル `report` をメールします。メールについての詳細は、`mail(1)` リファレンス・ページを参照してください。

6.2.1.2 出力のリダイレクト

すべてのシェルで、プロセスの標準出力を画面 (省略時の出力) からファイルにリダイレクトすることができます。この結果、コマンドによって生成されたテキストを新しいファイルまたは既存のファイルに格納することができます。

出力をファイルに送るには、1 つまたは 2 つの右山カッコ (`>` または `>>`) を使用します。

`>` を使用した場合、シェルは次のことを行います。

- ファイルが存在する場合、そのファイルの内容をコマンドの出力に置き換える。
- ファイルが存在しない場合、コマンドの出力を内容とするファイルを作成する。

`>>` は、コマンドの出力を既存のファイルの終わりに追加 (付加) します。ファイルが存在しない場合は、シェルはコマンドの出力を内容とするファイルを作成します。

次の例では、`ls` の出力は `file` という名前のファイルに書き込まれます。

```
$ ls > file
$
```

このファイルがすでに存在する場合、シェルはその内容を `ls` の出力に置き換えます。`file` が存在しない場合は、シェルはそのファイルを作成します。

次の例では、シェルは `ls` の出力を `file` という名前のファイルの終わりに追加します。

```
$ ls >> file
$
```

`file` が存在しない場合、シェルはそのファイルを作成します。

プロセスは標準出力のほかに、診断出力と呼ばれるエラー・メッセージや状態メッセージを出力することがよくあります。診断出力のリダイレクションについては、次の項を参照してください。

6.2.2 標準エラーのファイルへのリダイレクト

コマンドの実行が成功すると、その結果が標準出力に表示されます。コマンドの実行が成功しなかった場合は、省略時の標準エラー・ファイル、つまり画面にエラー・メッセージが表示されます。しかし、シェルは、プロセスの標準エラーを画面からファイルにリダイレクトすることができます。

リダイレクションの記号と構文はシェルによって異なります。以降の各項で、Korn および POSIX シェルおよび C シェルにおける標準エラーのリダイレクションについて説明します。

6.2.2.1 Bourne シェル、Korn シェル、および POSIX シェルのエラー・リダイレクション

Bourne、Korn、または POSIX シェルの標準エラー・リダイレクションの一般的な形式は次のとおりです。

command **2>** *errorfile*

command 変数はオペレーティング・システムのコマンドです。 *errorfile* 変数は、プロセスが標準エラーを書き込むファイルの名前です。 **2>** は、ファイル記述子数字と出力リダイレクション記号 (**>**) を組み合わせたものです。ファイル記述子数字はアクセスする標準ファイルをシェルに指示して、その内容をリダイレクトできるようにするものです。ファイル記述子数字 **2** は、標準エラー・ファイルがリダイレクトされることを示します。

Bourne、Korn、および POSIX シェルの場合、ファイル記述子数字は次のように、コマンドが通常使用する各ファイルと対応しています。

- ファイル記述子 **0** (**<** と同じ) は標準入力 (キーボード) を指定する。
- ファイル記述子 **1** (**>** と同じ) は標準出力 (画面) を指定する。
- ファイル記述子 **2** は標準エラー (画面) を指定する。

次の例では、`ls` コマンドが存在しないファイル `reportx` を表示しようとすると、エラーは `error` ファイルにリダイレクトされます。その後、ファイル `error` の内容を表示します。

```
$ ls reportx 2> error
$ cat error
reportx not found
$
```

上記の例では標準エラーだけがファイルにリダイレクトされますが、標準エラーと標準出力の両方をリダイレクトするのが普通です。詳細は 6.2.3 項を参照してください。

多くのコマンドの場合、標準出力と標準エラーとの違いを理解するのはむずかしいものです。たとえば、`ls` コマンドを使用して存在しないファイルを表示させようとする、エラー・メッセージが画面に表示されます。上記の例のようにエラー・メッセージをファイルにリダイレクトしても、出力は同じです。

6.2.2.2 C シェルのエラー・リダイレクション

C シェルにおける標準エラー・リダイレクションの一般的な形式は次のとおりです。

```
( command > outfile )>& errorfile
```

command 変数はオペレーティング・システムのコマンドです。 *outfile* 変数は、プロセスが標準出力を書き込むファイルの名前です。 `>` は標準エラーをファイルにリダイレクトします。 *errorfile* 変数は、プロセスが標準エラーを書き込むファイルの名前です。なお、このコマンド形式のカッコは必ず必要です。

6.2.3 標準エラーと標準出力の両方のリダイレクト

これまでは、標準出力と標準エラーを別々にリダイレクトする方法について説明しました。しかし、通常は標準出力と標準エラーを同時にリダイレクトします。標準出力と標準エラーは、異なるファイルに書き込むことも、同じファイルに書き込むこともできます。

Bourne, Korn, および POSIX シェルの場合、標準出力と標準エラーの両方を異なるファイルにリダイレクトするための一般形式は次のとおりです。

```
command > outfile 2> errorfile
```

command 変数はオペレーティング・システムのコマンドです。 *outfile* 変数はプロセスが標準出力を書き込むファイルの名前です。 `2>` 記号はエラー出力をリダイレクトします。 *errorfile* 変数はプロセスが標準エラーを書き込むファイルの名前です。C シェルの場合、標準出力と標準エラーの両方を異なるファイルにリダイレクトするための一般形式は次のとおりです。

```
( command > outfile )>& errorfile
```

command 変数はオペレーティング・システムのコマンドです。 *outfile* 変数はプロセスが標準出力を書き込むファイルの名前です。 *>&* 記号はエラー出力をリダイレクトします。 *errorfile* 変数は、プロセスが標準エラーを書き込むファイルの名前です。 なお、このコマンド形式のカッコは必ず必要です。 詳細は 6.2.2.2 項を参照してください。

Bourne, Korn, および POSIX シェルの場合、標準出力と標準エラーの両方を同じファイルにリダイレクトするための一般形式は次のとおりです。

```
command1 > outfile 2>&1 errorfile
```

command 変数はオペレーティング・システムのコマンドです。 1> 記号は標準出力をリダイレクトします。 *outfile* 変数は、プロセスが標準出力を書き込むファイルの名前です。 2>&1 記号は、標準エラー (ファイル記述子 2) を標準出力 (>&1) に対応するファイル、*outfile* に書き込むようにシェルに指示するものです。

C シェルの場合、標準出力と標準エラーの両方を同じファイルにリダイレクトするための一般形式は次のとおりです。

```
command>& outfile
```

command 変数はオペレーティング・システムのコマンドです。 *outfile* 変数は、プロセスが標準出力を書き込むファイルの名前です。 *>&* 記号は、標準出力と標準エラーを *outfile* で指定される同じファイルに書き込むようにシェルに指示するものです。

6.3 複数プロセスの同時実行

オペレーティング・システムは複数の異なるプロセスを同時に実行することができます。 この能力により、Tru64 UNIX オペレーティング・システムはマルチタスキング・オペレーティング・システムとなっています。 これは、複数のユーザのプロセスを同時に実行できるという意味です。

これらの異なるプロセスは、1 人のユーザのものであっても複数のユーザのものであってもかまいません。 シェルのプロンプトに対し一度にコマンドを入力する必要はありません。 フォアグラウンド・プロセスとバックグラウンド・プロセスの両方を同時に実行することもできます。 以降の各項で、フォアグラウンド・プロセスとバックグラウンド・プロセスについて説明します。

6.3.1 フォアグラウンド・プロセスの実行

通常、コマンド行にコマンドを入力すると、結果が画面に表示されるのを待ちます。シェルのプロンプトに対して1つずつ入力されたコマンドは、フォアグラウンド・プロセスと呼ばれます。

ほとんどのコマンドは1, 2秒という短い時間で実行されます。しかし、コマンドのなかにはもっと長い実行時間を要するものもあります。実行時間の長いコマンドをフォアグラウンド・プロセスとして実行すると、そのコマンドの実行が終了するまで、他のコマンドは実行できません。このような場合、実行時間の長いコマンドはバックグラウンド・プロセスとして実行します。次の項では、バックグラウンド・プロセスについて説明します。

6.3.2 バックグラウンド・プロセスの実行

バックグラウンド・プロセスは、実行に長い時間がかかるコマンドでは、きわめて有効です。実行時間の長いコマンドをフォアグラウンド・プロセスとして入力すると画面を塞いでしまうため、このようなコマンドはバックグラウンド・プロセスとして実行します。こうすれば、フォアグラウンドで引き続き他の作業を実行することができます。

バックグラウンド・プロセスを実行するときは、コマンドの終わりにアンパサンド (&) を付けます。プロセスをバックグラウンドで起動した後、ワークステーションから他のコマンドを入力することによって、別の作業を実行できます。

バックグラウンド・プロセスを作成すると、次のようになります。

- プロセス識別番号 (PID) が表示される。
システムで現在実行中のすべてのプロセスが追跡できるように、オペレーティング・システムはPIDを作成して割り当てます。Korn および POSIX シェルまたはCシェルでは、ジョブ番号も割り当てられます。
- プロンプトが表示され、別のコマンドを入力できる。
- Cシェルでは、バックグラウンド・プロセスが完了するとメッセージが表示される。

バックグラウンド・プロセスを作成したら、そのPID番号を書き留めておくてください。PID番号は、プロセスをモニタしたり終了したりするときに役に立ちます。詳しくは6.4節を参照してください。

バックグラウンド・プロセスは、システムが行っている作業総量を増加させるため、システムの処理速度が遅くなる場合があります。問題になるかどうかは、システムの処理速度がどの程度遅くなるかによります。また、システムのユーザが行っている他の作業の性質や、実行中のバックグラウンド・プロセスの数にもよります。

ほとんどのプロセスは、バックグラウンドで実行するときでも、出力を標準出力します。リダイレクトされない限り、標準出力はワークステーションに表示されます。バックグラウンド・プロセスからの出力がシステムの他の作業に干渉することもあるため、通常はバックグラウンド・プロセスからの出力をファイルやプリンタにリダイレクトするほうが良いでしょう。こうすれば、都合の良いときにいつでも出力を見ることができます。出力のリダイレクションについての詳細は、この章の後で説明する例、および 6.2 節を参照してください。

この章の後に示す例は、実行に数秒以上かかるコマンドを使用しています。

```
$ find / -type f -print
```

このコマンドはシステムにあるすべてのファイルのパス名を表示します。ここでは、`find` コマンドについて詳しく知っておく必要はありません。プロセスでどのように作業するかを具体的に示すために使用しているだけです。`find` コマンドについて詳しく知りたい場合は、`find(1)` リファレンス・ページを参照してください。

次の例では、`find` コマンドをバックグラウンドで実行し (`&` を入力)、その出力を `dir.paths` という名前のファイルにリダイレクトします (`>` 演算子の使用)。

```
$ find / -type f -print > dir.paths &  
24  
$
```

バックグラウンド・プロセスが起動すると、システムはそれに PID (プロセス識別) 番号 (この例では 24) を割り当て、それを表示してから、次のコマンドの入力を促すプロンプトを表示します。実際に表示されるプロセス番号は上の例と下の例で表示されている番号とは異なります。

Korn または POSIX シェルまたは C シェルを使用している場合には、ジョブ番号も割り当てられます。C シェルでは、上の例は次のようになります。

```
% find / -type f -print > dir.paths &  
[1] 24  
%
```

ジョブ番号 [1] は PID 番号の左側に表示されます。

`ps` (process status) または `jobs` コマンド (C シェル, Korn および POSIX シェル) を使用して, プロセスの状態をチェックすることができます。また, `kill` コマンドでプロセスを終了することもできます。これらのコマンドについては, 6.4 節を参照してください。

C シェルでは, バックグラウンド・プロセスが完了すると, 次のようなメッセージが表示されます。

```
[1] 24 Done find / -type f -print > dir.paths
```

完了メッセージはジョブ番号と PID, 状態 `Done`, および実行したコマンドを表示します。

6.4 プロセスのモニタと終了

どのプロセスが実行中であるかを調べて, そのプロセスに関する情報を表示するには, `ps` (process status) コマンドを使用します。Korn および POSIX シェルと C シェルでは, `jobs` コマンドを使用してバックグラウンド・プロセスをモニタすることもできます。

プロセスを終了する前に停止する必要がある場合は, `kill` コマンドを使用してください。

以降の項では, プロセスをモニタおよび終了する方法について説明します。

6.4.1 プロセス状態のチェック

`ps` コマンドを使用すると, フォアグラウンドとバックグラウンド両方のすべてのアクティブ・プロセスの状態がモニタできます。Korn および POSIX シェルと C シェルでは, `jobs` コマンドを使用して, バックグラウンド・プロセスだけをモニタすることもできます。以降の項で, `ps` コマンドと `jobs` コマンドについて説明します。

6.4.1.1 `ps` コマンド

`ps` コマンドの形式は次のとおりです。

`ps`

例 6-1 では, `ps` コマンドは, ユーザのワークステーションに関連するすべてのプロセスの状態を表示しています。

例 6-1: ps コマンドからの出力

\$ ps				
PID	TTY	STAT	TIME	CMD
29670	p4	I	0:00.00	-sh (csh)
515	p5	S	0:00.00	-sh (csh)
28476	p5	R	0:00.00	ps
790	p6	I	0:00.00	-sh (csh)
\$				
PID	プロセス識別子。システムは、各プロセスの起動時にプロセス識別番号 (PID 番号) を割り当てる。プロセスと特定の PID 番号との間には一切関係がない。つまり、同じプロセスを何度かスタートすれば、そのつど異なる PID 番号を持つことになる。			
TTY	制御端末デバイス名。複数のワークステーションがあるシステムでは、このフィールドにプロセスを起動したワークステーションを表示する。ワークステーションが 1 つしかないシステムでは、このフィールドには <code>console</code> という指定が、1 つまたは複数の仮想端末の指定が表示される。			
STAT	シンボリック・プロセス状態。システムは、4 文字までの英数字で、プロセスの状態を表示する。詳細は ps(1) リファレンス・ページを参照。			
TIME	コンピュータがこのプロセスに充てた時間が、ps を入力した時からの分、秒、100 分の 1 秒単位で表示される。			
CMD	プロセスをスタートしたコマンド (あるいはプログラム) の名前。			

-p フラグと PID 番号を指定して ps コマンドを使用すると、特定プロセスの状態をチェックすることもできます。特定プロセスの状態をチェックするための一般形式は次のとおりです。

ps -p *PIDnumber*

ps コマンドはバックグラウンド・プロセスの状態也表示します。実行中のバックグラウンド・プロセスがある場合には、フォアグラウンド・プロセスとともに表示されます。次の例では、find バックグラウンド・プロセスをスタートした後、その状態をチェックする方法を示します。

```
$ find / -type f -print > dir.paths &
25
$ ps -p25
PID    TTY          TIME CMD
 25    console    0:40.00  find
$
```

バックグラウンド・プロセス状態は、そのプロセスが実行されている間は
何度でもチェックできます。次の例では、`ps` コマンドで前述の `find` プ
ロセスの状態を 5 回表示します。

```
$ ps -p25
PID   TTY          TIME COMMAND
 25   console    0:18:00 find
$ ps -p25
PID   TTY          TIME COMMAND
 25   console    0:29:00 find
$ ps -p25
PID   TTY          TIME COMMAND
 25   console    0:49:00 find
$ ps -p25
PID   TTY          TIME COMMAND
 25   console    0:58:00 find
$ ps -p25
PID   TTY          TIME COMMAND
 25   console    1:02:00 find
$ ps -p25
PID   TTY          TIME COMMAND
$
```

なお、6 回目の `ps` コマンドは状態情報を返しません、これは、最後の `ps`
コマンドが入力される前に `find` プロセスが終了したためです。

通常は、ここで説明している単純な `ps` コマンドでも、プロセスについて必
要な情報はすべてわかります。ただし、フラグを使用すれば、`ps` コマンド
が表示する情報の種類を制御することができます。最も便利な `ps` フラグ
の 1 つは `-e` です。 `-e` を指定すると `ps` は、ユーザの端末またはワークス
テーションに関連するプロセスだけでなく、すべてのプロセスに関する情報
を返します。 `ps` コマンドの全フラグについての説明は、`ps(1)` リファレン
ス・ページを参照してください。

6.4.1.2 `jobs` コマンド

Korn および POSIX シェルと C シェルは、バックグラウンド・プロセスが
作成されると、ジョブと PID の両方を表示します。 `jobs` コマンドは、
ジョブ番号に基づいて、すべてのバックグラウンド・プロセスの状態だけ
を報告します。

`jobs` コマンドの形式は次のとおりです。

`jobs`

`-l` フラグを追加すると、ジョブ番号と PID の両方を表示します。

次の例では、C シェルにおいて find プロセスを起動し、その後、jobs -l コマンドを使用して状態をチェックする方法を示します。

```
% find / -type f -print > dir.paths &
[2] 26
% jobs -l
[2] +26 Running    find / -type f -print > dir.paths &
%
```

状態メッセージはジョブ ([2]) と PID 番号 (26) の両方、状態 Running、および実行されたコマンドを表示します。

6.4.2 フォアグラウンド・プロセスの取り消し (Ctrl/C)

フォアグラウンド・プロセスを取り消す (実行中のコマンドを停止する) ときは、Ctrl/C を押します。コマンドは実行を停止し、システムはシェル・プロンプトを表示します。なお、フォアグラウンド・プロセスを取り消すことは、コマンド実行を停止することと同じです (第 1 章で説明)。

大部分の単純なオペレーティング・システム・コマンドは、プロセスの取り消し方を示す例としては適当ではありません。コマンドの実行があまりに速く、取り消す間もなく終わってしまうからです。ただし、次の find コマンドは実行に長い時間がかかるので、プロセスが 2、3 秒間実行してから、Ctrl/C を押すことによって取り消すことができます。

```
$ find / -type f -print
/usr/sbin/acct/acctcms
/usr/sbin/acct/acctcon1
/usr/sbin/acct/acctcon2
/usr/sbin/acct/acctdisk
/usr/sbin/acct/acctmerg
/usr/sbin/acct/accton
/usr/sbin/acct/acctprc1
/usr/sbin/acct/acctprc2
/usr/sbin/acct/acctwtmp
/usr/sbin/acct/chargefee
/usr/sbin/acct/ckpacct
/usr/sbin/acct/dodisk
Ctrl/C
$
```

システムはシェル・プロンプトを画面に返します。ここで、別のコマンドを入力することができます。

6.4.3 バックグラウンド・プロセスの取り消し (kill コマンド)

バックグラウンド・プロセスを起動してから、そのプロセスを中止したくなった場合は、`kill` コマンドでプロセスを取り消すことができます。ただし、バックグラウンド・プロセスを取り消すには、そのプロセスの PID 番号が必要です。

プロセスの PID 番号を忘れてしまった場合は、`ps` コマンドを使用して、すべてのプロセスの PID 番号を表示させることができます。C シェルまたは Korn または POSIX シェルを使用している場合には、`jobs` コマンドを使用して、バックグラウンド・プロセスのみを表示させるほうが効率的です。

特定のプロセスを終了させるための一般形式は次のとおりです。

kill *PIDnumber*

ログインしてからの全プロセスを終了したい場合は、`kill 0` コマンドを使用します。`kill 0` を使用する場合には、PID 番号は必要ありません。このコマンドはすべてのプロセスを削除するため、注意して使用してください。

次の例では、別の `find` プロセスを起動し、その状態をチェックしてから、終了する方法を示します。

```
$ find / -type f -print > dir.paths &
38
$ ps
PID    TT     STAT      TIME    COMMAND
520    p4      I         0:11:10  sh
38     p5      I         0:10:33  find
1216   p6      S         0:01:14  qdaemon
839    p7      R         0:03:55  ps
$ kill 38
$ ps
38 Terminated
PID    TT     STAT      TIME    COMMAND
520    p4      I         0:11:35  sh
1216   p6      S         0:01:11  qdaemon
839    p7      R         0:03:27  ps
$
```

コマンド `kill 38` がバックグラウンドの `find` プロセスを停止したので、2 番目の `ps` コマンドは、PID 番号 38 に関する状態情報を返しません。次のコマンドを入力するまで、システムは終了メッセージを表示しません。

なお、この例では、`kill 38` と `kill 0` は同じ効果を持っています。なぜなら、この端末またはワークステーションからは1つのプロセスしか起動していないからです。

C シェルでは、`kill` コマンドの形式は次のとおりです。

kill % *jobnumber*

次の例では、C シェルで別の `find` プロセスを起動し、`jobs` コマンドでその状態をチェックしてから、終了する方法を示します。

```
% find / -type f -print > dir.paths &
[3] 40
% jobs -l
[3] +40 Running  find / -type f -print > dir.paths &
% kill %3
% jobs -l
[3] +Terminated  find / -type f -print > dir.paths
%
```

6.4.4 フォアグラウンド・プロセスの中断と再開 (C シェルのみ)

システム資源を消費するプロセスを長時間実行している途中で、他のタスクを急いで実行する必要があるときは、フォアグラウンド・プロセスを一旦中断してから再開するとよいでしょう。

プロセスが完了するのを待たなくても、そのプロセスを一時停止 (中断) し、もっと大切なタスクを実行してから、そのプロセスを再開することができます。プロセスを中断する機能は、C シェルのユーザしか利用できません。

プロセスを中断するには `Ctrl/Z` を押します。ジョブ番号、状態 `Suspended`、および実行したコマンドをリストしたメッセージが表示されます。

プロセスを再開する用意ができたなら、フォアグラウンド・タスクとして、次のようにジョブ番号 `n` を入力してください。

```
% n
```

バックグラウンドでプロセスを再開するときは、次のようにジョブ番号 `n` を入力してください。

```
% n &
```

次の例では、`find` プロセスをスタートさせて中断し、その状態をチェックしてから再開し、その後、終了するものです。

```
% find / -type f -print > dir.paths &
[4] 41
```

```
% jobs -l
[4] +41 Running  find / -type f -print > dir.paths &
% Ctrl/Z
Suspended
% jobs -l
[4] +Stopped    find / -type f -print > dir.paths
% 4 &
[4] find / -type f -print > dir.paths &
% kill %4
[4] +Terminated  find / -type f -print > dir.paths
```

中断したプロセスは、`fg` コマンドで再開できます。あるいは、現在実行中のプロセスがあまりに時間がかかり、キーボードを塞ぐ場合は、`bg` コマンドを使用してそのプロセスをバックグラウンドに入れて、他のコマンドを入力することもできます。

次の例では、`find` プロセスをスタートさせて中断し、そのプロセスをバックグラウンドに入れて、ファイルをコピーした後、そのプロセスをフォアグラウンドで再開するものです。

```
% find / -type f -print > dir.paths
Ctrl/Z
Suspended
% bg
[5]      find / -type f -print > dir.paths &
% cp salary1 salary2
% fg
find / -type f -print > dir.paths
%
```

6.5 ユーザとユーザのプロセスに関する情報の表示

Tru64 UNIX オペレーティング・システムには、現在システムを使用しているユーザと、そのユーザが何を実行しているか知らせる、次のようなコマンドがあります。

<code>who</code>	現在ログインしているユーザを表示する。
<code>w</code>	現在ログインしているユーザと、ユーザがワークステーションで現在何を実行しているかを表示する。
<code>ps au</code>	現在ログインしているユーザと、ユーザが実行しているプロセスに関する情報を表示する。

who コマンドを使用すると、システムにログインしているユーザを確認できます。たとえば、メッセージを送りたいときや、現在、その人の手を借りられるかどうか知りたいときは、特に有効でしょう。

例 6-2 では、現在ログインしているユーザ全員が表示されます。

例 6-2: who コマンドからの出力

```
$ who
juan    tty01    Jan 15    08:33
chang   tty05    Jan 15    08:45
larry   tty07    Jan 15    08:55
tony    tty09    Jan 15    07:53
lucy    pts/2    Jan 15    11:24    (boston)
$
```

who コマンドは、システムにログインしている各ユーザのユーザ名、使用しているワークステーション、およびその人がログインした時刻を表示します。また、ユーザがリモート・システムからログインしている場合には、そのシステム名も表示されます。たとえば、lucy は Jan 15 の 11:24 に、システム boston からリモートでログインしています。

who コマンドの実際の表示形式は、現在のロケールにより異なります。ロケールについての詳細は、付録 C を参照してください。

who -u コマンドは、who コマンドの全情報のほかに、各ユーザの PID 番号、ワークステーションでの作業を終了した時点からの時間と分数也表示します。1 分未満の作業はドット (.) で示されます。

例 6-3 では、現在ログインしているユーザ全員が表示されます。

例 6-3: who -u コマンドからの出力

```
$ who -u
juan    tty01    Jan 15    08:33    01:02    50
chang   tty05    Jan 15    08:45    .        52
larry   tty07    Jan 15    08:55    .        58
tony    tty09    Jan 15    07:53    01:20    60
lucy    pts/5    Jan 15    11:24    .        65    (boston)
$
```

例 6-3 では、juan と tony が 1 時間以上作業をしていないのに対し、chang、larry、および lucy は作業をやめてから 1 分未満です。

これで、だれがシステムで作業しているかを調べる方法がわかりました。次に、各ユーザが現在どんなコマンドを実行しているかを知りたい場合があるかもしれません。w コマンドは、各ユーザのワークステーションで現在実行中のコマンドを表示します。

例 6-4 では、全ユーザ (User) と現在のコマンド (what) が表示されます。

例 6-4: w コマンドからの出力

```
$ w
11:02 up 3 days, 2:40, 5 users, load average: 0.32, 0.20, 0.00
User      tty      login@   idle    JCPU    PCPU    what
juan      tty01     8:33am   12       54      14    -csh
chang     tty05     8:45am           6:20    26    mail
larry     tty07     8:55           1:58     8    -csh
tony      tty09     7:53     3:10     22     4    mail
lucy      tty02     11:24    1:40     18     4    -csh
$
```

- tty: ユーザのワークステーション
- login@: ユーザのログイン時刻
- idle: ユーザがコマンドを入力してから時間
- JCPU: 現在のログイン・セッション中に使用された全 CPU 時間
- PCPU: 現在実行中のコマンドによって使用された CPU 時間
- what: ユーザが現在実行しているコマンド

場合によっては、現在のプロセス (フォアグラウンドとバックグラウンドの両方) と、それを実行しているユーザの詳細なリストが必要になることもあります。このようなリストを表示させるには、ps au コマンドを使用します。

例 6-5 では、5 人のユーザとそのアクティブ・プロセスが表示されます。

例 6-5: ps au コマンドからの出力

```
$ ps au
USER      PID %CPU %MEM VSZ  RSS TTY  S  STARTED      TIME COMMAND
juan     26300 16.5  0.8 441  327 p3   R           0:02:01 ps au
chang    25821  7.0  0.2 149   64 p4   R           0:12:23 mail -n
larry    25121  6.1  0.2 107   83 p22  R          26:25:07 lpstat
tony     11240  4.5  0.6 741  225 p1   R           1:57:46 vi
lucy     26287  0.5  0.1  61   28 p1   S           0:00:00 more
```

例 6-5: `ps au` コマンドからの出力 (続き)

\$

一般ユーザにとって最も大切なフィールドは `USER` , `PID` , `TIME` , および `COMMAND` フィールドです。その他のフィールドに関する情報は , `ps(1)` リファレンス・ページを参照してください。



シェルの概要

この章では、オペレーティング・システムのシェルについて紹介します。この章では次のような項目について説明しています。

- C シェルおよび Bourne , Korn , および POSIX シェルの目的と一般機能の理解 (7.1 節)
- シェルの変更 (7.3 節)
- すべてのシェルに共通なコマンド入力エイドの使用 (7.4 節)
- シェル環境 , ログイン・スクリプト , 環境変数 , およびシェル変数の理解 (7.5 節)
- 環境変数とシェル変数の設定とクリア (7.7 節)
- シェルがシステム上でコマンドを見つける方法の理解 (7.8 節)
- ログアウト・スクリプトの作成 (7.9 節)
- 基本的なシェル・プロシージャの作成と実行 (7.10.1 項)

この章では、オペレーティング・システムのすべてのシェルに共通な機能を取り上げるとともに、シェル間の相違についてもある程度説明します。C シェルおよび Bourne , Korn , または POSIX シェルの特殊機能についての詳細は、第 8 章を参照してください。

7.1 シェルの目的

オペレーティング・システムへのユーザ・インタフェースはシェルと呼ばれます。シェルは、ユーザが入力するコマンドを解釈し、要求されたプログラムを実行して、その結果を画面に表示します。

Tru64 UNIX オペレーティング・システムには次のようなシェルがあります。

- Bourne シェル (システムの省略時のシェル)
- C シェル
- Korn シェル

- POSIX シェル

システムに設定されているセキュリティ制限や、Korn シェルのライセンス制限に合致すれば、いずれのシェルにもアクセスできます。すべてのシェルは同じような基本的な機能を実行します。ユーザが実行したコマンドは、シェルの介してシステムに伝えられます。

コマンドを解釈するだけでなく、シェルはプログラミング言語としても使用できます。ユーザはコマンドを含むシェル・プロシージャを作成することができ、このシェル・プロシージャは、プログラムと同じようにコマンド行上でシェル・プロンプトに対して実行できます。

シェル・プロシージャを実行するとき、ユーザの現在のシェルはサブシェルを作成 (spawn) します。サブシェルとは、現在のシェルがプログラムを実行するために新しく作成するシェルのことです。このため、シェル・プロシージャが実行するコマンド (たとえば `cd`) は、シェル・プロシージャを起動したシェルには影響を及ぼしません。

シェル・プロシージャを使用すると、長すぎるコマンドや大きく複雑なコマンド・シーケンス、および決まりきった反復的な作業を行うのが簡単になります。

シェル・プログラミングについての詳細は、7.10 節を参照してください。

7.2 C シェル, Bourne シェル, Korn シェル, および POSIX シェルの機能の要約

オペレーティング・システムには、コマンド実行とプログラミングを可能にするという 2 つの機能を兼ね備えた次のようなシェルがあります。

- Bourne シェル (`sh`)

これはプログラミングで手軽に使用できる単純なシェルです。通常ドル記号 (`$`) プロンプトが表示されます。このシェルは、C シェルや Korn および POSIX シェルのような対話機能や複雑なプログラミング構造 (配列や整数算術演算) を備えていません。

Bourne シェルには制限付きシェル (`Rsh`) もあります。詳しくは、7.2.2 項を参照してください。

- C シェル (`cs`)

このシェルは対話用に設計されています。通常パーセント記号 (%) がシステム・プロンプトとして表示されます。C シェルは、対話式にコマンドを入力する次のような機能を備えています。

- コマンド・ヒストリ・バッファ
- コマンド別名
- ファイル名補完
- コマンド行編集

これらの機能についての詳細は、7.2.1 項を参照してください。

- Korn シェル (ksh)

このシェルは、C シェルの使いやすさと Bourne シェルのプログラミングしやすさを組み合わせたものです。システム・プロンプトは通常ドル記号 (\$) です。Korn シェルは次のような機能を備えています。

- C シェルの対話機能
- Bourne シェルの単純なプログラミング構文
- コマンド行編集機能
- 最速実行時間
- Bourne シェルとの上位互換性

Bourne シェルプログラムのほとんどは Korn シェルで実行できます。

これらの機能について詳細は、7.2.1 項を参照してください。

- POSIX シェル (sh)

このシェルは、IEEE POSIX 規格に準拠しています。Korn シェルによく似ているので、本書では、Korn シェルと POSIX シェルをまとめて説明します。この規格についての情報は、sh(1p) および standards(5) リファレンス・ページを参照してください。

POSIX シェルは Korn シェルのもう 1 つの指名字です。

7.2.1 C シェルおよび Korn シェルまたは POSIX シェルの機能に関する詳細情報

C シェルおよび Korn または POSIX シェルは、次のような対話機能を備えています。

- コマンド・ヒストリ

コマンド・ヒストリ・バッファは入力されるコマンドを格納して、ユーザがいつでもそれを表示できるようにします。このため、以前に入力したコマンド、または以前に入力したコマンドの一部を選択して、それを再度実行することができます。この機能を使用すると、長いコマンドを改めて入力し直さなくても、再度使用できるため、時間が節約できます。C シェルでこの機能を利用するには、`.cshrc` ファイルにいくつかの設定を行わなければなりません、Korn および POSIX シェルでは、この機能は自動的に提供されます。

- コマンド別名

コマンド別名機能を使用すると、長いコマンド行を短縮したり、コマンドの名前を変更したりすることができます。これは、頻繁に使用する長いコマンド行の別名を定義することによって行います。たとえば、ディレクトリ `/usr/chang/reports/status` に何回も移動する必要があるとします。その場合は、たとえば別名 `status` を作成して、コマンド行に `status` と入力するだけで、いつでもそのディレクトリに移動できるようにします。さらに、別名機能を使用すれば、オペレーティング・システムのコマンドに対して、もっと説明的な名前を作成することもできます。たとえば、`mv` コマンドに対して `rename` という別名を定義することもできます。

- ファイル名補完

C シェルでは、ファイル名補完機能とは、ファイル名の一部を入力すると、シェルがユーザに代わってそれを補完してくれる機能で、入力の手間を省いてくれるものです。Escape キーを押すと、シェルがユーザに代わってファイル名を補完します。C シェルのファイル名補完機能についての詳細は、8.2.4 項を参照してください。

Korn または POSIX シェルでは、シェルに要求して、入力した部分名に一致するファイル名のリストを表示させることができます。その後、表示されたファイルの中からファイルを選択することができます。Korn または POSIX シェルのファイル名補完機能についての詳細は、8.4.5 項を参照してください。

Korn および POSIX シェルはインライン編集機能を備えています。この機能により、以前に入力したコマンドを検索して、それを編集することができます。

す。この機能を使用するには、`vi` や `emacs` のようなテキスト・エディタの使用方法を知っておく必要があります。

これらのシェル機能の詳細については、第 8 章を参照してください。

7.2.2 制限付き Bourne シェル

オペレーティング・システムでは、指定ユーザに対し、制限付き Bourne シェル (`Rsh`) で限定された機能を提供することによって、システム・セキュリティを拡張します。これらの指定ユーザがシステムにログインするときには、制限付き Bourne シェルだけにアクセスする権利が与えられます。だれが制限付き Bourne シェルにアクセスできるかは、システム管理者が決定します。

制限付きシェルは、より制御されたシェル環境を要求するインストレーションにとって有効です。このため、システム管理者は、特権と機能が制限されたユーザ環境を作成することができます。たとえば、システムのゲストであるユーザは全員、ユーザ名 `guest` の下でしかアクセスできないとします。システムにログインすると、ユーザ `guest` は制限付きシェルを割り当てられます。

`Rsh` の動作は、次の点を除いて `sh` の動作と同じです。

- ディレクトリの変更
`cd` コマンドは起動できません。
- `/` を含むパス名またはコマンド名の指定
- `PATH` 変数 または `SHELL` 変数の値の設定
これらの変数に関する詳細は、7.5.2 項を参照してください。
- `>` および `>>` を使用した出力のリダイレクション

`Rsh` に関する詳細は、`sh(1b)` リファレンス・ページを参照してください。システム管理者が制限付きシェルを作成する方法については、システム管理者に問い合わせてください。

7.3 シェルの変更

ユーザがシステムにログインすると、システム管理者が指定したシェルに自動的に入ります。ただし、システム上に設定されているセキュリティ機能によっては、次のことを行うコマンドが入力できます。

- 実行しているシェルの確認

- シェルの一時的変更
- シェルの永久的変更

以降の各項で、これらの操作について説明します。

7.3.1 実行しているシェルの確認

現在どのシェルを実行しているかを確認するときは、次のコマンドを入力します。

```
echo $SHELL
```

実行しているシェルのファイル名が表示されます。

Bourne シェル (sh) を実行している場合は、次のような結果が出力されます。

```
$ echo $SHELL
/usr/bin/sh
$
```

表 7-1 は、表示される各シェルのファイル名、および root 以外のユーザに対する省略時のシステム・プロンプトを示したものです。システムにより、システム・プロンプトは異なる場合があります。

表 7-1: シェルのファイル名と省略時のプロンプト

シェル	シェル名	省略時のプロンプト
Bourne	sh	\$
制限付き Bourne	Rsh	\$
C	csch	%
Korn	ksh	\$
POSIX	sh	\$

7.3.2 シェルの一時的変更

システムのセキュリティ機能で許されている場合は、他のシェルを使用することができます。

シェルを一時的に変更するときは、次のコマンドを入力します。

shellname

`shellname` は使用するシェルのファイル名です。コマンド行に入力する有効なシェルのファイル名については、表 7-1 を参照してください。シェルが呼び出されると、正しいシェル・プロンプトが表示されます。

新しいシェルでの作業が終了したら、`exit` を入力するか `Ctrl/D` を押すことによって、省略時のシェルに戻ることができます。

たとえば、Korn シェルが省略時のシェルの場合に、C シェルに移り、その後 Korn シェルに戻るには、次のコマンドを入力します。

```
$ /usr/bin/csh
% exit
$
```

注意

制限付き Bourne シェルを使用している場合は、別のシェルに移ることはできません。

7.3.3 シェルの永久的変更

システムのセキュリティ機能で許されている場合は、省略時のシェルの永久的に変更することができます。現在のシェルがC シェルである場合、省略時のシェルを変更するには `chsh` コマンドを使用します。C シェルを使用していない場合には、システム管理者に問い合わせ、省略時のシェルを変更してください。

C シェルでは、省略時のシェルを変更するには、次のコマンドを入力します。

```
% chsh
Changing login shell for user.
Old shell: /usr/bin/csh
New shell:
```

新しいシェルの名前を入力してください。コマンド行に入力する有効なシェル名については、表 7-1 を参照してください。

なお、`chsh` コマンドを入力した後、その変更を有効にするために、ログアウトしてから改めてログインしなければなりません。

7.4 コマンド入力支援機能

オペレーティング・システムのすべてのシェルには、ユーザの作業を支援する次のような機能があります。

- 複数コマンドおよびコマンド・リストを入力する機能
- パイプとフィルタ
- コマンドをグループ化する機能
- 引用

以降の各項で、これらの機能について説明します。

7.4.1 複数コマンドとコマンド・リストの使用

通常、シェルはコマンド行の最初のワードをコマンド名として解釈し、他のワードはすべてそのコマンドの引数として解釈します。シェルは普通、各コマンド行を単一のコマンドと考えます。ただし、表 7-2 に示す演算子を使用すれば、単一のコマンド行で複数のコマンドを実行することができます。

表 7-2: 複数コマンド演算子

演算子	動作	例
;	コマンドを順次実行させる。	<code>cmd1 ; cmd2</code>
&&	現在のコマンドが成功した場合に、次のコマンドを実行する。	<code>cmd1 && cmd2</code>
	現在のコマンドが失敗した場合に、次のコマンドを実行する。	<code>cmd1 cmd2</code>
	パイプラインを作成する。	<code>cmd1 cmd2</code>

以降の項で、コマンドの順次実行 (;)、条件付きのコマンド実行 (|| と &&)、およびパイプラインの使用 (|) について説明します。

7.4.1.1 セミコロン (;) を使用したコマンドの順次実行

コマンドをセミコロン (;) で区切ると、1 行に複数のコマンドを入力することができます。

次の例では、シェルは `ls` を実行して、それが終了するのを待ちます。`ls` が終了すると、シェルは `who` を実行し、以下同様に最後のコマンドまで実行します。


```
$ ls ; who ; date ; pwd
change file3 newfile
amy console/1 Jun 4 14:41
Tue Jun 4 14:42:51 CDT 1999
/u/amy
$
```

なお、コマンドのどれかが失敗しても、その他のコマンドは順次実行されます。

コマンド行を読みやすくするために、ブランクまたはタブを使用してコマンドとセミコロン (;) を分離します。この場合には、シェルはこのようなブランクやタブを無視します。

7.4.1.2 条件付きのコマンド実行

コマンドを 2 つのアンパサンド (&&) または縦線 (||) 演算子で接続すると、シェルは最初のコマンドを実行し、それから次の条件で残りのコマンドを実行します。

- &&

シェルは、現在のコマンドが完了した場合に限って、次のコマンドを実行します。コマンドがゼロの値を返すと、正常に完了したことを意味します。

- ||

シェルは、現在のコマンドが完了しない場合に限って、次のコマンドを実行します。

アンパサンド (&&) 演算子の構文は次のとおりです。

```
cmd1 && cmd2 && cmd3 && cmd4 && cmd5
```

cmd1 が成功すれば、シェルは *cmd2* を実行します。*cmd2* が成功すれば、*cmd3* を実行し、以下同様に、コマンドが失敗するか最後のコマンドが終了するまで続きます。コマンドのどれかが失敗すれば、シェルはコマンド行の実行を停止します。

2 本の縦線 (||) 演算子の構文は次のとおりです。

```
cmd1 || cmd2
```

cmd1 が失敗すれば、シェルは *cmd2* を実行します。*cmd1* が成功すれば、シェルはコマンド行の実行を停止します。

たとえば、コマンド `mysort` はソート・プログラムであり、ソート処理中に一時ファイル (`mysort.tmp`) を作成するとします。ソート・プログラムが正常に終了すれば、一時ファイルを削除して、クリーンアップ処理を行います。一方、プログラムが失敗した場合は、一時ファイルが削除されないことがあります。確実に `mysort.tmp` の削除を行うには、次のコマンド行を入力してください。

```
$ mysort || rm mysort.tmp
$
```

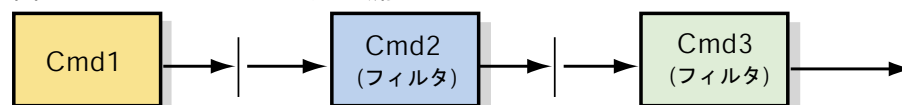
最初のコマンドが失敗した場合に限り、2 番目のコマンドが実行されます。

7.4.2 パイプとフィルタの使用

パイプとは、2 つの関連するコマンド間の一方向接続です。あるコマンドがその出力をパイプに書き込み、もう一方のプロセスがその入力をパイプから読み取ります。2 つ以上のコマンドがパイプ (`|`) 演算子によって接続されると、パイプラインを形成します。

図 7-1 は、パイプラインを通る入力と出力の流れを示しています。最初のコマンド (`cmd1`) の出力は 2 番目のコマンド (`cmd2`) の入力となり、2 番目のコマンドの出力は 3 番目のコマンド (`cmd3`) の入力になります。

図 7-1: パイプラインを通る流れ



ZK-0537U-AIJ

フィルタとは、標準入力を読み取り、その入力を変換し、変換済みの入力を標準出力に書き込むコマンドです。フィルタの典型的な使用例として、パイプライン中の中間コマンドとしてパイプ (`|`) 演算子で接続して使用する方法があります。たとえば、`ls` コマンドを使用して、現在のディレクトリから階層の最下位までの全ディレクトリの内容を再帰的にリストし、その結果を表示するには、次のコマンドを入力します。

```
$ ls -R | pg
```

この例では、`pg` コマンドがフィルタです。これは、`pg` コマンドが `ls -R` コマンドからの出力を変換して、一度に 1 画面ずつ表示するからです。

フィルタでないコマンドのなかにも、そのコマンドにフィルタと同じような機能を行わせるフラグを持つものがあります。たとえば、`diff` (ファイル比較) コマンドは、通常 2 つのファイルを比較し、その違いを標準出力に書き込みます。`diff` の普通の形式は次のとおりです。

diff *file1 file2*

しかし、どちらかのファイル名の代わりにダッシュ (`-`) フラグを使用すると、`diff` は標準入力を読み取って、それを指定されたファイルと比較します。

次のパイプラインでは、`ls` は現在のディレクトリの内容を標準出力に書き込みます。`diff` コマンドは `ls` の出力を `dirfile` というファイルの内容と比較して、相違箇所を一度に 1 ページずつ (`pg` コマンドで) 標準出力に書き込みます。

```
$ ls | diff - dirfile | pg
```

次の例では、別の種類のフィルタ・プログラム (`grep`) を使用しています。

```
$ ls -l | grep r-x | wc -l
    12
$
```

この例では、次のことが行われます。

- `ls -l` コマンドは現在のディレクトリの内容を詳細な形式でリストする。
- `ls -l` の出力が `grep r-x` の標準入力となる。
`grep r-x` はフィルタであり、標準入力中のファイルを検索して、許可が `r-x` のパターンを探し、そのパターンを含むすべての行を標準出力に書き込みます。
- `grep r-x` の標準出力は `wc -l` の標準入力となる。
`wc -l` は、標準入力中にある `grep` の基準に合致するファイル数を表示します。

パイプラインを使用しないで同じ結果を得るには、次のようにします。

1. `ls -l /user` の出力をファイルに送る。
たとえば、次のように入力します。

```
$ ls -l > file1
```
2. `file1` を `grep r-x` の入力として使用し、`grep` の出力を別のファイルにリダイレクトする。

たとえば、次のように入力します。

```
$ grep r-x file1 > file2
```

3. `grep` の出力ファイルを `wc -l` の入力として使用する。

たとえば、次のように入力します。

```
$ wc -l file2
```

上記の手順から明らかなように、同じオペレーションでも、パイプラインを使用すると、はるかに容易に行うことができます。

パイプライン中の各コマンドは、別個のプロセスとして実行されます。パイプラインは一方向(左から右)にのみ作用し、また、パイプライン中の全プロセスは同時に実行できます。読み取るべき入力がないとき、または次のプロセスへのパイプが一杯になっているとき、プロセスは一時停止します。

7.4.3 コマンドのグループ化

シェルには複数のコマンドをグループ化する方法が2通りあります。表 7-3 を参照してください。

表 7-3: コマンドをグループ化する記号

記号	動作
<code>(commands)</code>	シェルはサブシェルを作成して、グループ化された <code>commands</code> を別個のプロセスとして実行する。
<code>{commands}</code>	シェルはグループ化された <code>commands</code> をユニットとして実行する。中カッコ (<code>{}</code>) は Bourne, Korn, および POSIX シェルだけで使用できる。

以降の項で、表 7-3 に示すコマンドをグループ化する記号について詳細に説明します。

7.4.3.1 カッコ () の使用

次のコマンドのグループ化では、シェルはカッコで囲まれたコマンドを別個のプロセスとして実行します。

```
$ (cd reports;ls);ls
```

シェルはサブシェル(別個のシェル・プログラム)を作成して、ディレクトリ `reports` に移動し、そのディレクトリ内のファイルをリストします。サブシェル・プロセスが完了すると、シェルは現在のディレクトリ内のファイルをリストします (`ls`)。

このコマンドを () を指定しないで入力した場合には、元のシェルがディレクトリ `reports` に移動して、そのディレクトリ内のファイルをリストした後、そのディレクトリ内のファイルをもう一度リストします。 `cd reports;ls` コマンドのためのサブシェルも、別個のプロセスも作成されません。

シェルは、カッコ () がコマンド行のどこにあっても認識します。カッコを本来の意味で (すなわち、コマンドのグループ化を行わずに) 使用するときは、左カッコまたは右カッコの直前にバックスラッシュ (\) を付加して、カッコを引用してください。たとえば、 \ (のように指定します。

シェルにおける引用の詳細については、7.4.4 項を参照してください。

7.4.3.2 中カッコ { } の使用

中カッコ { } の使用は Bourne, Korn, および POSIX シェルのみで有効です。

複数のコマンドを中カッコ { } で囲んでグループ化すると、シェルはサブシェルを作成せずにそれらのコマンドを実行します。次の例では、シェルは `date` コマンドを実行し、その出力を `today.grp` ファイルに書き込んだ後、`who` コマンドを実行して、その出力を `today.grp` に書き込みます。

```
$ { date; who ;} > today.grp
$
```

コマンドが中カッコで囲まれていない場合、シェルは `date` コマンドの出力をディスプレイに表示し、`who` コマンドの出力をファイルに書き込みます。

シェルはパイプラインとコマンド・リストの中カッコ { } を認識しますが、左中カッコがコマンド行の先頭文字である場合に限りません。

7.4.4 引用

シェルにとって特殊な意味を持つ左山カッコ (<) , 右山カッコ (>) , パイプ (|) , アンパサンド (&) , アスタリスク (*) , 疑問符 (?) などの文字を予約文字といいます。オペレーティング・システムの各シェルの予約文字については、第 8 章を参照してください。

予約文字を本来の意味で (すなわち、特殊な意味なしに) 使用するには、表 7-4 に示すように、3 種類のシェル引用規約のいずれかを使用して、その文字を引用します。

表 7-4: シェル引用規約

引用規約	動作
\	バックスラッシュ --- 単一の文字を引用する。
' '	一重引用符 --- 文字列を引用する (一重引用符自体を除く)。
" "	二重引用符 --- 文字列を引用する (\$, ` , および \ を除く)。

以降の項では、表 7-4 に示す引用規約について詳細に説明します。

7.4.4.1 バックスラッシュの使用 (\)

単一の文字を引用するときは、次のようにその文字の直前にバックスラッシュ (\) をタイプします。

```
$ echo \?  
?  
$
```

このコマンドは単一の疑問符 (?) を表示します。

7.4.4.2 一重引用符の使用 (')

文字列を一重引用符で囲むと、シェルはその文字列内のすべての文字 (' 自体を除く) を本来の意味に解釈します。一重引用符は次の場合に役立ちます。

- ドル記号 (\$), 抑音符 (`), バックスラッシュ (\) などの予約文字をシェルに解釈させたくないとき
- シェルに変数名を解釈させたくないとき

次の例は、変数名をシェルに解釈させずに表示したい場合に、一重引用符を使用する方法を示しています。

```
$ echo 'The value of $USER is' $USER  
The value of $USER is amy  
$
```

echo コマンドは、変数名 `$USER` が一重引用符内にある場合には、その変数名をそのまま表示しますが、`$USER` が一重引用符の外にある場合には、その値を解釈します。

変数代入については、7.7.1 項を参照してください。

7.4.4.3 二重引用符の使用 (" ")

二重引用符 (" ") は特殊な引用形式を提供します。二重引用符内では、予約文字のドル記号 (\$), 抑音符 (), およびバックスラッシュ (\) は、それぞれの特な意味を持ちます。シェルは、二重引用符内のその他のすべての文字は本来の意味に解釈されます。二重引用符は変数代入において最も頻繁に使用されます。

次の例は、シェル変数の値を含むメッセージにおいて、山カッコ (通常は予約文字) をそのまま表示したい場合に、二重引用符を使用する方法を示しています。

```
echo "<<Current shell is $SHELL>>"
<<Current shell is /usr/bin/csh>>
$
```

変数代入については、7.7.1 項を参照してください。

7.5 シェル環境

ログインするといつも、省略時のシェルがユーザ固有の作業環境を定義して維持します。ユーザの環境は、ユーザ識別やシステムのどこで作業しているか、またどんなコマンドを実行しているかなどの特性を定義します。

作業環境は環境変数とシェル変数の両方で定義されます。省略時のログイン・シェルは環境変数を使用し、それらをユーザが作成するすべてのプロセスやサブシェルに渡します。シェル変数は現在のシェルだけで有効で、サブシェルには渡されません。

以降の項で、シェル環境、シェル環境の構成方法、およびシェル環境の調整方法について説明します。

7.5.1 ログイン・プログラム

ログインすると必ず、login プログラムが実行されます。このプログラムは /etc/passwd ファイルに格納されているデータを使用して、実際にログイン・セッションを開始します。/etc/passwd ファイルには各システム・ユーザについて記述した 1 行の情報が含まれています。この行には、ユーザ名、パスワード (暗号化された形式)、ホーム・ディレクトリ、および省略時のシェルが記述されています。/etc/passwd ファイルについて詳細は、第 5 章を参照してください。

login プログラムは、login: プロンプトに対してユーザ名を入力した後、実行されます。このプログラムにより次の機能が行われます。

- Password: プロンプトを表示する (パスワードがある場合)。
- ユーザが入力したユーザ名とパスワードを、/etc/passwd ファイルに格納されているデータと照合して確認する。
- シェル環境に省略時の値を割り当てる。
- シェル・プロセスの実行を開始する。
- システム・ログイン・スクリプトおよびユーザ作成のログイン・スクリプトを実行する。

詳細は 7.6 節を参照してください。

7.5.2 環境変数

シェル環境は、ユーザ固有の作業環境を定義して維持します。ユーザの作業環境の特性の大部分は環境変数によって定義されます。

環境変数は名前と値から構成されます。たとえば、ユーザのログイン・ディレクトリ用の環境変数は *HOME* という名前であり、その値はログインするときに自動的に定義されます。

環境変数のなかには login プログラムによって設定されるものもありますが、ユーザのシェルに合わせてログイン・スクリプトで定義できるものもあります。たとえば、C シェルを使用している場合、環境変数は .cshrc ログイン・スクリプトで設定するのが普通です。ログイン・スクリプトについての詳細は、7.6 節を参照してください。

表 7-5 に、オペレーティング・システムのすべてのシェルで使用できる主な環境変数の一覧を示します。これらの変数の値の大部分はログイン・プロセスで設定され、その後、セッション中に作成される各プロセスに渡されます。

表 7-5: 主なシェル環境変数

環境変数	説明
HOME	ログイン・ディレクトリ, つまりログインが完了すると現在のディレクトリになるディレクトリの名前を指定する。cd コマンドは省略値として HOME の値を使用する。この値は login プログラムで設定され, 個々のユーザが変更することはできない。
LOGNAME	ユーザのログイン名を指定する。
MAIL	メール・システムが新たなメールの到着を検出するために使用するファイルのパス名を指定する。login プログラムがユーザ名に基づいてこの変数を設定する。
PATH	システムがコマンドの探索, 発見, 実行に使用するディレクトリとディレクトリ順を指定する。この変数はユーザのログイン・スクリプトで設定される。
SHELL	省略時のシェルを指定する。この変数は, login プログラムにより, /etc/passwd ファイルのエントリに指定されたシェルを使用して設定される。
TERM	使用している端末のタイプを指定する。この変数は通常, ユーザのログイン・スクリプトで設定される。
TZ	現在の時間帯とグリニッジ平均時との差を指定する。この変数はシステム・ログイン・スクリプトで設定される。
LANG	システムのロケールを指定する。ロケールは言語, テリトリ, および文字コード・セットの3つの部分から構成される。省略値は C ロケールであり, これは言語が英語, テリトリが米国, コード・セットが ASCII である。LANG はログイン・スクリプトで設定できる。
LC_COLLATE	名前をソートするときとパターンに文字範囲が指定されたときに使用する照合順序を指定する。省略値は ASCII 照合順序。LC_COLLATE はログイン・スクリプトで設定できる。
LC_CTYPE	ctype関数で使用される現在のロケールの文字分類規則を指定する。省略値は ASCII 文字の分類。LC_TYPE 変数はログイン・スクリプトで設定できる。
LC_MESSAGES	yes/no プロンプトに使用する言語を指定する。省略値はアメリカ英語であるが, システムによって別の言語を指定することもできる。
LC_MONETARY	システムの通貨表記形式を指定する。省略値はアメリカの通貨表記形式。LC_MONETARY 変数はログイン・スクリプトで設定できる。

表 7-5: 主なシェル環境変数 (続き)

環境変数	説明
LC_NUMERIC	システムの数値形式を指定する。省略値はアメリカの数値形式。LC_NUMERIC 変数はログイン・スクリプトで設定できる。
LC_TIME	システムの日時形式を指定する。省略値はアメリカの日時形式。LC_TIME 変数はログイン・スクリプトで設定できる。

これらの環境変数の多くは、ログイン・プロセス中に、該当するログイン・スクリプトによって設定できます (7.6 節を参照)。ただし、これらの値は再設定することができます。また、省略値が定められていない変数についても設定することができます。詳しくは 7.7.1 項を参照してください。

LANG, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, LC_TIME の各変数についての詳細は、付録 C を参照してください。この付録では、言語および各国の習慣をサポートする他のシステム機能のコンテキストにおける変数について説明しています。

ユーザ自身の環境変数を作成することもできます。たとえば、システムによってはユーザが複数のメール・プログラムを利用できるものもあります。たとえば、システムで mail と mh のメール・プログラムが利用でき、それぞれに固有のパス名があるとします。その場合、各メール・プログラムのパス名について変数を定義することができます。

オペレーティング・システムの各シェルに特有な環境変数についての詳細は、第 8 章を参照してください。オペレーティング・システムでサポートするシェル環境変数の一覧については、sh(1b), sh(1p), csh(1), および ksh(1) のリファレンス・ページを参照してください。

7.5.3 シェル変数

シェル変数は現在のシェルに対してのみ有効であり、サブシェルには渡されません。したがって、シェル変数はそれらが定義されているシェルでしか使用できません。言い換えれば、シェル変数はローカル変数と考えることができます。

シェル変数は、環境変数にすることにより、現在のシェルの外部からアクセスできるようになります。環境変数についての詳細は、7.7.1 項を参照してください。

また、ユーザ自身のシェル変数を作成することもできます。たとえば、メール・プログラムのなかには、PAGER 変数を使用して、メールを表示するプログラムを定義するものがあります。メール・プログラムが `mailx` だとすると、PAGER 変数を定義して、メールの表示に `more` プログラムを使用するようにできます。

シェル変数の設定方法については、7.7.1 項を参照してください。

7.6 ログイン・スクリプトとユーザ環境

ログイン・スクリプトとは、ユーザ環境をセットアップするための一連のコマンドが記述されているファイルです。ログイン・スクリプトには次の 2 種類があります。

- 特定のシェルのすべてのユーザのためのシステム・ログイン・スクリプト

このスクリプトはすべてのユーザの省略時の環境を作成するものであり、システム管理者によって保守されます。Bourne、Korn、および POSIX シェルでは、`/etc/profile` というシステム・ログイン・スクリプトを使用します。C シェルでは `/etc/csh.login` というスクリプトを使用します。システム・ログイン・スクリプトのパス名については、表 7-6 を参照してください。

ログインすると、このファイルのコマンドがまず実行されます。

- 省略時のログイン・ディレクトリ内のローカル・ログイン・スクリプト

このスクリプトを使用すると、ユーザ環境を調整でき、該当するファイルはユーザが保守します。たとえば、省略時の探索パスやシェル・プロンプトを変更することができます。ユーザの省略時のシェルが Bourne、Korn、または POSIX シェルの場合 (7.3 節を参照) のログイン・スクリプトは `.profile`、C シェルの場合のログイン・スクリプトは `.login` です。ローカル・ログイン・スクリプトは、システム・ログイン・スクリプトの後に実行されます。

省略時のシェルが C シェルの場合 `.cshrc` ファイルを使用して環境を調整することもできます。`.cshrc` はログイン時 (`.login` の後) およびサブシェル

の作成 (spawn) 時に実行されます。 `.cshrc` ファイルを使用すると変数がサブシェルで自動的に有効になります。

Korn および POSIX シェルでは起動時に、`ENV` 環境変数によって設定されているファイルを実行します。この変数は通常 `.profile` ファイルで設定され、一般には `$HOME` ディレクトリにある別のファイル `.kshrc` または `.envfile` 等に対して設定されます。このような設定を行う場合には `.profile` ファイルに次のような記述を追加します。

```
ENV=~/.kshrc
```

このファイルには通常は、シェル変数、別名定義、関数定義などを記述します。本書では、このファイルを `.kshrc` とよびます。

システム・ログイン・スクリプトで基本的な環境は設定されるので、ユーザ自身のログイン・スクリプトは必ず作成しなくてはならないわけではありません。システム管理者がローカル・ログイン・スクリプトを作成している場合があるので、各ユーザはこれをエディタで編集して使用することができます。

システムに不慣れな場合は、すでに設定されている省略時の環境を使用したいと思うかもしれません。しかし、システムに慣れるにしたがい、自分のログイン・スクリプトを作成したり修正したいと思うようになるでしょう。

表 7-6 に、Tru64 UNIX オペレーティング・システムの各シェルのシステム・ログイン・スクリプトとローカル・ログイン・スクリプトを示します。システムにログインするとシェルに応じて、すべてのスクリプトが実行されます。さらに、任意のシェル・プロンプトに対して `cs` と入力すると、`.cshrc` ファイルが実行され、C シェルのサブシェルが作成されます。

表 7-6: システムおよびローカルのログイン・スクリプト

シェル	パス名	システム・ログイン・スクリプト	ローカル・ログイン・スクリプト
Bourne	/usr/bin/sh	/etc/profile	.profile
Korn	/usr/bin/ksh	/etc/profile	.profile ENV
POSIX	/usr/bin/posix/sh	/etc/profile	.profile ENV
C	/usr/bin/csh	/etc/csh.login	.login .cshrc

ホーム・ディレクトリにローカル・ログイン・スクリプトがあるかどうかを確認するためには、`ls -a` コマンドを使用します。このコマンドは、ドット(.) で始まるすべてのファイルを、他のエントリと一緒に表示します。

次のカスタマイズ機能は、一般にログイン・スクリプトで設定されています。

- 端末特性
- 探索パスとその他の環境変数
- シェル変数
- `umask` による新しいファイルに対する最大許可 (第 5 章を参照)
- ワークステーションへのメッセージの許可または停止
- `trap` コマンド (Bourne, Korn, および POSIX シェルのみ)
- コマンド別名, ヒストリ変数 (C シェルと Korn または POSIX シェルのみ)
- システム状態情報とその他のメッセージの表示
- メールの有無のチェック
- ニュースの有無のチェック

システム・ログイン・スクリプトの内容をチェックして、ローカル・ログイン・スクリプトで重複しないようにするとよいでしょう。たとえば、システム・ログイン・スクリプトでニュースの有無をチェックしていれば、ローカル・ログイン・スクリプトで同じことをする必要はありません。

ログイン・スクリプトの個々の例については、第 8 章を参照してください。

7.7 変数の使用

Tru64 UNIX オペレーティング・システムのすべてのシェルは、環境変数とシェル変数を使用して、ユーザ環境の特性を定義します。システム管理者は、セット・アップ処理の一環として、適切なログイン・スクリプトで省略時の環境変数およびシェル変数の値を設定しています。

大部分のユーザにとっては、省略時の環境変数およびシェル変数の値で十分です。しかし、システムに慣れるにしたがい、一部の変数の値を修正したいと思うようになるかもしれません。たとえば、シェル・プロンプトを定義する変数を再設定して、もっと自分独自のものにしたいと思うかもしれません。また、非常に長いディレクトリ・パス名を指定するシェル変数を設定して、そのディレクトリを使用するコマンドをタイプする時間を節約できるよ

うにしたいと思うかもしれません (7.7.1 項に示す例を参照)。あるいは、シェル・プロシージャを作成するとき、変数を設定したほうが便利だと思うかもしれません。このように変数を創造的に使用すれば、作業環境が改善できることがわかるでしょう。

環境変数のなかには再設定できるものもあれば、読み取り専用で再設定できないものもあります。つまり、これらの変数は使用はできますが、修正はできません。詳しくは、該当するシェルのリファレンス・ページ、`sh(1b)`、`sh(1p)`、`cs(1)`、または `ksh(1)` を参照してください。

環境変数を再設定したり、ユーザ自身のシェル変数を定義するには、次のいずれかを行います。

- ログインするときにいつもその値が設定されているようにするには、該当するローカル・ログイン・スクリプトを編集します。詳しくは、7.6 節を参照してください。
- 現在のログイン・セッションのためにだけその値を設定したい場合は、コマンド行で環境変数を設定します。

ユーザはいつでも、任意の変数の値を参照したり表示することができます。また、どの変数の値もクリアすることができます。

7.7.1 変数の設定

以降の各項で、変数値の設定、参照、表示、クリアの方法について説明します。

7.7.1.1 Bourne シェル、Korn シェル、および POSIX シェルのシェル変数

Bourne、Korn、および POSIX シェルでは、変数は代入文で設定します。変数設定のための一般形式は次のとおりです。

name = *value*

name には変数名を指定します。 *value* には、変数に代入する値を指定します。コマンド行にはスペースを入れないようにしてください。

シェル変数をサブシェルで使えるようにするには、次のように `export` コマンドを入力します。

export *name*

シェル変数をエクスポートすると環境変数になります。

Bourne シェルの場合は、2 つのステートメントが必要になります。 Korn および POSIX シェルでは、2 つのステートメントを次のように結合することができます。

export *name* = *value*

たとえば、*place* という変数を、U. S. A. という値を代入することによって作成できます。 その場合には、次の文を使用します。

```
$ place='U. S. A.'  
$
```

これ以後は、変数 *place* は、その値を使用するのとまったく同様に使用することができます。

もっといい例として、Bourne シェルを使用していて、シェル・プロンプトを一時的に自分独自のものにしたいとします。 省略時の Bourne シェル・プロンプトは、*PS1* 環境変数によって設定されるドル記号 (\$) です。 それを What Shall I Do Next? > に設定するには、次のコマンドを入力します。

```
$ PS1='What Shall I Do Next? >'  
What Shall I Do Next? >
```

このシェル・プロンプトをサブシェルでも利用できるようにするには、次のコマンドを入力します。

```
$ export PS1
```

この What Shall I Do Next? > というプロンプトはセッションを通して有効です。 この新しいプロンプトを永続的なものにしたい場合は、*.profile* ファイルに同じ代入文と *export* コマンドを入力してください。

もう 1 つ例を挙げると、タイプする時間を節約するため、何度も使用する長いパス名の変数を定義したいとします。 ディレクトリ */usr/sales/shoes/women/retail/reports* を表わす変数 *reports* を定義するには、次のように入力します。

```
$ reports=/usr/sales/shoes/women/retail/reports
```

変数を設定したのちに参照するには、変数名の前に \$ を入力します。 変数参照についての詳細は、7.7.2 項を参照してください。

このセッション中に入力する任意のコマンドで変数 *reports* を使用できます。 このプロンプトを永続的なものにする場合は、*.profile* ファイルに同じ代入文を入力してください。

7.7.1.2 C シェルの変数

C シェルでは、環境変数は `setenv` コマンドを使用して設定します。 `setenv` コマンドの一般形式は次のとおりです。

`setenv name value`

`name` には変数名を指定します。 `value` には、変数に代入する値を指定します。

`PATH` 環境変数を設定する例として、7.8 節を参照してください。

シェル変数は `set` コマンドで設定します。 `set` コマンドの一般形式は次のとおりです。

`set name = value`

`name` には変数名を指定します。 `value` には、変数に代入する値を指定します。 `value` にスペースを含む場合には、式全体を一重引用符 (`'`) で囲んでください。

たとえば、プロンプトを変更したいとします。省略時の C シェル・プロンプトは `%` です。それを `Ready? >` に変えたいときは、コマンド行に次のように入力します。

```
% set prompt='Ready? >'
Ready? >
```

`Ready? >` プロンプトはセッションを通して有効です。 `Ready? >` プロンプトから別のシェルを実行すると、その新しいシェルのプロンプトが表示されます。この新しいプロンプトを永続的なものにする場合は、`.cshrc` ファイルに同じコマンドを入力してください。

7.7.1.3 すべてのシェルにおける変数の設定

Tru64 UNIX オペレーティング・システムの任意のシェルで、環境変数またはシェル変数を設定あるいは再設定するときは、次のいずれかを行います。

- ログインするときにいつもその値が設定されているようにするには、該当するローカル・ログイン・スクリプトを編集します。詳しくは、7.6 節を参照してください。
- 現在のログイン・セッションのためにだけその値を設定したい場合は、コマンド行から設定します。

7.7.2 変数の参照 (パラメータ置換)

コマンド行で変数の値を参照するには、変数名の前にドル記号 (\$) を入力します。\$ によって、現在使用しているシェルは変数名を変数の値に置き換えます。これはパラメータ置換と呼ばれるものです。

たとえば、長いパス名 `/user/reports/Q1/march/sales` を表わす変数 `sales` をあらかじめ定義しておいて、この変数を `cd` コマンドで使用したいとします。これを行うには、次のように `sales` 変数を指定して `cd` コマンドを入力します。

```
$ cd $sales
$
```

その後、`pwd` コマンドを入力して、ディレクトリが変更されたことを確認します。

```
$ pwd
/user/reports/Q1/march/sales
$
```

この例では、シェルは変数名 `sales` をディレクトリの実際のパス名 `/user/reports/Q1/march/sales` に置き換えます。

7.7.3 変数値の表示

自分のシェルに現在設定されているすべての変数の値を表示させることができます。変数値は、単独で表示することもグループとして表示することもできます。

単一の変数の値を表示するには、`echo` コマンドを次の一般形式で使します。

echo *\$variable*

variable には、値を表示させる変数を指定します。

たとえば、Korn シェルを使用している場合に、`SHELL` 環境変数の値を表示させたいとします。これを行うには、次のコマンドを入力します。

```
$ echo $SHELL
/usr/bin/ksh
$
```

Bourne、Korn、および POSIX シェルの場合、現在設定されているすべての変数の値を表示するには、オプションなしで `set` コマンドを使用します。た

例えば、次の例は Bourne シェルに現在設定されている値の一覧を表示します (使用しているシステムにより出力は異なります)。

```
$ set
EDITOR=emacs
HOME=/users/chang
LOGNAME=chang
MAIL=/usr/mail/chang
PATH=:/usr/bin:/usr/bin/X11
PS1=$
SHELL=/usr/bin/sh
TERM=xterm
$
```

C シェルの場合、現在設定されているすべてのシェル変数の値を表示するには、オプションなしで `set` コマンドを使用します。現在設定されているすべての環境変数の値を表示するには、オプションなしで `setenv` コマンドまたは `printenv` コマンドを使用します。

7.7.4 変数値のクリア

現在設定されている変数値のほとんどは削除することができます。ただし、次の変数はクリアできないことに注意してください。

- `PATH`
- `PS1` (Bourne, Korn, および POSIX シェル)
- `PS2` (Bourne, Korn, および POSIX シェル)
- `MAILCHECK` (Bourne, Korn, および POSIX シェル)
- `IFS` (Bourne, Korn, および POSIX シェル)

これらの変数についての詳細は、該当するシェルのリファレンス・ページ、`sh(1b)`、`sh(1p)`、`csch(1)`、または `ksh(1)` を参照してください。

Bourne, Korn, および POSIX シェルでは、環境変数とシェル変数の両方を `unset` コマンドでクリアします。 `unset` コマンドの形式は次のとおりです。

unset *name*

name には変数名を指定します。

C シェルでは、環境変数は `unsetenv` コマンドでクリアします。 `unsetenv` コマンドの形式は次のとおりです。

unsetenv *name*

name には変数名を指定します。

シェル変数は **unset** コマンドでクリアします。 **unset** コマンドの形式は次のとおりです。

unset *name*

name には変数名を指定します。

たとえば、Korn シェルを使用している場合に、*place* という変数を作成し、その変数に U. S. A. という値を割り当てているとします。この変数をクリアするには、次のように入力します。

```
$ unset place
$
```

変数の設定と参照についての詳細は、該当するシェルのリファレンス・ページを参照してください。

7.8 シェルによるコマンドの見つけ方

ユーザがコマンドを入力すると、シェルはディレクトリのリストを探索して、コマンドを見つけます。このディレクトリのリストは *PATH* 環境変数によって指定します。

多くのインストールでは、システム管理者が新規ユーザに対して省略時の *PATH* ディレクトリを指定します。しかし、経験を積んだユーザは、これらの *PATH* ディレクトリを変更する必要があるかもしれません。

PATH 変数には、探索するディレクトリのリストが含まれ、各ディレクトリはコロン (:) で区切られてます。シェルは、ディレクトリがリストされている順に従って、ユーザの入力するコマンドを探索します。

PATH の値を確認するには、**echo** コマンドを使用します。たとえば、C シェルを使用しているときに、次のように入力したとします。

```
% echo $PATH
/usr/bin:/usr/bin/X11
%
```

この **echo** コマンドからの出力は、上記の例の探索順が次のとおりであることを示しています。なお、システムによって出力は異なります。

- */usr/bin* ディレクトリが最初に探索される。

- `/usr/bin/X11` ディレクトリが次に探索される。

一般に、`PATH` は適切なログイン・スクリプトで環境変数として設定されます。Bourne、Korn、および POSIX シェルでは、`PATH` 変数は通常 `.profile` スクリプトで設定されます。C シェルでは、通常、`.login` スクリプトで設定されます。

探索経路を変更したい場合は、`PATH` 変数に新しい値を割り当てることができます。たとえば、Bourne シェルを使用しているとき、いくつかのオペレーティング・システムのコマンドの代わりに、ユーザ独自のコマンドを使用することにします。そこで、探索パスに `$HOME/usr/bin/` を追加します。新しい `PATH` 変数値を現在のログイン・セッションについて有効なものにしたい場合は、コマンド行に次のように入力してください。

```
$ export PATH=$HOME/usr/bin:/usr/bin:/usr/bin/X11
$
```

この新しい `PATH` 変数値を将来の全セッションにわたって有効にしたい場合には、`.profile` スクリプトの `PATH` 変数を修正してください。`.profile` スクリプトに対して行った変更は、次回ログインしたときに有効になります。

7.9 ログアウト・スクリプトの使用

セッションを終了すると自動的に実行されるログアウト・スクリプトを作成することができます。ログイン・スクリプトと同様に、`.logout` ファイルはユーザのホーム・ディレクトリになければなりません。

ログアウト・スクリプトは次の目的に使用できます。

- 画面をクリアする。
- ログアウト・メッセージを表示する。
- ログアウト後、長いバックグランド・プロセスを実行する。
- ファイル・クリーンアップ・ルーチンを実行する。

ログアウト・スクリプトを作成するには、次の手順に従います。

1. テキスト・エディタを使用して、ホーム・ディレクトリに `.logout` というファイルを作成する。
2. 実行させるコマンドをそのファイルに入力する。

概略については 1.2 節を参照。

3. テキストを保管して、エディタを終了する。
4. 次のコマンドを入力して、`.logout` ファイルに適切な実行許可があることを確認する。

```
$ chmod u+x .logout
$
```

必ずしも `.logout` ファイルを使用する必要はありません。これはユーザの作業環境を改善する便利な手段です。

7.9.1 ログアウト・スクリプトとシェル

C シェルを使用している場合、ログアウトすると `.logout` スクリプトが自動的に実行されます。

Korn または POSIX シェルを使用しているときに、ログアウト・スクリプトを使用したい場合は、`.profile` スクリプトに特殊なトラップが設定されていることを確認しなければなりません。トラップとは、端末からの指定された信号を探して、指定のコマンドまたはコマンド・セットを実行するコマンド・シーケンスのことです。

`.profile` スクリプトに次の行が設定されていない場合には、テキスト・エディタを使用して追加しなければなりません。

```
trap $HOME/.logout 0
```

この文は、ゼロ (0) 信号を受信すると必ず `.logout` スクリプトを実行するようシステムに指示するものです。ゼロ (0) 信号は、ログアウトするときに自動的に発生します。

7.9.2 `.logout` ファイルの例

この項で示すサンプルの `.logout` ファイルは、次のことを行います。

- 画面をクリアする。
- システム名、ユーザ名、およびログアウト時刻からなるログアウト・メッセージを表示する。
- 別れのメッセージを表示する。
- ログアウト後、バックグラウンドでファイル・クリーンアップ・ルーチンを実行する。

番号記号(#)で始まる行は、その下のコマンドについて説明しているコメント行です。

```
# Clear the screen
clear

# Display the name of your system, your user name,
# and the time and date that you logged out
echo 'hostname' : 'whoami' logged out on `date`

# Run the find command in the background. This command
# searches your login directory hierarchy for all
# temporary files that have not been accessed in
# 7 days and then deletes them.
find ~ -name '*.tmp' -atime +7 -exec rm {} \; &

# A parting message
echo "Good Day. Come Back Soon"
```

7.10 シェル・プロシージャの使用 (スクリプト)

シェルは、コマンド行から入力されたコマンドを実行するだけでなく、ファイルに含まれているコマンドを読み取って実行することができます。このようなファイルは、シェル・プロシージャまたはシェル・スクリプトと呼ばれます。

シェル・プロシージャの作成は簡単で、しかもそれを使用すると、より効率的に作業を進めることができます。たとえば、頻繁に使用するコマンドを1つのファイルに入れておけば、プロシージャの名前だけを入力することによって、それらを実行することができます。このようなことから、シェル・プロシージャを使用するとたいへん便利なのがわかるでしょう。したがって、本来なら多数のコマンドをコマンド行に入力する必要がある反復的な作業を行う場合に役立ちます。

シェル・プロシージャはコンパイルする必要のないテキスト・ファイルであるため、作成と保守が簡単です。

各シェルには固有のネイティブ・プログラミング言語があります。すべてのシェルにあてはまるプログラミング言語の機能を次に示します。

- 変数に値を格納する。
- 事前定義条件を検査する。
- コマンドを反復して実行する。

- 引数をプログラムに渡す。

各シェルの特定のプログラミング機能についての詳細は，第 8 章を参照してください。

7.10.1 シェル・プロシージャの作成と実行

シェル・プロシージャの作成と実行は，次の手順に従って行います。

1. 作業に必要なコマンドのファイルを作成する。

このファイルは，一般のテキスト・ファイルと同様に作成します。つまり，vi または他の編集プログラムを使用して行います。このファイルには，任意のシステム・コマンドまたはシェル・コマンドを入れることができます。詳しくは，sh(1b)，sh(1p)，csh(1)，または ksh(1) のリファレンス・ページを参照してください。

2. chmod +x コマンドを使用して，ファイルに x (実行) 状態を付与する。

たとえば，コマンド chmod g+x reserve は，グループ (g) のすべてのユーザに対して，reserve という名前のファイルを実行可能にします。chmod コマンドについての詳細は，第 5 章を参照してください。

3. ファイル名を入力して，そのプロシージャを実行する。

そのプロシージャ・ファイルが現在のディレクトリにない場合はパス名を入力します。

次に示す例は，ls -l コマンドの出力をファイル・サイズによってソートする，lss という名前の単純なシェル・プロシージャです。

```
#!/usr/bin/csh
# lss: sort and list
ls -l | sort -n +4
```

表 7-7 で lss の各行について説明します。

表 7-7: シェル・スクリプト例の説明

シェル・コマンド	説明
#!/usr/bin/csh	シェル・プロシージャを実行するシェルの指定する。 ^a

表 7-7: シェル・スクリプト例の説明 (続き)

シェル・コマンド	説明
#lss: list and sort	シェル・プロシージャの目的を説明するコメント行。
ls -l sort -n +4	シェル・プロシージャのコマンド。このプロシージャはディレクトリ内のファイルをリストする (ls -l)。ls -l コマンドからの出力は sort コマンドにパイプされる (sort -n +4)。このコマンドは ls -l 出力の最初の 4 カラムを読み飛ばし、5 番目のカラム (ファイル・サイズ・カラム) を数値でソートしてから、その行を標準出力に書き込む。

^a詳細は 7.10.2 項を参照。

lss プロシージャを実行するには、lss と入力します。システムの出力例は次のようになります。

```
$ lss
-rw-rw-rw- 1 larry system 65 Mar 13 14:46 file3
-rw-rw-rw- 1 larry system 75 Mar 13 14:45 file2
-rw-rw-rw- 1 larry system 101 Mar 13 14:44 file1
$
```

7.10.2 実行シェルの指定

特にCシェルと他のシェルとの間には構文上の違いが存在することがあるため、シェル・プロシージャの実行に使用するシェルを指定したい場合があります。

省略時には、オペレーティング・システムは、ユーザが実行するシェル・プロシージャはログイン・シェルと同じシェルで実行されるものと想定します。たとえば、ログイン・シェルが Korn シェルであれば、省略時には、シェル・プロシージャはその同じシェルで実行されます。

省略値を無効にする機能は、多くのユーザが実行するシェル・プロシージャにとって非常に便利なものです。なぜなら、この機能によって、ユーザのログイン・シェルにかかわらず、シェル・プロシージャを正しいシェルで実行できるからです。この省略時の実行シェルを変更するには、次のコマンドをシェル・プロシージャの 1 行目に入れてください。

```
#!shell_path
```


`shell_path` には、そのプロシージャを実行させるシェルの完全パス名を指定します。

たとえば、シェル・プロシージャをC シェルで実行したい場合には、そのシェル・プロシージャの1行目は次のようになります。

```
#!/usr/bin/csh
```



シェルの機能

この章では、他の章のように概念や入門的な知識の紹介ではなく、各シェルについての簡略な参照情報を提供します。

この章は、第 7 章で説明しているシェルの概要については理解していることを前提に記述されています。

この章を読み終えると、次のことが可能になります。

- オペレーティング・システムのシェル相互間の主な違いを理解する (8.1 節)。
- オペレーティング・システムの各シェルの特殊機能を理解する (8.2 節)。
- 各シェルのローカル・ログイン・スクリプトの特殊性を理解する (8.2.1 項)。

8.1 C , Bourne , Korn , および POSIX シェルの機能の比較

表 8-1 は C シェルおよび Bourne , Korn , および POSIX シェルの主な機能を比較したものです。

表 8-1: C , Bourne , Korn , および POSIX シェルの機能

機能	説明	C	Bourne	Korn または POSIX
シェル・プログラミング	ループ、条件文、および変数などの機能を含むプログラム言語	有	有	有
シグナル・トラップ	オペレーティング・システムから送られる割り込みやその他のシグナルをトラップするメカニズム	有	有	有
制限付きシェル	制御されたシェル環境に限定された機能を与えるセキュリティ機能	無	有	無
コマンド別名	長いコマンド行の短縮やコマンド名の変更が行える機能	有	無	有

表 8-1: C , Bourne , Korn , および POSIX シェルの機能 (続き)

機能	説明	C	Bourne	Korn または POSIX
コマンド・ ヒストリ	コマンドを格納し、そのコマンド の編集と再使用が行える機能	有	無	有
ファイル名補完	ファイル名の一部を入力する と、システムが自動的にそれを 補完するか、あるいは選択肢の リストを提示する機能	有	無	有
コマンド行編集	現在のコマンド行または以前に入力 したコマンド行の編集が行える機能	有	無	有
配列	データをグループ化し、名前 で呼び出す能力	有	無	有
整数算術	シェル内で算術関数を実行する能力	有	無	有
ジョブ制御	バックグラウンド・プロセスをモ ニタし、それにアクセスする機構	有	無	有

シェルの機能についての詳細は、該当するシェルのリファレンス・ページ、
sh(1b)、sh(1p)、csh(1)、または ksh(1) を参照してください。

8.2 C シェルの機能

この節では、次のC シェルの機能について説明します。

- .cshrc スクリプトおよび .login スクリプトの例
- メタキャラクタ
- コマンド・ヒストリおよび別名
- 組み込み変数およびコマンド

8.2.1 .cshrc スクリプトおよび .login スクリプトの例

.cshrc ログイン・スクリプトは、ローカル・シェル・プロセスのための変
数とオペレーティング・パラメータを定義することによって、ユーザの C
シェル環境をセットアップします。.login スクリプトは、セッションの開
始時に実行したい変数やオペレーティング・パラメータ、および、現在のロ
グイン・セッション中、すべてのシェル・プロセスで有効にしたい変数や
オペレーティング・パラメータを定義します。

ログインすると、オペレーティング・システムは、まずホーム・ディレクトリにある `.cshrc` ファイルを実行し、次に `.login` ファイルを実行します。`.login` スクリプトは、ログインしたときにだけ実行されます。しかし、`.cshrc` ファイルは、サブシェルを作成するたびに実行されます。

次に示す `.cshrc` スクリプトでは、シェル変数、コマンド別名、およびコマンド・履歴変数が設定されています。表 8-2 で、スクリプトの各部分について説明します。

```
# Set shell variables
set noclobber
set ignoreeof
set notify

# Set command aliases
alias h 'history \!* | more'
alias l 'ls -l'
alias c clear

# Set history variables
set history=40
set savehist=40

# Set prompt
set prompt = "\! % "
```

表 8-2: `.cshrc` スクリプト例の説明

コマンド	説明
シェル変数	
<code>set noclobber</code>	ファイルへの重ね書きを禁止する。これが設定されていると、出力リダイレクションに制限を加えて、ファイルが誤って破壊されないようにし、 <code>>></code> リダイレクションが既存ファイルを参照するようにする。
<code>set ignoreeof</code>	ログイン・セッションを終了させるために <code>Ctrl/D</code> を使用できないことを指定する。代わりに、 <code>exit</code> または <code>logout</code> コマンドを使用しなければならない。
<code>set notify</code>	バックグラウンド・プロセスが完了したときにユーザに通知する。
コマンド別名	

表 8-2: .cshrc スクリプト例の説明 (続き)

コマンド	説明
alias h 'history \!* more'	more コマンドにパイプするコマンド・ヒストリ・バッファの内容を定義する。文字列 \!* は、すべてのヒストリ・バッファをパイプするように指定する。
alias l 'ls -l'	ディレクトリ・ファイルを詳細なフォーマットでリストする ls -l コマンドの簡略名として、l を定義する。
alias c clear	画面をクリアする clear コマンドの簡略名として、c を定義する。
ヒストリ変数	
history=40	最後の 40 個のコマンドをヒストリ・バッファに格納することをシェルに指示する。
savehist=40	最後の 40 個のコマンドを格納して、次のログイン・セッションの開始時ヒストリとして使用することをシェルに指示する。
プロンプト変数	
set prompt = "\! % "	プロンプトを現在のコマンドのコマンド番号を表示するプロンプトに変更する。

次の .login スクリプトでは、ファイル作成の許可を設定し、PATH 環境変数を設定し、エディタとプリンタを指定します。表 8-3 で、スクリプトの各部分について説明します。

```
# Set file creation permissions
umask 027

# Set environment variables
set path=/usr/bin:/usr/local/bin:
set cdpath=...:$HOME
setenv EDITOR emacs
setenv MAILHOST boston
setenv PRINTER sales
```

表 8-3: .login スクリプト例の説明

コマンド	説明
ファイル許可	
<code>umask 027</code>	プログラムによって新たに作成されるすべてのファイルに対して設定される省略時の許可から差し引く許可を指定する。umask の値は、777 (実行可能プログラムの場合) または 666 から差し引かれる。実行可能プログラムの場合、umask 値 027 を設定すると、所有者に対してはすべての許可を与え、同じグループのメンバに対しては読み取りおよび実行の許可を与え、それ以外のユーザに対してはどの許可も与えないことになる。
環境変数	
<code>set path \</code> <code>/usr/bin:/usr/local/bin:</code>	探索パスを指定する。この場合、まず /usr/bin を探索し、次に /usr/local/bin を探索する。
<code>set cdpath=...:\$HOME</code>	cdpath は、cd コマンドの探索パスを設定する変数である。この変数代入では、cd コマンドが、現在のディレクトリ(.)、親ディレクトリ(..)、ホーム・ディレクトリ(\$HOME)の順で、指定されたディレクトリを探索するように指定する。
<code>setenv EDITOR emacs</code>	ファイル編集を可能にするプログラムを実行するとき、emacs を省略時のエディタとして指定する。たとえば、各種のメール・プログラムでは、エディタを使用して、メッセージを作成したり編集できる。
<code>setenv MAILHOST boston</code>	boston をメール・ハンドリング・システムとして指定する。
<code>setenv PRINTER sales</code>	プリンタ sales を省略時のプリンタとして指定する。

8.2.2 メタキャラクタ

表 8-4 は C シェルのメタキャラクタについて説明したものです。メタキャラクタとは、シェルにとって特別な意味を持つ文字のことです。これらのメタキャラクタの意味は、シェル・スクリプト、ファイル名指定で使用される場合、別の文字の引用、入出力で使用される場合、あるいは、可変の置換を示すために使用される場合にグループ分けされます。

表 8-4: C シェルのメタキャラクタ

メタキャラクタ	説明
構文	
;	順次実行するコマンドを区切る。
	パイプラインを構成するコマンドを区切る。
&&	現在のコマンドが成功した場合に次のコマンドを実行する。
	現在のコマンドが失敗した場合に次のコマンドを実行する。
()	別個のプロセスとしてサブシェルで実行するコマンドをグループ化する。
&	コマンドをバックグラウンドで実行する。
ファイル名	
/	ファイルのパス名の各部分を区切る。
?	先頭のドット (.) を除く任意の単一文字に対応する。
*	先頭のドット (.) を除く任意の文字シーケンスに対応する。
[]	[] 内の文字のいずれかの文字に対応する。
~	ファイル名の先頭に使用する場合には、ホーム・ディレクトリを指定する。
引用符	
'...'	' ' で囲まれた文字をいずれも本来の意味で、つまり、シェルにとって特殊な意味を加えることなく解釈するように指定する。
"..."	特別な引用形式を提供する。\$ (ドル記号), ` (抑音符), および \ (バックスラッシュ) 文字はそれぞれの特殊な意味を保持するが, " " で囲まれた他のすべての文字は本来の意味に、つまり、シェルにとって特殊な意味を加えることなく解釈するように指定する。二重引用符は変数代入を行う際に有用である。
入出力	
<	入力のリダイレクトする。
>	出力を指定ファイルにリダイレクトする。
<<	入力のリダイレクトし、シェルが入力を指定された行まで読み取ることを指定する。
>>	出力をリダイレクトし、シェルが出力をファイルの終わりに追加することを指定する。

表 8-4: C シェルのメタキャラクタ (続き)

メタキャラクタ	説明
>&	診断および標準出力の両方をリダイレクトして、ファイルに付加する。
>>&	診断および標準出力の両方を、既存ファイルの終わりにリダイレクトする。
>!	出力をリダイレクトする。また、 <i>noclobber</i> 変数が設定されている (ファイルの重ね書きを禁止する) 場合、その設定を無視して、ファイルに重ね書きできるように指定する。
置換	
\$	変数置換を指定する。
!	ヒストリ置換を指定する。
:	置換修飾子の前に置く。
^	特殊な種類のヒストリ置換に使用する。
`	コマンド置換を指定する。

8.2.3 コマンド・ヒストリ

コマンド・ヒストリ・バッファは、入力されたコマンドを格納し、そのコマンドをいつでも表示できるようにします。このため、以前に入力したコマンドやその一部を選択して、再実行することができます。この機能を利用すれば、長いコマンドを再入力しなくても再使用できるため、時間の節約になります。

.cshrc ファイルに、次の 3 つのコマンドを記述することができます。

- `set history=n`
入力するコマンド行を格納するヒストリ・バッファを作成します。n には、ヒストリ・バッファに格納するコマンド行の数を指定します。
- `set savehist=n`
現在のログイン・セッション中に入力したコマンド行を保存して、次のログイン・セッションで利用できるようにします。n には、ログアウト時にヒストリ・バッファに格納するコマンド行の数を指定します。
- `set prompt=[\!] %`
C シェル・プロンプトに、各コマンド行の番号を表示させます。

コマンド・ヒストリ・バッファの内容を見るときには，history コマンドを使用します。すると，次のような出力が表示されます (実際の表示は異なります)。

```
[18] % history
      3 set history=15
      4 pwd
      5 cd /usr/sales
      6 ls -l
      7 cp report report5
      8 mv /usr/accounts/new .
      9 cd /usr/accounts/new
     10 mkdir june
     11 cd june
     12 mv /usr/accounts/new/june .
     13 ls -l
     14 cd /usr/sales/Q1
     15 vi earnings
     16 cd /usr/chang
     17 vi status
     18 history
[19] % _
```

コマンド・ヒストリ・バッファ内のコマンドを再実行するには，表 8-5 に示すコマンドを使用します。各コマンドは感嘆符 (!) で始まっています。これは，ユーザがヒストリ・バッファ内のコマンドを使用していることを C シェルに知らせるものです。

表 8-5: ヒストリ・バッファ内のコマンドの再実行

コマンド	説明
!!	直前のコマンドを再実行する。
! <i>n</i>	<i>n</i> で指定したコマンドを再実行する。たとえば，前述の例で示したヒストリ・バッファの場合，! <i>5</i> によって <code>cd /usr/sales</code> コマンドが再実行される。
! <i>-n</i>	現在のコマンドに相対して以前のコマンドを再実行する。たとえば，前述の例で示したヒストリ・バッファの場合，! <i>-2</i> によってコマンド番号 17 の <code>vi status</code> コマンドが呼び出される。

表 8-5: ヒストリ・バッファ内のコマンドの再実行 (続き)

コマンド	説明
<code>!<i>string</i></code>	<i>string</i> によって指定される文字列で始まる最後に実行されたコマンドを再実行する。たとえば、前述の例で示したヒストリ・バッファの場合、 <code>!cp</code> によってコマンド番号 7 の <code>cp report report5</code> が呼び出される。
<code>!?<i>string</i></code>	<i>string</i> によって指定される文字列と同じ文字列を含む最後に実行されたコマンド行を再実行する。たとえば、前述の例で示したヒストリ・バッファの場合、 <code>!?Q1</code> によってコマンド番号 14 の <code>cd /usr/sales/Q1</code> が呼び出される。

コマンド・ヒストリ・バッファを使用すると、以前のコマンド引数を再使用することも以前のコマンド行を修正することもできます。これらの機能については、`cs(1)` リファレンス・ページを参照してください。

8.2.4 ファイル名補完

C シェルでは、シェル・プロンプトに対してファイル名またはパス名の一部を入力すると、シェルが自動的にその名前を照合して補完します。この機能により、長い一意のファイル名を表示する場合に、時間の節約ができます。

たとえば、現在のディレクトリに `meetings_sales_status` というファイルがあるとします。このファイルの詳細なリストを表示させるには、次のコマンドを入力します。

```
% ls -l meetings Escape
```

システムは同じコマンド行に次のように表示します。

```
% ls -l meetings_sales_status
```

ここで、Return キーを押せばコマンドを実行できます。

ファイル名補完についての詳細は、`cs(1)` リファレンス・ページを参照してください。

8.2.5 別名

コマンド別名機能を使用すると、長いコマンド行を短縮したり、コマンドの名前を変更することができます。これは、頻繁に使用する長いコマンド行に対して別名を作成することにより行います。

たとえば、`/usr/chang/reports/status` というディレクトリに頻繁に移動する必要があるとします。その場合には、`status` という別名を作成

して、コマンド行にその別名を入力するだけで、いつでもそのディレクトリに移動できるようにします。

さらに、別名を使用すると、コマンドに対してもっとわかりやすい名前を定義することができます。たとえば、`mv` コマンドに対して `rename` という別名を定義することもできます。

別名を作成するには、`alias` コマンドを使用します。 `alias` コマンドの形式は次のとおりです。

alias *aliasname command*

aliasname には、使用したい名前を指定します。 *command* には、元のコマンドあるいは一連のコマンドを指定します。 *command* が複数の部分に分かれている (間にスペースがある) 場合は、式全体を一重引用符 (') で囲んでください。

たとえば、`/usr/chang/reports/status` というディレクトリに移動するための別名 `status` を作成するには、次のコマンドを入力します。

```
% alias status 'cd /usr/chang/reports/status'
```

別名を定義する場合には、通常 `.cshrc` ファイルに記述して、ユーザ環境の永続的な要素にします。 こうしておけば、ログインしたり、新しいシェルを開始するたびに、その別名を使用することができます。 例については、8.2.1 項を参照してください。

すべての別名定義を表示するには、次のコマンドを入力します。

```
% alias
```

特定の別名定義を表示するには、次のように入力します。

```
% alias aliasname
```

aliasname には、定義を表示させる特定の別名を指定します。

現在のログイン・セッションについて別名を削除するには、`unalias` コマンドを使用します。 `unalias` コマンドの一般形式は次のとおりです。

unalias *aliasname*

aliasname には、削除する名前を指定します。

現在および将来のログイン・セッションのすべてについて別名を削除するには、次のようにします。

1. 次のコマンドを入力する。

```
% unalias aliasname
```

aliasname には、削除する別名を指定します。

2. `.cshrc` ファイルを編集して、その別名定義を削除した後、ファイルを保存する。
3. 次のコマンドを入力して、`.cshrc` ファイルを再実行する。

```
% source .cshrc
```

C シェルにおける別名使用の詳細については、`csh(1)` リファレンス・ページを参照してください。

8.2.6 組み込み変数

C シェルには、値を代入できる変数が用意されています。これらの変数は、後でコマンドで利用できる値を格納するのに非常に役立ちます。さらに、シェル自体が参照する変数を設定することによって、シェルの動作に直接影響を与えることができます。

表 8-6 では、一般ユーザにとって重要な C シェルの組み込み変数について説明しています。C シェルの組み込み変数の完全な一覧については、`csh(1)` リファレンス・ページを参照してください。

表 8-6: 組み込み C シェル変数

変数	説明
<code>argv</code>	シェルまたはシェル・スクリプトで利用できる 1 つまたは複数の値。
<code>cwd</code>	現在のディレクトリのパス名。この変数の値は、 <code>cd</code> コマンドを実行するたびに変更される。
<code>home</code>	ホーム・ディレクトリのパス名。この変数の省略値は、 <code>/etc/passwd</code> ファイルで指定される。
<code>ignoreeof</code>	システムからログアウトするために <code>Ctrl/D</code> を使用できるかどうかを指定する。これが設定されている場合は、ログアウトするには <code>logout</code> または <code>exit</code> を使用しなければならない。設定されていない場合には、 <code>Ctrl/D</code> を使用してログアウトできる。この変数は通常 <code>.cshrc</code> ファイルで設定される。
<code>cdpath</code>	<code>cd</code> 、 <code>chdir</code> 、または <code>pushd</code> コマンドでサブディレクトリを探するとき、システムが探索する代替ディレクトリを指定する。この変数は通常 <code>.login</code> ファイルで設定される。

表 8-6: 組み込み C シェル変数 (続き)

変数	説明
<i>noclobber</i>	ファイルに重ね書きできるかどうかを指定する。これが設定されている場合は、出力リダイレクション > に制限を加えて、ファイルが誤って破壊されないようにし、また >> リダイレクションが既存ファイルを参照するようにする。この変数が設定されていれば、ファイルに重ね書きができない。この変数は通常 <code>.cshrc</code> ファイルで設定される。
<i>notify</i>	バックグラウンド・プロセスが完了したことを知らせるかどうかを指定する。設定されている場合は、完了を通知する。設定されていない場合は、完了したことを通知しない。この変数は通常 <code>.cshrc</code> ファイルで設定される。
<i>path</i>	シェルがコマンドを見つけるために使用する探索パスを指定する。この変数は通常 <code>.login</code> ファイルで設定される。
<i>prompt</i>	C シェル・プロンプトのカスタマイズに使用できる。この変数は通常 <code>.cshrc</code> ファイルで設定される。
<i>shell</i>	プログラムが新しいサブシェルを作成するときに作成されるシェルを指定する。この変数は通常 <code>.login</code> ファイルで設定される。
<i>status</i>	最後に実行したコマンドがエラーなしで完了した (ゼロの値が返される) か、エラーが生じて完了した (非ゼロの値が返される) かを指定する。

8.2.7 組み込みコマンド

表 8-7 は、一般ユーザにとって重要な C シェルのコマンドを説明しています。C シェルの組み込みコマンドの完全な一覧については、`csh(1)` リファレンス・ページを参照してください。

表 8-7: 組み込み C シェル・コマンド

コマンド	説明
<code>alias</code>	別名定義を割り当てて、表示する。 ^a
<code>bg</code>	中断したプロセスをバックグラウンドに入れる。 ^b
<code>echo</code>	引数をシェルの標準出力に書き込む。
<code>fg</code>	現在バックグラウンドで実行しているプロセスをフォアグラウンドに入れる。 ^b
<code>history</code>	コマンド・ヒストリ・バッファの内容を表示する。 ^c
<code>jobs</code>	現在のバックグラウンド・プロセスのジョブ番号と PID 番号を表示する。 ^b
<code>logout</code>	ログイン・セッションを終了する。

表 8-7: 組み込み C シェル・コマンド (続き)

コマンド	説明
rehash	コマンド位置のハッシュ・テーブルを再計算するようシェルに指示する。シェルの探索パスにあるディレクトリにコマンドを追加したとき、シェルがそのコマンドを見つけられるようにしたい場合に、このコマンドを使用する。rehash を使用しない場合は、ハッシュ・テーブルが当初作成されたときにそのコマンドがディレクトリになかったため、そのコマンドは実行できない。
repeat	コマンドを指定回数だけ反復する。
set	シェル変数値を割り当てて、表示する。 ^d
setenv	環境変数値を割り当てる。 ^d
source	ファイル内のコマンドを実行する。これにより、現在のシェル環境を更新することができる。 ^e
time	指定コマンドの実行時間を表示する。
unalias	別名定義を削除する。 ^a
unset	変数に割り当てられていた値を削除する。 ^d
unsetenv	環境変数に割り当てられていた値を削除する。 ^d

^aalias および unalias コマンドについての詳細は、8.2.5 項を参照。
^bbg, fg, および jobs コマンドについての詳細は、第 6 章を参照。
^chistory コマンドについての詳細は 8.2.3 項を参照。
^dset, setenv, unset および unsetenv コマンドについての詳細は、第 7 章を参照。
^esource コマンドについての詳細は、8.2.5 項を参照。

8.3 Bourne シェルの機能

この節では、次の Bourne シェル機能について説明します。

- .profile ログイン・スクリプトの例
- メタキャラクタ
- 組み込み変数
- 組み込みコマンド

8.3.1 .profile ログイン・スクリプトの例

ログイン・シェルが Bourne シェルである場合，オペレーティング・システムは .profile ログイン・スクリプトを実行して，ユーザ環境をセットアップします。

エクスポートされる .profile ログイン・スクリプトの変数は，作成されるすべてのサブシェルおよびサブプロセスに渡されます。エクスポートされない変数は，ログイン・シェルだけで使用されます。

次の .profile ログイン・スクリプトでは，シェル変数を設定してエクスポートし，ログアウト・スクリプトのためにトラップを設定し，また情報を表示するようにシステムに指示します。表 8-8 で，スクリプトの各部分について説明します。

```
# Set PATH
PATH=/usr/bin:/usr/local/bin:
# Export global variables
export PATH
# Set shell variables
PS1='$LOGNAME $ '
CDPATH=.:.:$HOME
# Set up for logout script
trap "echo logout; $HOME/.logout" 0
# Display status information
date
echo "Currently logged in users:" ; users
```

表 8-8: Bourne シェルの .profile スクリプト例の説明

コマンド	説明
探索パスの設定	
<code>PATH=/usr/bin:/usr/local/bin:</code>	探索パスを指定する。この場合，まず /usr/bin が探索され，次に /usr/local/bin が探索される。
探索パスのエクスポート	
<code>export PATH</code>	実行するすべてのコマンドに探索パスが渡されるように指定する。
シェル変数の設定	

表 8-8: Bourne シェルの .profile スクリプト例の説明 (続き)

コマンド	説明
<code>PS1=' \$LOGNAME \$ '</code>	<code>PS1</code> 変数は Bourne シェル・プロンプトを指定する変数であり、その省略値は <code>\$</code> である。この変数代入は、プロンプトを <code>username \$</code> に変更するよう指定する。たとえば、ユーザ名が <code>amy</code> である場合には、プロンプトは <code>amy \$</code> になる。
<code>CDPATH=.:.::\$HOME</code>	<code>CDPATH</code> 変数は、 <code>cd</code> コマンドの探索パスを設定する。この変数代入では、 <code>cd</code> コマンドが、現在のディレクトリ (<code>.</code>)、親ディレクトリ (<code>..</code>)、ホーム・ディレクトリ (<code>\$HOME</code>) の順で、指定されたディレクトリを探索するように指定する。
ログイン・スクリプトのセットアップ	
<code>trap "echo logout; \$HOME/.logout" 0</code>	<code>trap</code> コマンドが終了信号 (0) を受け取ると、シェルが <code>logout</code> を表示して、 <code>.logout</code> スクリプトを実行するように指定する。 ^a
状態情報の表示	
<code>date</code>	日時を表示する。

^a`trap` コマンドについての詳細は、7.9.1 項を参照。

8.3.2 メタキャラクタ

表 8-9 は、Bourne シェルのメタキャラクタ (シェルにとって特別な意味を持つ文字) について説明しています。これらのメタキャラクタの意味は、シェル・スクリプト、ファイル名指定で使用される場合、別の文字の引用、入出力で使用される場合、あるいは、可変の置換を示すために使用される場合にグループ分けされます。

表 8-9: Bourne シェル・メタキャラクタ

メタキャラクタ	説明
構文	
<code> </code>	パイプラインを構成するコマンドを区切る。
<code>&&</code>	現在のコマンドが成功した場合に次のコマンドを実行する。

表 8-9: Bourne シェル・メタキャラクタ (続き)

メタキャラクタ	説明
	現在のコマンドが失敗した場合に次のコマンドを実行する。
;	順次実行するコマンドを区切る。
::	case 構造の構成要素を区切る。
&	コマンドをバックグラウンドで実行する。
()	別個のプロセスとしてサブシェルで実行するコマンドをグループ化する。
ファイル名	
/	ファイルのパス名の各部分を区切る。
?	先頭のドット (.) を除く任意の単一文字に対応する。
*	先頭のドット (.) を除く任意の文字列に対応する。
[]	[] 内のいずれかの文字に対応する。
引用符	
\	次の文字を本来の意味で、すなわち、シェルに対して特殊な意味を加えることなく解釈することを指定する。
'...'	'...'で囲まれた文字('を除く)をすべて本来の意味で、すなわち、シェルにとって特殊な意味を加えることなく解釈することを指定する。
"..."	特別な引用形式を提供する。\$ (ドル記号), ` (抑音符), および \ (バックスラッシュ) 文字はそれぞれの特殊な意味を保持するが, " " で囲まれた他の文字はすべて本来の意味に、すなわち、シェルに対する特殊な意味を加えることなく解釈することを指定する。二重引用符は変数代入を行う際に有用である。
入出力	
<	入力のリダイレクトする。
>	出力を指定ファイルにリダイレクトする。
<<	入力のリダイレクトし、シェルが入力を指定行まで読み取るように指定する。
>>	出力をリダイレクトし、シェルが出力をファイルの終わりに追加するように指定する。
2>	診断出力を指定ファイルにリダイレクトする。

表 8-9: Bourne シェル・メタキャラクタ (続き)

メタキャラクタ	説明
置換	
<code>\${...}</code>	変数置換を指定する。
<code>'...'</code>	コマンド出力置換を指定する。

8.3.3 組み込み変数

Bourne シェルには、値を代入できる変数が用意されています。シェルはこれらの変数のいくつかを設定し、ユーザは全部の変数を設定または再設定することができます。

表 8-10 に、一般ユーザにとって重要な Bourne シェルの組み込み変数を示します。Bourne シェルの組み込み変数に関する完全な情報については、sh(1b) リファレンス・ページを参照してください。

表 8-10: 組み込み Bourne シェル変数

変数	説明
HOME	ログイン・ディレクトリ、つまりログインが完了すると現在のディレクトリになるディレクトリの名前を指定する。cd コマンドは省略時の値として HOME の値を使用する。HOME は login コマンドで設定される。
PATH	システムがコマンドを探索して実行するディレクトリを指定する。シェルはここで指定された順序でこれらのディレクトリを探索する。通常、PATH 変数は .profile ファイルで設定される。
CDPATH	cd コマンドが cd に対して指定された引数を探索するディレクトリを指定する。cd コマンドの引数が空であるか、スラッシュ (/)、ドット (.)、またはドット・ドット (..) で始まる場合、CDPATH は無視される。通常、CDPATH は .profile ファイルで設定される。
MAIL	ユーザのメールを格納するファイルのパス名。ユーザは MAIL を設定する必要があり、これは通常 .profile ファイルで行う。
MAILCHECK	シェルがメールの有無をチェックする頻度を秒数で指定する (省略時の値は 600秒)。この変数の値を 0 に設定すると、シェルは各プロンプトを表示する前にメールの有無をチェックする。通常、MAILCHECK は .profile ファイルで設定される。
SHELL	省略時のシェルの指定する。この変数は .profile ファイルで設定されて、エクスポートされる。

表 8-10: 組み込み Bourne シェル変数 (続き)

変数	説明
PS1	省略時の Bourne シェル・プロンプトを指定する。省略時の値は \$。PS1 は通常 .profile ファイルで設定される。PS1 が設定されていない場合、シェルは標準一次プロンプト文字列を使用する。
PS2	二次プロンプト文字列を指定する。二次プロンプト文字列とは、ユーザがコマンド行を入力した後で、シェルがもっと多くの入力を必要とするときに表示する文字列である。標準二次プロンプト文字列は > の後にスペースが 1 つ続いたものである。PS2 は通常、ユーザの .profile ファイルで設定される。PS2 が設定されていない場合、シェルは標準二次プロンプト文字列を使用する。

8.3.4 組み込みコマンド

表 8-11 で、一般ユーザにとって重要な Bourne シェルのコマンドを示します。Bourne シェルの組み込みコマンドの完全な一覧については、sh(1b) リファレンス・ページを参照してください。

表 8-11: 組み込み Bourne シェル・コマンド

コマンド	説明
cd	ディレクトリの変更を可能にする。ディレクトリを指定しない場合には、HOME シェル変数の値が使用される。CDPATH シェル変数がこのコマンドの探索パスを定義する。 ^a
echo	標準出力に引数を書き込む。 ^b
export	指定された変数を、その後に実行されるコマンドの環境に自動的にエクスポートするためにマークする。
pwd	現在のディレクトリを表示する。 ^c
set	変数値を代入して表示する。 ^d
times	シェルから実行されたプロセスの累積ユーザおよびシステム時間を表示する。
trap	シェルが指定信号を受け取ると、指定コマンドを実行する。 ^d
umask	新しく作成されたすべてのファイルについて、差し引く許可を指定する。 ^e
unset	変数に代入されていた値を削除する。 ^d

^acd コマンドについての詳細は、第 4 章および sh(1) リファレンス・ページを参照。
^becho コマンドについての詳細は、8.3.1 項 および sh(1) リファレンス・ページを参照。
^cpwd コマンドについての詳細は、第 2 章を参照。
^dset, trap, および unset コマンドについての詳細は、第 7 章を参照。

表 8-11: 組み込み Bourne シェル・コマンド (続き)

^eumask コマンドについての詳細は、第 5 章および 8.2.1 項を参照。

8.4 Korn または POSIX シェルの機能

POSIX シェルは、IEEE POSIX.2 規格準拠を示す Korn シェルのもう 1 つの指名字です。この節では、次の Korn または POSIX シェルの機能について説明します。

- `.profile` および `.kshrc` ログイン・スクリプトの例
- メタキャラクタ
- コマンド・ヒストリ
- コマンド行編集
- ファイル名補完
- 別名
- 組み込み変数およびコマンド

8.4.1 `.profile` および `.kshrc` ログイン・スクリプトの例

ログイン・シェルが Korn または POSIX シェルである場合、オペレーティング・システムはホーム・ディレクトリにある `.profile` ログイン・スクリプトを処理します。`.profile` ログイン・スクリプトは環境変数を定義します。これらの変数は、ログイン・シェルで使用されるとともに、作成されたサブシェルやサブプロセスでも使用されます。`.profile` ログイン・スクリプトは、ログインしたときにだけ実行されます。

`.kshrc` ログイン・スクリプトは、ローカル・シェル・プロセスのための変数とオペレーティング・パラメータを定義することによって、Korn または POSIX シェル 環境をセットアップします。このログイン・スクリプトは、サブシェルを作成するたびに実行されます。

注意

ホーム・ディレクトリに `.kshrc` ファイルを作成する前に、`.profile` で `ENV=$HOME/.kshrc` 環境変数が設定されて、エクスポートされていることを確認してください。1 度これを行う

と、.kshrc ログイン・スクリプトはログイン時とサブシェルの作成時に必ず実行されます。

次の .profile ログイン・スクリプトでは、グローバル環境変数を設定してエクスポートするとともに、シェル変数を設定します。表 8-12 でスクリプトの各部分について説明しています。

```
# Set environment variables
PATH=/usr/bin:/usr/local/bin:
ENV=$HOME/.kshrc
EDITOR=vi
FCEDIT=vi
PS1="'hostname' [!] $ "

# Export global variables
export PATH ENV EDITOR FCEDIT PS1

# Set mail variables
MAIL=/usr/spool/mail/$LOGNAME
MAILCHECK=300
```

表 8-12: Korn または POSIX シェルの .profile ログイン・スクリプトの例の説明

コマンド	説明
環境変数の設定	
PATH=/usr/bin:/usr/local/bin	探索パスを指定する。この場合、まず /usr/bin を探索し、次に /usr/local/bin を探索する。
ENV=\$HOME/.kshrc	\$HOME/.kshrc をログイン・スクリプトとして指定する。
EDITOR=vi	vi を、シェル・プロンプトにおけるコマンド行編集とファイル名補完のための省略時のエディタとして指定する。
FCEDIT=vi	vi を、fc コマンドの省略時のエディタとして指定する。 ^a

表 8-12: Korn または POSIX シェルの .profile ログイン・スクリプトの例の説明 (続き)

コマンド	説明
PS1="'hostname' [!] \$ "	PS1 変数は Korn または POSIX シェルのプロンプトを指定する変数で、その省略時の値は \$ である。この変数を代入すると、プロンプトを変更して、hostname コマンドの出力の後に現在のコマンドのコマンド番号とドル記号 (\$) を続けて表示する。たとえば、システム名が boston で、現在のコマンドの番号が 30 である場合、プロンプトは boston[30] \$ となる。
グローバル変数のエクスポート	
export PATH ENV EDITOR FCEDIT PS1	PATH, ENV, EDITOR, FCEDIT, および PS1 の変数の値をすべてのサブシェルにエクスポートするように指定する。
メール変数の設定	
MAIL=/usr/spool/mail/\$LOGNAME	メール・システムが新たなメールの到着を検出するために使用するファイルのパス名を指定する。この場合、メール・システムは /usr/spool/mail ディレクトリの下にあるユーザ名のサブディレクトリを探索する。
MAILCHECK=300	シェルが 300 秒 (5 分) ごとにメールをチェックすることを指定する。

^afc コマンドについての詳細は、8.4.4 項を参照。

次の .kshrc ログイン・スクリプトでは、シェル変数、コマンド別名、およびコマンド・ヒストリ変数が設定され、作成されたファイルに対する許可も設定されます。表 8-13 でスクリプトの各部分について説明します。

```
# Set shell variables
set -o monitor
set -o trackall

# Set command aliases
alias rm='rm -i '
alias rename='mv '
alias l 'ls -l'
alias c clear

# Set history variables
HISTSZ=40
```

```
# Set file creation permissions
umask 027
```

表 8-13: .kshrc ログイン・スクリプトの例の説明

コマンド	説明
シェル変数	
set -o monitor	シェルがすべてのバックグラウンド・プロセスをモニタし、プロセス終了時に完了メッセージを表示するように指定する。
set -o trackall	ユーザの実行するすべてのコマンドをシェルが追跡するように指定する。コマンドを追跡すると、シェルはコマンドの位置を保存して、次にそのコマンドを入力するとき迅速に見つけてくれる。
コマンド別名	
alias rm='rm -i'	rm コマンドで、ファイル削除の確認を求めるプロンプトを表示する -i オプションの使用を指定する。
alias rename='mv'	rename を mv コマンドの新しい名前として指定する。
alias l='ls -l'	ディレクトリ・ファイルを詳細なフォーマットでリストする ls -l コマンドの簡略名を定義する。
alias c='clear'	画面をクリアする clear コマンドの簡略名を定義する。
ヒストリ変数	
HISTSIZE=40	最後の 40 個のコマンドをヒストリ・バッファに格納するようにシェルに指示する。
ファイル許可の設定	
umask 027	新しく作成されたすべてのファイルについての許可を指定する。このコマンドは所有者に対してはすべての許可を与え、同じグループのメンバに対しては読み取りおよび実行の許可を与え、それ以外のすべてのユーザに対してはどんな許可も与えない。umask はサブシェルへは継承されない。

8.4.2 メタキャラクタ

表 8-14 は Korn または POSIX シェルのメタキャラクタについて説明したものです。メタキャラクタとは、シェルにとって特別な意味を持つ文字のこと

です。これらのメタキャラクタの意味は、シェル・スクリプト、ファイル名指定で 사용되는場合、別の文字の引用、入出力で 사용되는場合、あるいは、可変の置換を示すために使用される場合にグループ分けされます。

表 8-14: Korn または POSIX シェルのメタキャラクタ

メタキャラクタ	説明
構文	
	パイプラインを構成するコマンドを区切る。
&&	現在のコマンドが成功した場合に次のコマンドを実行する。
	現在のコマンドが失敗した場合に次のコマンドを実行する。
;	順次実行するコマンドを区切る。
::	case 構造の構成要素を区切る。
&	コマンドをバックグラウンドで実行する。
()	別個のプロセスとしてサブシェルで実行するコマンドをグループ化する。
{ }	サブシェルを作成せずにコマンドをグループ化する。
ファイル名	
/	ファイルのパス名の各部分を区切る。
?	語頭のドット (.) を除く任意の単一文字に対応する。
*	語頭のドット (.) を除く任意の文字列に対応する。
[]	[] 内のいずれかの文字に対応する。
~	ファイル名の先頭に用いると、ホーム・ディレクトリを指定する。
引用符	
\	次の文字を本来の意味で、つまり、シェルに対する特殊な意味を加えることなく解釈することを指定する。
'...'	' ' で囲まれた文字 (' を除く) を本来の意味で、つまり、シェルに対する特殊な意味を加えることなく解釈することを指定する。
"..."	特別な引用形式を提供する。ドル記号 (\$), 抑音符 (^), バックスラッシュ (\), および右カッコ]) 文字はそれぞれの特殊な意味を保持するが, " " で囲まれた他のすべての文字は本来の意味に、つまり、シェルにとって特殊な意味を加えることなく解釈することを指定する。二重引用符 (" ") は変数代入を行う際に有用である。
入出力	

表 8-14: Korn または POSIX シェルのメタキャラクタ (続き)

メタキャラクタ	説明
<	入力のリダイレクトする。
>	出力を指定ファイルにリダイレクトする。
<<	入力のリダイレクトし、シェルが入力を指定された行まで読み取することを指定する。
>>	出力をリダイレクトし、シェルが出力をファイルの終わりに追加することを指定する。
>&	診断および標準出力の両方をリダイレクトして、ファイルに付加する。
置換	
\${...}	変数置換を指定する。
%	ジョブ番号の置換を指定する。
'...'	コマンド出力の置換を指定する。

8.4.3 コマンド・ヒストリ

コマンド・ヒストリ・バッファは、入力されたコマンドを格納し、いつでもそのコマンドを表示できるようにします。このため、以前に入力したコマンドやその一部を選択して、再実行することができます。この機能を利用すれば、長いコマンドを再入力しなくても再使用できるため、時間の節約になります。

コマンド・ヒストリ・バッファの内容を見るときには、history コマンドを使用します。すると、例 8-1 ような出力が表示されます (実際の出力は異なります)。

例 8-1: ksh ヒストリの出力例

```
[18] $ history
3 ls -l
4 pwd
5 cd /usr/sales
6 ls -l
7 cp report report5
8 mv /usr/accounts/new .
9 cd /usr/accounts/new
10 mkdir june
11 cd june
```

例 8-1: ksh ヒストリの出力例 (続き)

```
12 mv /usr/accounts/new/june .
13 ls -l
14 cd /usr/sales/Q1
15 vi earnings
16 cd /usr/chang
17 vi status
[19] $
```

コマンド・ヒストリ・バッファ内のコマンドを再実行するには、表 8-15 に示すコマンドを使用します。各コマンドは英字 `r` で始まっていることに注意してください。

表 8-15: ヒストリ・バッファ内のコマンドの再実行

コマンド	説明
<code>r</code>	直前のコマンドを再実行する。
<code>r n</code>	<code>n</code> で指定したコマンドを再実行する。たとえば、前述の例で示したヒストリ・バッファの場合、 <code>r 5</code> によって <code>cd /usr/sales</code> コマンドが実行される。
<code>r -n</code>	現在のコマンドに相対して以前のコマンドを再実行する。たとえば、前述の例で示したヒストリ・バッファの場合、 <code>r -2</code> によってコマンド番号 16 の <code>cd /usr/chang</code> コマンドが呼び出される。
<code>r string</code>	<code>string</code> によって指定される文字列で始まる最後に実行されたコマンドを再実行する。たとえば、前述の例で示したヒストリ・バッファの場合、 <code>r cp</code> によってコマンド番号 7 の <code>cp report report5</code> が呼び出される。

ヒストリ・バッファ・コマンドの再実行についての詳細は、`ksh(1)` リファレンス・ページを参照してください。

ヒストリ・バッファに格納するコマンドの数を増減したい場合は、`.profile` ファイルで `HISTSIZE` 変数を設定します。この変数の形式は次のとおりです。

HISTSIZE= `n`

`n` には、ヒストリ・バッファに格納するコマンド行数を指定します。

たとえば、履歴・バッファに 15 個のコマンドを格納するには、次のコマンドを使用します。

```
HISTSIZE=15
```

Korn または POSIX シェルでは、現在のコマンド行を編集するだけでなく、コマンド・履歴・バッファに格納されているコマンドを再使用することもできます。この機能を使用するには、`vi` や `emacs` のようなテキスト・エディタの使用方法を知らなければなりません。これらの機能についての詳細は、次の項を参照してください。

8.4.4 `fc` コマンドを使用したコマンド行編集

Korn または POSIX シェルでは、コマンド・履歴・バッファ内のコマンド行を表示したり、編集したり、あるいはその両方を行うことができます。このため、以前に実行したコマンド行の一部を修正してから、そのコマンドを再実行することができます。

Korn または POSIX シェルのコマンド行編集機能は広範囲にわたっています。この項では、最も基本的な機能だけを取り上げます。詳細については、`ksh(1)` または `sh(1p)` リファレンス・ページを参照してください。

コマンド・履歴・バッファを表示したり、その内容を編集したり、あるいはその両方を行うには、組み込みコマンドの `fc` (fix command) を使います。`fc` コマンドには、次の 2 つの形式があります。

```
fc [-e editor] [-nlr] [first] [last]
```

このコマンド形式を使用すると、バッファ内のコマンド行をいくつでも表示して編集することができます。

- `-e editor` 変数には、コマンド行の編集に使用したいエディタ (通常は `vi` または `emacs`) を指定します。`-e` を指定しなければ、`fc` コマンドはコマンド行を表示しますが、それを編集することはできません。
- `-n` フラグは、バッファ内のコマンド行を、番号を付けずに表示することを指定します。バッファ内のコマンド行を、番号を付けて表示することを指定するには、`-l` フラグを使用します。行番号や行番号の範囲を指定しない場合は、入力した最後の 16 行が表示されます。
- `-r` フラグは、バッファ内のコマンド行を逆順で表示することを指定します。

- *first* および *last* 変数には、バッファ内のコマンド行の範囲を指定します。指定は番号でも文字列でも行うことができます。

-e フラグに対して省略時のエディタを指定したい場合は、`.profile` スクリプトで `FCEDIT` 変数を定義します。たとえば、`emacs` を省略時のエディタにする場合は、次の変数定義を入力してください。

```
FCEDIT=emacs
```

fc -e - [old=new] [string]

このコマンド形式を使用すると、以前に入力した任意のコマンド行内の *old* 文字列を直ちに *new* 文字列に置き換えることができます。

- `-e -` は、置換を行うことを指定します。
- `old=new` は、*old* 文字列を直ちに *new* 文字列に置き換えることを指定します。
- *string* は、Korn または POSIX シェルが *string* を含むバッファ内の一番新しいコマンド行に対して編集を行うことを指定します。

次の項では、`fc` を使用する例をいくつか紹介します。

Korn または POSIX シェルでは、シェル・プロンプトで、`vi` または `emacs` エディタに類似したコマンド・セットを使用して、個々のコマンド行を編集することもできます。この機能についての詳細は、`ksh(1)` または `sh(1p)` リファレンス・ページを参照してください。

8.4.4.1 コマンド行編集の例

コマンド行 15 から 18 までを表示するには、次のように入力します。

```
$ fc -l 15 18
15 ls -la
16 pwd
17 cd /u/ben/reports
18 more sales
$
```

同じコマンド行を、行番号ではなくコマンド文字列を指定することによって、表示することもできます。次にその例を示します。

```
$ fc -l ls more
15 ls -la
16 pwd
```

```
17 cd /u/ben/reports
18 more sales
$
```

vi エディタを使用して、コマンド行 15 から 18 までを表示して編集するには、次のように入力します。

```
$ fc -e vi 15 18
ls -la
pwd
cd /u/ben/reports
more sales
~
~
~
~
```

編集を終えたら、:wq! コマンドを使用して、ファイルを書き込んで終了します。ファイル内のそのコマンド行は再実行することができます。

echo hello コマンドを入力した直後、hello を goodbye に置き換えたいとします。置換を行って、コマンド行を再実行するには、次のように入力します。

```
$ echo hello
hello
$ fc -e - hello=goodbye echo
echo goodbye
goodbye
```

fc コマンドとコマンド行編集についての詳細は、ksh(1) リファレンス・ページを参照してください。

8.4.5 ファイル名補完

Korn または POSIX シェルでは、シェル・プロンプトに対してファイル名またはパス名の一部を入力すると、シェルが自動的にその名前を照合して補完します。基準と一致するファイル名またはパス名が2つ以上存在する場合、シェルは基準と一致するものをすべて表示します。

ファイル名補完メカニズムを使用するには、.profile ファイルで EDITOR 変数を定義します。たとえば、vi エディタを使用したい場合には、.profile ファイルに次の変数定義を入力してください。

```
EDITOR=vi
```

ファイル名補完がどのように行われるかを具体的に示すため、使用しているエディタが vi であり、現在のディレクトリに salesreportjan、

salesreportfeb, salesreportmar という 3 つのファイルがあるとして
ます。ls -l コマンドでファイル補完機能を使用するには、次のように入
力します。

```
$ ls -l salesreport Escape =  
1) salesreportfeb  
2) salesreportjan  
3) salesreportmar  
$ ls -l salesreport
```

システムは入力されたコマンドを再表示し、カーソルは salesreport の
終わりにあります。ここで salesreportjan を選びたいとします。その
場合には、a (vi の append コマンド) とタイプした後に、jan と入力して
Return キーを押してください。すると、salesreportjan の詳細なリス
トが表示されます。

ファイル名補完についての詳細は、ksh(1) および sh(1p) リファレンス・
ページを参照してください。

8.4.6 別名

コマンド別名機能を使用すると、長いコマンド行を短縮したり、コマンドの
名前を変更することができます。これは、頻繁に使用するコマンド行に対し
て別名を作成することにより行います。

たとえば、頻繁にディレクトリ /usr/chang/reports/status に移動す
る必要があるとします。その場合には、別名 status を作成して、コマン
ド行にその別名を入力すれば、いつでもそのディレクトリに移動できる
ようにします。

さらに、別名を使用すると、コマンドに対してもっとわかりやすい名前を
付けることができます。たとえば、mv コマンドに対して rename という
別名を定義することもできます。

別名を作成するには、alias コマンドを使用します。alias コマンドの一
般形式は次のとおりです。

alias *aliasname= command*

aliasname には、使用したい名前を指定します。*command* には、元のコ
マンド、あるいは一連のコマンドを指定します。*command* が複数の部分
に分かれている (間にスペースがある) 場合は、式全体を一重引用符で囲
んでください。

たとえば、`/usr/chang/reports/status` というディレクトリに移動するための別名 `status` を作成するには、次のコマンドを入力します。

```
alias status='cd /usr/chang/reports/status'
```

別名の定義は、通常 `.kshrc` ファイルに入れておくことによって行います。こうしておくと、ログインしたり、新しいシェルを開始するたびに、その別名を使用することができます。例については、8.4.1 項を参照してください。

すべての別名定義を表示するには、次のコマンドを入力します。

```
$ alias
```

特定の別名定義を表示するには、次のように入力します。

```
% alias aliasname
```

`aliasname` には、定義を表示させる特定の別名を指定します。

Korn または POSIX シェルでは、ユーザが作成した別名をエクスポートすることができます。エクスポートした別名は、作成されるすべてのサブシェルにも渡されるため、シェル・プロシージャや新しいシェルを実行するときに、定義された別名を使用することができます。エクスポートしていない別名は、ログイン・シェルでしか使用することができません。

別名をエクスポートするには、次の形式の `alias` コマンドを使用します。

```
alias -x aliasname=command
```

`-x` フラグは別名をエクスポートすることを示します。`aliasname` には、使用したい名前を指定します。`command` には、元のコマンド、あるいは一連のコマンドを指定します。`command` が複数の部分に分かれている (間にスペースがある) 場合は、式全体を一重引用符で囲んでください。

たとえば、`rm` コマンドの別名定義をエクスポートするには、次のように入力します。

```
alias -x rm='rm -i '
```

このコマンドは、次の2つのいずれかの方法で入力することができます。

- ログインするたびに別名をエクスポートしたい場合は、`.kshrc` または `.profile` ファイルを編集する。
- 現在のログイン・セッションでのみ別名をエクスポートしたい場合は、コマンド行で別名をエクスポートする。

現在のログイン・セッションについて別名を削除するには、`unalias` コマンドを使用します。`unalias` コマンドの一般形式は次のとおりです。

unalias *aliasname*

aliasname には、削除したい別名を指定します。

現在および将来のログイン・セッションのすべてについて別名を削除するには、次のようにします。

1. 次のコマンドを入力する。

```
$ unalias aliasname
```

aliasname には、削除したい別名を指定します。

2. `.kshrc` ファイル (または、別名定義を含むファイル) を編集して、その別名定義を削除した後、ファイルを保存する。
3. 次のコマンドを入力して、`.kshrc` ファイルを再実行する。

```
$ . ~/.kshrc
```

Korn または POSIX シェルは、この他にも別名機能を備えています。Korn または POSIX シェルにおける別名使用についての詳細は、`ksh(1)` または `sh(1p)` リファレンス・ページを参照してください。

8.4.7 組み込み変数

Korn および POSIX シェルには、値を代入できる変数が用意されています。シェルはこれらの変数のいくつかを設定し、ユーザは全部の変数を設定または再設定することができます。

表 8-16 に、一般ユーザにとって重要な Korn または POSIX シェルの組み込み変数を示します。Korn または POSIX シェルの組み込み変数に関する完全な情報については、`ksh(1)` または `sh(1p)` リファレンス・ページを参照してください。

表 8-16: 組み込み Korn または POSIX シェル変数

変数	説明
<i>HOME</i>	ログイン・ディレクトリ名を指定する。cd コマンドは省略時の値として <i>HOME</i> の値を使用する。Korn または POSIX シェル・プロシージャでは、 <i>HOME</i> を使用すると、完全パス名を使用しなくてもすむ。これは、ログイン・ディレクトリのパス名を変更する際に特に有用である。 <i>HOME</i> 変数は login コマンドで設定される。
<i>PATH</i>	システムがコマンドを探索して実行するディレクトリを指定する。シェルはここで指定された順序でこれらのディレクトリを探索する。通常、 <i>PATH</i> は .profile ファイルで設定される。
<i>CDPATH</i>	cd コマンドが cd に対して指定された引数を探索するディレクトリを指定する。cd の引数が空であるか、スラッシュ (/), ドット (.), またはドット・ドット (..) で始まる場合、 <i>CDPATH</i> は無視される。通常、 <i>CDPATH</i> は .profile ファイルで設定される。
<i>MAIL</i>	ユーザのメールを格納するファイルのパス名。 <i>MAIL</i> は通常 .profile ファイルで設定される。
<i>MAILCHECK</i>	シェルがメールの有無をチェックする頻度を秒数で指定する (省略時の値は 600 秒)。この変数の値を 0 に設定すると、シェルは各プロンプトを表示する前にメールの有無をチェックする。通常、 <i>MAILCHECK</i> は .profile ファイルで設定される。
<i>SHELL</i>	省略時のシェルを指定する。この変数は .profile ファイルで設定されて、エクスポートされる。
<i>PS1</i>	省略時の Korn または POSIX シェル・プロンプトを指定する。省略時の値は \$。 <i>PS1</i> 変数は通常 .profile ファイルで設定される。
<i>PS2</i>	二次プロンプト文字列を指定する。二次プロンプト文字列とは、ユーザがコマンド行を入力した後で、シェルがもっと多くの入力が必要とするときに表示する文字列である。標準二次プロンプト文字列は > の後にスペースが続いたものである。 <i>PS2</i> 変数は通常 .profile ファイルで設定される。
<i>HISTFILE</i>	コマンド・ヒストリの格納に使用するファイルのパス名を指定する。この変数は通常 .profile ファイルで設定される。
<i>EDITOR</i>	シェル・プロンプトでのコマンド行編集とファイル名補完のための省略時のエディタを指定する。この変数は通常 .profile ファイルで設定される。

表 8-16: 組み込み Korn または POSIX シェル変数 (続き)

変数	説明
FCEDIT	fc コマンドの省略時のエディタを指定する。この変数は通常 .profile ファイルで設定される。
HISTSIZE	このシェルでアクセス可能な以前に入力したコマンドの数を指定する。省略時の値は 128。この変数は通常 .kshrc ファイルで設定される。

8.4.8 組み込みコマンド

表 8-17 に、一般ユーザにとって重要な Korn または POSIX シェルのコマンドを示します。シェルの組み込みコマンドの完全な一覧、または、リストされているコマンドの詳細については、ksh(1) または sh(1p) リファレンス・ページを参照してください。これらのコマンドのほとんどには、1.6.1 項で説明している方法でアクセスできるリファレンス・ページもあります。

表 8-17: 組み込み Korn または POSIX シェル・コマンド

コマンド	説明
alias	別名定義を代入して表示する。alias コマンドについての詳細は、8.4.6 項を参照。
cd	ディレクトリの変更を可能にする。ディレクトリを指定しなければ、HOME シェル変数の値が使用される。CDPATH シェル変数がこのコマンドの探索パスを定義する。cd コマンドについての詳細は、第 4 章 および csh(1) リファレンス・ページを参照。
echo	標準出力に引数を書き込む。
export	指定された変数を、その後に実行されるコマンドの環境に自動的にエクスポートするためにマークする。export コマンドについての詳細は、8.4.1 項を参照。
fc	コマンド・ヒストリ・バッファの内容を表示、編集、および再実行できる。fc コマンドについての詳細は、8.4.4 項を参照。
history	コマンド・ヒストリ・バッファの内容を表示する。history についての詳細は、8.4.6 項を参照。
jobs	現在のバックグラウンド・プロセスのジョブ番号と PID 番号を表示する。jobs コマンドについての詳細は、6.4.1 項を参照。

表 8-17: 組み込み Korn または POSIX シェル・コマンド (続き)

コマンド	説明
pwd	現在のディレクトリを表示する。pwd コマンドについての詳細は、第 2 章を参照。
set	変数値を代入して表示する。set コマンドについての詳細は、第 7 章を参照。
times	シェルから実行されたプロセスの累積ユーザおよびシステム時間を表示する。
trap	シェルが指定信号を受け取ると、指定コマンドを実行する。trap コマンドについての詳細は、第 7 章を参照。
umask	新しく作成されたすべてのファイルに対し、作成するプログラムによって設定される省略時の許可から、差し引く許可を指定する。umask コマンドについての詳細は、第 5 章および 8.4.1 項を参照。
unalias	別名定義を削除する。unalias コマンドについての詳細は、8.4.6 項を参照。
unset	変数に代入されていた値を削除する。unset コマンドについての詳細は、第 7 章を参照。

System V 動作環境の使用

この章では、System V 環境、コマンド、サブルーチン、システム・コールについて説明しています。この章で説明しているコマンドによって、次のことが可能になります。

- System V 動作環境の設定 (9.1 節)
- System V 動作環境へのアクセス (9.2 節)
- System V 動作コマンドの使用 (9.4 節)

System V 動作環境は、SVID (System V Interface Definition) に定義されている Base System および Kernel Extension のすべてのコンポーネントに関して、ソース・コード・インタフェースおよび実行時動作をサポートする代替バージョンのコマンド、サブルーチン、およびシステム・コールで構成されています。この System V 環境の実装により、すべての SVID 2 の機能と SVID 3 の機能がサポートされます。System V 環境には、すでに SVID の要件を満たしているシステム標準のコマンド、サブルーチン、およびシステム・コールの代替バージョンは含まれません。

System V 環境を設定すると、システム標準のコマンドおよび関数の代りに、対応する System V のコマンドおよび関数 (システム・コールおよびサブルーチン) が使用されるようになります。System V の環境には次の 2 つの方法でアクセスすることができます。

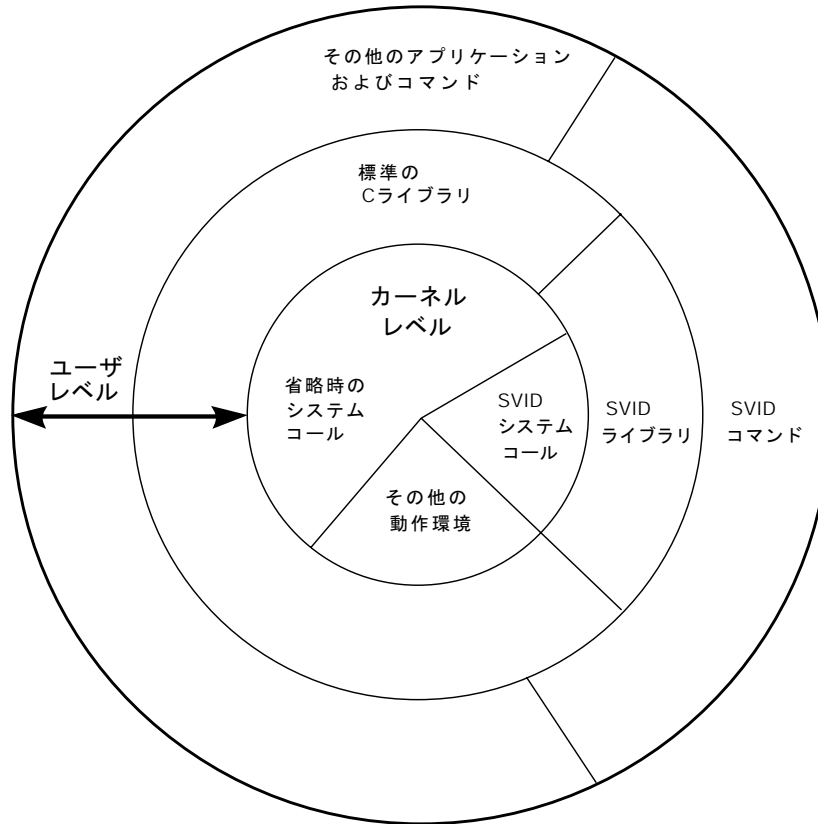
- System V コマンドおよびライブラリの絶対パス名を指定する。
- System V 環境のコマンドが、システム標準のコマンドより前に検索されるように、PATH 環境変数を定義する。

PATH 環境変数を設定するには、9.1 節の説明に従って、`.profile` または `.login` および `.cshrc` ファイルを変更します。

System V のシステム・コールはシステム標準のシステム・コールに依存してないので、System V のシステム・コールを使用して構築されたアプリケーションを実行しても、性能上の障害は発生しません。

図 9-1 はオペレーティング・システム内の System V 機能の位置付けを示した概念図です。この図から、System V のシステム・コールは、カーネル・レベルにあることがわかります。

図 9-1: System V 機能



ZK-0849U-AIJ

以降の各節で、System V 環境にアクセスするための設定方法と、機能について説明します。

9.1 環境の設定

ログイン時に自動的に System V の環境にアクセスするには、Bourne、Korn、または POSIX シェルを使用している場合は `.profile` ファイルに、C シェルを使用している場合は `.login` および `.cshrc` ファイルにそれぞれコマンド行を追加します。このコマンド行は `PATH` 環境変数を変更し、`/bin` または `/usr/bin` 等のシステム標準のコマンドが格納されている

9-2 System V 動作環境の使用

ディレクトリよりも先に、System V のコマンドが格納されているディレクトリが検索されるように設定します。次のような System V 環境設定スクリプトが利用できます。

- /etc/svid2_profile ファイル (Bourne, Korn, または POSIX シェル用) および /etc/svid2_login ファイル (C シェル用)
それぞれ .profile または .login および .cshrc ファイルに追加すると、SVID 2 の動作を実行します。
- /etc/svid3_login ファイル (Bourne, Korn, または POSIX シェル用) および /etc/svid3_profile ファイル (C シェル用)
それぞれ .profile または .login および .cshrc ファイルに追加すると、SVID 3 の動作を実行します。

たとえば、Bourne, Korn, または POSIX シェルを使用し、SVID 2 の機能を指定する場合、.profile ファイルに次の行を追加します。

```
if [ -f /etc/svid2_profile ]
then
    . /etc/svid2_profile
fi
```

C シェルを使用し、SVID 2 の機能を指定する場合、.login および .cshrc ファイルに次の行を追加します。

```
if ( -e /etc/svid2_login ) then
    source /etc/svid2_login
endif
```

ピリオド (.) と source コマンドはシェル固有です。シェル・コマンドについての詳細は、該当するシェルのリファレンス・ページを参照してください。

9.2 環境設定スクリプトによる PATH 環境変数の設定

SVID 2 の動作を指定するスクリプトまたは SVID 3 の動作を指定するスクリプトのいずれを選択しても、次のように System V 環境を設定します。

- System V のみの変数 SVID2PATH を /etc/svid2path または /etc/svid3path の内容に定義する。
/etc/svid2path および /etc/svid3path ファイルには、それぞれ SVID 2 および SVID 3 のパスが定義されています。
- PATH の先頭に SVID2PATH/bin および SVID2PATH/sbin を追加する。

- SVID2PATH および PATH をエクスポートする。

- TZ 変数を適切な値に設定する。

SVID 2 の機能を選択した場合にこの値を設定すると TZC 変数が適切な値に設定されます。タイム・ゾーン・フォーマットの詳細については、『システム管理ガイド』を参照してください。

以後、System V コマンドが格納されているディレクトリ名は、
/etc/svid2path または /etc/svid3path ファイルの内容を cat(1) コマンドで表示して確認することができます。

System V 環境設定スクリプトを使用して PATH 環境変数を変更すると、システム管理者が各ユーザの .profile または .login および .cshrc ファイルを変更しなくても、System V 環境のパスが変更できます。システム管理者は /etc/svid2path および /etc/svid3path ファイルを変更するだけで、グローバル定義を行うことができます。

System V 環境設定スクリプトが PATH を設定する方法については、次に示す .profile を参照してください。これは、SVID 2 用の System V 環境設定スクリプトを指定します。

```
stty erase DEL kill ^U intr ^C quit ^X echo
TERM=vt100
PATH=$HOME/bin:/usr/lib:/bin
MAIL=/usr/mail/$LOGNAME
EDITOR=vi
export MAIL PATH TERM EDITOR
if [ -f /etc/svid2_profile ]
then
    . /etc/svid2_profile
fi
```

この例では、/etc/svid2path にある System V コマンドのパスが /usr/opt/s5 で、ログイン・ディレクトリが /usr/users/xxx であるとして、前述の .profile ファイルを使用してログインした後、PATH を表示すると、次のように System V コマンドへのパスが .profile ファイルの 3 行目の PATH 設定の先頭に追加されて表示されます。

```
% echo $PATH Return
/usr/opt/s5/bin:/usr/opt/s5/sbin:/usr/users/xxx/bin:/usr/lib:/bin
```

以後、シェル・コマンドを発行すると System V のコマンドを格納しているディレクトリが最初に検索されます。このパスにコマンドが見つからない場合は指定されたパスが順次検索されます。

9.3 シェル・スクリプトの互換性

シェルの `PATH` 環境変数を変更することによって、シェル・スクリプトの互換性が達成されます (9.1 節で説明)。このため、System V のコマンドのパスがシステム標準のコマンドのパスより先に検索されます。`PATH` 環境変数が System V 環境用に設定されていると、C シェルまたは Bourne、Korn、または POSIX シェルのいずれを使用している場合でも、シェル・スクリプトは System V との互換性を持ちます。

9.4 System V のコマンド概略

表 9-1 はシステム標準のコマンドとはオプションまたは機能が異なる System V のユーザ・コマンドの機能の概略です。詳細については、各コマンドのリファレンス・ページを参照してください。

表 9-1: ユーザ・コマンドのまとめ

コマンド	System V 動作
<code>chmod(1)</code>	シンボリック・モードで保護モードを変更する場合、許可を与えるユーザ群 (u, g, o) を省略すると <code>umask</code> を無視し、a (すべてのユーザ) を指定した場合と同じ動作をします。
<code>df(1)</code>	<code>-t</code> オプションを指定すると、ディスク領域の合計が表示されます。このとき、オプションでファイル・システム名またはデバイス名を指定できます。
<code>ln(1)</code>	<code>-f</code> オプションを指定すると、指定されたリンクを作成する前に既存のデスティネーションのパス名が削除されます。
<code>ls(1)</code>	<code>-C</code> オプションが指定されている場合に限りマルチ・カラムの出力を行います。また <code>-s</code> オプションが指定されている場合、通常の 1024 バイト単位ではなく、512 バイト単位でファイルのサイズを出力します。
<code>mailx(1)</code> および <code>Mail(1)</code>	System V の <code>mailx</code> コマンドの機能をインクルードします。
<code>sum(1)</code>	省略時は語単位のアルゴリズムを使用し、 <code>-r</code> オプションが指定された場合にバイト単位のアルゴリズムを使用します。System V の <code>sum</code> コマンドでは、省略時の <code>checksum</code> アルゴリズムがシステム標準の <code>sum</code> コマンドのものと反対になります。
<code>tr(1)</code>	<code>-c</code> オプションを指定した場合、必ず <code>-A</code> オプションをインクルードします。 <code>-A</code> オプションは ASCII コードが 8 進数の 1 から 377 の文字だけを対象にします。



ネットワーク・ユーザおよびホストに関する情報の収集

この章では、通信やファイル転送処理に必要なローカルおよびリモートのユーザまたはホストに関する情報を検索する方法について説明します。この章で説明するコマンドを使用すると、次のことを行うことができます。

- 自身のネットワーク接続に関する情報の取得 (`who am i` コマンド, 10.1 節)
- 現在ローカル・システムにログインしているユーザ, およびそのユーザがログインした端末の確認 (`who` コマンド, 10.1 節)
- 別のユーザに関する追加情報 (たとえば, フル・ネーム, オフィスの所在地, 電話番号, プロジェクト, 計画など) の検索 (`finger` コマンド, 10.2.1 項)
- `talk` コマンドまたは `write` コマンドで相手の端末にアクセスできるかどうかの判定 (`finger` コマンド, 10.2.1 項および 10.2.2 項)
- リモート・ホストの使用状況に関する情報の分析およびソート (`ruptime` コマンド, 10.3 節)
- 現在リモート・ホストにログインしているユーザの確認 (`rwho` コマンド, 10.4 節)
- リモート・ホストがオンラインであるかどうかの判定 (`ping` コマンド, 10.5 節)

注意

この章で説明するコマンドは、すべての TCP/IP オペレーションと同様に、ローカル・ホストおよびリモート・ホストのセキュリティ機能の影響を受けます。この章や関連するリファレンス・ページの説明のとおりコマンドが動作しない場合には、システム管理者に問い合わせてください。

10.1 ローカル・ホスト上のユーザの確認

ユーザ名とパスワードを入力してホスト・コンピュータにログインすると、ユーザには固有の ID が割り当てられます。接続しているネットワークに関してこの情報を確認するには、`who am i` という `who` コマンドを使用してください。このコマンドを実行すると、次に示すユーザ情報が表示されます。

- ログイン名
- 端末名 (回線)
- ログインした時刻
- ネットワークに接続しているコンピュータ名

次の例は、`lennon` というユーザが、システム・プロンプト (%) から `who am i` コマンドを入力して出力された結果です。

```
% who am i
lennon      tty0      Jul 15 14:17      (walrus)
```

この例は、ユーザ `lennon` が 7 月 15 日の午後 2 時 17 分に `walrus` というホストからログインしたことを表しています。`tty0` は使用した回線で、`walrus` はネットワークに接続しているこの回線に対する名前です。

`who am i` コマンドを使用すると、ワークステーションで実行しているセッションを追跡することができます。セッションの中には、ユーザ自身による別のホストへのリモート・ログインや、一緒に作業をしている人によるリモート・ログインがあります。`who am i` コマンドについての詳細は、`who(1)` リファレンス・ページを参照してください。

同じローカル・ホストに別のユーザがログインしているかどうかを確認するには、`who` コマンドを使用してください。次の例では、`lennon` がローカル・ホスト `london` のプロンプトに対して `who` コマンドを入力しています。この例では、現在、他に 3 人のユーザが別々のノードから `london` にログインしていることが表示されています。

```
london% who -M
lennon      tty0      Jul 15 08:17      (walrus)
elvis       tty2      Jul 15 07:55      (velvet)
burdon      tty1      Jul 15 09:02      (animal)
sarjan      tty4      Jul 14 16:47      (pepper)
```

`who` コマンドによる出力は、`who am i` コマンドによる出力と同じフォーマットです。

10-2 ネットワーク・ユーザおよびホストに関する情報の収集

10.2 ネットワーク・ユーザに関する情報の取得

`finger` コマンドおよびそのオプションを使用すると、ローカル・ホストまたはリモート・ホストにアカウントを持つユーザの情報を表示することができます。指定するホストは、`fingerd` デモン・サーバを実行しているか、または `fingerd` を起動するように構成された `inetd` デモンがなければなりません。`finger(1)` リファレンス・ページの説明のとおり `finger` コマンドが動作しない場合は、システム管理者にお問い合わせください。

`finger` コマンドの構文は、次のとおりです。

```
finger [ [ option... ] [ user... ] [ user@host_name... ] ]
```

オプションまたはユーザ名を指定しないで `finger` コマンドを実行すると、ログインしているローカル・ホスト上のすべてのユーザに関して、次の情報が表示されます。ただし、これらの情報が表示されるのは、情報が `/etc/passwd` ファイルにあるユーザに限ります。

- ログイン名
- フル・ネーム
- 端末の回線名、およびその回線が、`write` コマンド (11.8 節を参照) または `talk` コマンド (11.9 節を参照) で、他のユーザからメッセージを受信できるかどうか
- アイドル時間
- ログイン時刻
- ユーザのオフィスの所在地

10.2.1 特定のユーザに関する情報の取得

`finger` コマンドを使用する際に、ローカル・ホスト上のユーザのログイン名を指定すると、指定しない場合よりも詳しい情報が表示されます。表示される追加情報は、次のとおりです。

- ユーザのホーム・ディレクトリおよびログイン・シェル
- ユーザのホーム・ディレクトリにある `.plan` ファイルおよび `.project` ファイルの内容 (存在する場合)

次の例では、`finger` コマンドを使用して、ローカル・ホストにアカウントを持つユーザ `smith` に関する情報を表示しています。

```
% finger smith
Login name: smith      (messages off)  In real life: John Smith
Office: LV05-3/T24
Directory: /usr/netd/r2/smith          Shell: /bin/csh
On since Apr  9 16:20:56 on ttyrb from wombat.lv5.dec.c
18 seconds Idle Time
Project: manual, "Communicating with Network Users"
Plan:
```

出力結果の最初の行にある `messages off` は、ユーザ `smith` が `mesg n` コマンドを `.login` ファイルに記述していることを表しています。このコマンドを記述しておくで、ユーザ `smith` の端末は、他のユーザから `write` コマンドまたは `talk` コマンドによって送信される邪魔なメッセージを受信しなくなります。

この例では、ユーザ `smith` が自分のホーム・ディレクトリに作成した `.project` ファイルおよび `.plan` ファイルの内容も表示されています。`.project` ファイルには1行しか入力することができません。`.plan` ファイルには、ファイル・システムの許容限度まで行を入力することができます。`finger` コマンドは、ファイルの終端 (EOF) までのすべての行を表示します。

10.2.2 リモート・ホスト上のユーザに関する情報の取得

次の例は、`finger` コマンドによって表示された、リモート・ホスト `boston` 上のユーザに関する情報を示しています。

```
% finger @boston
[boston]
Login      Name           TTY Idle   When      Office
amy        Amy Wilson     p0    4 Thu 10:00 345
chang      Peter Chang    *p1 2:58 Thu 10:16 103
```

この出力結果の最初の行にはリモート・ホスト名 `boston`、2行目には出力の各フィールドに表示される情報の種類、3行目以降には各ユーザに対して1行ずつ情報が表示されます。アスタリスク (*) は、ユーザ `chang` が `mesg n` コマンドを `.login` ファイルに記述して、他のユーザからの `write` コマンドまたは `talk` コマンドによるメッセージを受信しないようにしていることを示します。

10.2.3 リモート・ホスト上の各ユーザに関する情報の取得

リモート・ホスト `havana` 上のユーザ `luis` に関する情報を表示するには、次のように `finger` コマンドを実行してください。

10-4 ネットワーク・ユーザおよびホストに関する情報の収集

```
% finger luis@havana
Login name: luis                      In real life: Luis Aguilera
Directory: /users/luis                Shell: /bin/csh
On since May 24 10:16:07 on tty2 from :0.0
58 minutes Idle Time
Project: baseball game simulation software
Plan:
Distribute with linked statistics module.
```

10.2.4 **finger** コマンドからの出力のカスタマイズ

finger コマンドには、必要なデータに応じて出力結果を変更できるオプションが用意されています。表 10-1 に、各オプションとその説明を示します。

表 10-1: **finger** コマンドのオプション

オプション	説明
-b	簡易形式で出力する。
-f	各フィールドのタイトルを表示しない。
-h	ユーザの .project ファイルを表示しない。
-i	ユーザとアイドル時間のリストを表示する。
-l	他のオプションを無視して、詳細なフォーマットで出力する。
-m	user がアカウント名であるとみなす。
-p	ユーザの .plan ファイルを表示しない。
-q	ユーザのログイン名、端末名、およびログイン時間のみを表示する。
-s	他のオプションを無視して、簡潔なフォーマットで出力する。
-w	他のオプションを無視して、短い簡潔なフォーマットで出力する。

finger コマンドについての詳細は、**finger(1)** リファレンス・ページを参照してください。

10.3 リモート・ホストおよびユーザに関する情報の取得

本書で説明するコマンドを使用して、ネットワーク上でメッセージの送信やファイル転送を実行する前に、受信側のホストが現在オンラインであるかどうかを確認しなければなりません。これを行うには、ローカルのネットワー

クで rwhod デーモンを実行しているホストに対して動作する `ruptime` コマンドを使用します。

`ruptime` コマンドを実行すると、次の情報が表示されます。

- ホスト名
- オンライン状態 (オンラインの場合は `up` , オフラインの場合は `down`)
- ホストがオンライン (またはオフライン) の状態になっている日数 (1 日より長い場合) , 時間と分
- 現在 , ホストにログインしているユーザの数 (オプションを指定すると , セッションが 1 時間以上アイドル状態であるユーザを含むことができる)
- `ruptime` 要求の前の , 5 分 , 10 分 , または 15 分間隔での負荷平均の統計

`ruptime` コマンドの構文は、次のとおりです。

`ruptime` [[*option* ...] [*sort_option*]]

オプションを指定しないで `ruptime` コマンドを実行すると、ローカル・ネットワーク上のホストに関する状態レポートは、ホスト名のアルファベット順でソートされて表示されます。このレポートでは、ホストがオンライン (またはオフライン) 状態になっている時間は、時間;分という形式で示されます。正の符号 (+) が含まれている場合は、時間が 1 日 (24 時間) を超えています。次に表示の例を示します。

```
% ruptime
apple      up 102+05:07    4 users,   load 0.09, 0.04, 0.04
byblos     up   3+03:17,   3 users,   load 0.08, 0.07, 0.04
carpal     up      2:28,   0 users,   load 7.01, 5.02, 3.03
dull       down  9+21:59
eager      down 23+22:45
foobar     up   3+01:44,   9 users,   load 0.01, 0.02, 0.03
garlic     up  14+01:35,   1 user,    load 0.06, 0.12, 0.11
hiccup     up   4+22:14,  19 users,   load 6.37, 3.90, 2.71
jackal     up  13+10:32,  26 users,   load 0.70, 0.92, 0.95
starry     up  16+21:08,   1 user,    load 0.22, 0.14, 0.07
travel     up  13+23:44,   7 users,   load 1.01, 1.19, 0.5
trekky     down 23+03:53
tribbl     up   8+21:43,   0 users,   load 0.00, 0.00, 0.00
trubbl     up  14+02:34,   0 users,   load 0.00, 0.00, 0.00
tunnel     down 14+02:34
warp9      up   8+01:24,   9 users,   load 0.01, 0.02, 0.03
```


ある 1 台のホストがオンラインであるかどうかの確認だけが必要になることがあります。このような場合には、次に示すホスト `trekky` の例のように、`ruptime` コマンドの後にホスト名を入力してください。

```
% ruptime trekky
trekky    down 23+03:53
```

この出力結果は、ホスト `trekky` が、現在オンラインではないことを表しています。

10.5 節で説明する `ping` コマンドを使用しても、ホストがオンラインであるかどうかを確認することができます。`ping` コマンドは、TCP/IP ネットワーク構成の各ホストに対して動作します。

リモート・ホストでコマンドを実行する場合には (第 13 章で説明)、`-l` オプションを指定して `ruptime` コマンドを実行し、ホストのリソースが十分かどうかを確認してください。このコマンドを実行すると、ホストが負荷平均の降順でソートされます。次に、`ruptime -l` コマンドを実行した結果の出力例を示します。

```
% ruptime -l
carpal    up      2:28,    0 users,    load 7.01, 5.02, 3.03
hiccup    up  4+22:14,   19 users,    load 6.37, 3.90, 2.71
travel    up 13+23:44,    7 users,    load 1.01, 1.19, 0.5
jackal    up 13+10:32,   26 users,    load 0.70, 0.92, 0.95
:
```

この例では、`carpal` と `hiccup` 以外のホストで使用量が少なくなっています。したがって、`travel` か `jackal` が使用目的に適していれば、どちらかのホストにリモートでログインできます。

リモート・ホストを長い時間使用する必要がある場合は、セッションが 1 時間以上使用中のユーザ数だけでなく、そのホストを使用しているユーザの総数を知らなければなりません。リモート・ホスト上のユーザの総数を表示するには、`-a` オプションを指定して `ruptime` コマンドを使用してください。次の 2 つの例のうち、最初の例はホスト `travel` 上の、2 番目の例はホスト `jackal` 上のユーザの総数を示します。

```
% ruptime -a travel
travel    up 13+23:44,   32 users,    load 1.01, 1.19, 0.5
```

```
% ruptime -a jackal
jackal    up 13+10:32,   29 users,    load 0.70, 0.92, 0.95
```

-a オプションおよび -l オプション (前述の例を参照) を指定したruptime コマンドの実行結果から、ホスト travel および jackal には、ほぼ同数のユーザがいることがわかりますが、現在のホスト travel の使用状況としては、(総数 32 人のうち) ここ 1 時間内にセッションを使用しているユーザは 7 人しかいません。これに対し、ホスト jackal は、ユーザの総数は travel より少ないのですが、総数 29 人のユーザのうち 26 人がここ 1 時間内にセッションを使用しています。以上のことから、時間の経過にともない、ホスト travel の使用量は、ユーザがログインするにつれて増加する可能性があるが、ホスト jackal の使用量は、ユーザのほとんどがすでにログインしているため、変わらないか減少する可能性があることがわかります。

他のオプション (-r オプションを除く) は、別の出力フィールドによりアルファベットの降順にソートします。昇順にソートするには、コマンド行で各オプションの後に -r オプションを指定してください。ruptime コマンドのオプションは、-r オプション以外のオプションと組み合わせて指定しないでください。-r オプション以外のオプションと組み合わせて指定すると、コマンド行の最後に指定されたオプションのみが有効になります。表 10-2 に、各オプションについて説明します。

表 10-2: ruptime コマンドのオプション

オプション	説明
-a	セッションが 1 時間以上アイドル状態であるユーザを含め、すべてのユーザの情報を表示する。
-l	5, 10, および 15 分間隔の負荷平均によって出力をソートする。
-r	ソート順を逆にする。
-t	ホストのオンライン時間の長さによって出力をソートする。
-u	ユーザ数によって出力をソートする。

ruptime コマンドについての詳細は、ruptime(1) リファレンス・ページを参照してください。

10.4 リモート・ホスト上のユーザに関する情報の取得

メッセージの送信またはファイル転送のコマンドを実行する前に、受信側のユーザがログインしているかどうかを確認する必要があります。ローカルのネットワーク上のリモート・ホストにユーザがログインしているかどうかを確認するには、1 人または複数のユーザ名を指定して rwho コマンド

を実行してください。 `rwho` コマンドは、`rwhod` デーモンを実行しているホストに対してのみ動作します。 不明な点がある場合は、システム管理者に問い合わせてください。

`rwho` コマンドを実行すると、次の情報が表示されます。

- ユーザ名
- ホスト名
- 開始の日付と時刻
- ユーザのセッションがアイドル状態であった時間 (分数)

`rwho` コマンドの構文は、次のとおりです。

`rwho` [`[-a]` [*user...*]]

オプションを指定しないで `rwho` コマンドを実行すると、1 時間以上アイドル状態であるユーザを除いて、ローカル・ネットワーク上のホストに現在ログインしているすべてのユーザが表示されます。 ローカル・ネットワークでは、通常、ユーザ数は数十人にもものぼるため、情報が必要なユーザだけを指定するようにしてください。

`-a` オプションを指定すると、アイドル状態が 1 時間以上のユーザを含めてすべてのユーザが表示されますが、特定のユーザのみを指定することもできます。 これによって、ユーザが 1 時間以上アイドル状態かどうかに関係なく、リモート・ユーザがログインしているかどうかを確認することができます。 次に、`-a` オプションを指定して `rwho` コマンドを実行し、ユーザ `wally`、`becky`、および `smith` についての情報を表示する例を示します。

```
% rwho -a wally becky smith
becky  cygnus:pts0      Jan 17 11:20 :12
smith  aquila:ttyp0       Jan 15 09:52 :22
wally   lyra:pts7        Jan 17 13:15 1:32
wally   lyra:pts8        Jan 17 14:15 1:01
```

この例のように、`rwho` コマンドを実行した結果の出力は、アルファベット順にユーザ名、次にホスト名が表示されます。 アイドル時間の合計は、1 時間以上の場合、各セッションの開始時刻と日付の後にある最後のフィールドに表示されます。 `-a` フラグが指定されていたので、出力には、1 時間以上アイドル状態のユーザも含まれます。 `-a` フラグを指定しないで `rwho` コマンドを実行すると、ユーザ `wally` に関する情報は表示されません。

rwho コマンドについての詳細は、rwho(1) リファレンス・ページを参照してください。

10.5 リモート・ホストがオンラインであるかどうかの判定

ping コマンドは、システム管理者がネットワーク伝送の問題を解決するために使用し、TCP/IP ネットワークに構成されたホストに対して動作します。ネットワーク・ユーザは、このコマンドを使用して、リモート・ホストが現在オンラインであるかどうかを判断することができます。たとえば、リモート・ホスト moon がオンラインであるかどうかを判定するには、ローカルのシステム・プロンプトから ping コマンドを入力してください。リモート・ホストがオンラインであることを確認する出力は、Ctrl/C を押すまで、次の例のように表示されます。

```
% ping moon
PING moon (130.180.4.108): 56 data bytes
64 bytes from 130.180.4.108: icmp_seq=0 ttl=255 time=42 ms
64 bytes from 130.180.4.108: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 130.180.4.108: icmp_seq=2 ttl=255 time=0 ms
64 bytes from 130.180.4.108: icmp_seq=3 ttl=255 time=0 ms
Ctrl/C
----moon PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/11/42 ms
```

11

メッセージの送受信

この章では、次のコマンドのいずれか 1 つを使用して行う、ネットワーク上でのメッセージの送受信について説明します。

- `mailx` または `Mail` (11.2 節)
- `write` (11.8 節)
- `MH` (メッセージ・ハンドラ) プログラム (11.7 節)
- `talk` (11.9 節)

この章の例では、`Mail` ではなく `mailx` プログラムを使用します。`mailx` を使用すると、次のようなタスクを実行することができます。

- ユーザへのメッセージの送信
- 送信前のメッセージの編集
- メッセージへのファイルの読み込み
- 着信メッセージの保存、および分類

第 12 章で説明しているように、`mailx` を使用してファイル全体を送信することもできます。

`write` コマンドおよび `talk` コマンドは対話形式で実行されます。この場合、受信側がログインしていなければなりません。これらの対話式コマンドを使用する前には、第 10 章で説明している次のコマンドを使用して、ユーザやホストの名前およびアクセスが可能かどうかを確認します。

- `finger` または `who`
ローカル・ホスト上のユーザを検索する。
- `finger`, `rwho` または `ruptime`
リモート・ホスト上のユーザを検索する。
- `ping` または `ruptime`
現在アクセス可能なホストを検索する。

11.1 メール・メッセージのアドレス指定

`mailx` を使用すると、次の場所にいる 1 人または複数のユーザに対して、メッセージを送信することができます。

- ユーザのローカル・ホスト
- TCP/IP を介してユーザのローカル・ホストに接続されているリモート・ホスト
- TCP/IP, DECnet, または UUCP アドレッシングを介した別のネットワーク上のホスト

`mailx` コマンドの構文は次のとおりです。

```
mailx user [ @ { host | domain | host.domain } ] ...
```

ローカル・ホスト上のユーザにメールを送信するには、`mailx` コマンドを入力し、各ユーザに対してパラメータ `user` を指定してください。リモート・ホスト上のユーザに送信する場合は、アットマーク (@) の後に、ホストの位置を示す情報を追加しなければなりません。

たとえば、ホスト `orange` から同じローカル・ホスト上のユーザ `smith` と `jones` にメールを送信するには、次のコマンドを入力してください。

```
orange% mailx smith jones
```

同じドメイン内の別のホスト `pluto` 上のユーザ `hobbes` にメールを送信するには、次のコマンドを入力してください。

```
orange% mailx hobbes@pluto
```

この例のユーザ `hobbes` が `planets` という別のドメインにいる場合は、次のコマンドに示すように、リモート・ネットワーク・ドメインの名前を追加します。

```
orange% mailx hobbes@pluto.planets
```

ドメインは、名前をピリオド (.) で区切って宛先をさらに細かく区分することがあります。ネットワーク管理者によるネットワークの構成方法によっては、次に示すコマンドの例のように、ドメイン名だけを指定して、別のドメインにあるリモート・ホスト上のユーザのアドレスを指定することができます。

```
orange% mailx hobbes@planets
```

メール・メッセージのアドレス指定については、必要に応じてシステム管理者に問い合わせてください。

11.2 mailx を使用したメール・メッセージの送信

この節では、mailx を使用してローカル・ホスト上のユーザへメッセージを送信し、そのメッセージのコピーを他のユーザへ送信する方法について説明します。例を開始するにあたって、まず、ユーザ Jones は、ローカルのシステム・プロンプト orange% から mailx コマンドを次のように入力します。

```
orange% mailx suzuki Return
```

Return キーを押すと、Subject: プロンプトが表示されます。もう一度 Return キーを押すと、表題は空白のままになります。この例では、ユーザ Jones は次のように、メッセージの表題を入力してから Return キーを押し、続いてメッセージを書き始めます。

```
Subject: Baseball question Return
```

```
Are there any Japanese baseball simulation games?  
I want to compare Sadharu Oh's hitting statistics  
to those of Hank Aaron. To do this, I need to set  
up a simulated baseball season having each hitter  
play for one year in the other player's league.
```

次に示すように、ユーザ Jones は、空白行にピリオド (.) をタイプし、Return キーを押して、メッセージの入力を終了します。

```
      :  
      :  
play for one year in the other player's league.  
.  
Cc:
```

この例では、メッセージはまだ送信されず、代わりにプロンプト Cc: (carbon copy) が表示されます。これは、ユーザ Jones が、自分のホーム・ディレクトリにある .mailrc ファイルに set askcc を追加して、メール・セッションをカスタマイズしているためです。メール・セッションのカスタマイズについては、11.6 節および付録 D を参照してください。

Cc: プロンプトによって、他のユーザにコピーを送信することができます。コピーを送信しない場合には、Return キーを押して mailx を終了し、メッセージを送信してください。すると、テキストの終端を示すメッセージ (EOT) と、その後にシステム・プロンプトが表示されます。

この例では、ユーザ Jones は、Cc: プロンプトに対して、ローカル・ユーザの cranton と、リモート・ユーザの gillis および vincep のそれぞれ適切なアドレスを指定します。Return キーを押して mailx を終了し、メッセージを送信すると、コピーが送信されます。

⋮

```
Cc: cranton gillis@strato vincep@mlb.bbs.com Return
EOT
orange%
```

mailx プログラムは、アドレス指定エラーから回復することができます。たとえば、ローカル・ホスト上の受信者が cranton である場合に、間違っ
て crantom とタイプしたとすると、次のメッセージがすぐに画面に表示されます。

```
crantom... User unknown
```

メッセージはユーザに送り返されます。ユーザはそれを保存して、正しい宛先に送信し直すことができます。

リモート・ホスト上の未知のユーザにメッセージを送信すると、mailx がそれをユーザに送り返すまでに最長で 3 日間かかることがあります。送り返されたメッセージの保存と再送信の方法については、11.3.3.1 項を参照してください。

11.2.1 メッセージの編集

メール・メッセージを送信する前に編集するには、Subject: プロンプトに
応答した後、次のエスケープ・コマンドのうちのいずれか 1 つを入力して、
mailx 内でエディタを起動します。

- ~v の入力

ユーザの .mailrc ファイルに set VISUAL を指定して設定したスクリーン・エディタを起動する。

- ~e の入力

ユーザの .mailrc ファイルに set EDITOR を指定して設定したテキスト・エディタを起動する。

~e コマンドを使用して、mailx 内でテキスト・エディタを起動するには、新しい行の先頭に ~e を入力してください。チルダ (~) が表示されるまでは数回入力する必要があるかもしれません。たとえば次のようになります。

```
Subject: network documentation meeting at 2 PM
Everyone, please bring the Table of Contents
for your manual so that we can look for areas
of overlapping subject matter and
```

11-4 メッセージの送受信

~e

ユーザの `.mailrc` ファイルに `set EDITOR=/usr/ucb/vi` が記述されている場合、ユーザは `vi` エディタを使用して、最初の行の綴りの間違いを訂正して、メッセージを書き終えることができます。編集セッションを終了すると、`mailx` に戻ります。ユーザは、メッセージを書き終わって `mailx` を終了するか、または、`vi` を再び呼び出して、メッセージを書き続けることができます。

11.2.2 メッセージの打ち切り

メッセージの送信を中断することがあります。メッセージを送信する前に打ち切る方法には 2 通りあります。

11.2.2.1 Ctrl/C を使用したメッセージの打ち切り

ユーザは、メッセージ内の任意の場所で `Ctrl/C` を 2 回押して、メール・メッセージを打ち切ることができます。`Ctrl/C` を 1 回押すと、次のメッセージが表示されます。

```
(Interrupt -- one more to kill letter)
```

この時点で、メッセージを打ち切るかどうかをもう一度検討することができます。打ち切らない場合は、続けてテキストを入力してください。メッセージを打ち切る場合は、もう一度 `Ctrl/C` を押します。すると、次のメッセージが表示されます。

```
(Last Interrupt -- letter saved in dead.letter)
```

これでメッセージは打ち切られ、`mailx` を終了して、システム・プロンプトが表示されます。

省略時の設定では、打ち切られたメッセージは、ユーザのホーム・ディレクトリの `dead.letter` ファイルに保存されます。打ち切られたメッセージを保存しない場合は、`set nosave` をユーザの `.mailrc` ファイルに入力します。詳細については、11.6 節および付録 D を参照してください。

`dead.letter` ファイルには最後に打ち切られたメッセージだけが保存されます。メール・メッセージに打ち切られたメッセージを取り込むと、編集や再送信ができます。メッセージにファイルを取り込む方法についての詳細は、11.2.3 項を参照してください。

次に、`Ctrl/C` を押してメール・メッセージを打ち切る例を示します。

```
orange% mailx sally
Subject: Update to reference page files
What should the mailx(1) reference page include
about sending to remote users? Ctrl/C
(Interrupt -- one more to kill letter)
Ctrl/c
(Last Interrupt -- letter saved in dead.letter)
orange%
```

11.2.2.2 エスケープ・コマンドによるメッセージの打ち切り

空白行に `~q` または `~x` のエスケープ・コマンドを入力することによって、メール・メッセージを打ち切ることができます。 `Ctrl/C` を押してメッセージを打ち切る方法とは異なり、このコマンドでは、再考するためのプロンプトは表示されず、直ちにメッセージを打ち切ります。 `~q` エスケープ・コマンドは、打ち切られたテキストをユーザのホーム・ディレクトリの `dead.letter` ファイルに保存しますが、`~x` は、`.mailrc` ファイルに `set save` と設定していても、保存しません。

次の例は、`~x` エスケープ・コマンドを使用して、メール・メッセージを打ち切る方法を示しています。

```
orange% mailx sally
Subject: Update to reference page files
What should the mailx(1) reference page include
~x
orange%
```

チルダ (`~`) が表示されるまでに数回入力する必要があるかもしれません。

11.2.3 メッセージへのファイルの取り込み

メール・メッセージ内に、任意のファイル (未変換のバイナリ・ファイルは除く) を取り込むことができます。アドレスの指定を誤って返却されたメールを保存して再送信 (11.3.3.1 項参照) する場合や、ユーザのホーム・ディレクトリの `dead.letter` ファイルに保存している打ち切られたメッセージを編集して再送信する場合に、この方法が使用されます。

前述の例では、`dead.letter` ファイルには次のテキストが含まれています。

```
What should the mailx(1) reference page include
about sending to remote users?
```

追加情報を加えた後、このファイルをユーザ `sally` に再送信すると仮定します。 `mailx` 内では、`~d` エスケープ・コマンドを使用すると、現在の

作業ディレクトリに関係なく、`dead.letter` のテキストがメール・メッセージに自動的に追加されます。

例 11-1 では、ユーザ `sally` へのメッセージを開始してから、`~d` コマンドによって `dead.letter` のテキストを追加します。

例 11-1: `dead.letter` ファイルへの取り込み

```
orange% mailx sally
Subject: the mailx(1) reference page
The uucp(1) reference page has formatting
information for sending to remote users.
~d
"/usr/staff/r2/sally/dead.letter" 2/76
```

ファイルを取り込むと、完全パス名とともに、そのファイルの行数 (2) および文字数 (76) (Return キーまたは各行末の制御文字を含む) が表示されます。この後、終了すること、続けてメッセージを書き込むこともできますが、`~p` エスケープ・コマンドで表示されないインクルード・ファイルの内容を参照したり、テキスト・エディタを使用してインクルード・ファイルの内容を変更したりすることもできます (たとえば、`vi` を使用したい場合は `~v` と入力します)。

注意

`dead.letter` ファイルには、最後に打ち切られたメール・メッセージだけが保存されます。送信したいテキストが含まれていることを確認しなければなりません。

メール・メッセージにファイル (`dead.letter` を含む) を取り込むには、`~<`、または `~r` のいずれかのエスケープ・コマンドを指定してから、ファイル名を入力してください。この 2 つのコマンドはどちらも同様に動作します。ファイルのディレクトリが、`mailx` を入力したディレクトリと異なる場合には、ファイル名の前に完全パス名または現在のディレクトリに対する相対パス名を指定しなければなりません。

次の例 11-2 では、`~<` エスケープ・コマンドを使用して、現在の作業ディレクトリの下 `environ` ディレクトリから、`strato_prob` というファイルを挿入します。

例 11-2: mailx コマンドを使ったファイルの取り込み

```
orange% mailx sally
Subject: Dan, here's the stratosphere data file
~< environ/strato_prob
"environ/strato_prob" 41/1309
```

mailx ユーティリティによって、対話形式を使用しないでファイルを転送する方法については、12.1.3 項を参照してください。

11.3 メール・メッセージの受信

メール・メッセージを受信する場合には、次のいずれかを選択できます。

- 任意のメッセージを読み込むかまたは削除する。
- メールの送信者およびその他の受信者に応答する。
- ファイルにメッセージを保存する。
- フォルダと呼ばれるメッセージの保存用ファイルに、トピックごとにメッセージを分類する。

ユーザがログインしたとき、オペレーティング・システムのコマンドを入力したとき、または Return キーを押したときに、新しいメールが到着していれば、mailx プログラムはそれを通知します。また、システム・プロンプトから mailx コマンドを入力して、新しいメールが到着しているかどうかを調べることができます。

例 11-3 では、ホスト orange のユーザ Jones が mailx を入力して、2 件のメッセージが到着していることを確認します。

例 11-3: mailx 環境に入る

```
orange% mailx
Mail $Revision: 1.1.6.4 $ Type ? for help.
"/usr/spool/mail/jones": 2 messages 1 new 1 unread
  U  1 root                Mon Jul 20 10:39 14/438  "System news"
>N  2 root                Mon Jul 20 11:30 11/292  "Welcome"
?
```

この例では、システム管理者 (root) からの 2 件のメッセージが到着しています。1 件は、以前の mailx のセッションから未読のメッセージ (1 カラム目に U と示される) であり、もう 1 件は、新しく到着したメッセージです (N で示される)。メッセージの最後に表示される疑問符 (?) は、mailx のプロンプトです。この表示のヘッダおよび 11.4 節で説明するように、このプロンプトからもう 1 つ疑問符を入力すると、使用可能な mailx のコマンドのリストを表示することができます。

mailx プロンプトで Return キーを押すと、未読のメッセージ・リストに (>) で示されている、メッセージ 2 を読むことができます。たとえば、例 11-4 のように実行できます。

例 11-4: mailx プロンプトでのメッセージ開封

```
? Return
Message 2:
From root Wed Aug 4 11:17:36 1999
Date: Wed, 4 Aug 1999 11:17:29 -0400
From: root (system administrator)
To: jones
Subject: Welcome

Welcome to the company computer network. I'm the
person who manages this system. If you have
questions or problems, send mail to root. You
can also send mail to manager or admin; messages
will be forwarded to me.

I will be on vacation for the next two weeks after
this week... starting Monday, August 10. I'll be
stdin Space
back on Monday, August 24.
?
```

例 11-4 では、PAGER 変数が more (省略時の設定) に設定されているため、stdin が表示されています。PAGER 変数が pg に設定されている場合には、何も表示されません。また、上記の例では、ユーザ Jones が set crt=15 を .mailrc ファイルに追加して、mailx 環境をカスタマイズしているため、stdin はメッセージを 15 行表示した後に表示されます。.mailrc ファイルの set crt= は、ページャ (pg または more) を呼び出す前に、残りのメッセージの 1 回に表示する行数を指定します。例に示すように、メッセージが 15 行を超える場合、set crt は、15 行を表示した後にページャ

を呼び出すよう mailx に指示します。スペース・バーを押すと、次の 15 行が表示されます。set crt= を使用して、自分の mailx 環境をカスタマイズするようにしてください。カスタマイズしない場合には、長いメール・メッセージが高速でスクロールするため、Hold Screen キーをすばやく押さなければなりません。mailx をカスタマイズする方法についての詳細は、11.6.2 項および付録 D を参照してください。

他のメッセージを読むには、mailx プロンプトに対してメッセージ番号を入力してください。メッセージのリストを再び表示するには、mailx プロンプトで h を入力します。例 11-5 では、ユーザ Jones は、h コマンドを使用してメール・メッセージのリストを表示します。最初のメッセージが未読であることを確認すると、? プロンプトから 1 を入力してそれを読みます。

例 11-5: mailx プロンプトでの他のメッセージの開封

```
? h
  U  1 root    Mon Jul 20 10:39 14/438  "System news"
>   2 root    Mon Jul 20 11:30 11/292  "Welcome"
? 1
Message 1:
From root Mon Jul 20 11:30:07 1999
Date: Mon, 20 Jul 1999 11:30:04 -0400
From: root (system administrator)
To: jones
Subject: System news

The newest release of the text processing
software will be installed after 5 o'clock
today. Send mail if you have questions or
concerns before or after the installation.

?
```

ユーザが現在読んでいるメッセージを現在のメッセージと呼びます。再読するには、Return キーを押してください。次のメッセージを読むには、n を押します。すると、このメッセージが現在のメッセージになります。すべてのメッセージを連続して読む場合は、各メッセージを読み終えるごとに n を押します。これは、付録 D で説明しているように、gonext 変数を修正することによって変更することができます。

11.3.1 メッセージの削除

ファイルまたはフォルダに格納する前後にユーザが削除しない限り、メッセージは `mailx` に保存されています。現在のメッセージを読後に削除するには、`mailx` プロンプトに対して `d` を入力してください。別のメッセージを削除する場合は、`mailx` プロンプトで、`d` コマンドの後にメッセージ番号を入力してください。`d` コマンドの後にメッセージ番号を並べて入力すると、複数のメッセージを一度に削除することができます。たとえば、メッセージ 7 および 9 を削除するには、`mailx` プロンプトで次のコマンドを入力してください。

```
? d 7 9
```

ある範囲のメッセージを削除するには、削除する最初のメッセージの番号と最後のメッセージの番号の間にハイフンを使用して指定します。たとえば、メッセージ 7 から 11 までを削除する場合は、`mailx` プロンプトで次のコマンドを入力してください。

```
? d 7-11
```

メッセージを誤って削除した場合は、`u` (`undelete`) コマンドを使用すると、そのメッセージを回復することができます。たとえば、メッセージ 7 を回復するには、`mailx` プロンプトから次のコマンドを入力してください。

```
? u 7
```

`x` の代わりに、`q` または `quit` を入力して `mailx` を終了した場合、すでに読んだメッセージで削除していないものは、ホーム・ディレクトリにある、それまでに削除していないメッセージを保存するための `mbox` というファイルの最後に追加されます。

11.3.2 メッセージへの応答

メール・メッセージへの応答は、メール・メッセージへの送信と同様に実行できます。11.2 節で説明しているように、メッセージの編集、打ち切り、メッセージへのファイルの取り込みを同様に選択できます。

メッセージを読んですぐに送信者に応答する場合は、例 11-6 のように、メール・プロンプトから大文字の `R` (`reply`) コマンドを入力してください。

例 11-6: メール・メッセージへの応答

```
Message 3:
From deedee Mon Jul 20 14:13:32 1999
Date: Mon, 20 Jul 1999 14:13:05 -0400
From: deedee (DeeDee Smith)
To: jones, mays@sf24.usernet, susannah@artwrk
Subject: Testing text-processing software

I think we should test the new text processing
software on the older machines as well as the
newer. Remember that many customers still have
the older models.

? R
To: deedee
Subject: Re: Testing text-processing software

I agree. Also, we should test different machine
configurations to determine if, for example,
it performs satisfactorily when run remotely.
.
EOT
?
```

R を入力すると、受信者および表題が表示されて、受信者を確認することができます。

小文字の r を入力すると、元のメールの送信者のほかに受信者にも応答が送信されます。例 11-6 では、deedee に応答が送信されるとき、mays@sf24.usernet および susannah@artwrk にも送信されます。

注意

メール・メッセージの送信者にだけ応答する場合は、メール・プロンプトで大文字の R を入力してください。メール・メッセージの送信者とすべての受信者に応答する場合は、プロンプトで小文字の r を入力してください。

例 11-6 では、ユーザ Jones の .mailrc ファイルに set askcc コマンドが記述されていないため、Cc: プロンプトは表示されません。

11.3.3 メッセージの保存

x または exit コマンドではなく、q コマンドを入力して mailx を終了する場合、読み終わったファイルは、ユーザのホーム・ディレクトリにある mbox ファイルに保存されます。

もっと有効な方法でメール・メッセージを格納するためには、以下の項で説明するように、個別に名前を付けたファイルやフォルダにそれらのメッセージを保存することができます。

11.3.3.1 ファイルへのメッセージの保存

ファイルに保存するメールには次のような種類のものがあります。

- ユーザが読む短い重要なメッセージ
- mailx によって返送されたアドレス指定を誤ったメール・メッセージ
- 後で印刷してから読む長いメッセージ

ユーザが読む短いメッセージ、または mailx によって返送されたメール・メッセージを保存する場合は、mailx プロンプトで s コマンドを入力して、ファイル名を指定してください。

次の例では、mailx コマンドによる出力の 2 番目の項目に、返送されたメッセージが表示されています。

```
orange% mailx
Mail $Revision: 1.1.6.4 $ Type ? for help.
"/usr/spool/mail/jones": 2 messages 1 new 1 unread
  U 1 root    Mon Jul 20 10:39 14/438  "System news"
>N 2 MAILER-DAEMON Wed Aug  5 09:39 19/498 "Returned mail: User unknown"
?
```

次の例で示すように、ユーザ Jones は、この例の返送されたメッセージを verify-resend というファイルに保存して、再送信する前に正しいアドレスを確認することを覚えておくようにします。

```
      :
      :
>N 2 MAILER-DAEMON Wed Aug  5 09:39 19/498 "Returned mail: User unknown"
? s verify-resend
```

ファイル verify-resend は、明示的にパス名を指定しない限り、現在のディレクトリに保存されます。たとえば、ユーザ Jones は、次のようにコマンドを入力すると、サブディレクトリ fix-later に、このファイルを保存することもできます。

```
? s fix-later/verify-resend
```

長いメッセージの場合に、オンラインでテキスト全部を読まずに保存するには、Ctrl/C を押してメッセージのスクロールを中断し、mailx プロンプトを表示させます。この時点でメッセージを保存 (または削除) することができます。次の例では、ユーザ Jones は、20 ページのレポートからなるメッセージ 1 を受信します。Ctrl/C を押して mailx プロンプトを表示させ、ファイルを保存するためのコマンドを入力します。

```
? 1
Message 1:
From smith Wed Aug 5 16:43:42 1999
Date: Wed, 5 Aug 1999 16:43:41 -0400
From: smith (Cassandra Smith)
To: jones
Subject: 20-page report: host configuration results
```

Mortimer,

Here's the report on host configuration that the

Ctrl/C

Interrupt

```
?s sys-config-report
```

```
"sys-config-report" [New file] 2147/48353
```

```
?dp
```

```
⋮
```

前の例では、ファイルを作成した後、ユーザ Jones は、mailx プロンプトで dp (delete-proceed: 削除して続行) を入力して、この 20 ページの大きなファイルが mbox に保存されないようにし、次のメール・メッセージの読み込みを開始します。mailx コマンドの d (delete) を入力してから Return キーを押しても、同様の処理を行うことができます。

11.3.3.2 フォルダへのメッセージの保存

メッセージを参照情報で分類して、mbox ファイル (それ自身がフォルダ) のサイズを最小限にするには、フォルダと呼ばれるファイルにメッセージを保存します。mbox 以外のフォルダを使用する前には、ユーザのホーム・ディレクトリにフォルダのサブディレクトリを作成し、.mailrc ファイルにフォルダへのポインタを追加する必要があります。たとえば、ホーム・ディレクトリに sys-config というディレクトリを作成する場合は、set folder=sys-config という行をユーザの .mailrc ファイルに追加しなければなりません。

フォルダにメッセージを追加するには、`mailx` コマンド `s` の後にフォルダ名を指定してください。たとえば、ホスト構成についてのメッセージをそのトピックに関する他のメッセージと一緒に保存するには、次の例のように、`sys-config` というフォルダにそのメッセージを書き込みます。

```
Message 7:
From smith Thu Aug  6 09:32:09 1999
Date: Thu, 6 Aug 1999 09:32:08 -0400
From: smith (Cassandra Smith)
To: jones
Subject: host configuration testing
```

```
According to the report that each LAN ...
? s sys-config
"sys-config [New file] 11/235
```

フォルダに最初のメッセージを保存すると、`mailx` はそのメッセージを格納して、`New file` というメッセージを表示します。そのフォルダにさらに別のメッセージを保存すると、`mailx` はそれをファイルの最後に追加して、`Appended` というメッセージを表示します。

`mbox` 以外のフォルダに保存されたメッセージを読むには、次の 2 通りの方法があります。

- シェル・プロンプトで、`-f` オプションとフォルダ名を指定して `mailx` を起動する。

たとえば、`sys-config` フォルダを読むには、次のコマンドを入力してください。

```
orange% mailx -f sys-config
```

- ユーザがすでに `mailx` にいる場合は、別のフォルダに切り替える `folder` コマンドを使用する。

たとえば、`sys-config` フォルダを読んでいるときに、`meetings` フォルダを読みたい場合は、次のコマンドを入力します。

```
? folder meetings
```

フォルダを切り替える場合には、`mailx` は、クローズするフォルダに対して変更を加えてから、新しいフォルダをオープンします。

引数を指定しないで `folder` コマンドを使用すると、現在使用しているフォルダを確認することができます。たとえば次のようになります。

```
? folder
"sys-config": 17 messages
```

11.3.4 メッセージの転送

例 11-7 は、mailx の m コマンドと ~f エスケープ・コマンドを使用して、サブジェクト行とコメントを含めてメッセージ番号 3 のメッセージをユーザ deedeede に転送する例です。

例 11-7: Forwarding a Message

```
? m deedeede 1
Subject: forwarding a message 2
I received this note from Gary. Do you agree? 3
~f 3 4
Interpolating: 3 5
(continue)
~p 6
-----
```

Message contains:

To: deedeede

Subject: forwarding a message

I received this note from Gary. Do you agree?

From gary Wed Mar 4 16:10:48 1999

Date: Wed, 4 Mar 1999 16:10:48 -0500

From: To: csug@myhost.mydomain

Subject: Forwarding

Cc: smith

I think forwarding should be tomorrow's topic.

Gary

(continue)

.

EOT

1 m コマンドでメッセージの作成を開始します。

2 新しいメッセージの件名を入力します。

3 新しいテキストを入力します。

4 mailx の ~f エスケープ・コマンドでメッセージ番号 3 のメッセージを転送します。

5 mailx コマンドは、メッセージの読み取り中であることを示し、その後、入力の受け付けが可能になるとプロンプトを表示します。

6 転送するメッセージが正しく取り込まれていることを確認する場合は ~p エスケープ・コマンドを使用します。

この例に示すように、`~f` および `~p` コマンドを入力した後、書き込みを続けるか、または、メッセージを終了することができます。現在のメッセージを転送する場合は、`~f` の後に番号を入力しないでください。

11.4 mailx からのヘルプ情報の参照

`mailx` を起動したときにメッセージが到着している場合は、次の行がヘッダの先頭に表示されます。

```
Mail $Revision: 1.1.6.4 $ Type ? for help.
```

これは、`mailx` プロンプトで疑問符 (?) をタイプすると、例 11-8 に示すように、使用できる `mailx` コマンドの簡単な説明が表示されることを、ユーザに知らせるためのものです。

例 11-8: mailx の help コマンドからの出力

```
? ?
Control Commands:
  q             Quit - apply mailbox commands entered this session.
  x             Quit - restore mailbox to original state.
  ! <cmd>       Start a shell, run <cmd> and return to mailbox.
  cd [<dir>]     Change directory to <dir> or $HOME.
Display Commands:
  t [<msg_list>] Display messages in <msg_list> or current message.
  n             Display next message.
  f [<msg_list>] Display headings of messages.
  h [<num>]      Display headings of group containing message <num>.
Message Handling:
  e [<num>]      Edit message <num> (default editor is e).
  d [<msg_list>] Delete messages in <msg_list> or current message.
  u [<msg_list>] Recall deleted messages.
  s [<msg_list>] <file> Append messages (with headings) to <file>.
  w [<msg_list>] <file> Append messages (text only) to <file>.
  pre [<msg_list>] Keep messages in system mailbox.
Creating New Mail:
  m <addrlist>  Create/send new message to addresses in <addrlist>.
  r [<msg_list>] Send reply to senders and recipients of messages.
  R [<msg_list>] Send reply only to senders of messages.
  a            Display list of aliases and their addresses.
===== Mailbox Commands =====
```

11.5 メールの終了

`mailx` の終了に使用できるコマンドは、3 つあります。

- `q` コマンド

シェル・プロンプトに戻り、読後削除していないメッセージを、ユーザのホーム・ディレクトリの `mbox` ファイルに保存します。

- `x` コマンドおよび `exit` コマンド

この2つのコマンドは同じです。どちらも、ユーザのメールボックスを変更しないでシェル・プロンプトに戻ります。

11.6 メール・セッションのカスタマイズ

新規ユーザのアカウントを設定するとき、システム管理者は、`mailx` の省略時の設定を `/usr/share/lib/Mail.rc` ファイルに定義します。オペレーティング・システムで提供されるファイルには、次のような `mailx` の設定が含まれていますが、ユーザはこれを変更することができます。

- `set ask`
Subject: プロンプトを表示します。
- `set noaskcc`
Cc: プロンプトを表示しません。
- `set dot`
空白行に1つのドット (.) を入力すると、メール・メッセージが終了するようにします。
- `set nokeep`
システムのメールボックスが空になったときに削除されます。この設定は、ほとんどのユーザにとっては重要ではありません。
- `set save`
打ち切られたメッセージが、`dead.letter` ファイルに保存されます。

ユーザは、ホーム・ディレクトリの `.mailrc` ファイルに、別名を定義したり変数を設定することにより、`mailx` セッションをカスタマイズすることができます。次の例 11-9 では、`.mailrc` ファイルの例を示します。

例 11-9: `.mailrc` ファイルの例

```
alias sue susannah
alias wombats tom, jeff, craig, jim, ken
set ask
set askcc
set prompt=>
unset dot
set record=/usr/users/hale/outgoing
set folder=folder
```

例 11-9: .mailrc ファイルの例 (続き)

```
set crt=20
```

11.6.1 メールの別名の作成

mailx の alias コマンドを使用して、ユーザまたはユーザ・グループの代替名を作成できます。

注意

mailx の alias は、シェルが使用する alias コマンドとは別のコマンドです。シェルの alias コマンドでメール・コマンドを変更することはできません。

継続的に使用するメールの別名を定義するには、.mailrc ファイルに alias コマンドを記述して、別名と1つまたは複数のログイン名を指定します。次の .mailrc ファイルでは、2つの別名が定義されています。

```
alias sue susannah
alias wombats tom, jeff, craig, jim, ken
```

最初の別名は、sue という名前がユーザ susannah を意味するように定義されています。これによって、ユーザは、sue という名前を使用して、susannah にメールを送信することができます。2番目の別名は、メッセージを wombats にアドレス指定することによって、Wombats というチームのメンバである tom, jeff, craig, jim, および ken にメールを送信することができます。この行を別の方法で .mailrc ファイルに記述すると、次のようになります。

```
alias wombats tom,\
                jeff,\
                craig,\
                jim,\
                ken
```

バックスラッシュ (\) を使用すると、1つの長いコマンドを複数行に入力することができます。

mailx セッションで、引数を指定しないで `alias` を使用すると、定義されている別名を確認することができます。また、その mailx セッションのみで有効な一時的な別名を、mailx プロンプトから定義することもできます。

11.6.2 メール変数の設定

メール変数は、ユーザの `.login` ファイルに設定されている変数と同様です。バイナリ、文字列、または数字のいずれかになります。

`.mailrc` ファイルにバイナリのメール変数を設定するには、`set` コマンドの後にオプション名を指定して入力してください。サンプルの `.mailrc` ファイルには、次のバイナリ変数が含まれています。

```
set ask
set askcc
unset dot
```

- `ask` を設定すると、mailx は、送信するメッセージの表題の入力を求めるプロンプトを表示します。
- `askcc` を設定すると、mailx は、送信するメッセージのコピーの受信者を指定するように求めてきます。
- `dot` の設定を解除すると、行にピリオドだけを入力することによって mailx のメッセージを終了することはできません。この場合には、`Ctrl/D` を押してメッセージを終了しなければなりません。

文字列のメール変数は、文字または数を値として受け入れます。サンプルの `.mailrc` ファイルには、次の 3 つの文字列変数があります。

```
set folder=folder
set record=/usr/users/hale/folder/outgoing
set crt=20
```

- `folder` 変数は、ユーザのメール・フォルダを入れるためのサブディレクトリを定義します。この変数を設定すると、mailx ユーティリティは、`save folder` コマンドを使用してメッセージを保存する際に、このディレクトリにファイルとしてフォルダを作成します。mailx ユーティリティは、このファイル名をユーザのホーム・ディレクトリのサブディレクトリとして解釈します。

- `record` 変数は、送信する各メッセージのコピーを、指定するファイルに入れるように `mailx` に指示します。この変数を設定しなければ、自動的にレコードが保持されることはありません。この例では、`mailx` が通常のフォルダとして扱うファイルを指定しています。レコード・ファイルを選択するには、次のコマンドを使用します。

```
orange% mail -f outgoing
```

- `crt` 変数は、ページャ・プログラムを呼び出す前に表示するメッセージの行数を `mailx` に指示します。

11.7 MH プログラム

`mailx` プログラムの代わりとして、MH (メッセージ・ハンドラ) プログラムがあります。MH プログラムは、シェル・プロンプトから実行したいコマンドを入力して使用する、小さなメール処理プログラムの集まりです。

MH プログラムはオプションであるため、ユーザのホストにインストールされていないこともあります。MH が利用できるかどうかを確認するには、`/usr/bin/mh` ディレクトリを検索してください。

MH を使用するには、`.cshrc` ファイルまたは `.login` ファイルの `set path` 行を編集して、パスに `/usr/bin/mh` ディレクトリを追加しなければなりません。その後、ログアウトして再びログインするか、または次のコマンド (C シェルの場合) を入力することにより、シェルにパスの変更を通知します。

```
orange% source .login
```

パスが `.cshrc` に設定されている場合は、このコマンドで、`.login` ではなく `.cshrc` を使用してください。

Bourne, Korn, または POSIX シェルのいずれかを使用している場合には、この情報を `.profile` ファイルに追加し、次のコマンドを入力することによってシェルに通知します。

```
orange$ . .profile
```

MH プログラムを使用すると、フォルダは、`mailx` のフォルダとは異なる編成になります。新たに届いたメールや未読のメールは、`inbox` と呼ばれるフォルダに保存されますが、ここには、ユーザが `inc` コマンドを使用して、システムのメールボックスに届くメールを移動します。新しいメールの挿入は、毎回 `inc` コマンドを入力して行わなければなりません。

フォルダの選択は `folder` コマンドで行います。フォルダ名を指定せずにこのコマンドを入力すると、`folder` は現在選択されているフォルダを表示します。

`folder` コマンドにオプション `-all` を指定すると、フォルダのリストおよびそれぞれのフォルダ内にあるメッセージ数を表示します。

`show`、`prev`、および `next` コマンドを使用すると、それぞれ現在のフォルダの、現在のメッセージ、前のメッセージ、および次のメッセージを読むことができます。`show` コマンドの後にメッセージ番号を指定すると、そのメッセージが現在のメッセージになります。たとえば次のようになります。

```
orange% show 7
Message 7:
From deedee Mon Jul 23 10:02:10 1999
Date: Mon, 23 Jul 1999 10:01:25 edt
To: hale
Subject: Cafeteria hours
Cc:
Status: R

I'm sorry you didn't ask that sooner. The cafeteria
closes its breakfast service at 10. Lunch starts
at 11:30.
```

`rmm` コマンドは、現在のフォルダからメッセージを削除します。引数を指定しないで `rmm` コマンドを使用すると、現在のメッセージを削除します。1 つまたは複数のメッセージ番号を指定すると、指定されたメッセージが削除されます。たとえば、メッセージ 2、5、および 7 を削除する場合は、次のコマンドを入力してください。

```
orange% rmm 2 5 7
```

表 11-1 に、ほとんどの MH コマンドをリストにします。完全なリストは、`mh(1)` リファレンス・ページを参照してください。各 MH コマンドの詳細については、それぞれのコマンドのリファレンス・ページを参照してください。

表 11-1: MH プログラムのコマンド

コマンド	説明
<code>ali</code>	指定された別名のファイルを検索して、別名に対応するアドレスを表示する。
<code>anno</code>	メッセージに注釈を付けて、メッセージの配信、転送、および応答を記録する。

表 11-1: MHプログラムのコマンド (続き)

コマンド	説明
burst	転送されたメッセージから元のメッセージを取り出し、転送ヘッダを破棄して、元のメッセージを現在のフォルダの最後に追加する。
comp	新規にメール・メッセージを作成して、テンプレートを提供し、エディタを起動してメッセージを完成できるようにする。
dist	現在のメッセージを、元の配信リストにないアドレスに再配信する。
folder	フォルダを選択するか、または現在のフォルダを表示する。
folders	すべてのフォルダと、各フォルダに含まれるメッセージの数をリストする。
forw	元のアドレス指定にはなかった受信者にメッセージを転送する。メッセージはカプセル化され (Forwarded Message の通知を含む)、ヘッダが追加される。
inc	システム・メールボックスのメールを、ユーザの inbox フォルダに挿入する。
mark	現在のフォルダにある一連のメッセージに名前を付ける。こうしておくで、pick コマンドを使用して、この方法でマークしたメッセージを選択できる。
mhl	フォーマットされた MH メッセージをリストする。more コマンドの代わりに、メッセージの表示に使用できる。
mhmail	指定されたユーザにメールを送信する。ユーザを指定しない場合、mhmail は、inc コマンドと同じ動作をする。
msgchk	システム・メールボックスや、新しく到着するメールを受信するファイルをユーザに代わってチェックして、新しいメッセージを検索する。新しいメッセージが届いている場合、msgchk はそれを通知する。
next	現在のフォルダまたは指定されたフォルダの、次のメッセージを表示する。
packf	フォルダを単一のファイルに圧縮する (各メッセージは通常、個別のファイルとして保存される)。packf コマンドを pack コマンドと混同してはならない。
pick	内容、シーケンス名、または他の基準に基づいてメッセージを選択する。
prev	現在のフォルダの、前のメッセージを表示する。

表 11-1: MHプログラムのコマンド (続き)

コマンド	説明
<code>prompter</code>	メッセージ作成用の簡易エディタを呼び出す。 <code>prompter</code> コマンドは、 <code>comp</code> 、 <code>dist</code> 、 <code>forw</code> 、および <code>repl</code> によって呼び出されるため、 <code>prompter</code> を直接呼び出す必要はない。
<code>rcvstore</code>	標準入力からのメッセージを直接フォルダに挿入する。
<code>refile</code>	現在のフォルダから、1 つまたは複数のフォルダにメッセージを移動する。
<code>repl</code>	現在のメッセージまたは指定するメッセージに応答する。
<code>rmf</code>	フォルダ内のすべてのメッセージを削除した後、フォルダ自身も削除する。
<code>rmm</code>	フォルダからメッセージを削除する。メッセージ・ファイルは実際には消去されない。代わりに、 <code>rmm</code> は、ファイル名の先頭に番号記号 (#) を付加することによってファイル名を変更する。ほとんどのホストでは、名前が番号記号で始まるファイルは、1 日に 1 度自動的に削除される。実際に削除されるまでは、 <code>mv</code> コマンドを使用してファイル名を変更することにより、削除されたメッセージを回復することができる。
<code>scan</code>	フォルダ内のメッセージのリストを表示する。
<code>send</code>	<code>comp</code> 、 <code>prompter</code> 、またはその他のエディタを使用して作成したメッセージを送信する。
<code>show</code>	メッセージの内容を表示する。
<code>sortm</code>	フォルダ内のメッセージを、メッセージ・ヘッダの <code>Date:</code> フィールドに基づいて、時間順にソートする。
<code>whatnow</code>	作成したばかりのメッセージの処理方法を指定するように求める。ユーザは、応答する元のメッセージを再度調べたり、新しいメッセージの編集を再開したり、または、メッセージの送信に関連する他のタスクを実行することができる。
<code>whom</code>	メッセージのヘッダをアドレスにまで拡張し、オプションとしてメッセージがそれらのアドレスに送信されたかどうかを検査する。

次の例は、MH の `msgchk` コマンドによる新規メッセージの報告を示しています。

```
orange% msgchk
You have new mail waiting, last read on date
```

ユーザの最上位ディレクトリに `.mh_profile` ファイルを作成して、MH の機能を変更することができます。MH リファレンス・ページに、ユーザが変更できる機能についての説明があります。

11.8 write を使用したメッセージの送受信

`write` コマンドを使用すると、同一または異なるホスト上の 2 人のユーザが、ビデオ表示端末、またはメッセージを紙に印刷する方式のビデオ以外のデバイス (テレタイプライタなど) 上での通信が可能になります。

`write` コマンドでは、電話による通信が不可能な相手にメッセージを送ることができます。特に、応答を必要としない場合に便利です。11.9 節で説明する `talk` コマンドも参照してください。

`write` コマンドは受信者の端末画面にメッセージを表示します。ユーザは、ホーム・ディレクトリの `.login` ファイルに `mesg n` コマンドを入力すると、`write` および `talk` を介して通信できないようにすることができます。スーパーユーザ特権を持つユーザからのメッセージは、着信不能にすることはできません。

ローカル・ホストのユーザが `write` および `talk` コマンドからのメッセージの受信を不能にしているかどうかを確認するには、`finger` コマンドを使用します。出力結果の最初の行に `messages off` 句が表示されているかどうかを調べてください。次に例を示します。

```
Login name: smith      (messages off)  In real life: John Smith
      :
```

リモート・ホストのユーザでは、`write` および `talk` が使用不能の場合、出力行の TTY フィールドにアスタリスク (*) が表示されます。次に例を示します。

```
      :
Login      Name      TTY Idle      When      Office
chang      Peter Chang  *p1 2:58 Thu 10:16  103
```

詳細は 10.2.2 項および `finger(1)` リファレンス・ページを参照してください。

送信しようとしている相手が、干渉を防ぐために、一時的に `write` コマンドを使用不能にするコマンドを実行していることがあります。その場合、

送信者は、受信者が明示的に `write` を無効にしている場合と同様、次のようなメッセージを受け取ります。

```
Write: Permission denied
```

これは、受信者が明示的に `write` を使用不能にした場合と同じです。

`write` コマンドは、受信者がログインしている場合にだけ使用できます。10.1 節で説明しているように、`who` コマンドを使用して現在のユーザをリストします。たとえば、ユーザ `smith` がログインしていないときに、`write` を使用してメッセージを送信すると、次のようなメッセージがユーザの端末画面に表示されます。

```
smith is not logged on
```

次の手順は、ユーザ `wang` とユーザ `chung` がともにローカル・ホスト `dancer` にログインしている場合に、ユーザ `wang` がユーザ `chung` にメッセージを送信する方法を示しています。

1. ユーザ `wang` は、システム・プロンプトから `write` コマンドを入力する。

```
dancer% write chung
```

`write` プログラムはベルを鳴らし、次のメッセージを `chung` の端末画面に送信します。

```
message from wang tty04 Feb 14 10:32:45
```

接続が確立されると、ユーザ `wang` の端末のベルが鳴ります。

2. ユーザ `wang` はメッセージをタイプする。

各行ごとに `Return` キーを押し、`Ctrl/D` を押してメッセージを終了します。たとえば、`wang` は、ユーザ `chung` に、次のような 2 行のメッセージを送信します。

```
The double-sided lab printer is working. Return  
Re-send your job and I'll check it. Return  
Ctrl/D
```

3. `wang` が `Ctrl/D` を押す。

ユーザ `chung` の画面に EOF (ファイルの終端) シグナルが表示されて、メッセージの終わりを知らせます。

注意

Ctrl/D で受信者の画面に end-of-file (EOF) シグナルが表示されない場合、または、送信者の端末でベルが鳴らない場合は、システム管理者に問い合わせてください。

新しいメッセージ行の先頭で感嘆符 (!) を使用すると、シェル・プロンプトにアクセスして、任意のオペレーティング・システムのコマンド (write コマンドを含む) を実行することができます。たとえば、wang が、chung に現在のディレクトリからファイルを検索するように指示しなければならない場合に、現在のディレクトリ名を忘れてしまっても、次のように !pwd コマンドを入力して、調べることができます。

```
dancer% write chung
You can copy the network user files from: Return
!pwd Return
/ufs/usr/staff/r0/net-dir/network_comm
!
/ufs/usr/staff/r0/net-dir/network_comm
Ctrl/D
dancer%
```

write コマンドは対話式で使用できますが、メッセージの送信者および受信者ともに、いつ相手のメッセージが終了して、応答を待っているかどうかを判断するのは容易ではありません。たとえば、wang が次のコマンドを入力したとします。

```
dancer% write chung
```

Wang は、この時点で chung の応答を待ちますが、chung も、wang が引き続きメッセージを送信すると思って待つかもしれません。

問題を最小限に抑えるために、write を対話式で使用するときは、簡潔で一時的なプロトコルを決めるとよいでしょう。たとえば、ユーザ wang は、chung に送信するメッセージを次のように始めます。

```
dancer% write chung
I'll mark the end of each message with 'ZZZ' Return
and wait for a reply. Please do the same. Return
I'll install a driver for the new printer. Return
Do you want to test it? ZZZ Return
```

詳細は、`write(1)` リファレンス・ページを参照してください。

11.9 `talk` を使用したメッセージの送受信

`talk` コマンドを使用すると、同一のホストまたはリモート・ホスト上の別のユーザに、対話式で、`write` を使用するよりも容易に、メッセージを送信することができます。ただし、`talk` は、ビデオ表示端末だけで使用できます。

`write` コマンド同様、`talk` を使用して、電話で連絡できない相手にメッセージを送信することができます。また、`write` コマンドと同様に、`talk` コマンドは、メッセージを直接端末に送信して、応答が入力されるまで送信を続行するため、受信者の作業を中断させる場合があります。

`talk` (および、前述のように `write`) からのメッセージ (スーパーユーザ特権を持つユーザからのメッセージを除く) の受信を不能にするには、`.login` ファイルに `mesg n` コマンドを入力します。受信者がこれを実行しているかどうかを確認するには、10.2.1 項または `finger(1)` リファレンス・ページで説明しているように、`finger` コマンドを使用してください。

オンラインの `talk` セッション時には、送信ウィンドウおよび受信ウィンドウが各ユーザの端末でオープンされます。`talk` が受信ウィンドウに他のユーザがタイプしている内容を表示する間に、ユーザは送信ウィンドウにタイプすることができます。

たとえば、同じローカル・ホスト `apple` 上のユーザ `hoover` にメッセージを送信するには、ユーザ `coolidge` は、次の `talk` コマンドを入力します。

```
apple% talk hoover
```

すると、プログラムは、`coolidge` の端末画面を二分割して、上半分を `coolidge` に、下半分を `hoover` にそれぞれ割り当てます。

その後、次のメッセージが画面の最上部に表示されます。

```
[No connection yet]
```

接続が確立すると、次のメッセージが表示されます。

```
[Waiting for your party to respond]
```

このメッセージ後に、`hoover` の端末でベルが鳴り、次のメッセージが表示されます。

```
Message from Talk_Daemon@apple at 16:18 ...
talk: connection requested by coolidge@apple
```



```
talk: respond with: talk coolidge@apple
```

hoover がすぐに応答しない場合は、次のメッセージが coolidge の画面に表示されます。

```
[Ringing your party again]
```

hoover が応答すると、接続が確立したことを通知するメッセージが coolidge の画面に表示されます。各ユーザはこの時点でテキストを入力することができます。画面がいっぱいになると、talk は、画面の初め部分からテキストを重ね書きしていきます。どちらのユーザも、Ctrl/C を押して通信を終了することができます。talk セッションの終了時には、次のように表示されます。

```
[Connection closing. Exiting]
```



他のホストへのファイルのコピー

この章では、オペレーティング・システムのコマンドを使用して、次のタスクを実行する方法を説明します。

- ローカル・ホストとリモート・ホストの間でファイルをコピーする (12.1 節)。
- ローカル・ホストとリモート・ホストの間でディレクトリ全体 (サブディレクトリを含む) のファイルをコピーする (12.2 節)。
- 2つのリモート・ホスト間でファイルをコピーする (12.3 節)。

ファイルをコピーするためにリモート・ホストのホスト名またはオンライン状態を確認する場合は、第 10 章で説明した `finger`、`who`、`rwho`、`ping`、または `ruptime` コマンドを使用してください。

この章で説明する方法の他にも、第 14 章で、UNIX 間コピー・プログラム (UUCP) を使用して、リモート・システム間でファイルをコピーする方法について説明します。

注意

リモート・ホストのセキュリティ機能によって、ファイルをコピーできるかどうかが決まります。ファイルをコピーできない場合は、システム管理者に問い合わせてください。

12.1 ローカル・ホストとリモート・ホスト間でのファイルのコピー

次のコマンドを使用して、ローカル・ホストとリモート・ホストの間でファイルをコピーすることができます。

- `rcp` コマンド
- 12.1.1 項で説明します。ディレクトリ全体のファイルをコピーする場合は、12.2 節を参照してください。Tru64 UNIX オペレーション

グ・システムを実行しているホストは、他の UNIX ベースのオペレーティング・システムを実行しているホストとの間で、`rcp` コマンドを使用することができます。

- `ftp` コマンド

12.1.2 項で説明します。`ftp` を使用すると、`ftp` をサポートするオペレーティング・システムを使用しているホスト間で、ファイルをコピーすることができます。

- `mailx` コマンド

12.1.3 項で説明します。

- `write` コマンド

12.1.4 項で説明します。

12.1.1 `rcp` を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー

`rcp` を使用して、ローカル・ホストからリモート・ホストに、またはリモート・ホストからローカル・ホストにファイルをコピーする際には、次の `rcp` 構文のように、最初にコピー元のファイルを指定し、次に、コピー先のファイルを指定します。

`rcp` [*option...*] *localfile* *hostname:file*

localfile 変数には、コピーするローカル・ファイルを指定します。
hostname:file 変数については、*hostname* にリモート・ホストを指定し、コロン (:) に続けて、*file* にローカル・ファイルのコピー先のファイル名を指定します。

次の例では、`rcp` を使用して、ローカル・ファイル `YTD_sum` を、ローカル・ホストの `/usr/reports` ディレクトリから、リモート・ホスト `moon` の `/usr/acct` ディレクトリの `year-end` ファイルにコピーします。

```
% rcp /usr/reports/YTD_sum moon:/usr/acct/year-end
```

ローカル・ホスト上のファイルをリモート・ホストのユーザへ転送することもできます。次の例では、ファイル `YTD_sum` を、ローカル・ホスト上の `/usr/reports` ディレクトリから、リモート・ホスト `moon` 上の `jones` というユーザのホーム・ディレクトリの `acct_summaries` ファイルにコピーします。

12-2 他のホストへのファイルのコピー

```
% rcp /usr/reports/YTD_sum jones@moon:acct_summaries
```

前述の各例のように使用すると、rcp コマンドは、元のファイルから作成したファイルに、新しい作成日と作成時間を割り当てます。また、rcp コマンドは、新しく作成したファイルが置かれているホストまたはユーザ・ディレクトリに応じて、ファイルの読み取り、書き込み、実行の許可も割り当てます。

コピーした元のファイルの作成日時およびアクセス許可モードを新しいファイルでも変えたくない場合、次の例のように -p オプションを指定することにより、YTD_sum の元の作成日時およびファイル・アクセス許可を year-end ファイルに継承することができます。

```
% rcp -p /usr/reports/YTD_sum moon:/usr/acct/year-end
```

-p オプションを指定しない場合には、新しい作成日時が割り当てられ、ファイルのアクセス許可は、システム管理者がリモート・ホスト moon に割り当てた省略時の値に設定されます。

次の例では、-p オプションにより、元のファイル YTD_sum と同じファイル作成日時およびアクセス許可が acct_summaries ファイルに継承されます。

```
% rcp -p /usr/reports/YTD_sum jones@moon:acct_summaries
```

-p オプションを指定しない場合は、新しい日付と時間が割り当てられますが、前の例とは異なり、ファイルのアクセス許可は、ユーザ jones が .login または .profile ファイル内に umask コマンドを使用して省略時の値を設定している場合には、その値に設定されます。umask が .login または .profile ファイルに設定されていない場合は、ファイルのアクセス許可モードは、リモート・ホスト moon の省略時の値になります。umask の設定についての詳細は、umask(1) を参照してください。

リモート・ホストからローカル・ホストにファイルをコピーする場合、次のような rcp 構文を使用してください。localfile がコピー先ファイルであるために、コマンド行の最後に localfile が記述されている点以外、コマンド構文はローカル・ファイルをリモート・ホストにコピーする場合と同様です。

```
rcp [option...] hostname:file localfile
```

12.1.2 ftp を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー

ftp コマンドはファイル転送プロトコル (FTP) へのインタフェースです。このコマンドには、ファイルのコピーというメイン・タスクをサポートする 拡

張サブコマンド (表 12-1, 表 12-2, および表 12-3 で説明) があります。
ftp コマンドを使用すると, ftp をサポートする 2 つのホスト間で, ファイルをコピーすることができます。

ftp コマンド・オプションの詳細については, ftp(1) リファレンス・ページを参照してください。これらのオプションは, 主にネットワーク管理タスクに使用します。

FTP を使用してファイルをコピーするには, 次の手順で行います。

1. リモート・ホスト上にセッションを確立する。
2. ファイルをコピーする。
3. セッションを切断する。

ftp コマンドの構文は, 次のとおりです。

ftp *host_name*

host_name 変数には, 目的のホスト名を指定します。コマンド行に *host_name* を指定しない場合は, ftp サブコマンド open (表 12-1 で説明) を使用して, リモート・ホストと接続しなければなりません。

ftp と入力すると, ftp> プロンプトが表示されます。これで, リモート・ホストにログインできています。ここで, ftp サブコマンドを使用して, 次のタスクを実行することができます。

- ファイルをコピーする (表 12-1 参照)。
- ローカル・ファイルをリモート・ファイルに追加する (表 12-1 参照)。
- 複数のファイルをコピーする (表 12-1 参照)。
- リモート・ディレクトリの内容をリストする (表 12-2 参照)。
- リモート・ホスト上の現在のディレクトリを変更する (表 12-2 参照)。
- リモート・ホスト上のファイルを削除する (表 12-2 参照)。
- ローカル・シェルにエスケープしてコマンドを実行する (表 12-3 参照)。

例 12-1 に, ローカル・ホスト earth 上のユーザ alice がリモート・ホスト moon にログインし, ftp サブコマンドを使用して, 現在の作業ディレクトリのチェック, その内容の一覧表示, バイナリ・ファイルのコピー, およびセッションの終了を行う方法を示します。

例 12-1: ftp を使用したファイルのコピー

```
earth% ftp moon [1]
Connected to moon
220 moon FTP server (Version . . .) ready [2]
Name (moon:alice): Return [3]
Password: [4]
230 User alice logged in [5]
ftp> binary [6]
200 Type set to I
ftp> pwd [7]
257 "u/alice" is current directory
ftp> ls -l [8]
200 PORT command successful.
150 Opening data connection for /bin/ls (192.9.200.1,1026) (0 bytes)
total 2

-rw-r--r--  1 alice    system      101 Jun  5 10:03 file1

-rw-r--r--  1 alice    system      171 Jun  5 10:03 file2

-rw-r--r--  1 alice    system     1201 Jun  5 10:03 sales
ftp> get sales newsales [9]
200 PORT command successful.
150 Opening data connection for sales (192.9.200.1,1029) (1201 bytes)
226 Transfer complete.
local:sales remote:newsales
ftp> quit [10]
221 Goodbye.
earth%
```

- [1] ユーザ alice は、ローカル・ホスト earth のプロンプトから ftp コマンドを入力して、リモート・ホスト moon との ftp セッションを開始します。
- [2] 接続を確認するメッセージが、ローカル・ホスト上に表示されます。
- [3] ユーザ alice は、自分のログイン名がリモート・ホスト上のログイン名と一致しているので、プロンプトに対して Return キーを押します。
- [4] Password: プロンプトに対し、ユーザ alice は、パスワードを正しく入力しますが、パスワードは表示されません。
- [5] リモート・ホストへのログインが確認され、ftp> プロンプトが表示されて、リモート・ホストとの ftp セッションが確立されます。
- [6] ユーザ alice が ftp> プロンプトに対し binary サブコマンドを入力して、ファイル転送タイプをバイナリに設定すると、FTP によって、200 Type set to I という確認のメッセージが表示されます。

- 7 ユーザ `alice` は、`pwd` サブコマンドを入力して、現在の作業ディレクトリを確認すると、FTP によって、`u/alice is current directory` というメッセージが表示されます。
- 8 ユーザ `alice` は、`ls -l` サブコマンドを入力して、現在の作業ディレクトリに入っている、`file1`、`file2`、および `sales` を一覧表示します。
- 9 ユーザ `alice` は、`get` サブコマンドを使用して、ファイル `sales` をリモート・ホストからローカル・ホスト上の `newsales` というファイルにコピーします。
- 10 ユーザ `alice` は、`quit` サブコマンドを入力して `ftp` セッションを終了し、ローカルのシステム・プロンプトに戻ります。

注意

ファイル転送は、リモート・ホスト上のセキュリティ機能に依存します。 ファイルをコピーできない場合は、システム管理者に問い合わせてください。

表 12-1 に、ファイルをコピーしたり、`ftp` を終了させる `ftp` サブコマンドを示します。 ここで説明する `binary`、`get`、および `quit` サブコマンドは、例 12-1 で使用されています。

表 12-1: ホストへの接続とファイルのコピーを行う `ftp` サブコマンド

サブコマンド	説明
<code>account [password]</code>	Tru64 UNIX リモート・ホストから、そのホストのリソースにアクセスするために必要なパスワードを送信する。 <code>password</code> を指定しなければ、ユーザはパスワードの入力を要求される。パスワードは画面上には表示されない。
<code>ascii</code>	ファイル転送タイプを省略時の値であるネットワーク ASCII に設定する。たとえば、PostScript ファイルは ASCII ファイルである。
<code>binary</code>	ファイル転送タイプをバイナリ・イメージに設定する。これは、非 ASCII ファイルをコピーするときに必要である。たとえば、実行可能ファイルは非 ASCII ファイルである。

表 12-1: ホストへの接続とファイルのコピーを行う ftp サブコマンド (続き)

サブコマンド	説明
bye	ファイル・コピー・セッションを終了して、FTP を出る。quit と同じ。
get remfile locfile	リモート・ファイル <i>remfile</i> をローカル・ホストのファイル <i>locfile</i> にコピーする。 <i>locfile</i> を指定しない場合は、リモート・ファイル名が、ローカル・ファイル名にそのまま使用される。runique サブコマンドも参照。
mget remfile [locfile]	1 つまたは複数の指定ファイル (<i>remfile</i>) を、リモート・ホストからローカル・ホストの現在のディレクトリの <i>locfile</i> にコピーする。ワイルドカードや、パターン照合を行うメタキャラクタもサポートされている。
mput locfile [remfile]	1 つまたは複数の指定ファイル (<i>locfile</i>) を、ローカル・ホストからリモート・ホスト上の <i>remfile</i> にコピーする。ワイルドカードや、パターン照合を行うメタキャラクタもサポートされている。
open host [port]	<i>host</i> をコマンド行で指定しなかった場合に、そのホストとの接続を確立する。 <i>port</i> を指定すると、FTP はそのポートでサーバとの接続を試みる。autologin 機能が設定されている場合には (省略時の設定)、FTP はユーザをリモート・ホストへログインさせようとする。
put locfile [remfile]	ローカル・ホスト上のファイル <i>locfile</i> を、リモート・ホスト上の <i>remfile</i> にコピーする。 <i>remfile</i> を指定しない場合、FTP はリモート・ファイル名としてローカル・ファイル名を使用する。sunique サブコマンドも参照。
quit	ファイル・コピー・セッションを終了して、FTP を出る。bye と同じ。
recv remfile [locfile]	リモート・ホスト・ファイル <i>remfile</i> を、ローカル・ホストのファイル <i>locfile</i> にコピーする。recv は、get と同様に動作する。

表 12-1: ホストへの接続とファイルのコピーを行う ftp サブコマンド (続き)

サブコマンド	説明
runique	<p>get 処理時に、ローカルのコピー先ファイルに対して一意のファイル名を作成するかどうかの切り替えを行う。この一意のローカル・ファイル名機能がオフ(省略時の設定)の場合、FTP は、ローカル・ファイルを重ね書きする。オンの場合は、ローカルのコピー先ファイルに指定したものと同名のローカル・ファイルがあれば、FTP は、指定したローカルのコピー先ファイル名に拡張子 .1 を付加する。</p> <p>すでに .1 の付いた同名のローカル・ファイルがある場合には、FTP は、指定した名前に拡張子 .2 を付加する。この処理は、99 になるまで同様に繰り返される。それでもFTPが一意の名前を付けることができない場合には、エラーが報告され、ファイルのコピーは行われない。</p> <p>なお、runique は、シェル・コマンドから作成したローカル・ファイル名には影響しない。</p>
send locfile [remfile]	<p>ローカル・ファイル locfile を、リモート・ホストの remfile ファイルに格納する。</p> <p>send は、put と同様に動作する。</p>
sunique	<p>put 処理時に、リモートのコピー先ファイルに対して一意のファイル名を作成するかどうかの切り替えを行う。この一意のリモート・ファイル名機能がオフ(省略時の設定)の場合、FTP は、リモート・ファイルを重ね書きする。オンの場合は、リモートのコピー先ファイルに指定したものと同名のリモート・ファイルがあれば、runique と同様に、リモートの FTP server が、リモートのコピー先ファイルの名前を修正する。ただし、FTP server が、リモート・ホスト上でサポートされていないなければならない。</p>

表 12-2 に、ファイルをコピーする前に、必要であれば、現在のディレクトリの確認、変更、作成、あるいはディレクトリ内のファイルの一覧表示を行う ftp サブコマンドについて説明します。ここで説明する pwd および ls サブコマンドは、例 12-1 で使用されています。

表 12-2: ディレクトリおよびファイル进行操作するための ftp サブコマンド

サブコマンド	説明
<code>cd remotedir</code>	リモート・ホスト上の作業ディレクトリを <i>remotedir</i> に変更する。
<code>cdup</code>	リモート・ホスト上の作業ディレクトリを現在のディレクトリの親ディレクトリに変更する。
<code>delete remfile</code>	指定したリモート・ファイルを削除する。
<code>dir [remdir] [locfile]</code>	リモート・ディレクトリ <i>remdir</i> にあるファイルのリストを、ローカル・ホスト上のファイル <i>locfile</i> に書き込む。
<code>lcd [directory]</code>	ローカル・ホスト上の作業ディレクトリを変更する。 <i>directory</i> を指定しない場合は、ユーザのホーム・ディレクトリに移動する。
<code>ls [remdir] [locfile]</code>	リモート・ディレクトリ <i>remdir</i> の簡潔なフォーマットの一覧表示を、ローカル・ホスト上のファイル <i>locfile</i> に書き込む。
<code>mkdir [remdir]</code>	指定したディレクトリをリモート・ホストに作成する。
<code>pwd</code>	リモート・ホストの現在のディレクトリ名を表示する。
<code>rename from to</code>	リモート・ホスト上のファイルの名前を変更する。
<code>rmdir remdir</code>	リモート・ホストからリモート・ディレクトリ <i>remdir</i> を削除する。

表 12-3 に、直接、または ftp 内からシェルを呼び出すことによって、ヘルプおよび状態情報を提供する ftp サブコマンドについて説明します。

表 12-3: ヘルプおよび状態情報を得るための ftp サブコマンド

サブコマンド	説明
<code>!command [option]</code>	ローカル・ホスト上に対話式のシェルを呼び出す。
<code>?</code>	<i>subcommand</i> についてのヘルプ・メッセージを表示する。 <i>subcommand</i> を指定しない場合、FTP は利用できるサブコマンドの一覧を表示する。 <code>help</code> サブコマンドも参照。
<code>help [subcommand]</code>	ヘルプ情報を表示する。 <code>?</code> サブコマンドも参照。

表 12-3: ヘルプおよび状態情報を得るための ftp サブコマンド (続き)

サブコマンド	説明
status	現在の転送モード (ASCII またはバイナリ)、接続状態、タイムアウト値などを含む、ftp の現在の状態を表示する。
verbose	詳細 (verbose) モードを切り替える。詳細モードがオン (省略時の設定) の場合、FTP は、リモートの FTP server から受け取るすべての応答を表示する。また、終了したすべてのファイル転送の統計情報も表示する。

ファイルをコピーする別の方法として、簡易ファイル転送プロトコル (TFTP) へのインタフェースである tftp コマンドがあります。ftp とは異なり、ファイルのコピー以外のタスクを実行するためのサブコマンドはありません。スーパーユーザまたはオペレーティング・システムのインストール担当者のタスク (たとえば、オペレーティング・システムのカーネルのコピーなど) を実行する場合にだけ使用してください。これは、リモートの tftp サーバ・デーモン tftpd へのファイル・アクセス権が制限されているためです。詳細は、tftp(1) リファレンス・ページを参照してください。

12.1.3 mailx を使用したローカル・ホストとリモート・ホスト間での ASCII ファイルのコピー

mailx コマンドは、第 11 章で説明しているように、メール・メッセージを送受信するために最も頻繁に使用されますが、ASCII ファイルをローカル・ホストまたはリモート・ホストにコピーすることもできます。次の構文で示すように、左山カッコのリダイレクション記号 (<) を使用すると、mailx により、ASCII ファイルを 1 人または複数のユーザにコピーすることができます。

mailx [*option...*] *recipient...* < *filename*

recipient 変数には、ファイル *filename* の送り先である 1 人または複数のユーザ名、あるいは mailx の別名を指定します。

たとえば、ファイル *schedule* を複数のユーザに送信するには、次のように、メッセージの表題を示す -s オプションを指定して、mailx コマンドを使用します。

```
% mailx -s "games" tom jeff craig jim ken < schedule
```

Wombats というチームの 5 人のメンバに対して `wombats` というメール別名を作成する (11.6.1 項参照) と、次に示すように、ファイルをその別名に送信することができます。

```
% mailx -s "games" wombats < schedule
```

12.1.4 `write` を使用したローカル・ホストとリモート・ホスト間でのファイルのコピー

`write` コマンドは、11.8 節で説明しているように、他のユーザにメッセージを書き込むために最もよく使用されますが、ローカル・ホストまたはリモート・ホストにファイルをコピーすることもできます。`write` とタイプした後に、受信者のユーザ名、左山カッコのリダイレクション記号 (<)、および送信するファイル名を入力してください。たとえば、現在のディレクトリにある `letter` という名前のファイルをユーザ `maria` に送信するには、ホストのプロンプトで次のコマンドを入力します。

```
% write maria < letter
```

12.2 ローカル・ホストとリモート・ホスト間でのディレクトリのコピー

`-r` オプションを指定して `rcp` コマンドを使用すると、ファイルの入っているディレクトリ全体 (つまり、すべてのサブディレクトリ内のファイルおよびディレクトリも含む) を、ローカル・ホストとリモート・ホストの間で再帰的にコピーすることができます。

ローカル・ホストからリモート・ホストにディレクトリを再帰的にコピーするには、次の構文を使用します。

```
rcp -r localdirectory hostname:directory
```

`localdirectory` 変数には、再帰的にコピーするローカル・ディレクトリを指定します。`hostname:directory` 変数については、`hostname` にリモート・ホストを指定し、コロンに続けて、`directory` にローカル・ディレクトリのコピー先のリモート・ディレクトリ名を指定します。

次の例では、`rcp -r` を使用して、`/usr/reports` ディレクトリを、ローカル・ホストからリモート・ホスト `moon` 上の `/user/status/newdata` ディレクトリに再帰的にコピーします。

```
% rcp -r /usr/reports moon:/user/status/newdata
```

ローカル・ホスト上のディレクトリを、リモート・ホストのユーザに再帰的にコピーすることもできます。次の例に、ローカル・ホスト上のディレクトリ `/usr/reports` を、リモート・ホスト `moon` のユーザ `smith` のホーム・ディレクトリにある `/user/status/newdata` というディレクトリにコピーする方法を示します。この例では、12.1.1 項で説明しているように、`-p` オプションを使用して、新しいディレクトリにコピーされるディレクトリおよびファイルに元の作成日時とアクセス許可モードを残しています。

```
% rcp -p -r /usr/reports smith@moon:/user/status/newdata
```

リモート・ホストからローカル・ホストにディレクトリを再帰的にコピーするには、次に示すような `rcp` 構文に従ってください。 `localdirectory` がコピー先ファイルであるためにコマンド行の最後に記述されている以外は、このコマンドの構文は、ディレクトリをローカル・ホストからリモート・ホストに再帰的にコピーする場合と同じです。

```
rcp -r hostname:directory localdirectory
```

12.3 2つのリモート・ホスト間でのファイルのコピー

ローカル・ホストから `rcp` を使用して、あるリモート・ホスト上のファイルを別のリモート・ホスト上のファイルにコピーすることができます。これは、次の `rcp` 構文を使用して行います。

```
rcp remhost1:filesend remhost2:file-recv
```

`remhost1` 変数には、送信するファイルのあるリモート・ホストを指定します。この後ろにコロンの (:), および送信するファイル `filesend` が続きます。 `remhost2:file-recv` については、`remhost2` にコピー先のリモート・ホストを指定し、`file-recv` に `remhost1` からのファイルのコピー先であるファイル名を指定します。 `file-recv` にディレクトリ名だけを(次の例に示すように)指定すると、`filesend` は、同じファイル名でそのディレクトリにコピーされます。

次の例では、`rcp` を使用して、`spark` ファイルを、リモート・ホスト `flint` 上の `/u/cave/fred` ディレクトリから、リモート・ホスト `stone` 上の `/u/hut/barney` ディレクトリにコピーしています。

```
% rcp flint:/u/cave/fred/spark stone:/u/hut/barney
```

リモート・ホストでの作業

この章では、次の作業を可能にするコマンドの使用方法について説明します。

- ローカル端末からリモート・ホストにログインする (13.1 節)。
- リモート・ホスト上で指定したコマンドを実行する (13.2 節)。
- Telnet プロトコルを使用して、リモート・ホストにログインする (13.3 節)。 `rlogin` がサポートされていない場合は、代わりに `telnet` を使用する。

注意

すべてのリモート・ログインは、リモート・ホスト上のセキュリティ機能の影響を受けます。リモート・ホストにログインできない場合は、システム管理者に問い合わせてください。

これらのコマンドを使用する前に、正確なホスト名や、リモート・ホストへのアクセスが現在可能であるかどうかを調べる必要があります。第 10 章で説明している `finger`、`who`、`rwho`、`ruptime`、および `ping` コマンドを使用して、これらの情報を取得してください。

13.1 `rlogin` を使用したリモート・ホストへのログイン

`rlogin` を使用して、リモート・ホストにログインできます。コマンドの構文は次のとおりです。

`rlogin` `[-l user]` *host_name*

`-l` オプションを使用すると、自分のローカル・ユーザ名以外のリモート・ユーザ名を指定できます。*host_name* 変数には、リモート・ホストを指定します。

次に、ローカル・ホストと同じログイン名があるリモート・ホスト `boston` にログインする手順を示します。

1. リモート・ホスト名を指定して、`rlogin` コマンドを入力する。

次に例を示します。

```
% rlogin boston
Password:
```

2. パスワードを入力する。

システム・プロンプトが表示されると、リモート・ホストにログインできたので、コマンドを入力することができます。

3. 接続をクローズしてローカル・ホストに戻るには、`Ctrl/D` を押す。

ローカル・ホストとはログイン名が異なるリモート・ホスト上にアカウントがある場合は、次の手順に示すように、`-l` オプションを使用して、リモート・ホストにログインしなければなりません。

1. リモート・ログイン名とリモート・ホスト名を指定して、`rlogin -l` コマンドを入力する。

次に例を示します。

```
% rlogin -l celtic boston
Password:
```

2. ログイン名 `celtic` 用のパスワードを入力する。

システム・プロンプトが表示されると、リモート・ホストにログインできたので、コマンドを入力することができます。

3. 接続をクローズしてローカル・ホストに戻るには、`Ctrl/D` を押す。

次の場合には、`rlogin` はパスワードの入力を求めません。

- リモート・ホスト上の `/etc/hosts.equiv` ファイル内にローカル・ホストがリストされている場合
- リモート・ホスト上のホーム・ディレクトリにある `.rhosts` ファイルにホスト名 (および必要に応じてユーザ名) がリストされている場合

`rlogin` についての詳細は、`rlogin(1)` リファレンス・ページを参照してください。

13.2 rsh を使用したリモート・ホスト上でのコマンドの実行

rsh コマンドを使用すると、ログインしなくても、UNIX ベースのリモート・ホスト上で1つのコマンドを実行することができます。複数のコマンドを続けて実行する必要がある場合は、rlogin または telnet を使用して、リモート・ホストにログインしなければなりません。

rsh コマンドの構文は次のとおりです。

rsh [-l *user*] *host_name* *command*

-l オプションを使用すると、ログイン名 *user* がローカル・ホストとは異なるリモート・ホストにログインすることができます。-l オプションを指定しなければ、rsh は、ローカル・ホストとリモート・ホストのログイン名が同じであるとみなします。*host_name* 変数には、リモート・ホスト名を指定します。*command* 変数には、実行するコマンドを指定します。

注意

リモートで実行するコマンドを指定しない場合、rsh はリモート・ホストへのログイン情報の入力を求めます。

rsh を使用するには、次に示す条件のどちらかがあてはまらなければなりません。

- リモート・ホスト上の /etc/hosts.equiv ファイルにローカル・ホストがリストされている。
- リモート・ホスト上のホーム・ディレクトリにある .rhosts ファイルにホストがリストされている。

次の例では、rsh は、リモート・ホスト上のファイルをローカル・ホスト上のファイルに追加します。ホスト remhost2 上のリモート・ファイル remfile2 は、ローカル・ファイル locfile に追加されます。

```
% rsh remhost2 cat remfile2 >> locfile
```

13.3 telnet を使用したリモート・ホストへのログイン

telnet コマンドを使用して、リモート・ホストにログインすることができます。telnet コマンドは Telnet プロトコルをインプリメントしています。

telnet を使用すると、次の処理を行うことができます。

- リモート・ホストにログインする。
- リモート・ホスト上のオペレーティング・システム・コマンドを実行する。
- telnet サブコマンド (表 13-1 参照) を入力して、リモート・セッションの管理を行う。

telnet コマンドの構文は次のとおりです。

telnet [*host_name* [*port*]]

host_name 変数には、リモート・ホストを指定します。ホスト名を省略した場合は、Telnet ユーティリティの起動後に、open サブコマンドを使用して、接続を確立することができます。例 13-1 に、telnet コマンドを使用してリモート・ホスト star にログインし、telnet サブコマンドの status を使用し、接続を終了する手順を示します。

port を指定しない場合、Telnet プロトコルは、省略時のポートで Telnet サーバと接続しようとします。

例 13-1: telnet コマンドの使用

```
% telnet star[1]
Trying 16.69.224.1...[2]
Connected to star.milkyway.galaxy.com.[3]
Escape character is '^]'.

(star.milkyway.galaxy.com) (ttyra)[4]

login: username[4][5]
Password: [6][7]

% ^][8]
telnet> status[8][9]
Connected to star.milkyway.galaxy.com.[10]
Operating in single character mode
Catching signals locally
Remote character echo
Local flow control
Escape character is '^]'.
[11]
% ^D[12]
Connection closed by foreign host.[13]
% telnet star[1][14]
Trying 16.69.224.1...
Connected to star.milkyway.galaxy.com.
Escape character is '^]'.

(star.milkyway.galaxy.com) (ttyra)

login: username
Password:
```

13-4 リモート・ホストでの作業

例 13-1: telnet コマンドの使用 (続き)

```
% ^] 8  
telnet> q 15  
Connection closed. 16
```

- 1 `host_name` に `stor` を指定して `telnet` コマンドを入力しています。省略時のポートが使用されます。
 - 2 `telnet` ユーティリティは、接続使用としているアドレスを認識しています。
 - 3 `telnet` ユーティリティは接続を完了し、接続したホストを認識しています。
 - 4 リモート・ホスト・システムは自分自身を認識しており、ユーザ・ログインのためのプロンプトを表示しています。
 - 5 そのホスト・システムにおけるユーザ名が入力されています。
 - 6 リモート・システムは、ユーザ・パスワードのためのプロンプトを表示しています。
 - 7 そのホスト・システムにおけるユーザ・パスワードを入力しています。セキュリティ上の理由で、入力したパスワードは画面には表示されません。
 - 8 省略時のエスケープ・シーケンス `Ctrl/]` が押され、`telnet` サブコマンド・プロンプト `telnet>` が表示されています。
 - 9 プロンプトに対して `status` サブコマンドが入力されています。
 - 10 何行かの状態情報が表示されています。表示される内容はシステムの構成によって異なります。
 - 11 Return キーを押すとリモート・ホスト・プロンプトが表示されます。
 - 12 ホストのプロンプトに `Ctrl/D` を入力し、Telnet セッションを終了させています。
 - 13 リモート・ホストによって接続がクローズされています。
 - 14 接続とログインが繰り返されています。
 - 15 `telnet>` サブコマンド・プロンプトに `q` サブコマンドが入力され、Telnet セッションが終了しています。 `Ctrl/D` を使用することもできます。
 - 16 ローカルの `telnet` ユーティリティは接続をクローズしています。
-

引数を指定しないで `telnet` コマンドを入力すると、`telnet>` プロンプトで示される `telnet` サブコマンド・モードにアクセスすることができます。

`telnet` サブコマンドについては、表 13-1 で説明します。サブコマンドを入力する前には、エスケープ・シーケンス `Ctrl/]` を入力しなければなりません。このシーケンスは、`telnet` プログラムに対して、続いて入力する情報がテキストでないことを知らせます。エスケープ・シーケンスを入力しなければ、`telnet` はサブコマンドをテキストとして解釈します。

各サブコマンドは、コマンドを一意に識別できるだけの英字を入力するだけでかまいません。たとえば、quit コマンドについては、q を入力するだけで十分です。telnet サブコマンドの全リストについては、telnet(1) リファレンス・ページを参照してください。

表 13-1: telnet サブコマンド

サブコマンド	説明
? [subcommand]	ヘルプ情報を表示する。サブコマンドを指定すると、そのサブコマンドについての情報が表示される。
close	telnet 接続をクローズして、telnet コマンド・モードへ戻る。
display [argument]	引数を指定しなければ、すべての設定値および切り替え値を表示する。指定した場合は、引数に対応する値だけをリストする。
open host [port]	指定したホストへの接続をオープンする。ホスト指定は、ホスト名またはドット表記 10 進数形式のインターネット・アドレスのどちらでもよい。ポートが指定されなければ、telnet は、省略時のポートで telnet のサーバに接続しようとする。
quit	接続をクローズし、telnet プログラムを終了する。コマンド・モードで Ctrl/D を押しても、接続をクローズして、終了することができる。
status	現在のモードや現在接続されているリモート・ホストを含めて、telnet の状態を示す。
z	SHELL 環境変数で指定するローカル・ホスト上のシェルをオープンする。Ctrl/D を押してシェルを終了すると、telnet はリモートのセッションに戻る。

UUCP ネットワーク・コマンド

この章では、UNIX 間コピー・プログラム (UUCP) について説明します。UUCP を使用すると、次の処理を行うことができます。

- リモート・ホスト上でタスクを実行する (14.3.2.3 項)。
- ローカル・ホストとリモート・ホスト間でファイルの転送を行う (14.5 節)。
- バックグラウンド・モードで作業を行う。
- ローカル・ホストとリモート・ホストの切り換えを行いながら、同時に、どちらか一方または両方のホストでタスクを実行する。

UUCP についての補足情報は、`uucp_intro(7)` リファレンス・ページを参照してください。UUCP システム管理およびタスクの概要については、『ネットワーク管理ガイド』を参照してください。

UUCP を使用すれば、非同期式ハード配線回線、ネットワーク、または電話回線 (両端でモデムを使用) を、次のものに接続することができます。

- 別のワークステーション
- UNIX ベースのオペレーティング・システムを実行する別のコンピュータ
- UNIX ベース以外のオペレーティング・システムを実行するコンピュータ (この場合には特別なハードウェアおよびソフトウェアが必要)。

14.1 UUCP パス名規約

UUCP パス名は、次に示す例外を除いて、使用するオペレーティング・システムの規約に準拠します。

- 相対パス名は、すべての UUCP コマンドで使用できるわけではありません。使用できない場合には、コマンドを再入力して、完全パス名を使用してください。

- UUCP をサポートするホスト上では、他のホストとの間でのデータ転送用に `/usr/spool/uucppublic` ディレクトリがセットアップされます。このディレクトリの簡潔な指定形式は `~uucp` です。
- リモート・ホスト上のファイルのパス名には、ホスト名の後ろに感嘆符 (!) を付けなければなりません。たとえば、`sea!/research/new` は、ホスト `sea` 上のディレクトリ `/research` にあるファイル `new` を指定します。
- ファイルを指定するときに、ネットワーク・パスの他にリモート・ホスト名が必要な場合があります。Bourne, Korn, または POSIX シェルでファイルを指定する場合には、各ホスト名の後に感嘆符 (!) を付けてください。たとえば、`gem!car!sea!/research/cells` は、ホスト `sea` 上のディレクトリ `/research` にあるファイル `cells` を指定します。これは、ホスト `car` を介して到達可能であり、`car` は、ホスト `gem` を介して到達可能です。C シェルでは、感嘆符 (!) の前にバックスラッシュ (\) を付けなければ、誤って解釈されることがあります。次のように記述してください。

```
gem\!car\!sea\!/research/cells
```

14.2 UUCP をサポートするホストの検索

ローカル・ホストから UUCP コマンドを使用してリモート・ホストと通信するには、UUCP プロトコルをサポートしているホストを確認しなければなりません。ローカル・ホストから UUCP コマンドを使用して通信できるすべてのホストのリストを表示するには、UUCP `uname` ユーティリティを使用します。次に、`uname` コマンドを使用した場合の出力例を示します。

```
% uname
elvis
fab4
Y107
```

この例では、3 つのリモート・ホストが、UUCP を使用してローカル・ホストにアクセス可能であることを示しています。ローカル・ホストを識別するには、`-l` オプションを指定して `uname` コマンドを使用します。たとえば、次のように入力します。

```
% uname -l
music
```

互換性にあるホスト間で UUCP コマンドを使用することにより、ホスト music 上のユーザは、ホスト elvis、fab4 または Y107 との間でファイルの送受信を行うことができます。

詳細は `uname(1)` リファレンス・ページを参照してください。

14.3 リモート・ホストへの接続

UUCP コマンドを実行する前に、ローカル・ホストをリモート・ホストに接続しなければなりません。リモート・ホストに接続するには、次の 3 のコマンドを使用することができます。

- `cu` コマンドおよび `tip` コマンド

これらのコマンドは全二重接続を確立するため、リモート・ホストに直接ログインしているような感触を得ることができます。この接続によって、ホスト間でのデータの同時転送が可能になります。

- `ct` コマンド

接続されている端末にダイヤルし、モデムおよび電話回線を介してログインすることにより、リモート接続を確立することができます。

注意

リモート接続は、ローカル・ホストおよびリモート・ホスト上のセキュリティ機能の影響を受けます。詳細についてはシステム管理者に問い合わせてください。

14.3.1 `cu` を使用したリモート・ホストへの接続

`cu` コマンドとそのオプションを使用すると、リモート・ホストに接続して、ログインした後、ローカル・ホストからリモート・ホスト上でタスクを実行することができます。2 つのホスト間で切り換えを行うことによって、両方のホスト上でタスクを実行することができます。ローカル・ホストとリモート・ホストが同じオペレーティング・システムを使用している場合には、ローカル・ホストからリモート・ホストにコマンドを入力することもできます。

14.3.1.1 cu を使用した名前によるリモート・ホストへの接続

次の手順は、cu コマンドを使用して、ローカル・ホスト earth からリモート・ホスト moon に接続し、moon にログインして、そこからコマンドを入力する方法を示しています。

1. ローカルのシステム・プロンプトで、次の cu コマンドを入力する。

メッセージが表示されて接続が確認されます。

```
earth% cu moon
Connected
```

リモート・ホストのログイン・プロンプトが表示されます。

リモート・ホストによっては、ログイン・プロンプトが表示されるまでに、Return キーを何回か押さなければならない場合があります。

2. ログイン・プロンプトからホスト moon にログインする。

ホスト moon のシステム・プロンプトが表示されます。

3. ホスト moon がサポートするコマンドを入力する。

たとえば、/usr/geog/crater/earthside ディレクトリの内容を表示するには、システム・プロンプトで次のコマンドを入力します。

```
moon% ls /usr/geog/crater/earthside
copernicus.dat
tycho.dat
moon%
```

注意

この例は、すべての cu 接続で利用できるわけではありません。
簡単な一般例として示しています。詳細については、システム管理者に問い合わせてください。

リモート・ホストにログインした後は、リモート・ホストとローカル・ホストは同時に実行しているため、両ホストの間で切り換えを行うことができます。ローカル・ホストに戻ってコマンドを入力するには、コマンドの前にチルダと感嘆符(~!) を付けて入力するか、あるいはローカル・ホストのプロンプトが表示されるのを待ってから、コマンドを入力してください。リモート・ホストに戻るには、Ctrl/D を押します。

14.3.1.2 cu を使用した直接接続のリモート・ホストの指定

直接接続のリモート・ホストに接続するには、`cu` コマンドで `-l` オプションを使用して、2 台のコンピュータを接続しているハード配線回線名を指定してください。このような回線名のほとんどは、標準デバイス名 `tty` のバリエーションです。

`-l` オプションを指定して `cu` コマンドを実行し、名前のわからない (ただしハード配線デバイス `ttyd0` を使用する) リモート・ホストに接続するには、次のように入力します。

```
earth% cu -lttyd0
Connected
```

接続が確立されると、リモート・ホスト `moon` にログインして、コマンドを実行することができます。14.3.1.1 項の手順を参照してください。

ローカル・ホストに戻るには、コマンドの前にチルダと感嘆符 (`~!`) 付けてを入力するか、あるいはローカルのシステム・プロンプトが表示されるのを待ってから、コマンドを入力してください。リモート・ホストに戻るには、`Ctrl/D` を押します。

注意

`-l` オプションを使用し、さらにリモート・ホスト名を入力しても、エラー・メッセージは表示されません。この場合 `cu` は、`-l` オプションで指定された回線を無視して、要求されたホスト名で最初に接続可能な回線に接続しようとします。接続できても、希望するホストではないことがあります。

14.3.1.3 cu を使用した電話回線によるリモート・ホストへの接続

リモート・ホストが UUCP を介してローカル・ホストと通信できるようにセットアップされていない場合には、`cu` を使用すると、電話回線を介してリモート・ホストに接続できます。この接続のためには、次の条件が満たされていない必要ありません。

- ローカル・ホストおよびリモート・ホストの両方がモデムに接続されている。

- リモート・モデムの電話番号を知っていて、そのホストに有効なログイン権限がある。

次の例は、cu コマンドを使用して、外線の電話番号が 1-612-555-6789 であるリモート・ホストに接続する方法を示しています。-s オプションで、伝送速度 300 ボーを指定します。外線番号の 9 をダイヤルする必要がある場合には、ローカルのシステム・プロンプトから、次のように cu コマンドを入力します。

```
earth% cu -s300 9=16125556789
Connected
```

接続が確立されたら、リモート・ホストにログインして、コマンドを実行することができます。14.3.1.1 項で説明している手順を参照してください。

ローカル・ホストに戻るには、コマンドの前にチルダと感嘆符 (~!) を付けて入力するか、あるいはローカルのシステム・プロンプトが表示されるのを待ってから、コマンドを入力してください。リモート・ホストに戻るには、Ctrl/D を押してください。

-s オプションを使用して伝送速度を指定しない場合は、省略時の設定により、/usr/lib/uucp/Devices のデータから適切な速度が選択されます。

セキュリティ機能を追加する場合には、-n オプションを使用して、電話番号の入力を求めるプロンプトを表示させます。このオプションは、ps コマンドによる電話番号の表示を抑制します。-n オプションを指定しない場合には、入力した番号を cu コマンドで表示します。

表 14-1 に、cu コマンドのオプションとその説明をまとめます。詳細は、cu(1) リファレンス・ページを参照してください。

表 14-1: cu コマンドのオプション

オプション	説明
-sspeed	リモート・ホストへのデータ伝送速度を指定する。ほとんどの作業では、UUCP のインストール時に設定され、/usr/lib/uucp/Devices のデータに基づく省略時の速度で十分である。
-e -o	リモート・ホストに送信するデータのパリティに対し、偶数の場合には -e、奇数の場合には -o を指定する。
-h	-h を指定すると、ローカル・エコーをエミュレートして、端末が半二重モードに設定されている他のホストへの呼び出しをサポートする。

表 14-1: cu コマンドのオプション (続き)

オプション	説明
-d	-d を指定すると、診断トレースを出力する。
-n	-n を指定すると、cu により電話番号の入力を求めるプロンプトを表示する (セキュリティ機能の追加)。
-l line	2 台のコンピュータ間での通信用のデバイス (回線) 名を指定する。省略時の設定は、非同期のハード配線回線、またはモデムなどの自動ダイヤラが接続されている電話回線である。サイトに複数の通信回線がある場合には、cu リンク用に特定の回線を指定することもある。 通常は、回線またはデバイスを指定する必要はない。UUCP インストール時に設定された省略時の設定で十分である。リモートのコンピュータに接続したいが、その名前がわからない場合には、-l オプションと標準デバイス名 tty のバリエーション (たとえば -ltty1) を指定して、cu コマンドを入力する。サイトのデバイス名については、システム管理者に問い合わせる。
-t	「自動応答」を設定している端末にダイヤルして、carriage return を carriage return/linefeed にマップする。
host_name	接続を確立するリモート・ホスト名を指定する。
telno	モデムを使用してリモート接続する際の電話番号を指定する。

14.3.1.4 ローカルの cu コマンド

cu コマンドでリモート・ホストに接続すると、ローカルの cu コマンドを使用して、次のタスクを実行できます。

- ローカル・ホストおよびリモート・ホスト間で切り換えを行う。
- ローカル・ホスト上のディレクトリを変更する。
- ローカル・ホストおよびリモート・ホスト間でファイルをコピーする。
- リモート接続を終了する。

一時的にローカル・ホストに戻って作業を行うには、リモートのシステム・プロンプトに対してチルダと感嘆符 (~!) を入力し、次の形式のローカルのシステム・プロンプトが表示されるのを待ちます。この場合、local はローカル・ホストの名前です。

~ [*local*] !

ローカルのシステム・プロンプトが表示されるまで待たなくても、ローカル・ホストにアクセスする ~! を入力した直後に、コマンドを入力することもできます。たとえば、cu でリモート・ホスト moon に接続しているときに、次のコマンドを入力すると、ローカル・ホスト earth に戻り、cat コマンドを使用して、/usr/crew/r2/asimov/AI ファイルを読むことができます。

```
moon% ~!cat /usr/crew/r2/asimov/AI
```

頻繁に実行するタスクで使用する、次の 3 つの cu ローカル・コマンドがあります。リモート・ホストで作業を続けながら、これらのコマンドをリモート・ホストから入力すると、ローカル・ホスト上でタスクを実行できます。これらのコマンドの前には、チルダとパーセント記号を付けます。

~%cd <i>directory_name</i>	ローカル・ホスト上のディレクトリを変更する。
~%take from [<i>to</i>]	リモート・ホストからローカル・ホストへファイルをコピーする。
~%put to [<i>from</i>]	ローカル・ホストからリモート・ホストへファイルをコピーする。

たとえば、cu を使用してリモート・ホスト moon に接続しているとき、次のコマンドを入力することにより、ローカル・ホスト earth の現在のディレクトリを、/usr/geog/ocean から /usr/geog/ocean/pacific に変更できます。

```
moon% ~%cd pacific
```

cu でリモート・ホスト moon に接続しているとき、リモートのシステム・プロンプトから、次の ~%take ローカル・コマンドを入力することにより、ファイル /usr/ETI/clavius を、ローカル・ホスト earth 上の /usr/NASA/decode へコピーできます。

```
moon% ~%take /usr/ETI/clavius /usr/NASA/decode
```

cu でリモート・ホスト moon に接続しているとき、リモートのシステム・プロンプトから、次の ~%put ローカル・コマンドを入力することにより、ローカル・ファイル /usr/NASA/jupiter を、リモート・ホスト上の /usr/ETI/clavius/hal9 へコピーできます。

```
moon% ~%put /usr/NASA/jupiter /usr/ETI/clavius/hal9
```

注意

~%take および ~%put コマンドを使用する前には、コピー先ディレクトリが存在することを確認してください。uucp コマンドとは異なり、これらの cu ローカル・コマンドは、ファイル転送の間に中間ディレクトリを作成しません。

~%take および ~%put を使用して転送できるのは ASCII ファイルだけです。たとえば、PostScript ファイルは ASCII ファイルですが、実行可能ファイルは ASCII ファイルではありません。

14.3.1.5 cu を使用したローカル・コンピュータの複数のリモート・コンピュータへの接続

cu コマンドを入力して、ホスト X をホスト Y に接続し、ホスト Y にログインしてから、そこで再び cu コマンドを入力して、ホスト Z に接続できます。これによって、1 つのローカル・ホスト・コンピュータ X と、2 つのリモート・ホスト・コンピュータ Y および Z を使用できるようになります。

ホスト Z にログインすると、そのホスト上でオペレーティング・システムのコマンドを実行できます。次のようにすると、Z から他のホスト上でコマンドを実行できるようになります。

- ホスト X でコマンドを実行するには、コマンドの前にチルダ (~) を 1 つ付ける。
- ホスト Y でコマンドを実行するには、コマンドの前に 2 つのチルダ (~) を付ける。

表 14-2 に、最も一般的に使用される cu ローカル・コマンドを示します。他の cu ローカル・コマンドについての詳細は、cu(1) リファレンス・ページを参照してください。

表 14-2: ローカルの **cu** コマンド

コマンド	説明
~.	リモート・コンピュータからログアウトして、リモート接続を終了する。 モデムを使用し、電話回線を介してリモート・ホストと接続する場合には、このコマンドは常に動作するとは限らない。動作しない場合には、Ctrl/D を押してログアウトする。その後、プロンプトからチルダとピリオド (~.) を入力し、Return キーを押して、リモート接続を終了する。
~!	セッションをリモート・ホストからローカル・ホストへ戻す。プロンプトに対してチルダと感嘆符 (~!) をタイプした後、任意のコマンドを入力する。リモート・ホストに戻るには、Ctrl/D を押す。 cu 接続が確立されると、~! (リモートからローカルへ)、または Ctrl/D (ローカルからリモートへ) を入力することによって、2 つのホストを切り換えることができる。
~%cd <i>directory_name</i>	ローカル・ホストの現在のディレクトリを、 <i>directory_name</i> 変数で指定したディレクトリに変更する。ディレクトリ名を指定しなければ、ホーム・ディレクトリに戻る。
~%take <i>source</i> [<i>dest</i>]	リモートからローカル・ホストにファイルをコピーする。ローカル・ホスト上の <i>dest</i> コピー先ファイルを指定しなければ、~%take コマンドは、リモート・ファイルをローカル・ホストにコピーして、同じファイル名を割り当てる。
~%put <i>source</i> [<i>dest</i>]	ローカルからリモート・ホストにファイルをコピーする。リモート・ホスト上の <i>dest</i> コピー先ファイルを指定しなければ、~%put コマンドは、ローカル・ファイルをリモート・ホストにコピーして、同じファイル名を割り当てる。
~\$ <i>cmd</i>	ローカル・ホスト上で <i>cmd</i> コマンドを実行し、リモート・シェルで実行するために、出力をリモート・ホストに送信する。

14.3.2 tip を使用したリモート・ホストへの接続

tip コマンドとそのオプションを使用すると、リモート・ホストに接続して、ログインし、ローカル・ホストからリモート・ホスト上でタスクを実行

できます。2つのホスト間で切り換えを行うことによって、各ホスト上でタスクを実行することができます。ローカル・ホストとリモート・ホストが同じオペレーティング・システムを使用している場合には、ローカル・ホストからリモート・ホストにコマンドを入力することができます。

14.3.2.1 tip を使用した名前によるリモート・ホストへの接続

次の手順は、tip コマンドを使用して、ローカル・ホスト earth から、リモート・ホスト moon に接続し、moon にログインして、そこからコマンドを入力する方法を示しています。

1. ローカルのシステム・プロンプトで、次の tip コマンドを入力する。

メッセージが表示されて接続が確認されます。

```
earth% tip moon
Connected
```

リモート・ホストのログイン・プロンプトが表示されます。

リモート・ホストによっては、ログイン・プロンプトが表示されるまでに、Return キーを何回か押さなければならない場合があります。

2. ログイン・プロンプトからホスト moon にログインする。

ホスト moon のシステム・プロンプトが表示されます。

3. システム・プロンプトが表示されるのを待って、ホスト moon がサポートするコマンドを入力する。

たとえば、/usr/geog/crater/darkside ディレクトリの内容を表示するには、システム・プロンプトから次のコマンドを入力します。

```
moon% ls /usr/geog/crater/darkside
copernicus.dat
tycho.dat
moon%
```

注意

この例は、すべての tip 接続で使用できるわけではありません。簡単な一般例として示しています。詳細については、システム管理者に問い合わせてください。

リモート・ホストにログインした後は、リモート・ホストとローカル・ホストは同時に実行しているため、両ホストの間で切り換えを行うことができます。ローカル・ホストに戻ってコマンドを入力するには、コマンドの前にチルダと感嘆符 (~!) を付けて入力するか、あるいはローカル・ホストのプロンプトが表示されるのを待ってから、コマンドを入力してください。リモート・ホストに戻るには、Ctrl/D を押します。

14.3.2.2 tip を使用した電話回線によるリモート・ホストへの接続

次の条件を満たしている場合には、tip コマンドを使用し、電話回線を介してリモート・ホストに接続できます。

- ローカル・ホストおよびリモート・ホストの両方がモデムに接続されている。
- リモート・モデムの電話番号がわかっているか、あるいはリモート・ホストのエントリが `/etc/remote` に存在する。

次の手順では、tip コマンドを使用して、内線の電話番号が 555-1234 のリモート・ホストに伝送速度 300 のボー・レートで接続しています。

1. ローカル・プロンプト `jupiter` から、次の tip コマンドを入力する。
メッセージが表示されて接続が確認されます。

```
jupiter% tip -300 5551234  
Connected
```

2. Return キーを押す。

リモート・ホストによっては、リモート・ホストのログイン・プロンプトが表示されるまでに、Return キーを数回押さなければならない場合があります。

3. リモート・ホストのログイン・プロンプトからログインする。

ローカル・ホストとの接続は引き続きオープンされているので、ローカル・ホストまたはリモート・ホストの両方で作業ができます。

注意

伝送速度を指定しなければ、省略時の設定により、tip コマンドは 1200 ボー・レートを使用します。

次の手順では、tip コマンドを使用して、外線の電話番号が 1-612-555-9876 のリモート・ホストに伝送速度 300 のボー・レートで接続しています。

1. 外線番号の 9 をダイヤルする必要がある場合には、ローカル・ホスト earth のプロンプトから、次の tip コマンドを入力する。

メッセージが表示されて接続が確認されます。

```
earth% tip -300 9,16125559876
Connected
```

2. Return キーを押す。

リモート・ホストによっては、リモート・ホストのログイン・プロンプトが表示されるまでに、Return キーを数回押さなければならない場合があります。

3. リモート・ホストのログイン・プロンプトからログインする。

ローカル・ホストとの接続は引き続きオープンされているので、ローカル・ホストまたはリモート・ホストの両方で作業ができます。

/etc/remote ファイルおよび /etc/phones ファイルのカスタマイズについての詳細は、『ネットワーク管理ガイド』、および remote(4) と phones(4) のリファレンス・ページを参照してください。

表 14-3 に、tip コマンド・オプションとその説明を示します。詳細は、tip(1) リファレンス・ページを参照してください。

表 14-3: tip コマンドのオプション

オプション	説明
<code>-baud_rate</code>	リモート・ホストへのデータ伝送速度を指定する。省略時の値は 1200 ボー。 UUCP のインストール時に設定され、サイト用にカスタマイズされたボー・レートは、接続に使用するハードウェアに従って構成される。
<code>-v</code>	<code>.tiprc</code> ファイルから読み取った (詳細モード) 任意の変数を表示する。
<code>host_name</code>	接続する リモート・ホストを指定する。ローカル・ホストとリモート・ホスト間でのシステム通信のセットアップ方法に従って、 <code>tip</code> コマンドは、ハード配線回線、またはモデムと電話回線を使用して接続する。
<code>telno</code>	モデムを使用してリモート接続する際の電話番号を指定する。リモート・ホスト名が <code>tip</code> で認識されない (つまり、 <code>/etc/remote</code> ファイルにエントリがない) 場合には、この方法を使用する。

14.3.2.3 ローカルの tip コマンド

`tip` コマンドでリモート・ホストに接続しているときに、ローカル・コマンドを使用して、次のタスクを実行できます。

- ローカル・ホストとリモート・ホスト間で切り換えを行う。
- リモート・ホストからローカル・ホスト上のディレクトリを変更する。
- ローカル・ホストとリモート・ホスト間でファイルをコピーする。
- リモート接続を終了する。

一時的にローカル・ホストに戻ってコマンドを入力するには、リモートのシステム・プロンプトからチルダと感嘆符 (~!) を入力します。ローカルのシステム・プロンプトが次の形式で表示されます。ここで、`shell` はローカルのシェル名であり、`pmt` はローカルのシェル・プロンプトです。シェル・プロンプトは、C シェルに対しては %、Bourne、Korn、または POSIX シェルに対しては \$ が表示されます。

~ [`shell`] `pmt`!

リモート・ホストに戻るには、ローカルのシステム・プロンプトで Ctrl/D を押します。 `tip` プロセスを終了するには、チルダをタイプして Ctrl/D (~^D) を押します。

リモート・ホストで作業を続けながら、リモート・ホストから次の `tip` コマンドを使用すると、ローカル・ホストでタスクを実行できます。これらコマンドは前にチルダを付けて使用します。

<code>~c directory_name</code>	ローカルのディレクトリを変更する。
<code>~t from [to]</code>	リモート・ホストからローカル・ホストにファイルをコピーする。
<code>~<</code>	リモート・ホストからローカル・ホストにファイルをコピーする。
<code>~p from [to]</code>	ローカル・ホストからリモート・ホストにファイルをコピーする。
<code>~></code>	ローカル・ホストからリモート・ホストにファイルをコピーする。

たとえば、`tip` でリモート・ホスト `moon` に接続しているとき、次のコマンドを入力して、ローカル・ホスト `earth` 上の現在のディレクトリを、`/usr/geog/polar` から `/usr/geog/polar/arctic` に変更できます。

```
moon% ~c arctic
```

`tip` でリモート・ホスト `moon` に接続しているとき、次のコマンドを入力して、`/usr/darkside/temp/dat` ファイルを、ローカル・ホスト `earth` 上の `/usr/NASA/bios/temp` ファイルにコピーできます。

```
moon% ~t /usr/darkside/temp/dat /usr/NASA/bios/temp
```

`tip` でリモート・ホスト `moon` に接続しているとき、次のコマンドを入力して、ローカルの `/usr/NASA/bios/warn` ファイルを、リモート・ホスト上の `/usr/darkside/temp/change` ファイルにコピーできます。

```
moon% ~p /usr/NASA/bios/warn /usr/darkside/temp/change
```

注意

`~t` および `~p` コマンドで転送できるのは ASCII ファイルだけです。たとえば、PostScript ファイルは ASCII ファイルですが、オブジェクト (コード) ファイルは ASCII ファイルではありません。

`~t` も `~p` も、ファイル転送エラーをチェックしませんが、`uucp` コマンドには、この確認の機能があります。

14.3.2.4 tip コマンドを使用したローカル・ホストから複数のリモート・ホストへの接続

tip コマンドを入力して、ホスト *x* をホスト *y* に接続し、ホスト *y* にログインしてから、そこで tip コマンドを入力し (*y* が tip コマンドをサポートする場合)、ホスト *z* に接続することができます。これによって、1 つのローカル・ホスト・コンピュータ *x* と、2 つのリモート・ホスト・コンピュータ *y* および *z* を使用できるようになります。

ホスト *z* がホストである場合には、ログインすると、ホスト *z* 上でオペレーティング・システムのコマンドを実行できます。次のようにすると、*z* から他のホスト上でコマンドを実行できるようになります。

- ホスト *x* 上でコマンドを実行するには、コマンドの前にチルダ (~) を 1 つ付ける。
- ホスト *y* 上でコマンドを実行するには、コマンドの前に 2 つのチルダ (~) を付ける。

注意

チルダ (~) で始まるコマンド・シーケンスは、チルダがコマンド行の先頭にある場合にのみ、tip によって解釈されます。

表 14-4 に、最も一般的に使用される tip ローカル・コマンドを示します。他の tip ローカル・コマンドについての詳細は、tip(1) リファレンス・ページを参照してください。

表 14-4: ローカル tip コマンド

コマンド	説明
<code>~Ctrl/D</code> <code>~.</code>	リモート・コンピュータからログアウトして、リモート接続を終了する。 モデムを使用し、電話回線を介してリモート・ホストと接続する場合には、このコマンドは常に動作するとは限らない。動作しない場合には、Ctrl/D を押してログアウトする。その後、プロンプトから <code>~Ctrl/D</code> を入力し、Return キーを押して、リモート接続を終了する。
<code>~!</code>	セッションをリモート・ホストからローカル・ホスト上のシェルへ戻す。プロンプトに対してチルダと感嘆符 (<code>~!</code>) をタイプした後、任意のコマンドを入力する。リモート・ホストに戻るには、Ctrl/D を押す。 tip 接続が確立されると、 <code>~!</code> (リモートからローカルへ)、あるいは Ctrl/D (ローカルからのリモートへ) を入力することによって、2 つのホストを切り換えることができる。
<code>~c directory_name</code>	ローカル・ホスト上の現在のディレクトリを、 <code>directory_name</code> 変数で指定したディレクトリに変更する。ディレクトリ名を指定しなければ、ホーム・ディレクトリに戻る。
<code>~t source [dest]</code>	リモートからローカル・ホストにファイルをコピーする。ローカル・ホスト上の <code>dest</code> コピー先ファイルを指定しなければ、 <code>~t</code> コマンドは、リモート・ファイルをローカル・ホストにコピーして、同じファイル名を割り当てる。
<code>~p source [dest]</code>	ローカルからリモート・ホストにファイルをコピーする。リモート・ホスト上の <code>dest</code> コピー先ファイルを指定しなければ、 <code>~p</code> コマンドは、ローカル・ファイルをリモート・ホストにコピーして、同じファイル名を割り当てる。

表 14-4: ローカル tip コマンド (続き)

コマンド	説明
~<	リモートからローカル・ホストにファイルをコピーする。 tip コマンドは、リモート・ホスト上で リモート・ファイルやローカル・ファイル名の表示に使用するコマンド文字列 (たとえば, <code>cat filename</code>) の入力を求めるプロンプトを表示する。
~>	ローカルからリモート・ホストにファイルをコピーする。 tip コマンドはローカル・ファイル名の入力を求めるプロンプトを表示し、そのファイルを標準入力と同様にリモート・ホストへ送信する。 ~> コマンドを実行する前に、この入力を認識するよう、リモート・ホストのコマンドをセットアップしなければならない。たとえば、 <code>remote% cat > destination-file</code> のように行う。

注意

tip プログラムは、システム・プロンプトとシステムの割り込み文字列に対応する文字列を使用して、~< または ~> コマンドによるファイル転送の終了をシグナル通知します。これらの値は、`/etc/remote` ファイルに定義されます。 詳細は、`remote(4)` リファレンス・ページを参照してください。

14.3.3 ct を使用したモデムによるリモート端末への接続

モデムを備えたりリモート ASCII 端末のユーザは、`ct` コマンドを使用すると、モデムと電話回線を介して、ローカル・ホストと通信できます。 リモート端末のユーザは、接続後、ローカル・ホストにログインして、作業を行うことができます。 使用できる電話回線がない場合には、`ct` コマンドはメッセージを表示して、回線が使用できるまで待つかどうかを尋ねます。

`ct` コマンドは、次のような場合に有効です。

- 機密性の高い通信が必要な場合

ローカル・ホストはリモート端末と接続しているため、リモート・ユーザは、ローカル・ホストの電話番号を知る必要がありません。 ローカ

ル・ユーザは、`ct` を入力することによって、リモート・ユーザの作業をモニタすることができます。

- 電話接続の費用が、ローカル・サイトまたはリモート端末の特定のアカウントに (コレクト・コールのように) チャージされる場合
-h オプションを省略すれば、コレクト・コールをエミュレートすることができます。 リモート端末のユーザは、-h オプションを指定しないで `ct` コマンドを入力できます。

次の `ct` の機能は、特定の環境で有効です。

- 接続が確立されるか、あるいは設定された時間が経過するまで、番号をダイヤルし続けるように、`ct` に指示することができる。
- 2 つ以上の電話番号を指定して、接続が確立されるまで、`ct` が各モデムにダイヤルするように設定できる。

注意

通常、リモート端末のユーザは、ローカル・ホストのユーザを呼び出して、`ct` セッションを要求します。このような接続が頻繁に発生する場合には、システム管理者は UUCP を設定して、指定された時間に、1 つまたは複数の指定端末に対し、ローカル・ホストが自動的に `ct` を実行するようにします。UUCP のカスタマイズについての詳細は、システム管理者に問い合わせてください。

たとえば、同じサイトにあるリモート端末のモデムに接続するには、次のコマンドを入力してください。リモート・モデムの電話番号は 7-6092 とします。

```
earth% ct 76092
Allocated dialer at 1200 baud
Confirm hang_up? (y to hang_up)
```

コマンドを入力すると、接続を確認するメッセージが表示され、現在使用中の他の電話回線を中断するか、あるいはコマンドを取り消すかを入力するプロンプトが表示されます。

次の例は、`ct` コマンドを使用して、市内電話番号が 555-0043 であるリモート端末モデムに接続する方法を示しています。この例では、外線番号とし

て 9 を指定し、モデム回線を 2 分間待つように指示する `-w` オプションを指定しています。

```
earth% ct -w2 9=5550043
Allocated dialer at 1200 baud
Confirm hang_up? (y to hang_up)
```

前述の例と同様に、現在使用中の他の電話回線を中断するか、あるいはコマンドを取り消すかを入力するプロンプトが表示されます。

次の例は、`ct` コマンドを使用して、ローカル・ホスト `earth` から、市外電話番号が 1-201-555-7824 であるリモート端末のモデムに接続する方法を示しています。この例では、外線番号として 9 を指定し、モデム回線を 5 分間待つように指示する `-w` オプションを指定しています。

```
earth% ct -w5 9=12015557824
Allocated dialer at 1200 baud
Confirm hang_up? (y to hang_up)
```

現在使用中の他の電話回線を中断するか、あるいはコマンドを取り消すかを入力するプロンプトが表示されます。

詳細は、`ct(1)` リファレンス・ページを参照してください。

表 14-5 に、`ct` コマンドのオプションと必要なエントリを示します。

表 14-5: `ct` コマンドのオプション

オプション	説明
<code>-wminutes</code>	<code>ct</code> コマンドが回線を待つ最大時間を分単位で指定する。 <code>ct</code> コマンドは、接続が確立されるか、指定された時間が経過するまで、リモートのモデムに 1 分間隔でダイヤルする。 <code>-w</code> オプションを指定すると、接続が確立できなかった場合に、通常 <code>ct</code> が表示するメッセージを抑制する。
<code>-xnumber</code>	ローカル・ホスト上でのコマンド実行時の標準エラー出力についての詳細情報を作成する。これは、デバッグの際に参照する。デバッグ・レベル <code>number</code> は、0 ~ 9 の数字。推奨する省略時の値は 9。
<code>-v</code>	<code>ct</code> コマンドで、実行中のナレーティブを標準エラー出力に送信できるようにする。
<code>-h</code>	<code>ct</code> によって、現在の接続が切断されないようにする。
<code>-sspeed</code>	<code>ct</code> のデータ伝送速度を指定する。省略時の値は 1200 ボー。 ボー・レートは、接続する端末のボー・レートに設定する。

表 14-5: ct コマンドのオプション (続き)

オプション	説明
<code>telno</code>	リモート・モデムの電話番号を指定する。市内電話番号または市外電話番号を入力し、外線用の 9 や、その他のアクセス・コードを表す 2 次ダイアル・トーンを指定する。 2 次ダイアル・トーンの後ろには等号 (=) を使用し (9=) , 555-5092 のように、遅延にはハイフン (-) を指定する。電話番号は 31 文字まで、また下記のいづれも含むことができる。 0 ~ 9 の数字 ハイフンまたはダッシュ (-) 等号 (=) アスタリスク (*) 番号記号またはボンド記号 (#)

14.4 uux を使用したリモート・ホスト上でのコマンドの実行

`uux` コマンドを使用すると、ローカル・ホスト上で作業しながら、リモート・ホスト上でコマンドを実行することができます。リモート・ホストにコマンドが存在しない場合には、`uux` は実行されず、リモート・ホストがそのことをメールで通知します。コマンドが実行されて出力を生成する場合 (たとえば `cat` や `diff`) には、`uux` に、指定のホスト上のファイルに出力を書き込むようにさせることができます。

注意

機密保護のため、`uux` を介したコマンドの使用を制限しているサイトがあります。また、ローカル・ホストにエンハンスド・セキュリティ機能が設定されている場合には、`uux` を介したリモート・ホスト上での特定のコマンドの実行に影響があります。詳細については、システム管理者に問い合わせてください。

`uux` コマンドの構文は、使用するシェルのコマンド・インタプリタが特殊文字を処理する方法によって決まります。この構文は、`Bourne`、`Korn`、および `POSIX` シェルでは同じですが、`C` シェルでは異なります。

どのシェルから `uux` を使用するかに関係なく、デスティネーションの指定には次の 2 つの方法があります。

```
uux [option...] "commandstring > destination"
```

```
uux [option...] commandstring \{ destination \}
```

最初の構文では、右山カッコ (リダイレクション記号) (>) は、リモート・コマンドの出力を、デスティネーションのディレクトリまたはファイルに向けます。二重引用符 (" ") は、リダイレクション記号の右山カッコ (>) が特殊文字であるため、コマンド全体を囲みます。コマンド行で次のいずれかの文字を使用する場合には、必ずその文字またはコマンド全体を二重引用符 (" ") で囲んでください。

- 左山カッコ (<)
- 右山カッコ (>)
- セミコロン (;)
- 垂直バーつまりパイプ (|)
- プラス記号 (+)
- 左大カッコ ([)
- 右大カッコ (])
- 疑問符 (?)

2 番目の構文では、デスティネーション名を中カッコ ({ }) で囲みます。中カッコは、シェル・コマンド・インタプリタでの特殊文字であるため、各中カッコの前にはバックスラッシュ (\) をタイプしなければなりません。バックスラッシュがなければ、中カッコが誤って解釈される恐れがあります。

デスティネーション・ファイルのパス名を指定する場合には、完全パス名または `~user` に続くパス名を使用できます。この場合、`user` はユーザのログイン・ディレクトリ名です。

出力ファイルには「書き込み」許可のステータスが必要です。あるターゲット出力ファイルの許可がどのように設定されているか不明な場合には、コマンドの結果を `/usr/spool/uucppublic` ディレクトリへ出力してください。シェルからは簡単に、このディレクトリを `~uucp` と指定することができます。

14.4.1 Bourne シェル , Korn シェル , または POSIX シェルからの uux の使用

次の例は、uux コマンドが、オペレーティング・システムのコマンド cat を使用して、ホスト gem にある /u/doc/F1 ファイルを、ホスト sky にある /usr/doc/F2 ファイルと連結する方法を示したものです。結果はホスト gem 上の /u/doc/F3 ファイルに入ります。

```
uux "gem!cat gem!/u/doc/F1 sky!/usr/doc/F2 > gem!/u/doc/F3"
```

次の例では、タスクは前のコマンドと同じですが、リダイレクション記号ではなく、中カッコ ({ }) を使用して、uux コマンド行にデスティネーションを指定しています。タスクは前のコマンドと同じですが、出力先は暗黙指定です。

```
uux gem!cat gem!/u/doc/F1 sky!/usr/doc/F2 \{gem!/u/doc/F3\}
```

14.4.2 C シェルからの uux の使用

前の項と同じ操作を C シェルで行うには、次の uux コマンドのいずれか 1 つを入力してください。

```
uux "gem\!cat gem\!/u/doc/F1 sky\!/usr/doc/F2 > gem\!/u/doc/F3"
```

次の例では、暗黙のデスティネーション出力ファイルを使用しています。

```
uux gem\!cat gem\!/u/doc/F1 sky\!/usr/doc/F2 \{gem\!/u/doc/F3\}
```

次の 2 つの例では、uux コマンドは cat コマンドを使用して、acct6 ファイルをリモート・ホスト boston から、ローカル・ホスト上のパブリック・ディレクトリにある acct6 ファイルへの出力として送信します。

```
uux "cat boston\!/reports/acct6 > ~uucp/acct6"
```

次の例では、暗黙のデスティネーション出力ファイルを使用します。

```
uux cat boston\!/reports/acct6 \{~uucp/acct6\}
```

14.4.3 その他の uux の機能およびヒント

uux コマンドでは、ローカル・ホストを省略時のホストとするため、コマンド行にローカル・ホストを指定する必要はありません。たとえば、diff コマンドを実行して、ホスト car 上の /u/F1 ファイルを、ホスト sea 上の /u/F2 ファイルと比較し、その結果をローカル・ホストの /u/F3 ファイルに入れるには、次のコマンドを使用します。

```
uux "diff car!/u/F1 sea!/u/F2 > /u/F3"
```

また、次の例のように、感嘆符だけを使用して、ローカル・ホストを表すことができます。

```
uux "!diff car!/u/F1 sea!/u/F2 > !/u/F3"
```

diff または cat などのコマンドでパス名ソース・ファイルを指定する場合には、リモート・ホストが解釈するシェルのパターン照合文字を含めることができます。

- 疑問符 (?)
- アスタリスク (*)
- 左大カッコ (()
- 右大カッコ ()

これらの文字は、2つの バックスラッシュ (\ ... \) または二重引用符 (" ... ") で囲み、uux がコマンドをリモート・ホストに送信する前に、ローカルのシェルがその文字を解釈しないようにします。 デスティネーション名には、パターン照合文字を使用しないでください。

左山カッコ (<), 右山カッコ (>), セミコロン (;) およびパイプ (|) のシェル文字を使用する場合には、それらの文字をバックスラッシュ (\ ... \) または二重引用符 (" ... ") で囲むか、コマンド行全体をバックスラッシュまたは二重引用符で囲んでください。

注意

シェルのリダイレクション文字、2つの左山カッコ (<<) および2つの右山カッコ (>>) は、UUCP では使用できません。

表 14-6 に、uux コマンドのオプションおよび必要なエントリをまとめます。詳細は、uux(1) リファレンス・ページを参照してください。

表 14-6: uux コマンドのオプション

オプション	説明
-n	コマンドがデスティネーション・ホスト上で実行できなかった場合に、通常 mailx を介して送られる通知を取り消す。 -n と -z オプションは、相互に排他的である。
-z	デスティネーション・ホスト上でコマンドが実行できなかった場合に、mailx を介してメッセージを送信する。 -n と -z オプションは、相互に排他的である。

表 14-6: uux コマンドのオプション (続き)

オプション	説明
-j	リモートのコマンドを実行する uux が要求するジョブ識別番号を表示する。この番号は uustat コマンドで使用する。詳細は 14.8.1 項を参照。
cmd_string	指定したホストが受けつけるコマンドを指定する。コマンド・フォーマットについての詳細は、14.4 節を参照。
dest_name	リモート・ホストで実行するコマンドの出力を格納するためのホストおよびファイルを指定する。たとえば、リモート・ホスト上のディレクトリにあるすべてのファイルの一覧を表示する場合は、uux コマンドを使用して適切なデスティネーション名を入力すれば、ローカル・ホスト上のファイルに結果を入れることができる。デスティネーション・フォーマットの詳細は、14.4 節を参照。

14.5 UUCP を使用したファイルの送受信

UUCP プロトコルをサポートする UNIX ベースのコンピュータでは、uucp コマンドを使用して、あるコンピュータから別のコンピュータに、1 つまたは複数のファイルをコピーできます。uucp を使用すると、次のようにファイルをコピーできます。

- ローカル・ホストとリモート・ホスト間
- 2 つのリモート・ホスト間
- 中間ホストを介した 2 つのホスト間
- ローカル・ホスト内

多くのサイトでは、ファイル転送を円滑に行うために、パブリックの UUCP ディレクトリ `usr/spool/uucppublic` を作成して、利用できるようにしています。このディレクトリは、すべてのユーザによる読み取りおよび書き込みアクセスが可能で、セキュリティ制限を回避します。このディレクトリは、uucp コマンドでは、`~uucp` または `~/` で簡単に指定できます。

注意

uucp を介したファイル転送は、転送を行うホスト上のセキュリティ機能の影響を受けます。uucp ユーティリティは、ファイル転送が失敗した場合に、エラー・メッセージを表示しません。詳細は、システム管理者に問い合わせてください。

システム管理者は、リモート・ユーザによる不正な使用を防止するために、セキュリティ制限を定義しているので、ファイル転送でアクセスできるのは、一部のディレクトリおよびファイルのみです。

14.5.1 Bourne シェル, Korn シェル, および POSIX シェルでの UUCP を使用したファイルのコピー

Bourne, Korn, または POSIX シェルからは、コピー先ファイルのホスト名の前に指定する感嘆符 (!) の前にバックスラッシュ (\) を付けなくても、uucp ファイル名を指定できます。たとえば、ローカル・ホスト earth の star ファイルを、リモート・ホスト sky 上のパブリック・ディレクトリにある /sun/stats ファイルにコピーするには、次のコマンドを入力してください。

```
earth% uucp star sky!~/sun/stats
```

同じファイルをコピーするときに、明示的に /usr/spool/uucppublic ディレクトリを識別するには、次のコマンドを入力します。

```
earth% uucp star sky!usr/spool/uucppublic/sun/stats
```

ローカル・ホストからはアドレスがわからないリモート・ホストに、ファイルをコピーする場合は、そのリモート・ホストのアドレスがわかっている別のホストを介して行います。この方法でローカル・ファイルをリモート・ホストにコピーするには、まず、1 つまたは複数の中間ホストにファイルを送信します。このとき、各ホスト名は感嘆符 (!) で区切ります。たとえば、ローカル・ファイル star を、リモート・ホスト sky 上の /sun/stats ファイルへコピーする場合に、まず、中間ホスト mlkway を介して送信するには、次のコマンドを入力してください。

```
earth% uucp star mlkway!sky!~uucp/sun/stats
```

ローカル・ホストから uucp を使用して、リモート・ホストからローカル・ホストにファイルをコピーできます。たとえば、リモート・ホスト biochem から /cells/type1 ファイルを、ローカル・ファイル /dna/sequence にコピーするには、ローカル・ホスト earth から次のコマンドを入力してください。

```
earth% uucp biochem!/cells/type1 /dna/sequence
```

複数のファイルをリモート・ホストからローカル・ホストにコピーする場合には、パターン照合文字を使用してファイルを指定することができます

す。たとえば、リモート・ホスト moon 上の /geog/survey ディレクトリから、名前が report で始まるすべてのファイルを、ローカル・ホスト earth 上のパブリック・ディレクトリ ~uucp へコピーするには、次のコマンドを入力します。

```
earth% uucp moon\!/geog/survey/report* ~uucp
```

14.5.2 C シェルでの UUCP を使用したファイルのコピー

C シェルでは、感嘆符 (!) は特別な意味を持っています。コマンド・インタプリタが誤って解釈しないように、パス名で使用する場合には、感嘆符の前にバックスラッシュ (\) を付けてください。

たとえば、ローカル・ホスト earth の /usr/NASA/ctrl-specs ファイルを、リモート・ホスト luna7 上のパブリック・ディレクトリ ~uucp にコピーするには、ローカル・ホストから次のコマンドを入力します。

```
earth% uucp /usr/NASA/ctrl-specs luna7\!~uucp
```

リモート・ホスト luna7 上の /usr/reports/exobio ディレクトリにある plan9 ファイルを、ローカル・ホスト earth のパブリック・ディレクトリ ~uucp にコピーするには、次のコマンドを入力します。

```
earth% uucp luna7\!/usr/reports/exobio/plan9 ~uucp
```

リモート・ホスト luna7 上の /sensory/visual/earthrise ディレクトリにある名前が msg で始まるすべてのファイルを、ローカル・ホスト earth のパブリック・ディレクトリ ~uucp にコピーするには、次のコマンドを入力します。

```
earth% uucp luna7\!/sensory/visual/earthrise/msg'*' ~uucp
```

ここで、ソース・ファイル名で使用しているパターン照合文字のアスタリスク (*) は、誤って解釈されないように一重引用符で囲みます。

次の例では、同じファイルが ~uucp にコピーされますが、誤って解釈されないように、ソース・ファイルのパス名全体を二重引用符で囲みます。

```
earth% uucp "luna7\!/sensory/visual/earthrise/msg*" ~uucp
```

表 14-7 に、uucp コマンドのオプションおよび必要なエントリをまとめます。詳細は、uucp(1) リファレンス・ページを参照してください。

表 14-7: UUCP コマンドのオプション

オプション	説明
-d	ソース・ファイルを，リモート・ホスト上のファイルにコピーする際に必要な，中間ディレクトリを作成する。デスティネーション・パス名を指定して <code>uucp</code> を入力すれば，必要なディレクトリが作成される。-d オプションは省略時の設定。
-f	ファイル転送中に，中間ディレクトリを作成しない。
-j	転送処理のジョブ識別番号を表示する。ジョブ識別番号は， <code>uustat</code> コマンドで使用して転送のステータスを確認したり，転送を終了するために <code>uustat -k</code> で使用する。詳細は 14.8.1 項を参照。
-m	リモート・ホスト上にファイルがコピーされたことを確認するために， <code>uucp</code> が要求者へメールを送信することを指定する。ローカル転送の場合には，メールは送信されない。
-nusername	リモート・ホスト上の受信者 <code>username</code> に，ファイルが送信されたことを通知する。ローカル転送の場合には，メールは送信されない。
source_file	送信または受信するファイルのパス名を指定する。UUCP パス名についての詳細は，14.1 節を参照。
destination_name	コピーを受信するファイル (またはディレクトリ) のパス名を指定する。デスティネーション・ファイルのパス名についての詳細は，14.5.1 項および 14.5.2 項を参照。

14.6 uuto と uupick を使用したファイルのコピー

`uuto` コマンドでは，指定したファイルをコピー先ホスト上のパブリック・ディレクトリにコピーし，受信者は `uupick` を介してそのファイルを手に入れます。`rmail` プログラムが，受信者にファイルの到着を通知します。

注意

すべてのファイル転送は，ローカル・ホストおよびリモート・ホスト上のセキュリティ機能の影響を受けます。詳細は，システム管理者に問い合わせてください。

たとえば、ローカル・ホスト moe から /usr/bin/data/junk ファイルを、リモート・ホスト stooge のユーザ curly に送信するには、次のコマンドを入力します。

```
moe% uuto /usr/bin/data/junk stooge!curly
```

uuto コマンドが、ファイルをホスト stooge 上の /usr/spool/uucppublic/receive/curly/moe ファイルにコピーします。次に、rmail ユーティリティにより、ファイルが到着したことを通知するメール・メッセージが、ユーザ curly に送信されます。メッセージを受信すると、ユーザ curly は、uupick コマンドを入力してファイルにアクセスし、保存、移動、または削除が行えるようになります。次の例では、ユーザ curly は、ホスト stooge で uupick コマンドを入力します。uupick からは、次のような応答が返ってきます。

```
stooge% uupick
from system moe: file junk
?
```

ユーザ curly は、uupick の疑問符 (?) プロンプトから d および q オプションを入力して、ファイルを削除し uupick を終了します。

```
? d
? q
```

表 14-8 は、uupick の ? プロンプトから入力する uupick ファイル処理オプションの一覧です。

表 14-8: uupick コマンドのオプション

オプション	説明
*	使用可能な uupick ファイル処理オプションを表示する。
Return	uupick が次のファイルに進むようにする。
a [dir]	パブリック・ディレクトリにあるすべての uuto ファイルを、ローカル・ホスト上の指定したディレクトリへ移動する。ディレクトリは完全パス名または相対パス名を使用して指定する。省略時の値は、uupick を入力したディレクトリ。
d	d オプションは、uupick で取得した現在のファイルを削除する。
m [dir]	完全パス名または相対パス名で指定したディレクトリにファイルを移動する。省略時の値は現在のディレクトリ。
p	ファイルを表示する。

表 14-8: uupick コマンドのオプション (続き)

オプション	説明
q Ctrl/D	パブリック・ディレクトリにあるファイルの表示，移動または削除を行うことなしに，uupick を終了できる。Ctrl/D を押しても終了できる。
! <i>command</i>	オペレーティング・システムのプロンプトに戻りコマンドを実行できる。コマンドを実行すると，制御はuupickに戻る。

詳細は，uupick(1) リファレンス・ページを参照してください。

14.7 uuto を使用したファイルのローカル送信

uuto を使用して，同じローカル・ホスト上の別のユーザに，ファイルを送信することもできます。ただし，受信者には，ファイルの転送を知らせるメール・メッセージは送信されません。たとえば，ユーザ shemp は，同じローカル・ホスト stooge 上のユーザ larry に，次のように，ファイル /usr/bin/data/status を送信できます。

```
stooge% uuto /usr/bin/data/status larry
```

表 14-9 は，uuto コマンドのオプションおよび必要なエントリの一覧です。詳細は，uuto(1) リファレンス・ページを参照してください。

表 14-9: uuto コマンドのオプション

オプション	説明
-m	uuto が，指定したユーザ名およびホストにソース・ファイルをコピーしたことを，送信者に通知する。
-p	通常，uuto は，ソース・ファイルを /usr/spool/uucp-public/receive /username/host/file にコピーする。-p オプションは，指定したホスト上のパブリック・ディレクトリにソース・ファイルのコピーを転送する前に，ローカル・ホスト上のスプール・ディレクトリにソース・ファイルを送信する。

表 14-9: uuto コマンドのオプション (続き)

オプション	説明
<i>file_name</i>	ソース・ファイルのパス名。
<i>destination_name</i>	ソース・ファイルをコピーする先のパス名。 <i>destination_name</i> には、ファイルを受信するユーザ名を必ず含み、その形式は <i>host:username</i> である。ここで、 <i>host</i> はリモート・コンピュータ名であり、 <i>username</i> は受信側のユーザ名である。ローカル・ホスト上にファイルをコピーする場合には、 <i>destination_name</i> は、ファイルを送信するユーザ名だけでもよい。

14.8 UUCP ユーティリティのジョブ状態の表示

UUCP ユーティリティには、`uustat`、`uulog` および `uumonitor` という 3 つのコマンドがあります。これらのコマンドは UUCP ジョブの状態情報を表示します。詳細は以降の項で説明します。

14.8.1 uustat コマンド

`uustat` コマンドは、次のような情報や機能を提供することによって、UUCP ジョブをサポートします。

- `uucp` および `uuto` によって要求されたファイル転送の状態情報
- `uux` によって要求されたコマンド実行の状態情報
- リモート・コンピュータ上で実行するためにキューに登録されているジョブの制限付き制御
- `uucp` からのコピー要求の取り消し

`uustat` による状態レポートは、この形式を基本として、ワークステーションの画面に表示されます。`uustat` オプションを使用すると、表示形式を変更することができます。

```
jobid date/time status system_name username size file
```

注意

すべての状態表示は、ローカル・ホストおよびリモート・ホスト上のセキュリティ機能の影響を受けます。詳細はシステム管理者に問い合わせてください。

オプションを指定しないで `uustat` を入力すると、保留状態のキューが最後にクリーンアップされてから入力された、すべての UUCP コマンドに関する状態情報が表示されます。

特定のユーザが要求したジョブの状態をレポートするには、次に示すように、`-u` オプションを使用します。ここでは、ユーザ `hugh` が要求したジョブ状態を表示します。

```
% uustat -u hugh
```

`uustat` のオプションを使用すると、現在のキューおよび保留状態のキューに関する 2 種類の情報を作成することができます。`uustat -q` コマンドでは、現在のキューに関する情報を出力し、1 つまたは複数のリモート・ホスト上で、実行するためにキューに入っている UUCP ジョブや、現在実行中の UUCP ジョブを表示します。`uustat -a` コマンドでは、保留状態のキューに関する情報を出力し、設定した時間内に実行されなかったすべてのジョブを表示します。

注意

設定時間が経過したら、保留状態のキューのエントリを、`uucleanup` コマンドを使用して手動で削除するか、または `uudemon.cleanup` スクリプトによって自動的に削除してください。`uudemon.cleanup` スクリプトには、`/etc/cron` デーモンで起動する `/usr/spool/cron/crontabs/uucp` にエントリがあります。UUCP キューのクリーンアップについての詳細は、`uucleanup(8)` リファレンス・ページを参照するか、システム管理者に問い合わせてください。

14.8.1.1 `uustat` のオプションを使用した保留状態のキュー出力の表示

保留状態のキューに入っているすべての UUCP ジョブの状態を確認するには、次のように `uustat -a` を実行してください。

```
% uustat -a
sunC3113 Thu Jun 04 17:47:25 1999 S sun doc 289 D.car471afd8
gemN3130 Thu Jun 04 09:14:30 1999 R gem geo 338 D.car471bc0a
seaC3120 Wed Jun 03 16:02:33 1999 S sea doc 828 /u/doc/tt
seaC3119 Wed Jun 03 12:32:01 1999 S sea msg rmail doc
```

この出力例には、次の 7 つのフィールドがあります。

- フィールド 1 -- 操作のジョブ ID。

ローカルのコンピュータ上のプロセスを取り消す必要がある場合には、次の例のように、`-k` オプションを指定して、`uustat` コマンドを入力します。

```
% uustat -k seaC3119
```

- フィールド 2 -- UUCP コマンドを入力した日時。
- フィールド 3 -- ジョブがファイルを送信する場合は S、受信する場合は R。
- フィールド 4 -- コマンドが入力されたホスト名。
- フィールド 5 -- コマンドを実行したユーザ名。
- フィールド 6 -- ファイルのサイズ。リモート実行の場合 (最後の出力行) は、リモート・コマンド名 (rmail)。
- フィールド 7 -- フィールド 6 にサイズが表示された場合 (最初の 3 行の出力行) は、このフィールドにはファイル名が表示される。

ファイル名は、`/u/doc/tt` などのユーザが指定した名前、または、`D.car471afd8` のように、リモート実行に関連して UUCP が内部的にデータ・ファイルに割り当てた名前です。

特定のホストによって要求された保留状態のキューにあるすべての UUCP ジョブの状態をレポートするには、次のように `uustat -s` を実行してください。ここで示されている出力例は、ホスト `sky` に関するものです。

```
% uustat -s sky
skyNlbd7 Wed Jun 03 12:09:30 1999 S sky doc 522 /user/doc/A
skyClbd8 Wed Jun 03 12:10:15 1999 S sky doc 59 D.3b2a12ce4924
skyC3119 Wed Jun 03 12:11:18 1999 S sky doc rmail msg
```

この出力は、コマンド `uustat -a -s sky` で生成される出力と同じです。

14.8.1.2 uustat オプションを使用した現在のキュー出力の表示

各ホスト上にある現在実行中、あるいは実行するためにキューに入っているすべての UUCP ジョブの状態を確認するには、次のように、`uustat -q` コマンドを実行します。

```
% uustat -q
sea 3C      Mon Jul 13 09:14:35 1999 NO DEVICES AVAILABLE
sun 2C      Mon Jul 13 10:02:22 1999 SUCCESSFUL
gem 1C (2)  Mon Jul 13 10:12:48 1999 CAN'T ACCESS DEVICE
```

この出力例には、次の 5 つのフィールドがあります。

- フィールド 1 -- ホスト名。
- フィールド 2 -- ホストの保留状態のキューに入っているコマンド・ファイル (C) または実行ファイル (X) の数。
- フィールド 3 -- ファイルが保留状態のキューに入っている日数 (1 日以上の場合)。
- フィールド 4 -- UUCP がフィールド 1 のホストと通信を試みた最後の日時。
- フィールド 5 -- 対話の状態メッセージ。

詳細は、`uustat(1)` リファレンス・ページを参照してください。

表 14-10 は、`uustat` コマンドのオプションおよび必要なエントリの一覧です。

表 14-10: `uustat` コマンドのオプション

オプション	説明
<code>-a</code>	元の UUCP コマンドを入力したユーザに関係なく、保留状態のキューに入っているすべてのジョブについて情報を表示する。
<code>-k jobid</code>	<code>jobid</code> で指定した UUCP プロセスを取り消す。 <code>jobid</code> で指定した UUCP コマンドを入力した場合にのみ、ジョブを取り消すことができる。 また、スーパーユーザ特権を持っていれば、UUCP 要求も取り消すことができる。

表 14-10: uustat コマンドのオプション (続き)

オプション	説明
-m	UUCP を使用して別のコンピュータと最後に通信しようとしたときの状態をレポートする。たとえば、UUCP による要求が実行された場合、状態は成功としてレポートされ、ジョブが完了しなかった場合には、Login failed などのエラー・メッセージがレポートされる。
-p	ロック・ファイルのすべての PID 番号に対して、ps -flp (process status: a full, long list of specified process IDs: 指定されたプロセス ID に関する完全で詳細なリストによる処理状態の表示) を実行する。このオプションの使用は、スーパーユーザ特権を持つユーザのみ使用可能。
-q	各ホストについて、現在キューに入っているジョブを一覧表示する。これらのジョブは、実行を待っているか、または実行中のいずれかである。ホストに状態ファイルがある場合には、UUCP は、その日付、時刻、および状態情報をレポートする。プロセスが完了すると、UUCP は現在のキューからジョブ・リストを削除する。
-r <i>jobid</i>	ジョブ識別番号で指定した UUCP プロセスを復活させる。このオプションを使用すると、保留状態のキューにあるファイルに現在の日時のマークを付けて、割り当てたジョブの変更時間が経過するまで、クリーンアップ処理によってこれらのファイルが削除されないようにすることができる。
-shost	指定したホスト上で実行するためにユーザが入力した、すべての UUCP 要求の状態をレポートする。
-uusername	<i>username</i> のユーザによって、実行するために入力されたすべての UUCP 要求の状態をレポートする。 -shost および -uusername オプションを指定して uustat コマンドを実行すると、特定のホスト上の特定のユーザによって入力されたすべての UUCP 要求についての状態レポートを得ることができる。

14.8.2 uulog コマンドを使用した UUCP ログ・ファイルの表示

ローカル・ホストによって uucp、uuto または uux コマンドが使用される場合には、UUCP ログ・ファイルが作成されます。ログ・ファイルは、各リ

モート・ホストおよび各デーモンに対して作成されます。uulog コマンドを使用すると、これらのログ・ファイルを表示できます。uulog は、ホストによる uucp, uuto および uux コマンド要求の要約を表示します。

uulog コマンドは、次に示すデーモンのいずれかのアクティビティに関するログ・ファイルの内容を表示します。

- uucp および uuto によって呼び出される uucico デーモン
このデーモンのアクティビティは、`/usr/spool/uucp/.Log/uucico/host` にログが取られます。
- uux によって呼び出される uuxqt デーモン
このデーモンのアクティビティは、
`/usr/spool/uucp/.Log/uuxqt/host` にログが取られます。

uuxqt ログ・ファイルだけを表示するには、次のように uulog の `-x` オプションを使用します。

```
% /usr/lib/uucp/uulog -x
```

uulog コマンドを使用すれば、uucico ログ・ファイル、または任意のホストに対するファイル転送のログ、あるいはどちらかのログ・ファイルの最後の指定された数行のみを表示することもできます。たとえば、ホスト sky に対する uucico ログ・ファイルを表示するには、次のように、`-s` オプションを使用します。

```
% /usr/lib/uucp/uulog -s sky
```

ホスト sky に対するファイル転送ログの最後の40行を表示するには、次に示すように、`-f` オプションと行数を指定します。

```
% /usr/lib/uucp/uulog -f sky -40
```

表 14-11 は、uulog コマンドのオプションおよび必要なエントリの一覧です。

表 14-11: uulog コマンドのオプション

オプション	説明
-f <i>host</i>	指定された <i>host</i> のファイル転送ログに対して <code>tail -f</code> を実行し、ログ・ファイルの最後を表示する。Interrupt キー・シーケンスを使用すると、ファイルを終了してプロンプトに戻る。
-s[<i>host</i>]	指定されたホストに関するコピー要求についての情報を出力する。ホストが指定されない場合には、すべてのホストについての情報が表示される。
-x[<i>host</i>]	指定されたホストに関する <code>uuxqt</code> ログ・ファイルを表示する。ホストが指定されない場合には、すべてのホストについての情報が表示される。
- <i>number</i>	ログ・ファイルの最終 <i>number</i> 行を表示する。行数は <i>number</i> によって決まる。このパラメータの適用についての詳細は、 <code>tail(1)</code> リファレンス・ページを参照してください。

14.8.3 UUCP 状態のモニタ

`uumonitor` コマンドは、ジョブのバックログ、一時的なシャットダウン、あるいは電話番号またはログイン・パスワードのいずれかの変更によって状態が変更したホストを検出する際に有効です。

`uumonitor` 出力は、次の 6 つのフィールドから構成されます。

- フィールド 1 -- ホスト名。
- フィールド 2 -- リモート・ホストで実行するためにキューに入っているコマンド・ファイルの数。

多過ぎる場合 (ホストにもよるが、たとえば 100 ~ 1000 個) には、バックログの原因を判断しなければなりません。
- フィールド 3 -- リモート・ホストからのリモート実行要求の数。
- フィールド 4 -- 最後に実行したりリモート・ホストへの接続試行の結果。
- フィールド 5 -- リモート・ホストへのログインの失敗数 (ダイヤルの失敗を除く)。

20 より大きい場合には、これ以上実行されません。
- フィールド 6 -- 最後の状態エントリの時刻。

詳細は、`uumonitor(8)` リファレンス・ページを参照してください。



この付録では、vi の機能の概要を説明します。さらに詳しく知りたい場合は、vi(1) リファレンス・ページを参照してください。また、vi についての書籍は多数出版されています。

この付録は、3 つのセクションに分かれています。最初のセクションでは vi の基本的な機能について説明しています。2 つ目のセクションでは、作業の効率を上げるためのより高度なテクニックについて説明しています。3 つ目のセクションでは、ex コマンドの利用について説明しています。

どのようなコンピュータ・システムでも、メモ書きから C プログラムの修正まで、テキスト・ファイルの編集が最も基礎的な作業となります。vi は、たいいていのコンピュータ・ユーザにとって、毎日の作業に特に役に立ちます。vi を使用すると、すばやく簡単にファイルをオープンして編集し、結果を保存することができます。

vi テキスト・エディタは、次のような主要な機能はすべて備えています。

- 迅速な処理、特に起動時とグローバル操作の速度において優れています。
- フル・スクリーン編集およびスクロール機能があります。
- テキスト入力と編集は、別々のモードで行われます。
- ex エディタ・コマンドを使用したグローバル置換と複合編集コマンドがあります。
- オペレーティング・システム・レベルのコマンドが使用できます。
- システム・パラメータやキーボード・マッピングのカスタマイズができます。

この付録では、vi の基本的な機能の使い方について説明します。これを読み終えると、次のことができるようになります。

- 新しいファイルを作成して保存する (A.1.1 項) および (A.1.3 項)。
- 既存のファイルをアクセス (オープン) する (A.1.2 項)。

- ファイル内でカーソルを移動する (A.1.4 項)。
- 新しいテキストを入力する (A.1.5 項)。
- 既存のテキストを変更する (A.1.6 項)。
- 文字列を検索する (A.2 節)。
- テキストの移動やコピーを行う (A.2 節)。
- グローバル置換を行う (A.3.1 項)。
- テキストの一部または全部をファイルに書き出す (A.3.2 項)。
- テキスト・ブロックの削除，移動，コピーを行う (A.2.2 項 および A.3.3 項)。
- 編集環境をカスタマイズする (A.3.4 項)。

A.1 入門

この節では，`vi` を使用して次のことを行う方法について説明します。

- ファイルを作成する。
- ファイルを保存する。
- ファイル内で自由にカーソルを移動させる。
- テキストを作成する。
- テキストを削除する。
- テキストを修正する。

A.1.1 ファイルの作成

この付録の例では，`my.file` というファイルを使用しています。このファイルを作成するには，次に示すように `vi` コマンドを入力します。

```
$ vi my.file Return
```

すると，画面は次のようになります。

```
~
~
~
~
~
~
~
```

```
~
~
"my.file" [New file]
```

行頭にチルダ (~) が表示されている行は、ファイル内の空白行を示しています。my.file は空のファイルであるため、ファイル内の全行の先頭にチルダ (~) が表示されます。

vi エディタには、コマンド・モードと入力モードの 2 つのモードがあります。vi を開始したときには、コマンド・モードになっています。コマンド・モードでは、入力する文字はテキスト操作のためのコマンドとして解釈されます。入力モードでは、入力する文字はテキストとして解釈されます。

vi コマンドで新しいファイルを作成するとき、vi エディタはコマンド・モードになっています。つまり、vi はコマンドが入力されるのを待っています。ここでは、my.file が空であるため、vi を入力モードにしてファイルにテキストを入力できるようにします。

vi を入力モードにするには、次のようにタイプします。

```
i
```

i コマンドは画面上には表示されません。vi エディタは入力モードになり、これからタイプする文字はすべてテキストとして解釈されます。

次に示すサンプルのテキストでは、最後に入力した行で Escape キーを使用し、:wq コマンドで、ファイルを保存して vi エディタを終了していることに注意してください。サンプルのテキストどおりにタイプしてください。間違えたときには、バックスペース・キーを使用して修正してください。Return の指示があるところでは、Return キーを押して次のテキスト行に移動してください。

```
You can use this text file Return
to experiment with vi. Return
The examples shown here Return
will teach you the basics of vi. Escape
~
~
~
~
~
:wq

"my.file" 4 lines, 108 characters
$
```

注意

使用している端末またはワークステーションのセットアップ方法によっては、Escape キーが異なる機能を実行するようにプログラムされていることがあります。その場合には、キーボードのファンクション・キーのいずれか (おそらく F11) が、エスケープ機能を実行するようにセットアップされている可能性があります。Escape キーが正しく動作しない場合には、システム管理者に問い合わせてください。

`vi` が入力モードのときに Escape キーを押すと、`vi` はコマンド・モードに戻ります。コマンド・モードに戻ると、タイプする文字はすべてコマンドとして解釈されます。`:wq` コマンドは、`my.file` という名前でファイルを現在のディレクトリに書き込んで (保存する)、`vi` エディタを終了します。

`:wq` コマンドは、他の `vi` コマンドと大きく異なっています。これは、`:wq` コマンドが `vi` コマンドではなく、`ex` コマンドであるためです。`vi` がコマンド・モードであるときにコロン (:) をタイプすると、コロンが画面の下に表示されるのがわかるでしょう。コロン (:) は、`vi` 内から `ex` コマンドを開始します。どの `ex` コマンドも、`vi` がコマンド・モードのときに実行されます。コマンドの後には Return キーを押して、コマンドの入力が終了したことを `ex` に知らせなければなりません。`ex` コマンドについての詳細は、A.3 節を参照してください。

`vi` がどちらのモードかわからなくなった場合には、Escape キーを数回押してコマンド・モードにいることを確認してください。システムによっては、`vi` がコマンド・モードのときに Escape キーを押すと、ベルが鳴るように構成されていることがあります。

`vi` での Escape キーの使用や別の終了方法については、この付録の後の節で詳しく説明します。

`my.file` に入力したテキストは、この付録の残りの例で使用します。

A.1.2 既存のファイルのオープン

`vi` では、新規にファイルを作成するにも、既存のファイルをオープンするにも、次に示す同じ構文を使用します。

`vi filename`

`my.file` ファイルをオープンするには、次のように `vi` コマンドを入力します。

```
$ vi my.file
```

画面は次のようになります。

```
You can use this text file
to experiment with vi.
The examples shown here
will teach you the basics of vi.
```

```
~
~
~
~
~
~
```

```
"my.file" 4 lines, 108 characters
```

ファイルに入力したテキストが画面の上に表示されます。チルダではじまる行はファイル内の空行を表しています。画面の最下行には、ファイルの名前、ファイルの行数および文字数が表示されます。

A.1.3 ファイルの保存と `vi` の終了

前述の例では、`:wq` コマンドでファイルを保存して、`vi` エディタを終了することを学びました。ファイルを保存して終了する方法には、次に示す方法もあります。

- ファイルを保存して、編集作業は継続する。
- ファイルを保存して、`vi` を終了する。
- ファイルに対して行った変更を保存しないで、`vi` を終了する。

大きなテキスト・ファイルに対して、多量の情報を追加したり、削除したり、大規模な変更を行っている場合には、頻繁に（たとえば 10 分ごと）保存して重要なデータが失われないようにします。 `write` コマンドを使用す

ると、ファイル全体が現在のディレクトリに書き込まれます。write コマンドの形式は次のとおりです。

:w filename

filename はオプションであり、別のファイル名で保存する場合にのみ指定します。*filename* を省略した場合には、自動的に現在のファイル名で保存されます。:w コマンドを入力すると、現在のファイル名、行数、および文字数が画面の下に表示されます。新しいファイル名を入力した場合には、その新しいファイル名が表示されます。

注意

:w コマンドに新しいファイル名を指定すると、新しいファイル名と元のファイル名の 2 つファイルが現在のディレクトリに保存されます。

ファイルの変更が終了すると、ファイルの保存と vi の終了を同時に行うことができます。write および quit コマンドの形式は次のとおりです。

:wq

:wq コマンドは、同じ名前でファイルを保存して vi を終了し、シェル・プロンプトに戻ります。

ファイルに対して行った変更を保存しないで vi を終了する方法もあります。これは、間違って多数の行を削除してしまい、初めからやり直したい場合などに役立ちます。この場合には、vi を終了すると、ファイルは元の状態に戻ります。ただし、ファイルが元の状態に戻るのは、現在の編集セッションでファイルを保存していない場合に限りです。変更を保存しないで vi を終了するには、次のように入力します。

:q!

:q! コマンドでファイルを終了しても、ディレクトリから削除されることはありません。ファイルはディレクトリに残っていますが、ファイルに対して行った変更は含まれていません。

表 A-1 に、ファイルの保存や vi エディタの終了に使用するコマンドをまとめています。

表 A-1: write および quit コマンドのまとめ

コマンド	説明
:w	ファイル全体を現在のファイル名で保存する。vi エディタは終了しない。
:w <i>filename</i>	ファイル全体を新しいファイル名で保存する。vi エディタは終了しない。新しいファイル名と元のファイル名がディレクトリに存在する。
:wq	ファイル全体を現在のファイル名で保存して、同時に vi エディタを終了する。
:q!	ファイルを終了し、vi エディタも終了する。ファイルを最後に保存した以降の変更は保存しない。

A.1.4 ファイル内での移動

`my.file` をクローズしている場合には、次のコマンドを使用して、再オープンします。

```
$ vi my.file
```

カーソルはファイルの最初の文字のところ、つまり、`You` の `Y` のところにあるはずです。

この付録で前述したように、`vi` を起動したときにはコマンド・モードになっています。コマンド・モードでは、入力した文字はファイルへのテキスト入力ではなく、コマンドとして扱われます。

A.1.4.1 カーソルの移動 (上, 下, 左, 右)

`vi` がコマンド・モードのときには、キーボード上のいくつかのキーが移動キーとして使用されます。次の英字キーでカーソル移動を制御します。

- `h` -- カーソルを 1 文字左へ移動させる。
- `j` -- カーソルを 1 行下へ移動させる。
- `k` -- カーソルを 1 行上へ移動させる。
- `l` -- カーソルを 1 文字右へ移動させる。

移動キーを使用して、カーソルを `experiment` という語の最初の文字のところへ移動させてみてください。次のようにタイプします。

```
lllj
```

キーボードに矢印キーがある場合には、矢印キーを使用してカーソルを上下左右に移動させることができます。ただし、`h`、`j`、`k`、`l`の各キーを使用すれば、指をキーボードの定位置から離さずに入力できるので、入力スピードを上げることができます。キーボードによっては、`h`、`j`、`k`、`l`の各キーは繰り返し型のキーになっており、キーを押したままにしておくと、キーを離すまでそのキーの動作を繰り返します。たとえば、`j`を押したままで、ファイルの行を迅速にスクロールすることができます。

コマンド・モードでは、Return キーはカーソル移動キーと同様に動作します。Return キーを押すと、カーソルは次の行の先頭に移動します。この動作は、カーソルを次行の同じ文字位置に移動させる `j` キーとは異なり、Return キーはカーソルを次行の先頭に移動させます。

コマンド・モードでは、ハイフン (`-`) はカーソルを前行の先頭に移動させます。この機能はファイル内で逆にスクロールする場合に役立ちます。この動作は、前行の同じ文字位置に移動させる `k` キーとは異なり、`-` はカーソルを前行の先頭に移動させます。

ここで説明したとおりにカーソルが動くことを確認したら、次の項に進む前に、カーソルを `experiment` の最初の文字に戻してください。

A.1.4.2 カーソルの移動 (語、行、文、パラグラフ)

`w` コマンドを使用して、カーソルを単語単位で移動させることもできます。`w` コマンドは、カーソルを次の文字の先頭に移動します。たとえば、`with` という単語の先頭にカーソルを移動させるには、次のようにタイプします。

`w`

また、`b` コマンドを使用すると、カーソルを 1 つ前の単語の先頭まで戻すことができます。たとえば、`experiment` という単語の先頭にカーソルを移動させるには、次のようにタイプします。

`b`

`b` コマンドを使用しないで、単語の先頭から次のようにタイプするとどうなるでしょうか。

`l111lb`

カーソルは、`experiment` の先頭に戻ります。

この単語移動コマンドは、必要な場合には、次テキスト行、または前テキスト行にラップします。次のようにタイプして、カーソルを `text` という単語の先頭に移動させてください。

`bbb`

カーソルが逆方向に移動して、前行にラップすることに注意してください。

ここで、他の移動コマンドも紹介しておきましょう。ゼロ (0) はカーソルを現在のテキスト行の最初にカーソルを移動させ、ドル記号 (\$) はカーソルを現在のテキスト行の最後に移動させます。

右カッコ (]) はカーソルを次の文の先頭に移動させ、左カッコ ([) はカーソルを前の文の先頭に移動させることができます。

右中カッコ (}) でカーソルは次のパラグラフの先頭に移動し、左中カッコ ({) で前のパラグラフの先頭に移動します。

A.1.4.3 ファイル内でのカーソルの移動とスクロール

大きなファイルでは、制御キーを押すことによって、画面単位でカーソルを移動させることができます。

- `Ctrl/F` -- カーソルを 1 画面先に移動させる。
- `Ctrl/B` -- カーソルを 1 画面後ろに移動させる。
- `Ctrl/D` -- 画面を半分下方 (前) にスクロールさせて、カーソルを移動させる。
- `Ctrl/U` -- 画面を半分上方 (後ろ) にスクロールさせて、カーソルを移動させる。

次に示す英大文字のコマンドは、カーソルを広範囲に移動させます。

- `H` -- カーソルをホーム (Home) 位置、つまりファイルの先頭に移動させる。
- `G` -- カーソルをファイルの最終行に移動 (Go) させる。

A.1.4.4 移動コマンドのまとめ

`vi` エディタには他にも多くの移動コマンドがあります。この付録で基本を習得した後、詳しくは `vi(1)` リファレンス・ページを参照してください。

表 A-2 にカーソル移動コマンドの一覧を示します。カーソル移動キーは vi がコマンド・モードのときのみ有効です。

表 A-2: カーソル移動コマンド一覧

コマンド	説明
h	カーソルを 1 文字左に移動させる。
j	カーソルを同じ文字位置で 1 行下に移動させる。
k	カーソルを同じ文字位置で 1 行上に移動させる。
l	カーソルを 1 文字右に移動させる。
Return キー	カーソルを次行の先頭に移動させる。
—	カーソルを前の行の先頭に移動させる。
w	カーソルを次の単語の先頭に移動させる。
b	カーソルを前の単語の先頭に移動させる。
0	カーソルを現在の行の先頭に移動させる。
\$	カーソルを現在の行の終わりに移動させる。
)	カーソルを次の文の先頭に移動させる。
(カーソルを前の文の先頭に移動させる。
}	カーソルを次のパラグラフの先頭に移動させる。
{	カーソルを前のパラグラフの先頭に移動させる。
Ctrl/D	画面を半分下方へスクロールさせる。
Ctrl/F	カーソルを 1 画面先へ移動させる。
Ctrl/B	カーソルを 1 画面後ろへ移動させる。
Ctrl/U	画面を半分上方へスクロールさせる。
H	カーソルをホーム位置(ファイルの先頭文字)に移動させる。
G	カーソルをファイルの最終行へ移動させる。

A.1.5 新しいテキストの入力

ファイルに新しいテキストを入力するには、vi は入力モードでなければなりません。入力モードでは、タイプした文字は直接ファイルのテキストと

して挿入されます。入力モードからコマンド・モードに戻るには、Escape キーを 1 回押します。

テキストの挿入に使用するコマンドはいくつかありますが、どのテキスト挿入のコマンドが入力されても、vi は自動的に入力モードになります。

この項の練習を始める前に、まず `my.file` をオープンし、カーソルを最初の行にある単語 `text` に移動させてください。

初めて `my.file` にテキストを入力したときのように、挿入コマンドを使用して、`text` の直前に `new` を挿入します。カーソルを `text` の最初の `t` の上に置き、次の挿入コマンドをタイプして、vi を入力モードにします。

```
i
```

次に、`new` という単語を入力して、スペース・バーを 1 回押します。

```
new Space
```

入力モードを出るには、Escape キーを押します。

```
Escape
```

カーソルは、`new` と `text` の間のスペースにあるはずです。

`i` コマンドを入力すると、カーソルのある文字の直前からテキストの挿入を開始します。このため、カーソルが単語の最初の文字にあるときにテキストを挿入する場合は、スペース・バーを押して単語と単語の間にスペースを入れるようにしなければなりません。

テキスト挿入に使用するもう 1 つのコマンドは、追加 (`a`) コマンドです。挿入コマンドとは異なり、`a` コマンドは入力した文字をカーソル位置の直後に追加します。`a` コマンドがどのように動作するかをみるため、カーソル移動キーを使用してカーソルを `you` の `u` に移動させた後、次のようにタイプしてください。

```
a  
, too, Escape
```

vi テキスト・エディタは入力されたテキストを `You` の後ろに追加します。カーソルは 2 つ目のコンマの上にあるはずです。

`o` コマンドは、カーソルのある行の下に新しい行を作成し、その行の先頭からテキストを挿入できるようにします。このファイルの終わりに文を追加するため、次のように Return キーを 3 回押して、カーソルをファイルの最終行に移動させます。

```
Return
Return
Return
```

カーソルは、`will` に移動します。ここで、次のようにタイプすると、カーソルのある行の下に新しい行が作成され、`vi` は自動的に入力モードになります。

○

次のテキスト例を入力してください。`Return` のところでは、必ず `Return` キーを押してください。入力が終了したら、`Escape` キーを押してコマンド・モードに戻ります。

```
New text can be easily entered Return
while in input mode. Escape
```

画面は次のようになっているはずです。

```
You, too, can use this new text file
to experiment with vi.
The examples shown here
will teach you the basics of vi.
New text can be easily entered
while in input mode.
~
~
~
~
~
~
```

○ コマンドを使用すると、現在の行の上に新しい行を作成して、その行の先頭からテキストを挿入することができます。このコマンドは、ファイルの先頭に新しいテキストを追加するときにたいへん便利ですが、ファイルの中のどこで使用してもかまいません。練習としてこのコマンドを使用し、新しい行を作成してテキストを入力するには、まず、カーソルをファイルの1行目に移動させます。カーソル移動コマンド `H` を使用できることを思い出してください。その後、次のようにタイプします。

○

```
Opening a new line is easy. Escape
```

`vi` テキスト・エディタは `Escape` キーが押されると、コマンド・モードに戻ります。

vi エディタには、入力モードに入るコマンドがもう 2 つあります。I コマンドと A コマンドです。I コマンドは、現在の行の最初の文字の前にテキストを挿入し、A コマンドは、現在の行の最後の文字の後にテキストを挿入します。

次のようにタイプして、行の先頭にテキストを挿入する練習をしてください。

```
I
Inserting text is easy. [Space][Escape]
```

行の終わりにテキストを追加するには、次のようにタイプします。

```
A
Really! [Escape]
```

画面は次のようになっているはずです。

```
Inserting a line is easy. Opening a new line is easy. Really!
You, too, can use this new text file
to experiment with vi.
The examples shown here
will teach you the basics of vi.
New text can be easily entered
while in input mode.
~
~
~
~
~
~
```

表 A-3 に、ファイルにテキストを挿入したり追加するために使用するコマンドの一覧を示します。これらのコマンドはコマンド・モードで実行しますが、これらのコマンドを入力すると、vi は自動的に入力モードになります。

表 A-3: テキスト挿入コマンド一覧

コマンド	説明
i	カーソル位置の直前にテキストを挿入する。
a	カーソル位置の直後にテキストを追加する。
I	カーソルのある行の先頭にテキストを挿入する。
A	カーソルのある行の終わりにテキストを追加する。
o	カーソルのある行の下に新しい行を作成する。
O	カーソルのある行の上に新しい行を作成する。

A.1.6 テキストの編集

ここまでは、新しいテキストをファイルに追加する方法を学習しました。ところで、テキストに変更を加える必要がある場合にはどうするのでしょうか。vi エディタには、テキストの削除や変更のためのコマンドが用意されています。たとえば、`my.file` の 6 行目にある `easily` という単語を削除するには、カーソルをこの単語の先頭文字に移動させて、次のようにタイプします。

```
dw
```

これは、削除コマンド `d` と、移動コマンド `w` を組み合わせたものです。実際、vi のコマンドの多くは、移動コマンドと組み合わせて、動作の持続期間を指定することができます。vi コマンドの一般形式は、次のとおりです。

```
[number] [command] motion
```

`command` は動作コマンドを表し、`motion` は移動コマンドを表します。`number` はオプションであり、コマンドを繰り返す回数を表します。この一般形式を使用して、カーソルをすばやく移動させることもできます。

これを説明する前に、まず、`H` をタイプして、カーソルを `my.file` の先頭に移動させてください。ここで、カーソルを 4 単語先に移動させてみましょう。次のように入力します。

```
4w
```

カーソルが 4 単語先に移動して、5 番目の単語 `easy` の最初の文字の上に来ます。

A.1.6.1 単語の削除

コマンドの一般形式を使用して、このテキスト・ファイルの最後の 5 文字を削除することができます。Return キーを数回押してカーソルを最後の行の先頭に移動させてから、次のように入力します。

```
5dw  
:w
```

1 行全体を削除するには、4 単語ではなく、5 単語の削除が必要です。これは、行の最後についているピリオドも 1 文字として数えるためです。単語削除コマンドを使用するときには、句読点もそれぞれ 1 単語として数えます。時々ファイルの保存を行うことを思い出してもらうため、この例では `:w` コマンドを使用してファイルを `write` (保存) しています。

単語の一部だけを削除する場合は、`x` コマンドを使用します。`x` コマンドは一度に 1 文字を削除します。カーソルを `examples` の `s` まで移動して、このコマンドを使用してみましょう。`x` キーを 1 回押すと、`s` が削除されます。

A.1.6.2 行の削除

`dd` コマンドは一度に 1 行全体を削除するためのコマンドです。`dd` コマンドに数字を指定して、複数行を削除することもできます。たとえば、カーソルを 6 行目 (`New` で始まる行) に移動させて、次のようにタイプしてください。

```
2dd
```

ファイルの 6 行目と 7 行目 (7 行目は空) が同時に削除されます。`dd` コマンドは、数字を指定しなくても使用できます。その場合には、一度に 1 行ずつ削除されます。

`D` コマンドは、現在カーソルがある行のカーソル位置からその行の終わりまでをクリアしますが、行そのものは削除しません。カーソルが行の先頭にある場合には、行全体がクリアされます。このコマンドを使用すると、`dw` コマンドなどのように語数を指定する必要がないため、作業をスピードアップすることができます。このコマンドは、行全体を書き換える場合に便利です。カーソルを行の先頭に置き、`D` コマンドに続けてテキスト挿入コマンド (`i`, `I`, `a`, `A` のいずれか) を入力すると、少ないキー入力で、現在の行のテキストをクリアして、新しいテキストを再入力することができます。

A.1.6.3 テキストの変更

テキストの変更コマンド `c` を使用すると、削除と入力モードへの移行が組み合わされた動作を行うことができます。このコマンドは `d` コマンドと同じ一般形式を取ります。`new text` というテキストを `almost new demo` に変更するには、まず、`new` の先頭の文字にカーソルを移動してから、次のコマンドを入力します。

```
2cw
```

テキストはすぐには消えません。ドル記号 (\$) が変更範囲の最後 (`text` の最後の `t`) に表示され、`vi` は自動的に入力モードになります。ここで入力したテキストがドル記号までの既存のテキストを重ね書きし、必要に応じて変更範囲は広がります。次の新しいテキストを入力してください。

```
almost new demo Escape
```

A.1.6.4 テキスト編集コマンドのまとめ

これまでの項で説明したように、テキスト編集コマンドは移動コマンドと一緒に使用することにより、編集力をアップすることができます。テキスト編集コマンドは数字と組み合わせて使用すると、複数の単語や行を同時に変更したり削除することができます。表 A-4 に、テキスト編集に使用するコマンドの一覧を示します。

表 A-4: テキスト編集コマンドのまとめ

コマンド	説明
<code>cw</code>	カーソルのある単語を新しく入力するテキストに変更する。必要な長さのテキストを新しく入力することができる。Escape キーを押して、変更の終了を知らせる。
<code>ncw</code>	n 個の単語を新しく入力するテキストに変更する。新しく入力するテキストは n 個の単語である必要はない。必要な単語数のテキストに変更できる。Escape キーを押して、変更の終了を知らせる。
<code>D</code>	カーソル位置からその行の終わりまでのテキストをクリアする。行そのものは削除しないため、テキストを追加できる。
<code>dd</code>	カーソルのある行を削除する。
<code>ndd</code>	カーソルのある行から n 行を削除する。
<code>dw</code>	カーソルのある単語を削除する。
<code>ndw</code>	カーソルのある単語から n 語を削除する。
<code>x</code>	カーソルのある文字を削除する。

A.1.7 コマンドの取り消し

変更を行った後で、それが間違っていることに気づいた場合、次のコマンドを実行していなければ、間違いを正すことができます。`u` コマンドは、最後に入力したコマンドを取り消します。次のようにタイプして、最後のコマンド `2cw` を取り消してみてください。

`u`

`2cw` コマンドを実行した後、別のコマンドを実行していない場合には、文字列 `almost new demo` が `new text` に戻ります。

大文字の U コマンドは、カーソルのある行に対して行われたすべての変更を取り消して、その行を元の状態に戻します。U コマンドは、カーソルを別の行に移動させていない場合にだけ有効です。

A.1.8 編集の終了

この付録の練習を終えたら、ファイルを保存して vi を終了します。変更内容を保存してから vi を終了させるには、次のように入力します。

```
:wq Return
```

変更内容を保存しないで vi を終了させるには、次のように入力します。

```
:q! Return
```

これで、vi を使用してファイルの編集ができるようになりました。次の各節で、生産性を上げるための高度なテクニックや環境をカスタマイズする方法を説明します。

A.2 高度なテクニックの使用

この節では、文字列を検索したり、テキストを移動させたり、コピーやペーストを行う方法について説明します。文書が大きくなるにつれ、こうしたテクニックを駆使すると、効率的に作業することができます。

A.2.1 文字列の検索

大きな文書では、特定の文字列をテキストの中で探すのは、たいへん時間のかかる作業です。スラッシュ (/) コマンドを使用して、ファイル内で文字列を検索することができます。スラッシュ (/) を入力すると、検索するテキスト文字列を入力するように求められます。Return キーを押すと、vi は入力した文字列が最初に現れる位置を検索します。

my.file をまだオープンしていない場合には、そのファイルを再オープンします。この付録で説明したカーソル移動キーを使用して、文書の先頭に移動します。文字列 th を検索するには、次のように入力します。

```
/th Return
```

スラッシュ (/) を入力するとすぐに、/ が画面の最下行に表示されます。これはコロン (:) を入力する場合と同様です。文字列 th を入力すると、画面の最下行にエコー (表示) されます。検索する文字列を入力し間違えた場合には、バックスペース・キーを使用して訂正できます。

Return キーを押すと、入力した文字列が最初に現れる位置にカーソルが移動します。この例の場合は、`this` という単語の `th` に移動します。

`n` コマンドは、前回検索した文字列が次に現れる位置を検索します。次のように入力してみてください。

`n`

カーソルは文字列が次に現れるところ、つまり、`with` の `th` の位置に移動します。

同様に、`N` コマンドも検索文字列が次に現れる位置を検索しますが、`n` コマンドと逆方向に検索します。

疑問符 (?) コマンドも、文字列の検索を開始するために使用されますが、`?` コマンドはファイルを逆方向に検索します。逆方向に検索する場合、`n` コマンドで、カーソルは逆方向に移動して前にある文字列を検索し、`N` で順方向に検索します。`/` で検索する場合の全く逆になります。

A.2.2 テキストの削除と移動

テキスト・ブロックを移動するには、まず、移動するテキストを選択します。選択する方法はすでに説明済みです。削除 (`d`) コマンドは、テキスト行を削除するだけでなく、ペースト・バッファへのコピーも行います。一度ペースト・バッファに入れると、カーソルの位置を決めた後、小文字の `p` コマンドを使用して、カーソル位置の後にテキストを移動 (ペースト) させることができます。

カーソルをファイルの 1 行目に移動させて、次のようにタイプします。

`dd`

するとその行は削除され、ペースト・バッファにコピーされます。カーソルは、ファイルの次の行に移動します。バッファにコピーされた行を、カーソルがある行の次の行に戻すには、次のようにタイプします。

`p`

大文字の `P` (Paste) コマンドは、テキストをカーソルのある行の下ではなくに上にペーストします。

文字や単語のブロックを削除すると、削除されたテキストは現在の行の新しい位置にペーストされます。たとえば、`can` という単語を、`with` という単

```

/
can Return
dw
/
with Return
P

```

テキストのコピーは、移動の場合と同様の方法で行います。ただし、テキスト削除コマンド `d` の代わりに、ヤंक (yank) コマンド `y` を使用します。`y` コマンドは、指定したテキストをファイルから削除することなくペースト・バッファにコピーします。構文は `d` コマンドの場合と同じです。また、`yy` コマンドを使用すると、`dd` と同じように、テキスト行全体をペースト・バッファにコピーすることもできます。

2yy
j
p

A.2.4 その他の vi の機能

J カーソルのある行と次の行を結合して 1 行にする。

- 直前のコマンドを繰り返す。

vi 入門 A-19

x	カーソル位置の文字を削除する。
~	カーソル位置の文字が小文字ならば大文字に，大文字ならば小文字に変換する。
!	オペレーティング・システムのコマンドを，カーソルがある行で実行し，テキストを出力されたものと置き換える。
Ctrl/L	画面表示に何か問題が発生した場合に，画面をクリアする。画面上の出力に混乱が生じた場合には，Ctrl/L を押す。

A.3 ex コマンドの利用

vi スクリーン・エディタは，ex ライン・エディタをベースに設計されています。ベースになっている ex ライン・エディタを利用することにより，テキスト・ファイル全体や，ファイル内の広範囲にわたる部分で一括変換を行うことができます。ex コマンドには，vi コマンド内からコロン (:) コマンドを使用してアクセスできます。:wq や :q! といった編集内容を書き込むコマンドやエディタの終了に用いる ex コマンドについては，この付録の前の節ですでに紹介しています。

コロン (:) コマンドを使用すると，ex によってエディタ画面の最下行にコロン (:) ではじまるコマンド行が表示されます。各 ex コマンドの最後には Return キーを押します。また，vi コマンド Q を使用すると，ex を半永久的に入力することができます。このコマンドは，ユーザが明示的に vi に戻るまで，処理を ex に引き渡します。偶然にこのような状態になることがよくあります。そのような場合に vi に戻るには，次のように，コロン (:) プロンプトに対して vi とタイプしてから，Return キーを押します。

```
:vi Return
```

: [*address*[, *address*]] *command*

<code>line number</code>	アドレスを絶対行番号で指定する。
--------------------------	------------------

`/pattern/` パターンを含む次の行を指定する。

カーソルのある行を指定する。

\$ ファイルの最後の行を指定する。

address±lines アドレス行からの相対オフセットを指定する。

% ファイルの全行を指定し、両方のアドレスの代わりに使用する。

A.3.1 置換

vi 入門 A-21

```
:%s/is/was/g Return
```

この置換コマンドは、% アドレスにより、ファイル内の全行に適用されます。スラッシュ (/) はセパレータとして使用されます。コマンドの終わりに g 引数が指定されている場合は、各行でパターンが見つかるたびにすべて (グローバルに) 置き換えられます。g 引数が指定されていない場合には、各行で 1 回しか置き換えられません。

置換を行うときは、目的どおりに実行されたかどうかを確認する必要があります。上記のコマンド行では、is がすべて was に変換されたため、this が thwas に変換されたことに注意してください。

c 引数を g 引数と一緒に指定すると、置換のたびに確認することができます。確認の形式は少し複雑ですが、グローバル置換をする問題がありそうな場合には、この方法をお勧めします。

置換時の確認の例として、次のコマンドを入力して thwas を this に戻します。

```
:%s/thwas/this/gc Return
```

画面の下には、次のプロンプトが表示されます。

```
You, too, use thwas new text file
      ^^^^^
```

次の例で示すように、y とタイプして Return キーを押します。すると、次の置換に関して確認が求められます。

```
You, too, use thwas new text file
      ^^^^^y Return
You, too, use thwas new text file
      ^^^^^
```

y をタイプして Return キーを押します。Hit return to continue というプロンプトに対しては、次のようにもう一度 Return キーを押します。

```
You, too, use thwas new text file
      ^^^^^y
You, too, use thwas new text file
      ^^^^^y Return
[Hit return to continue] Return
```

これで、2 箇所で見つかった thwas という単語が this に戻りました。vi はコマンド・モードに戻り、カーソルは最後に置換が行われた行の先頭に来ます。

次に、サンプルのファイルで、別の置換を試みましょう。まず、\$ (最後の行の先頭へ移動する)、o (新しい行を作成する)、yy (コピー)、p (ペースト) の各コマンドを使用して、次のように、新しいテキスト行を 3 行ファイルに追加します。

```
:$ Return
o
Some new text with a misspelling. Escape
yy
p
p
p
```

これで、新しいテキストが 4 行増えたことになり、どの行にもスペルミスのある単語 `misspelling` が含まれています。

このスペルミスを訂正するには、次のコマンドのいずれかを入力します。

```
:1,$s/misspelling/misspelling/ Return
```

または、

```
:5,8s/misspelling/misspelling/ Return
```

最初の例では、アドレス `1,$` は、1 行目 (1) からファイルの最終行 (\$) まで置換を行うことを示しています。2 番目の例では、`5,8` は 5 行目から 8 行目までの行に対して置換を行うことを指定しています。変更は各行で 1 回ずつなので、`g` 演算子を使用する必要はありません。

A.3.2 ファイル全体または一部の書き込み

`:wq` コマンドは特殊な `ex` コマンドで、ファイル全体を書き込みます。このコマンドは、書き込みコマンド `w` と終了コマンド `q` の機能を組み合わせたものです。終了コマンドが取ることのできる引数は感嘆符 (!) だけです。このコマンドは、変更内容を保存しないで強制終了します。

`w` コマンドは、アドレスやファイル名の引数を取ることができるため、テキストの一部を別のファイルに保存することができます。たとえば、テキストの最初の 3 行を新しいファイル、`my.new.file` に保存するには、次のコマンドを使用します。

```
:1,3w my.new.file Return
"my.new.file" [New file] 3 lines, 130 characters
```

A.3.3 テキスト・ブロックの削除

ex の削除コマンドは、vi と同様、d です。現在の行からファイルの終わりまでを削除するには、次のコマンドを使用します。

```
:.,$d Return
```

A.3.4 ユーザ環境のカスタマイズ

ex エディタには、ユーザの vi 環境をカスタマイズするメカニズムが 2 種類用意されています。1 つは、:set コマンドを使用して環境変数を設定する方法であり、もう 1 つは、:map コマンドを使用して、キー・シーケンスを vi コマンド・キーにマップする方法です。

環境変数を設定する場合、ブール変数は option または no option として割り当てるか、あるいは、option=value として割り当てます。環境変数の完全な一覧は、ex(1) リファレンス・ページに記載されています。表 A-5 に、一般的な変数をいくつか示します。

表 A-5: vi の一般的な環境変数

変数	説明
errorbells	エラー発生時に、ベルを鳴らすことを指定する。これが省略時の設定。
ignorecase	検索時には大文字、小文字の区別を無視することを指定する。省略時の設定は noignorecase。
number	行番号を左端に表示することを指定する。省略時の設定は nonumber。
showmatch	対になった小カッコや中カッコを入力するときに、カーソルが対の一方の文字に移動してから改行することを指定する。省略時の設定は noshowmatch。
tabstop	タブ間のスペース数を指定する。省略時の値は tabstop=8。

表 A-5: vi の一般的な環境変数 (続き)

変数	説明
wrapscan	検索時に、ファイルの先頭または最後でラップすることを指定する。省略時の設定値は wrapscan。
wrapmargin	画面の端から指定文字数分だけ自動的に右マージンを作成する。カーソルが指定された右マージンに達すると、新しい行が自動的に作成され、入力中の単語は次の行に送られる。省略時の値は wrapmargin=0。 wrapmargin 変数は、ユーザの好みの値に設定する方がよい。設定しない場合、vi は省略時の値である 0 を使用する。省略時の値を使用すると、カーソルは画面の行末に来到次の行に移る。この場合、入力中の単語は 2 行にまたがって表示される。

次のコマンドを使用して、サンプル・ファイルに行番号を表示させてください。

```
:set number Return
```

行番号を削除するには、次のコマンドを入力します。

```
:set nonumber Return
```

:map コマンドは、vi コマンド・キーに vi コマンドのシーケンスを割り当てます。:map コマンドの構文は次のとおりです。

```
:map key sequence Return
```

このコマンド・シーケンスが、該当するキーに割り当てられているコマンドの代わりにキーに割り当てられます。コマンド・シーケンスは、マップしたいキー・ストロークと同じでなければなりません。ただし、Return キー、Escape キーなどの特殊キーや、Ctrl キーと同時に押すキーの前には、Ctrl/V をタイプします。q キーや v キーには関連するコマンドがないので、マップにはこれらのキーを使用することをお勧めします。

たとえば、テキストに "This space held for new text" という行を挿入するキー・シーケンスをマップするには、次のようなコマンドを使用します。

```
:map q oThis space held for new text Ctrl/VEscapeReturn
```

上記の例では、エスケープ文字の前に Ctrl/V をつけていることに注意してください。

A.3.5 カスタマイズした内容の保存

カスタマイズした環境を永久保存するには、該当する ex コマンドを、ホーム・ディレクトリにある `.exrc` という名前のファイルに入れます。このファイルに入れたコマンドは、vi や ex を入力するたびに有効になります。このファイルの中では、vi コマンドの `(:)` を使用する必要はありません。なぜなら、これらのコマンドは基礎となる ex エディタから直接読み取られるからです。

たとえば、ファイルの行番号が常に表示され、前の項で示したシーケンスをマップし、5 文字分の自動右マージン設定を行うように環境をカスタマイズするには、vi を使用して、ホーム・ディレクトリの `.exrc` ファイルをオープンし、次の行を追加します。

```
set number
map q oThis space held for new text Ctrl/V Escape
set wrapmargin=5
```

このファイルに書き込んだ後、サンプル・ファイルをオープンして、望みどおりの環境設定が行われたことを確認してください。

B

ed によるファイルの作成および編集

この付録では、行編集プログラム ed を使用して、テキスト・ファイルを作成、編集 (修正)、表示、保存する方法を説明します。システムに他の編集プログラムがある場合には、そのプログラムでこれらの作業を行ってもかまいません。

ed の操作方法を学習するには、この付録にある例題をシステム上で実際に実行してみるといいでしょう。例は次第に内容が難しくなるので、1 題ずつ順序正しく進んで行くことが大切です。また、例題どおりに実行して、画面の表示が本書の例と一致するようにしてください。

例題の中では、ユーザがタイプするところはすべて太字で印刷されています。説明の中で何かを入力するように指示がある場合は、その行の情報をすべてタイプして、Return キーを押します。

ed は行エディタなので、ファイルの内容は 1 行ずつしか編集できません。画面には他のテキストが表示されていても、編集できるのは現在の行だけです。画面編集プログラムを使用した経験がある場合には、そのプログラムと ed との違いをよく認識しておいてください。たとえば、ed プログラムでは、矢印キーを使用して、現在の行を変更することはできません。

B.1 テキスト・ファイルと編集バッファ

ファイルとは、ある名前の下で、コンピュータにまとめて格納されたデータの集まりです。普通のファイル・フォルダが、コンピュータに入ったものだと考えてください。ファイルには、手紙やレポート、あるいはその他の文献を収めたり、コンピュータ・プログラムのソース・コードを入れたりします。

編集バッファは一時的な記憶域であり、作業中のファイルを保持するところです。これは、一般の机の上に当たる部分です。テキスト・ファイルで作業を行う場合、テキスト・ファイルを編集バッファに置き、そのファイルに変更を加え (編集し)、その後、バッファの内容を永久的な記憶域に転送 (コピー) します。

この付録では、ed エディタを使用して、テキスト・ファイルを作成、表示、保存、編集 (修正) する方法を説明します。

B.2 テキスト・ファイルの作成と保存

テキスト・ファイルを作成して保存するには、次の手順に従ってください。詳細については、以降の項で説明します。

1. シェル・プロンプトから、次のコマンドを入力する。

```
$ ed filename
```

filename には、作成または編集するファイルの名前を指定します。

2. `? filename` のメッセージが表示された場合には、次の追加コマンドを入力する。

```
a
```

3. テキストを入力する。

4. テキストの追加を終了するには、新しい行の先頭でドット (.) を入力する。

5. 次のコマンドを入力して、編集バッファの内容をファイル *filename* にコピーする。

```
w
```

6. ed プログラムを終了するには、次のコマンドを入力する。

```
q
```

B.2.1 ed プログラムの起動

ed プログラムを起動するには、`ed filename` 形式のコマンドをシェル・プロンプト (\$) の後に入力します。

次の例では、`ed afile` コマンドにより ed プログラムを起動して、*afile* という名前のファイルで作業を行うことを示します。

```
$ ed afile
?afile
```

```
—
```

ed プログラムは、`?afile` というメッセージを表示します。これは、そういう名のファイルが存在しないことを意味します。ここで、`a` (追加) サブコマ

ンドを使用して、`afile` を作成し、テキストを入力することができます。このサブコマンドについては、次の項目で説明します。

B.2.2 テキストの入力 – `a` (追加) サブコマンド

ファイルにテキストを追加するには、`a` を入力します。サブコマンド `a` は、ユーザが入力するテキストを編集バッファに追加するように `ed` に対して指示します。そのファイルにすでにテキストが入っている場合には、サブコマンド `a` は新しいテキストをファイルの終わりに追加します。

テキストをタイプし、各行末で `Return` キーを押します。テキストを全部入力し終わったら、新しい行の先頭にドット (`.`) を入力します。

注意

行末で `Return` キーを押さない場合には、行末まで入力すると、`ed` プログラムが自動的にカーソルを次の行に移動します。ただし画面上で何行入力しても、`Return` キーが押されるまで `ed` プログラムは入力されたテキストをすべて 1 行と見なします。つまり、入力行が (ワークステーションの表示設定に基づいて) 次の行にラップされるということです。

次の例は、`afile` ファイルにテキストを入力する方法を示しています。

```
a
The only way to stop
appending is to enter a
line that contains only
a dot.
.
```

バッファにテキストを追加するのを一旦やめてから、また追加したくなった場合には、もう一度 `a` サブコマンドを入力します。テキストをタイプし、入力が終了したら、新しい行の先頭にドット (`.`) を入力して、テキスト・バッファへの追加を停止します。

テキストの入力中に間違えてタイプした場合には、`Return` キーを押す前に訂正できます。バックスペース・キーを使用して、間違えて入力した文字を消します。その後、正しい文字をその位置に入力します。

B.2.3 テキストの表示 – p (プリント) サブコマンド

p (プリント) サブコマンドを使用して、編集バッファの内容を表示させることができます。

1 行だけ表示させるには、サブコマンド np を使用します。n には行番号を指定します。たとえば、次のように入力します。

```
2p
appending is to enter a
—
```

数行を表示させるには、n, mp サブコマンドを使用します。n には開始行番号、m には終了行番号を指定します。たとえば、次のように入力します。

```
1,3p
The only way to stop
appending is to enter a
line that contains only
—
```

特定の行からバッファの終わりまでをすべて表示させるには、n, \$p サブコマンドを使用します。n は開始行番号、\$ はバッファの最終行を意味します。次の例では、1, \$p でバッファ内の全行が表示されます。

```
1,$p
The only way to stop
appending is to enter a
line that contains only
a dot.
—
```

注意

この付録の多くの例では、1, \$p を使用してバッファの内容を表示しています。これらの例では、サブコマンド 1, \$p はオプションですが、便利です。このコマンドを使用すると、例中のサブコマンドが意図したとおりに動作するかどうかを確認することができます。他に、ed には、もう 1 つ、p という便利な用法があります。これは、1, \$p と同等のサブコマンドであり、バッファの内容を表示します。

B.2.4 テキストの保存 – w (書き込み) サブコマンド

w (書き込み) サブコマンドは、バッファの内容をファイルに書き込んだり、コピーしたりします。ファイルの全部または一部を、もとの名前、または新しい名前で保存することができます。いずれの場合にも、ed は指定されたファイルのもとの内容をバッファからコピーしたデータで書き換えます。

B.2.4.1 同じファイル名でのテキストの保存

もとのファイル名でバッファの内容を保存するには、w サブコマンドを入力します。

```
w
78
—
```

ed プログラムは、バッファの内容を afile という名前のファイルにコピーして、ファイルにコピーされた文字数 (78) を表示します。この数字には、空白や、Return (改行とも呼ばれる) などの画面上では見えない文字も含まれます。

w サブコマンドは、編集バッファの内容には影響を与えません。ファイルのコピーを保存した後、バッファの内容の編集作業を続けることができます。

格納されたファイルは、次に w を使用してバッファの内容をそのファイルにコピーするまで、変更されません。安全のため、ファイルの編集作業中には、定期的にファイルを保存することをおすすめします。こうしておくと、変更を行ったが保存したくない場合や間違いがあった場合に、以前に保存したファイルを使用して、そこからやり直すことができます。

注意

u (取り消し) サブコマンドは、ed サブコマンドで最後に修正した前の状態にバッファを戻します。u で取り消すことのできるサブコマンドは、a, c, d, g, G, i, j, m, r, s, t, v および V です。

B.2.4.2 別のファイル名でのテキストの保存

同じファイルのコピーが複数必要なことがよくあります。たとえば、手紙のオリジナル文を 2 つのファイルに入れておき、1 つは保管用、もう 1 つは変更用にすることができます。

これまでに説明した例を実行している場合には、オリジナルのテキストが入っている `afile` という名前のファイルがあります。このファイルのコピーをもう1つ作成するには(内容をバッファに入れたままで)、`w filename` 形式のサブコマンドを、次の例のように使用します。

この時点では、`afile` と `bfile` は、同じバッファの内容をコピーしたものであるため、内容は同じです。ただし、`afile` と `bfile` は別個のファイルなので、相互の内容に影響を与えずに一方の内容を変更することができます。

ファイルの一部を保存するには、`n,mw filename` 形式のサブコマンドを使用します。このサブコマンドでは、変数は次のように使用されます。

m 保存するファイルの部分の最終行番号 (または, 1 行だけ保存する場合はその行の番号) を指定する。

次の例では、`w` サブコマンドは、バッファの 1 行目と 2 行目を `cfile` という名前の新しいファイルにコピーします。

ed は、cfile に書き込まれた文字数 (44) を表示します。

注意

ed プログラムを終了すると、バッファの内容は失われます。バッファ内のデータを保存するには、ed プログラムを終了する前に、w サブコマンドを使用して、バッファの内容をファイルにコピーします。

q サブコマンドを入力すると、シェル・プロンプト (\$) に戻ります。

バッファを変更したが、内容のコピーをまだ保存していない場合、q サブコマンドは ? というエラー・メッセージを表示します。このときには、w サブコマンドでバッファの内容を保存するか、再び q を入力して、バッファの内容をコピーしないで ed プログラムを終了します。

B.3 ファイルの編集バッファへのロード

ファイルを編集する前に、そのファイルを編集バッファにロードしなければなりません。ファイルのロードは、ed プログラムを起動するとき、あるいは、プログラムの実行中に行うことができます。

ed プログラムの起動時にファイルをロードするには、次のコマンドを入力します。

```
ed filename
```

このコマンドにより、ed が起動し、filename というファイルが編集バッファにロードされます。

ed プログラムの実行中にファイルを編集バッファにロードするには、次のうちどちらかのコマンドを入力します。

- e filename

バッファの以前の内容をすべて消去してから、filename ファイルを編集バッファにロードします。

- nr filename

filename ファイルをバッファの n 行目の後に読み取ります。n を指定しない場合、ed はそのファイルを編集バッファの終りに追加します。

B.3.1 ed (編集) コマンド

ed プログラムの起動時にファイルを編集バッファにロードするには、ed コマンドの後にファイル名を入力します。次の ed コマンドは、ed プログラムを起動して、ファイル `afile` を編集バッファにロードします。

```
$ ed afile
78
—
```

ed プログラムは、編集バッファに読み取った文字数 (78) を表示します。

ed がファイルを見つけることができなかった場合、`?filename` というメッセージが表示されます。そのファイルを新規に作成する場合には、`a` (追加) サブコマンド (B.2.2 項参照) と、`w` (書き込み) サブコマンド (B.2.4 項参照) を使用します。

B.3.2 e (編集) サブコマンドの使用

ed プログラムを起動すると、`e` (編集) サブコマンドを使用してファイルをバッファにロードすることができます。`e` サブコマンドは、バッファの内容を新しいファイルで書き換えます。`e` サブコマンドを、B.3.3 項で説明する `r` サブコマンドと比較してみてください。`r` サブコマンドは、新しいファイルをバッファに追加します。

注意

新しいファイルをバッファにロードすると、バッファの以前の内容は新しいファイルで重ね書きされてしまいます。`w` サブコマンドを使用してバッファのコピーを保存してから、新しいファイルをバッファに読み取るようにしてください。

次の例では、サブコマンド `e cfile` は、`cfile` という名前のファイルを編集バッファに読み取り、`afile` を置換します。次に、`e afile` サブコマンドで `cfile` が消去され、`afile` が再びバッファに読み取られます。`e` サブコマンドを実行するたびに、ed プログラムはバッファに読み取った文字数を返します (44 と 78)。

```
e cfile
44
e afile
78
```

ed がファイルを見つけることができなかった場合、`? filename` というメッセージを表示します。そのファイルを新規に作成する場合には、`a` (追加) サブコマンド (B.2.2 項を参照) と、`w` (書き込み) サブコマンド (B.2.4 項を参照) を使用します。

ed プログラムを終了しなくても、複数のファイルを一度に 1 つずつ編集することができます。 `e` サブコマンドを使用して、ファイルをバッファにロードし、ファイルの修正が終了したら、`w` サブコマンドを使用して、修正したファイルのコピーを保存します。 `w` サブコマンドの詳細については、B.2.4 項を参照してください。次に、再び `e` サブコマンドを使用して、別のファイルをバッファにロードします。

B.3.3 `r` (読み取り) サブコマンドの使用

ed プログラムを起動すると、`r` (読み取り) サブコマンドを使用して、ファイルをバッファに読み取ることができます。 `r` サブコマンドはファイルの内容をバッファの内容に追加します。 `r` サブコマンドは、バッファの内容を消去しません。 `r` サブコマンドを、B.3.2 項で説明した `e` サブコマンドと比較してみてください。 `e` コマンドは、新しいファイルを読み取る前にバッファの内容を削除します。

`r` サブコマンドを使用して、ファイルをバッファ内の特定の場所に読み取ることができます。たとえば、`4r cfile` サブコマンドは、行番号 4 の後に `cfile` というファイルを読み取ります。ed プログラムは、その後バッファ内の行番号をつけ直します。行番号を指定しない場合には、`r` サブコマンドは、新しいファイルをバッファの内容の終りに追加します。

次の例では、`r` サブコマンドを行番号を指定して使用方法を示します。

```
1,$p
The only way to stop
appending is to enter a
line that contains only
a dot.
3r cfile
44
1,$p
The only way to stop
appending is to enter a
line that contains only
The only way to stop
appending is to enter a
```

```
a dot.
```

```
—
```

サブコマンド `l,$p` で `afile` の 4 行を表示します。次に、`3r cfile` サブコマンドが、`cfile` の内容をバッファ内の行番号 3 の後にロードして、バッファに 44 文字が読み取られたことを示します。次のサブコマンド `l,$p` がバッファの内容を再び表示して、`r` サブコマンドでバッファ内の行番号 3 の後に `cfile` が読み込まれたことが確認できます。

システム上で例題を実行している場合には、次に進む前に、以下の作業を行ってください。

1. バッファの内容を `cfile` ファイルに保存する。

```
w cfile
```

2. バッファに `afile` ファイルをロードする。

```
e afile
```

B.4 現在の行の表示と変更

`ed` プログラムは行エディタです。つまり、`ed` では、バッファの内容を一度に 1 行ずつ編集します。ある時点において作業できる行を現在の行といい、この行はドット (.) で示されます。ファイルの別の部分を編集するには、現在の行を変更しなければなりません。

現在の行を表示させるには、次のサブコマンドを入力します。

```
p
```

現在の行の行番号を表示するには、次のサブコマンドを入力します。

```
. =
```

注意

矢印キーを使用して、現在の行を変えることはできません。現在の行を変えるには、以降の項で説明する `ed` サブコマンドを使用します。

バッファ内で位置を変更するには、次のうちのいずれかを行います。詳細については、以降の項で説明します。

1. 現在の行を行番号 n に設定するには、次のサブコマンドを入力する。

n

2. 現在の行をバッファ内で 1 行ずつ前に進めていくには、Return キーを押す。

3. 現在の行をバッファの中で 1 行ずつ後に戻していくには、ダッシュ (-) を入力する。

4. 現在の行をバッファの中で n 行前に進めていくには、次のサブコマンドを入力する。

$.+n$

5. 現在の行をバッファの中で n 行後ろに戻していくには、次のサブコマンドを入力する。

$.-n$

B.4.1 バッファ内での位置付け

最初にファイルをバッファにロードする場合、ファイルの最後の行が現在の行になります。ファイルで作業を行っているときには、通常、現在の行を何回も変えます。現在の行およびその行番号はいつでも表示することができます。

現在の行を表示するには、 p を入力します。

```
p
a dot.
—
```

p サブコマンドは、現在の行 (a dot.) を表示します。afile を読み取ってから現在の行を変更していないので、現在の行はバッファ内の最後の行です。

現在の行の行番号を表示するには、 $.=$ を入力します。

```
. =
4
—
```

`afile` には 4 行あり、現在の行がバッファ内の最後の行であるため、`.=` サブコマンドは 4 を表示します。

また、`$` (バッファの最後の行を表わす記号) を `=` サブコマンドと一緒に使用して、バッファ内の最後の行を判断することもできます。

```
$=  
4
```

—

`$=` サブコマンドを使用すると、バッファ内の行数を簡単に知ることができます。ed の `$` 記号はシェル・プロンプト (`$`) とは何の関係もありません。

B.4.2 バッファ内での位置の変更

バッファ内の位置の変更 (現在の行の変更) は、次のいずれかの方法で行います。

- 行番号 (絶対位置) を指定する。
- 現在の行に相対して前または後に移動する。

現在の行を特定の行に移すには、行番号を入力します。ed は新しい行番号を表示します。次の例では、`afile` の最初の行が現在の行になります。

```
1  
The only way to stop  
—
```

Return キーを押すとバッファ内を 1 行ずつ前に進んでいき、新しい現在の行を次の例のように表示します。

```
appending is to enter a  
  
line that contains only  
  
a dot.  
  
?  
—
```

バッファの最後の行の次に移動しようとする、ed は、エラー・メッセージ `?` を表示します。バッファの最後の行の次には移動できません。

現在の行を、バッファの最後の行に設定するには、`$` を入力します。

現在の行を、バッファ内で 1 行ずつ後ろに戻すには、次に示すようにマイナス符号 (`-`) を次々と入力します。


```
-
line that contains only
-
appending is to enter a
-
The only way to stop
-
?
```

バッファの最初の行より上に移動しようとする、ed は、エラー・メッセージ? を返します。バッファの最初の行より先には移動できません。

現在の行を、バッファの中で一度に 2 行以上前に移動するには、`.n` を入力します。`n` は移動する行数を指定します。

```
.2
line that contains only
-
```

`.2` は `.+2` の簡略な形式です。

現在の行を、バッファの中で一度に 2 行以上後に移動するには、サブコマンド `.-n` を入力します。`n` は移動する行数を指定します。

```
.-2
The only way to stop
-
```

B.5 テキストの探索

特定の単語や文字列がある行の行番号がわからない場合、文脈探索を使用してその行を探すことができます。

文脈探索は、次のいずれかを行います。

- 順方向に探索する場合には、次のサブコマンドを入力する。

```
/string to find/
```

- 逆方向に探索する場合には、次のサブコマンドを入力する。

```
?string to find?
```

次の項で、テキストを探索する方法について詳しく説明します。

B.5.1 バッファ内での順方向探索

バッファ内を順方向に探索するには、文字列をスラッシュ (//) で囲んで入力します。

```
/only/  
line that contains only
```

—

文脈探索 (/only/) は、現在の行の次の行から始まり、文字列 "only" を含む次の行を探索して表示します。そして、その行が現在の行になります。

ed が該当する文字列を探索の先頭行からバッファの最終行までの間で見つけることができなかった場合は、探索を 1 行目から現在の行まで続けて行います。ed がバッファ全体を探索しても文字列が見つからない場合には、エラー・メッセージ ? を表示します。

```
/random/  
?
```

—

文字列を一度探索すると、// を入力することで同じ文字列をもう一度探索することができます。次の例では、only という文字列を一度探索した後、同じ文字列をもう一度探索します。

```
/only/  
The only way to stop  
//  
line that contains only
```

—

B.5.2 バッファ内での逆方向探索

バッファ内を逆方向に探索するのは、順方向に探索する場合と同様に行います。ただし、逆方向の探索では、文字列を疑問符 (?) で囲みます。

```
?appending?  
appending is to enter a
```

—

文脈探索は、現在の行の 1 行前から始まり、文字列 appending を含む最初の行を探します。そして、その行が現在の行になります。ed がバッファ全体を探索しても該当する文字列を見つけない場合は、現在の行で探索を中止して、エラー・メッセージ ? を表示します。

逆方向に文字列探索した後、?? を入力すると、同じ文字列をもう一度逆方向に探索することができます。これは、ed が探索する文字列を記憶しているためです。

B.5.3 探索方向の変更

スラッシュ (/) と疑問符 (?) の探索文字を交互に使用することにより、ある文字列の探索の方向を変えることができます。

```
/only/  
line that contains only  
??  
The only way to stop  
—
```

文字列を探索するときに行き過ぎてしまった場合などに、探索方向が変えられるので便利です。

B.6 置換 – s (置換) サブコマンド

s (置換) サブコマンドを使用すると、ある文字列 (1 つ以上の文字) を別の文字列で置き換えることができます。s サブコマンドは、一回に 1 行または複数行に対して処理を行います。このサブコマンドは、特に、タイプミスやスペルミスを訂正する場合に便利です。

置換を行うには、次のいずれかを行います。

- 現在の行で最初に見つかる *oldstring* (旧文字列) を、*newstring* (新文字列) に置き換えるには、次のサブコマンドを入力する。

```
s/ oldstring / newstring /
```

- 行番号 *n* で最初に見つかる *oldstring* を、*newstring* に置き換えるには、次のサブコマンドを入力する。

```
n s/ oldstring / newstring /
```

- 行番号 *n* から *m* までの各行で最初に見つかる *oldstring* を、*newstring* に置き換えるには、次のサブコマンドを入力する。

```
n,m s/ oldstring / newstring /
```

次の各項で、置換方法について詳しく説明します。

B.6.1 現在の行での置換

現在の行で置換を行う前に、まず変更したい行が現在の行であることを確認します。次の例では、`/appending/` (探索) サブコマンドで、変更する行を探します。次に、`s/appending/adding text/p` (置換) サブコマンドで、現在の行にある文字列 `appending` を `adding text` に置き換えます。その後、`p` (プリント) サブコマンドで、変更した行を表示します。

```
/appending/  
appending is to enter a  
s/appending/adding text/p  
adding text is to enter a  
—
```

注意

操作を簡単にするため、`p` (プリント) サブコマンドを `s` サブコマンドに追加することができます。たとえば、`s/appending/adding text/p` のように指定します。これにより、置換の結果を確認するために `p` サブコマンドを別にタイプする手間が省けます。

`s` サブコマンドは、該当する行で最初に見つけた文字列だけを置き換えます。1 行にあるすべての該当する文字列を置き換える方法については、B.6.4 項を参照してください。

B.6.2 特定の行での置換

特定の行で置換を行うには、次の形式のサブコマンドを使用します。

`n s/ oldstring / newstring /`

n は、置換を行う行の番号です。次の例では、`s` サブコマンドで行番号 1 に移動し、文字列 `"stop"` を `"quit"` に置き換えて、新しい行を表示します。

```
1s/stop/quit/p  
The only way to quit  
—
```

`s` サブコマンドは、指定行で最初に見つけた該当文字列だけを置き換えます。1 行にあるすべての該当する文字列を置き換える方法については、B.6.4 項を参照してください。

B.6.3 複数行での置換

複数行で置換を行うには、次の形式のサブコマンドを使用します。

n,m s/ oldstring / newstring /

n は置換を行う先頭行であり、*m* は最終行です。次の例では、*s* サブコマンドは、バッファ内の各行で最初に見つかった文字列 "to" を "TO" に置き換えます。

```
1,$s/to/TO/  
1,$p  
The only way TO quit  
adding text is TO enter a  
line that contains only  
a dot.  
—
```

1,\$p サブコマンドはバッファの内容を表示するので、置換が行われたことを確認できます。

B.6.4 すべての該当文字列の置換

通常、*s* (置換) サブコマンドは、指定行で最初に探索された該当文字列のみを置き換えます。ただし、*g* (グローバル) 演算子を指定すると、1 行または複数行にあるすべての該当文字列を置き換えることができます。

1 行についてグローバル置換を行うには、次の形式のサブコマンドを使用します。

n s/ oldstring / newstring /

次の例では、3*s/on/ON/gp* は、行番号 3 で探索されたすべての文字列 "on" を "ON" に置き換えて、新しい行を表示します。

```
3s/on/ON/gp  
line that cONtains ONly  
—
```

複数行についてグローバル置換を行うには、次の形式のサブコマンドで、行のグループを指定します。

n,m s/ oldstring / newstring /g

次の例では、1,\$*s/TO/to/g* は、バッファ内のすべての行で、文字列 "TO" を文字列 "to" に置き換えます。

```
1,$s/TO/to/g
1,$p
The only way to quit
adding text is to enter a
line that cONTains ONLY
a dot.
—
```

B.6.5 文字の削除

`s` サブコマンドを使用して、文字列を削除する（つまり、文字列を空列で置き換える）ことができます。文字を削除するには、`s/oldstring//` 形式でサブコマンドを指定します。このとき、最後の 2 つの `/` の間にはスペースを入れないでください。

次の例では、`ed` は文字列 "adding" を行番号 2 から削除し、変更した行を表示します。

```
2s/adding//p
text is to enter a
—
```

B.6.6 行頭と行末での置換

次の 2 つの特殊文字を使用すると、行頭あるいは行末で置換を行うことができます。

`^` (カレット) 行頭で置換を行う。

`$` (ドル記号) 行末で置換を行う。この場合には、ドル記号 (`$`) はバッファ内の最後の行を表していない。

行頭で置換を行うには、`s/newstring/` (新文字列) サブコマンドを使用します。次の例では、最初の `s` サブコマンドで、文字列 "Remember" を行番号 1 の先頭に追加します。2 番目の `s` サブコマンドは、文字列 "adding" を行番号 2 の先頭に追加します。

```
1s/^/Remember,/p
Remember, The only way to quit
2s/^/adding/p
adding text is to enter a
—
```

行末で置換を行うには、`s/$/newstring` (新文字列) 形式のサブコマンドを使用します。次の例では、`s` サブコマンドで、文字列 "Then press Enter." を行番号 4 の末尾に追加します。

```
4s/$/ Then press Enter./p
a dot. Then press Enter.
```

—

Then の前にスペースを 2 つ入れて、2 つの文を分けていることに注意してください。

B.6.7 文脈探索の使用

変更する行の行番号がわからない場合には、文脈探索を使用して探すことができます。文脈探索についての詳細は、B.5 節を参照してください。

操作を簡単にするため、次の形式のように、文脈探索と置換を組み合わせると一つのサブコマンドにすることができます。

```
/string to find/ s/ oldstring / newstring /
```

次の例では、`ed` は文字列 ", The" を含む行を探して、その文字列を ", the" に置き換えます。

```
/, The/s/, The/, the/p
Remember, the only way to quit
```

—

探索文字列を置換する文字列として使用することもできます。この場合は、`/string to find/s//newstring/` という形式のサブコマンドを使用します。次の例では、`ed` は、文字列 "cONtains ONLY" を含む行を探し、その文字列を "contains only" に置き換えて、変更された行を表示します。

```
/cONtains ONLY/s//contains only/p
line that contains only
```

—

B.7 行の削除 – `d` (削除) サブコマンド

`d` (削除) サブコマンドを使用して、バッファから 1 つ以上の行を削除できます。`d` サブコマンドの一般形式は、次のとおりです。

```
starting line,ending line d
```

行を削除した後、`ed` は現在の行を、削除された行の次の行に設定します。バッファの最終行を削除した場合には、バッファに残っている最後の行が

現在の行となります。削除後、ed はバッファに残っている行の番号を付けなおします。

バッファから行を削除するには、次のようにします。

- 現在の行を削除するには、次のサブコマンドを入力する。
`d`
- 行番号 n をバッファから削除するには、次のサブコマンドを入力する。
`nd`
- 行番号 n から m までをバッファから削除するには、次のサブコマンドを入力する。
`n,m d`

以降の各項で、行を削除する方法について詳しく説明します。

B.7.1 現在の行の削除

現在の行を削除する場合には、`d` を入力します。次の例では、`1,$p` サブコマンドはバッファの内容を全部表示し、`$` サブコマンドがバッファの最終行を現在の行にします。

```
1,$p
Remember, the only way to quit
adding is to enter a
line that contains only
a dot. Then press Enter.
$
a dot. Then press Enter
d
—
```

最後に、`d` サブコマンドが、現在の行 (この場合、バッファ内の最後の行) を削除します。

B.7.2 特定の行の削除

削除する行の行番号がわかっている場合は、`nd` 形式のサブコマンドを使用します。次の例では、`2d` サブコマンドでバッファから行番号 2 を削除します。

```
2d
1,$p
Remember, the only way to quit
line that contains only
—
```


1, \$p サブコマンドによりバッファの内容を表示して、その行が削除されたことを確認します。

B.7.3 複数行の削除

複数行をバッファから削除するには、 n, md 形式のサブコマンドを使用します。 n は削除する複数行の先頭行番号であり、 m は最終行番号です。

次の例では、1, 2d サブコマンドは行番号 1 および 2 を削除します。

```
1, 2d
1, $p
?
```

1, \$p サブコマンドはメッセージ ? を表示して、バッファが空であることを示します。

システムで実際に例題を実習している場合は、次の節に進む前にバッファの内容をリストアします。次の例で、バッファの内容をリストアする方法を示します。

```
e afile
?
e afile
78
```

この一連のコマンドは、元のファイル afile のコピーをバッファ内に読み込みます。

B.8 テキストの移動 – m (移動) サブコマンド

m (移動) サブコマンドを使用して、複数行をバッファの中のある位置から別の位置に移動することができます。移動後、移動された最後の行が現在の行になります。

テキストを移動するには、次の形式のサブコマンドを入力します。

$x, y \text{ m } z$

x は移動するグループの先頭行番号、 y は移動するグループの最終行番号、 z は移動先の直前の行番号です。

次の例では、1, 2m4 サブコマンドは、バッファの最初の 2 行を行番号 4 の直後に移動します。

```
1,2m4
1,$p
line that contains only
a dot.
The only way to stop
appending is to enter a
—
```

1,\$p サブコマンドは、バッファの内容を表示して、移動が完了したことを示します。

複数行をバッファの先頭に移動するには、移動先の行番号として、ゼロ (0) を指定します。次の例では、3,4m0 サブコマンドにより、行番号 3 と 4 をバッファの先頭に移動します。

```
3,4m0
1,$p
The only way to stop
appending is to enter a
line that contains only
a dot.
—
```

1,\$p サブコマンドは、バッファの内容を表示して、移動が行われたことを示します。

複数行をバッファの末尾に移動するには、移動先の行番号として、\$ を指定します。

```
1,2m$
1,$p
line that contains only
a dot.
The only way to stop
appending is to enter a
—
```

B.9 テキスト行の変更 – c (変更) サブコマンド

c (変更) サブコマンドを使用して、1 行または複数の行を、1 行または複数の新しい行と置き換えることができます。c サブコマンドは、まず置き換えたい行を削除した後、a (追加) サブコマンドと同様に新しい行を入力できるようにします。新しいテキストを入力し終わったら、次の行にドット (.) だけを入力します。

c サブコマンドの一般形式は、次のとおりです。

starting line, ending line **c**

テキスト行を変更するには、次のようにします。

1. 次の形式のサブコマンドを入力する。

n, m **c**

n は削除するグループの先頭行番号、*m* は削除するグループ (または 1 行だけ) の最終行番号を指定します。

2. 新しい行をタイプして、各行の終わりでは Return キーを押す。
3. 最後の行にドット (.) だけを入力する。

次の各項で、テキストを変更する方法について詳しく説明します。

B.9.1 1 行のみの変更

テキストを 1 行だけ変更するには、**c** (変更) サブコマンドに行番号を 1 つだけ指定します。この 1 行を必要な行数で置き換えることができます。

次の例では、**2c** サブコマンドで行番号 2 をバッファから削除した後、新しいテキストを入力することができます。

```
2c
appending new material is to
use the proper keys to create a
.
1,$p
The only way to stop
appending new material is to
use the proper keys to create a
line that contains only
a dot.
```

行にドット (.) だけを入力すると、**ed** によるバッファへのテキストの追加が終了します。**1,\$p** サブコマンドは、バッファの内容全体を表示して、変更が行われたことを示します。

B.9.2 複数行の変更

テキスト行を 2 行以上変更するには、**c** サブコマンドで変更したい範囲の先頭行番および最終行番号を指定します。この複数行は 1 行以上で置き換えることができます。

次の例では、`2,3c` サブコマンドで行番号 2 と 3 をバッファから削除した後、新しいテキストを入力しています。

```
2,3c
adding text is to enter a
.
1,$p
The only way to stop
adding text is to enter a
line that contains only
a dot.
```

—

行にドット (.) だけを入力すると、ed によるバッファへのテキストの追加が終了します。`1,$p` サブコマンドは、バッファの内容全体を表示して、変更が行われたことを示します。

B.10 テキストの挿入 – `i` (挿入) サブコマンド

`i` (挿入) サブコマンドを使用して、バッファに 1 行以上の新しい行を挿入できます。新しい行を挿するバッファの位置を決めるには、行番号を指定するか、または文脈探索を使用します。`i` サブコマンドは、指定された行の直前に新しい行を挿入します。`i` サブコマンドを、B.2.2 項で説明している `a` サブコマンドと比較してください。`a` サブコマンドは、新しい行を指定行の後に挿入します。テキストを挿入するには、次のようにします。

1. 次のいずれかのサブコマンドを入力する。

```
ni
```

`n` は、新しい行が直前に挿入される行の行番号です。

```
/string/i
```

`string` は、新しい行が直前に挿入される行に含まれている文字列を指定します。

2. 新しい行を入力する。
3. 最後の行の先頭にドット (.) を入力する。

次の各項で、テキストを挿入する方法について詳しく説明します。

B.10.1 行番号の使用

新しい行を挿入する先の行番号がわかっている場合は、挿入サブコマンドを `ni` 形式で使用できます。`n` には行番号を指定します。入力した新しい行

はバッファ内の行番号 n の前に挿入されます。 `i` サブコマンドを終了するには、最後にドット (.) だけの行を入力します。

次の例では、まず、`1,$p` サブコマンドでバッファの内容を表示します。つぎに、`4i` サブコマンドが行番号4の直前に新しい行を挿入します。

```
1,$p
The only way to stop
adding text is to enter a
line that contains only
a dot.
4i
--repeat, only--
.
1,$p
The only way to stop
adding text is to enter a
line that contains only
--repeat, only--
a dot.
-
```

`4i` を入力した後に、新しいテキスト行を入力し、次の行にドットだけを入力して、`i` サブコマンドを終了します。2 回目の `1,$p` サブコマンドでバッファの内容が表示され、新しいテキストが挿入されたことを示します。

B.10.2 文脈探索の使用

`i` サブコマンドで新しい行を挿入する位置を指定するもうひとつの方法に文脈探索があります。 `/string/i` 形式のサブコマンドを使用して `string` を含む行を探し、その行の直前に新しい行を挿入します。新しい行の挿入が終了したら、行の先頭でドットを入力します。

次の例では、`/dot/i` サブコマンドにより、文字列 "dot" を含んでいる行の直前に新しいテキストを挿入します。

```
/dot/i
and in the first position--
.
1,$p
The only way to stop
adding text is to enter a
line that contains only
--repeat, only--
and in the first position--
a dot.
-
```

1,\$p サブコマンドでバッファの内容をすべて表示し、新しくテキストが挿入されたことを示します。

B.11 行のコピー – t (転送) サブコマンド

t (転送) サブコマンドを使用して、バッファ内のある位置にある行を、他の位置にコピーすることができます。t サブコマンドは、元の行に影響を与えません。

t サブコマンドの一般形式は、次のとおりです。

starting line,ending line t line to follow

行をコピーするには、次の形式のサブコマンドを入力します。

n,m tx

n はコピーする範囲の先頭行番号、*m* はコピーする範囲の最終行番号、*x* はコピー先の直前の行番号です。

行をバッファの先頭にコピーするには、コピー先の直前の行番号として、ゼロ (0) を指定します。行をバッファの最後にコピーするには、コピー先の直前の行番号として、ドル記号 (\$) を指定します。

次の例では、1,3t4 サブコマンドで行番号 1 から 3 までをコピーし、その内容を行番号 4 の後に挿入します。

```
1,3t4
1,$p
The only way to stop
adding text is to enter a
line that contains only
--repeat, only--
The only way to stop
adding text is to enter a
line that contains only
and in the first position--
a dot.
—
```

1,\$p サブコマンドはバッファの内容をすべて表示して、ed により行のコピーが挿入され、もとの行はそのまま残っていることを示します。

B.12 ed からのシステム・コマンドの使用

ed プログラムを終了せずにシステム・コマンドを使用する方が便利なことがあります。その場合には、感嘆符 (!) を使用すると、ed プログラムから一時的に出ることができます。

ed からシステム・コマンドを使用するには、次のように入力します。

```
!command
```

次の例では、!ls コマンドで一時的に ed プログラムを停止して、ls (リスト) システム・コマンド (現在のディレクトリ内のファイルをリストするコマンド) を実行します。

```
!ls
afile
bfile
cfile
!
```

ls コマンドは現在のディレクトリにあるファイルの名前 (afile, bfile, cfile) を表示した後、! を表示します。ls コマンドが実行され、続けて ed を使用できます。

ed プログラム内からどのシステム・コマンドでも使用できます。別の ed プログラムを実行して、ファイルを編集した後、元の ed プログラムに戻ることもできます。2 番目の ed プログラムから、3 番目の ed プログラムを実行したり、システム・コマンドを使用したりすることもできます。

B.13 ed プログラムの終了

これで ed プログラムの紹介を終わります。ファイルを保存して、ed プログラムを終了するには、次の手順に従ってください。

1. w コマンドを次のように入力する。

```
w
```

2. q コマンドを次のように入力する。

```
q
```

w サブコマンドおよび q サブコマンドの詳細については、それぞれ、B.2.4 項および B.2.5 項を参照してください。

他の ed の機能については、ed(1) リファレンス・ページを参照してください。

ed で作成したファイルの印刷方法の詳細については、第 3 章を参照してください。

C

国際化機能の使用

この付録では、Tru64 UNIX オペレーティング・システムの国際化機能について説明します。これらの機能により、ユーザは、それぞれの言語、習慣、地域 (ロケール) に応じた方法で、データを処理したり、システムと対話することができます。

この付録を読み終えると、次のことができるようになります。

- ロケールの概念の理解 (C.1 節)
- ロケールにより影響を受ける関数の理解 (C.2 節)
- ロケールが設定されているかどうかの判断 (必要な場合) (C.3 節)
- ロケールの設定 (必要な場合) (C.4 節)
- ロケールあるいはロケールのアスペクトの変更 (必要な場合)

ユーザ・サイトが米国内にあり、アメリカ英語とその表記規約を使用する場合は、システムの省略時の設定がアメリカ英語であるため、ロケールを設定する必要はありません。

ユーザ・サイトが米国外の場合は、多くの場合、システム管理者がロケールを設定しています。ロケールがすでに設定されている場合には、この付録にざっと目を通して国際化機能の概要について学ぶだけでかまいません。しかし、ロケールが設定されていない場合には、この付録の情報は非常に重要です。必ず目を通してください。

C.1 ロケール

Tru64 UNIX は国際化されたオペレーティング・システムなので、さまざまな方法で情報を表示することができます。ロケールを指定することにより、言語、国、文化習慣に適した方法で情報を処理したり表示するように、オペレーティング・システムに指示することができます。ロケールの指定方法については、C.4 節を参照してください。

ロケールは一般に、言語、テリトリ、コード・セットという 3 つの要素から構成されます。3 つとも、情報の処理および表示方法を指定する上で重要な要素です。

- 言語 使用言語を指定する (例: ドイツ語、フランス語、英語)。
- テリトリ 地域を指定する (例: ドイツ、フランス、英国)。
- コード・セット ロケールに使用するコード化文字セットを指定する (例: ISO 8859/1, ISO Latin-1 コード・セット)。

ここで、コード・セットについて簡単に説明します。

UNIX システムでは従来から、ASCII コード・セットを使用してアメリカ英語を表記しています。アルファベット (A から Z, a から z) の各英字、数字、制御文字、および記号を、標準バイト 8 ビット中の 7 ビットによって一意に表現します。しかしながら、新しいコード・セットの追加、または従来のコード・セットの拡張により、英語以外の文字のサポートが必要になりました。大半のプログラムは ASCII に依存しているため、一般に使用されるコード・セットはすべて ASCII から始まり、そこから作成されていきます。

標準バイトの 8 ビットすべてを使用すると、1 つのコード・セットで多数のアルファベット使用言語をサポートすることができます。最も普及しているコード・セットは、ISO 8859 と呼ばれる一連のコード・セットです。最初のコード・セットは ISO 8859/1、2 番目は ISO 8859/2 と続き、ISO 8859/10 まであります。ISO 8859/1 コード・セットは、一般に Latin-1 (ラテン 1) と呼ばれ、英語およびその他の西ヨーロッパの言語をサポートしています。

中国語や日本語などのアジアの言語の表意文字をすべてサポートするには、文字のコード化に 2 バイト以上を必要とします。複数バイトの文字コード化を使用する数多くのコード・セットがアジアの言語に対して開発されていますが、複数バイトの文字コード化は一連の ISO 8859 コード・セットではサポートされていません。

Unicode および ISO/IEC 10646 標準では、アジア圏の言語を含めすべての言語に関して同じルールで文字を処理することができるユニバーサル文字セット (UCS) を定義しています。32 ビット値をベースにした UCS-4 エンコーディング・フォーマットは、ほとんどすべての言語をサポートし、UNIX システムに加えて Windows NT でも使用されているため、各ロケール用のコード・セットとして注目されまじめています。

C.2 ロケールがデータの処理と表示に与える影響

これまで説明したように、情報がどのように処理されて表示されるかは、システムで指定されているロケールによって決まります。ロケールは特に次の点に関してソフトウェアの動作に影響を与えます。

- データの照合 (ソート)
- 日付/時刻の表示フォーマット
- 通貨/数字の表記フォーマット
- 表示メッセージ
- yes/no 応答プロンプト

次の各項で、上記のそれぞれに関して詳しく説明します。

C.2.1 照合

照合とは、一連の要素を特定の順序に整列させる処理です。照合は必ず、各種のルールに従います。英語では使用されない照合のルールを必要とする言語もいくつかあります。

- マルチ・レベル

言語によっては、同じ 1 次位置にソートされる文字グループを持つものがあります。この場合には、追加のルールを適用して、この同じグループ内の文字の順序を決定します。たとえば、フランス語では、a, á, à, â はすべて、同じ 1 次位置にソートされます。これらの文字で始まる単語は同じ位置で照合され、その後、グループ内でソートされます。次の例では、フランス語における正しい順序で単語をソートしています。

```
a
á
abord
âpre
après
âpreté
azur
```

- 1 対 2 の文字マップ

言語の中には、特定の単一文字が 2 文字として処理されるものがあります。たとえば、ドイツ語の ß は ss としてソートされます。

- 多対 1 の文字マップ

言語によっては、1 つの文字列が単一の要素として処理されます。たとえば、スペイン語では、ch と ll のシーケンスはアルファベットの中で独特の要素として処理されます。次の例では、伝統的なスペイン語の正しい順序で単語をソートしています。

```
canto
construir
curioso
chapa
chocolate
dama
```

- 無視される文字

照合ルールによっては、特定の文字を無視します。たとえば、ハイフン (-) を無視する文字として定義している場合、文字列 re-locate と relocate は同じ位置にソートされます。

注意

(A から Z , a から z) の範囲にアルファベットのすべての文字が含まれるとは限りません。たとえば、デンマーク語ではアルファベットの z の後にソートされる文字が 3 文字あります。

C.2.2 日付/時刻フォーマット

世界中のそれぞれの国では、それぞれ異なるフォーマット規約によって日付と時刻を表現します。日付と月を指定する場合、アメリカでは一般に次のフォーマットで表現します。

Tuesday, May 22, 1996

一方、フランスでは次のようなフォーマットを使用します。

mardi, 22 mai 1996

次の例は、1996 年 3 月 20 日を表すために使用するフォーマットです。各フォーマットは必ずしもその国で唯一の表記方法というわけではありません。

- 3/20/96 (アメリカ)
- 20/3/96 (イギリス)

C-4 国際化機能の使用

- 20.3.96 (フランスおよびドイツ)
- 20-III-96 (イタリア)
- 96/3/20 (日本)
- 8/3/20 (日本, 年号を使用)

日本の年号を使用したフォーマットでは, 年 (上の例では 8) は現在の天皇が即位してからの年数です。

日付と同じように, 時刻を表現する規約も数多くあります。アメリカでは a.m. と p.m. を指定して 12 時制で表しますが, 他の多くの国では, 24 時制で表します。

12 時制/24 時制表示の違いだけでなく, 句読点の使い方も異なります。たとえば, 次のような表記方法があります。

- 3:20 p.m. (アメリカ)
- 15h20 (フランス)
- 15.20 (ドイツ)
- 15:20 (日本)

利用可能な日付/時刻フォーマットとその使用方法については, `date(1)` リファレンス・ページを参照してください。

C.2.3 数字と通貨のフォーマット

数字や通貨の値をフォーマットするために使用する文字は, 国によって異なります。アメリカでは, 小数点 (整数と小数を区切る文字) としてピリオド (.) を使用し, 千の位を表す区切り文字としてコンマ (,) を使います。ヨーロッパのほとんどの国々では, この逆です。次に, 数字のフォーマットの例を示します。

- 1,234.56 (アメリカ)
- 1.234,56 (フランス)

次に, 通貨フォーマットの例を示します。

- \$1,234.56 (アメリカ, ドル)
- kr1.234,56 (ノルウェイ, クローネ)
- SFrs.1,234.56 (スイス, スイス・フラン)

また、通貨単位では 3 桁以上の小数桁が必要なこともあります。

C.2.4 メッセージ

プログラムには英語で書かれたメッセージが埋め込まれていることがあります。国際化されたプログラムでは、メッセージはプログラムとは別のファイルに収められ、プログラム内でメッセージ・システムへの呼び出しが行われてメッセージが表示されます。別のファイルに収められたメッセージを翻訳することにより、プログラムに翻訳メッセージを表示させることができます。翻訳済みのメッセージが利用できる場合には、ユーザはそれぞれの国の言語でシステムと対話することができます。

C.2.5 Yes/No プロンプト

多くのプログラムは、肯定または否定で応答する質問を行います。このようなプログラムでは通常、英語文字列の `y` または `yes`、`n` または `no` を入力します。国際化されたプログラムでは、それぞれの言語に対応した文字や単語を入力することができます。たとえば、フランス語では、`o` または `oui` が入力できるようにします。

C.3 ロケールが設定されているかどうかの判断

システムが自国の言語や規約に従って動作している場合は、ロケールが正しく設定されていると考えることができます。

ロケールが正しく設定されているかどうかわからない場合には、次のように `locale` コマンドを入力して、ロケール環境変数の現在の設定を表示させます。

```
% locale
LANG=fr_FR.ISO8859-1
LC_COLLATE="fr_FR.ISO8859-1"
LC_CTYPE="fr_FR.ISO8859-1"
LC_MONETARY="fr_FR.ISO8859-1"
LC_NUMERIC="fr_FR.ISO8859-1"
LC_TIME="fr_FR.ISO8859-1"
LC_MESSAGES="fr_FR.ISO8859-1"
LC_ALL=
```

ロケール環境変数 (C.4.1 項を参照) は、メッセージ、照合、コード・セット、数字フォーマット、通貨フォーマット、日付/時刻フォーマット、yes/no プロンプトなどに使用するロケール名を定義します。

```
LANG
LC_COLLATE
LC_CTYPE
LC_NUMERIC
LC_MONETARY
LC_TIME
LC_MESSAGES
LC_ALL
```

LANG 変数だけにロケールが設定されている場合は、LANG の設定が、LC_COLLATE、LC_CTYPE、LC_NUMERIC、LC_MONETARY、LC_TIME および LC_MESSAGES の各ロケール・カテゴリ変数に省略時の値として設定されます。この場合、ロケール・カテゴリ変数に LANG の値とは別の値を設定することも可能です。LC_ALL 変数が設定されている場合は、その他のすべてのロケール変数の設定が無視されます。

C.4 ロケールの設定

ロケールを指定する場合には、言語、テリトリ、およびコード・セットを示すロケール名を指定します。Tru64 UNIX オペレーティング・システムでは、ロケール名は次のフォーマットに従います。

lang_terr.codeset

各変数は、次のとおりです。

lang

言語名を 2 文字の小文字に省略したもの。省略形は、『ISO 639 Code for the Representation of Names of Languages』に指定されている。たとえば、en (English)、fr (French)、de (ドイツ語の *Deutsch*)、ja (Japanese)。

terr

テリトリ名を 2 文字の大文字で省略したもの。省略形は『ISO 3116 Code for the Representation of Names of Countries』に指定されている。たとえば、US (United States)、NL (the Netherlands)、FR (France)、DE (ドイツ語の *Deutschland*)、JP (Japan)。

`codeset`

コード・セットを識別する文字列。たとえば、ISO8859-1 (ISO 8859/1)、SJIS (Shift Japanese Industrial Standard)、AJEC (Advanced Japanese EUC)。

完全なロケール名は、`en_US.ISO8859-1` (英語、アメリカ向けの統合規約)、`fr_FR.ISO8859-1` (フランス語、フランス向けの統合規約)、`de_DE.ISO8859-1` (ドイツ語、ドイツ向けの統合規約) などのようになります。

ロケールは、各ユーザまたはシステム管理者のどちらでも設定することができます。システム管理者がサイトのロケールを設定した場合には、そのシステムだけでなく、サイト内の全システムに対して省略時のロケールがすでに指定されていると考えられます。必要であれば、ユーザは省略時の指定を変更することができます。

ロケールを設定するには、1 つまたは複数の環境変数にロケール名を割り当てます。LANG 環境変数は、ロケールのすべての要素 (コード・セット、照合順序、数字、通貨、および日付/時刻フォーマット、メッセージなど) をカバーするので、この変数にロケール名を割り当てるのが最も簡単な設定方法です。

表 C-1 は、単一バイトのヨーロッパ向けロケールのサブセットをインストールした場合に使用できるロケールの一覧です。他の言語用のソフトウェア・サブセットがインストールされている場合には、利用できるロケールが異なることがあります。使用できるロケールについては、`l10n_intro(5)` リファレンス・ページを参照してください。

表 C-1: ロケール名

言語	国	コード・セット	ロケール名
–	–	ASCII	C
–	–	ASCII	POSIX
デンマーク語	デンマーク	Latin-1	da_DK.ISO8859-1
ドイツ語	スイス	Latin-1	de_CH.ISO8859-1
ドイツ語	ドイツ	Latin-1	de_DE.ISO8859-1
ギリシャ語	ギリシャ	Latin-7	el_GR.ISO8859-7
英語	イギリス	Latin-1	en_GB.ISO8859-1

表 C-1: ロケール名 (続き)

言語	国	コード・セット	ロケール名
英語	アメリカ	Latin-1	en_US.ISO8859-1
スペイン語	スペイン	Latin-1	es_ES.ISO8859-1
フィンランド語	フィンランド	Latin-1	fi_FI.ISO8859-1
フランス語	ベルギー	Latin-1	fr_BE.ISO8859-1
フランス語	カナダ	Latin-1	fr_CA.ISO8859-1
フランス語	スイス	Latin-1	fr_CH.ISO8859-1
フランス語	フランス	Latin-1	fr_FR.ISO8859-1
イタリア語	イタリア	Latin-1	it_IT.ISO8859-1
オランダ語	ベルギー	Latin-1	nl_BE.ISO8859-1
オランダ語	オランダ	Latin-1	nl_NL.ISO8859-1
ノルウェー語	ノルウェー	Latin-1	no_NO.ISO8859-1
ポルトガル語	ポルトガル	Latin-1	pt_PT.ISO8859-1
スウェーデン語	スウェーデン	Latin-1	sv_SE.ISO8859-1
トルコ語	トルコ	Latin-9	tr_TR.ISO8859-9

C ロケールは、システムにロケールが設定されていない場合の省略時の設定です。POSIX ロケールは C ロケールと同等であり、POSIX ロケールと C ロケールに対して指定されている ASCII コード・セットには、英語のアルファベットの文字だけが含まれます。

C.4.1 ロケール・カテゴリ

表 C-2 に、ロケール関数に影響を及ぼす環境変数を示しています。

表 C-2: ロケール関数に影響を及ぼす環境変数

環境変数	説明
LC_COLLATE	文字列のソート、およびパターン内の文字範囲の決定に使用する照合順序を指定する。
LC_CTYPE	文字分類 (コード・セット) 情報を指定する。
LC_MONETARY	通貨フォーマットを指定する。

表 C-2: ロケール関数に影響を及ぼす環境変数 (続き)

環境変数	説明
LC_NUMERIC	数字フォーマットを指定する。
LC_MESSAGES	翻訳版が利用できる場合に、システム・メッセージを表示する言語を指定する。さらに、この変数は肯定/否定プロンプトに使用する文字列も指定する。
LC_TIME	日付/時刻フォーマットを指定する。
LC_ALL	前述のすべての変数および LANG 環境変数の指定を変更する。

LANG 変数と同様、表 C-2 に示しているすべての環境変数にはロケール名を代入することができます。たとえば、会社は米国にあるが、従業員の多くはスペイン語を使用しているとします。この場合には、LANG 環境変数をスペイン語のロケール名に設定し、LC_NUMERIC および LC_MONETARY 変数をアメリカ英語のロケール名に設定することができます。明示的に LC_NUMERIC および LC_MONETARY 変数を設定すると、LANG によって暗黙に設定される指定を変更します。ただし、LC_CTYPE、LC_MESSAGES、LC_TIME、LC_COLLATE 変数は、スペイン語ロケールに設定されたままです。C シェルの場合、上記のように設定するには、次の変数を割り当ててください。

```
setenv LANG es_ES.ISO8859-1
setenv LC_NUMERIC en_US.ISO8859-1
setenv LC_MONETARY en_US.ISO8859-1
```

Bourne、Korn、および POSIX シェルの場合には、次のように割り当てます。

```
LANG=es_ES.ISO8859-1
export LANG
LC_NUMERIC=en_US.ISO8859-1
export LC_NUMERIC
LC_MONETARY=en_US.ISO8859-1
export LC_MONETARY
```

特定の言語やソフトウェア・アプリケーションの要求を満たすために、同じロケールの異なるバージョンがローカルに用意されていることがあります。このようなロケール名には後ろにアットマーク (@) と修飾子フィールドが付いています。たとえば、電話帳で使用する照合順序が辞書で使用する照合順序と異なる言語があります。ある言語の標準的なロケールが辞書の照合順序を定義している場合には、そのロケールにもう 1 つのバージョンを用意し

て、電話帳の照合順序をサポートします。この場合、ロケールの代替バージョンの名前は、`en_RF.ISO8859-1@phone` のようになります。

C.4.2 ロケール設定の制約

ロケールを設定すれば環境を調整することができますが、間違って設定してしまうことがないとはいえません。以降の項で、ロケール変数を定義する場合に起こる可能性がある問題点について説明します。

C.4.2.1 ロケール設定が有効でない

ロケールの種々のアスペクトに対して、ありそうもないロケール名の組み合わせを定義してしまうことがあります。これを防止する方法はありません。たとえば、`LANG` 環境変数をフランス語のロケールに設定し、`LC_CTYPE` 変数をノルウェー語のロケールに設定したとします。おそらく望ましくない結果が生じてしまうでしょう。フランス語に訳されたメッセージには、ノルウェー語で指定されていない文字が含まれています。`LANG` の他にロケール変数を定義した場合には、ユーザの責任においてロケール設定の組み合わせが有効であることを確認しなければなりません。

C.4.2.2 ファイル・データがロケールに対応していない

ファイルの作成時に設定されていたロケールをシステムが識別する方法はありません。したがって、システムでは、異なるロケールを使用したファイルのデータ処理を防ぐことはできません。たとえば、`LANG` 変数がドイツ語のロケールに設定されていたときに作成されたファイルを自分のシステムにコピーしたとします。自分のシステムで `LANG` がフランス語のロケールに設定されている場合に、そのファイルに対し `grep` コマンドを使用して文字列を検索しようとする、`grep` コマンドはフランス語の照合およびパターン照合規則をドイツ語のデータに適用することになります。ファイルに入っているデータの言語を調べ、それに合わせてロケールを設定するのはユーザの責任です。

C.4.2.3 `LC_ALL` の設定は他のロケール変数を上書きする

`LC_ALL` 変数に指定した値は、その後に `LC_COLLATE` など、そのカテゴリだけに作用する変数を設定しても、すべてのロケール環境変数の値として使用されます。`LC_ALL` の影響を取り消す唯一の方法は、`unsetenv LC_ALL` コマンドを入力することによって、この変数の定義を解除することです。

LC_ALL 変数は System V 環境に慣れているユーザのための変数です。
System V 環境では LC_ALL を設定するかもしれない。個々のすべてのロケール・
カテゴリ変数を設定して、ロケールの設定を行います。

D

mailx セッションのカスタマイズ

表 D-1 に記載している設定を `.mailrc` ファイルに記述しておくことで、`mailx` セッションを永久的にカスタマイズすることができます。一時的な設定については、付録 F の `unset` コマンドを参照してください。

表 D-1: `mailx` セッションをカスタマイズする変数

変数	型	説明
<code>allnet</code>	バイナリ	ネットワーク内でログイン名が一致するものを、すべて同じものとして扱う。
<code>append</code>	バイナリ	メッセージを受信した順序で <code>mbox</code> ファイルに保存する。つまり、最初に受信したメッセージがそのファイルの最初のメッセージになる。この変数を設定しなければ、メッセージは逆順で保存され、ファイルの最初のメッセージが最新のものになる。この変数を設定すると、 <code>mailx</code> プログラムの実行速度が向上する。
<code>ask</code>	バイナリ	メッセージを送信する際に表題の入力を求めるプロンプトを表示する。空白行を入力すると、表題を付けずにメッセージが送信される。
<code>askcc</code>	バイナリ	送信するメッセージのコピーの受信者の入力を求めるプロンプトを表示する。
<code>autoprint</code>	バイナリ	現在のメッセージを削除すると、次のメッセージが自動的に表示される。この変数を設定しなければ、メッセージを削除しても <code>mailx</code> は次のメッセージを表示しない。いずれの場合にも、次のメッセージが新たに現在のメッセージになる。
<code>bang</code>	文字列	<code>vi</code> のように、感嘆符 (!) の特殊処理がエスケープ・コマンド行でできる。
<code>cmd</code>	文字列	縦線つまりパイプ () コマンドを使用する際に実行する省略時のコマンドを指定できる。

表 D-1: mailx セッションをカスタマイズする変数 (続き)

変数	型	説明
conv	文字列	sendmail 用に UUCP 形式のアドレスを変換する方法を指定できる。
crt	数値	ビデオ・ディスプレイ (CRT) 端末に対して使用する。more プログラムを使用して、メールを一画面ずつ表示する。指定する値により、ページャを起動する前に表示するメッセージの行数を mailx に通知する。たとえば、次のように設定する。 set crt=20
DEAD	文字列	dead.letter に対して別のディレクトリを指定できる。省略時の設定では、デッド・レターは \$HOME/dead.letter に書き込まれる。
debug	バイナリ	デバッグ情報を表示する。
dot	バイナリ	ピリオドだけが入力された行をメッセージの終了として解釈する。unset dot と set ignoreeof は同時には指定しない。
EDITOR	文字列	edit コマンドまたは ~e エスケープを使用する際に使用するテキスト・エディタのパス名を指定する。たとえば、次のとおり。 set EDITOR=/usr/ucb/ex CRT 端末である場合は、この変数にスクリーン・エディタを指定することができる。この表の後で説明する VISUAL 変数を参照。
escape	文字列	エスケープ文字 (メッセージの書き込み中にエスケープ・コマンドを実行する文字) を指定できる。省略時の設定はチルダ (~)。単一の文字を指定する。
excode	文字列	発信メール・メッセージの文字変換を行う際に使用するロケールを指定できる。

D-2 mailx セッションのカスタマイズ

表 D-1: mailx セッションをカスタマイズする変数 (続き)

変数	型	説明
folder	文字列	メール・フォルダを格納するためのディレクトリを指定する。 /usr/users/hale のようにスラッシュで始まる名前は、絶対パス名である。スラッシュで始まっていない名前は、ホーム・ディレクトリに対する相対パス名である。たとえば、コマンド <code>set folder=folder</code> は、 <code>/usr/users/hale/folder</code> ディレクトリを示す。
gonext	バイナリ	これが設定されている場合は、 <code>Return</code> だけを押すと、次のメール・メッセージが表示される。設定されていない場合は、現在のメッセージが再表示される。
header	バイナリ	mailx の起動時に、メッセージのヘッダを出力する。
hold	バイナリ	読み終わったメッセージを、mbox ではなく、ユーザのシステム・メールボックスに保存する。
ignore	バイナリ	Ctrl/C の割り込みを無視して、アットマーク (@) としてエコーする。この変数は、付録 F で説明する ignore コマンドとは別のものである。
ignoreeof	バイナリ	発信メッセージの終了を示す Ctrl/D を無視する。set ignoreeof と unset dot は同時には設定しない。
indentprefix	文字列	~m コマンドを使用して、取り込んだメール・メッセージのテキストの各行頭に挿入する文字列を指定できる。
keep	バイナリ	ユーザのシステム・メールボックスが空の場合に、mailx がそれを削除するのではなく、切り詰めるようにする。これは、セキュリティの観点から、システム・メールボックスに特別な許可を設定している場合に便利である。この変数が設定されていなければ、システム・メールボックスは空になると削除される。したがって、新たにシステム・メールボックスを作成するときは、必要な許可を再設定しなければならない。

表 D-1: mailx セッションをカスタマイズする変数 (続き)

変数	型	説明
keepsave	バイナリ	メールを終了しても、保存されたメッセージが削除されないようにする。通常、メッセージを他のファイルまたはフォルダに保存すると、mailx プログラムはメッセージにマークを付け、ユーザが mailx を終了すると、システム・メールボックスからそのメッセージを削除する。この変数を設定すると、mailx はこれらのメッセージをシステム・メールボックスに残しておく。
lang	文字列	メール・メッセージの表示に使用するロケールを指定できる。
LISTER	文字列	folders コマンドによって実行されるコマンドを指定できる。
MBOX	文字列	mbox フォルダのディレクトリを指定できる。通常、mbox フォルダは \$HOME/mbox にある。
metoo	バイナリ	ユーザ自身がメンバである別名にメールを送信する際、受信者リストにユーザ自身も含める。この変数を設定しなければ、ユーザ自身がメンバである別名に送信したメッセージのコピーは受信しない。
noheader	バイナリ	mailx の起動時に、ヘッダおよびバージョン識別子を表示しない。
nosave	バイナリ	mailx が、打ち切られたメッセージを dead.letter としてホーム・ディレクトリに保存しないようにする。
onehop	バイナリ	他にも受信者がいるメッセージに回答する場合、その受信者のアドレスは発信人のアドレスに関連していることがある。この変数を指定すると、強制的に、受信時のパスを通らず、メッセージが直接配信されるため、効率が向上する。
outfolder	バイナリ	mailx が、folder で指定したディレクトリに発信メール・メッセージを保存するようにする。
page	バイナリ	パイプ () コマンドによって処理されるメッセージの間にフォーム・フィードを挿入する。

表 D-1: mailx セッションをカスタマイズする変数 (続き)

変数	型	説明
PAGER	文字列	<p>メッセージを表示する際に使用されるページング・プログラムを指定できる。たとえば、次のように指定する。</p> <pre>PAGER=/usr/bin/more</pre> <p>または</p> <pre>PAGER=/usr/bin/pg</pre>
prompt	文字列	mailx の起動時に mailx プロンプトを変更できる。たとえば、次のとおり。 prompt=>>>
quiet	バイナリ	最初に起動したときのバージョンと、type コマンドを使用するときのメッセージ番号を表示しない。
record	文字列	mailx がすべての発信メッセージのコピーを保存するファイルの名前を指定する。
Replayall	バイナリ	reply コマンドと Reply コマンドの機能を逆にする。
save	バイナリ	メール・メッセージを dead.letter に保存できるようにする。
sendwait	バイナリ	メーラによってメッセージが処理されるまで、mailx を待機させる。このオプションを指定すると、メッセージが配信されるまでユーザは待たなければならないため、ユーザにとっては効率が低下することになる。
SHELL	文字列	~ コマンドまたは ~! コマンドの起動時に使用するシェルを指定できる。
screen	数値	headers コマンドの使用時に、一画面に表示されるメッセージ数を指定する。
sendmail	文字列	メール・メッセージの送信に使用するプログラムのパス名を指定する。この変数を指定しなければ、mailx は省略時の配信システムを使用する。代替りの送信システムについては、システム管理者に問い合わせる。
showto	バイナリ	発信人名ではなく受信者名をメッセージ・ヘッダに表示する。

表 D-1: mailx セッションをカスタマイズする変数 (続き)

変数	型	説明
sign	文字列	~a コマンドの使用時に、メール・メッセージに挿入する文字列を指定できる。
Sign	文字列	~A コマンドの使用時に、メール・メッセージに挿入する文字列を指定できる。
toplines	数値	top コマンドで出力する行数を指定する。省略時の値は 5 行。
verbose	バイナリ	mailx を詳細モードで起動する。実際のメッセージの配信は端末に表示される。これは、コマンド行で -v フラグを使用するのと同様である。この変数は、主にデバッグ目的で使用される。例 D-1 では、verbose 変数の使用方法を示している。
VISUAL	文字列	visual コマンドまたは ~v エスケープを使用する際に使用されるスクリーン・エディタのパス名を指定する。たとえば、次のように指定する。 set VISUAL=/usr/ucb/vi CRT 端末である場合は、EDITOR 変数にスクリーン・エディタを指定することもできる。すると、edit (~e) または visual (~v) のどちらも同じエディタを呼び出す。

次の例は、表 D-1 で説明した verbose 変数の使用方法を示しています。この変数により、mailx がメッセージを送信すると、展開された別名が表示されます。

例 D-1: mailx 詳細モード

```
? set verbose 1
? alias 2
smith csug@solo.my.company.com smith@my.company.com smith
? mailx tg 3
Subject: Conference Room
Starting tomorrow, our weekly meeting will be
moved to Meeting Room 4.

DAL
```

例 D-1: mailx 詳細モード (続き)

```
. 3
EOT
csug@solo.my.company.com... Connecting to (local)...
about to exec
csug@solo.my.company.com... Sent
smith,smith@my.company.com... Connecting to
your.company.com (smtp)...
220 your.company.com ESMTP Sendmail 8.7.6/UNIX 1.7
(1.1.10.5/28Jun99-0151PM) Tue, 25 Nov 1999
09:52:10 -0500 (EST)
>>> HELO solo.my.company.com
250 your.company.com Hello solo.my.company.com
[255.255.255.0],
pleased to meet you
>>> MAIL From:250 ... Sender ok
>>> RCPT To:250 Recipient ok
>>> RCPT To:250 Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 JAA0000022475 Message accepted for delivery
>>> QUIT
221 your.company.com closing connection
smith@my.company.com,smith... Sent 4
? q 5
```

- 1 verbose 変数を設定しています。
 - 2 alias コマンドをパラメータなしで実行すると、別名の内容が表示されます。
 - 3 別名 tg にメッセージを送信しています。
 - 4 メッセージが送信されると別名が展開されて表示されます。
 - 5 q コマンドで mailx セッションを終了しています。
-



E

mailx セッションでのエスケープ・コマンドの使用

エスケープ・コマンドまたはエスケープと呼ばれる特殊なコマンドがあります。これらのコマンドは、メッセージの作成プロセスにおいて、さまざまな機能を実行できます。

エスケープ・コマンドを使用するには、コマンドの前にチルダ(~)を付け、行に単独で入力します。チルダは、現在の編集環境からエスケープしてその後続くコマンドを実行することを mailx にシグナル通知するため、エスケープ文字と呼ばれます。escape メール変数を設定することにより、エスケープ文字を変更することができます。メッセージ行の1文字目にチルダを入力する場合は、チルダを2つ入力してください。

表 E-1 に、エスケープ・コマンドを説明します。

表 E-1: メール用エスケープ・コマンド

コマンド	機能
~~	メール・メッセージの本文にチルダ(~)文字を入力できる。
~!command	ユーザが入力するシェル <i>command</i> を実行する。
~?	エスケープ・コマンドの要約を出力する。
~:command	指定されたメール・コマンドを実行する。
~_command	メッセージの再表示のようなハウスキーピング・タスクの実行に有効。 たとえば、~:10 と入力すると、mailx プロンプトでメッセージ番号 10 を入力したときのように、メッセージ番号 10 が選択されて表示される。
~a	sign に設定された文字列をメール・メッセージに挿入する。

表 E-1: メール用エスケープ・コマンド (続き)

コマンド	機能
<code>~A</code>	<code>Sign</code> に設定された文字列をメール・メッセージに挿入する。
<code>~baddress_list</code>	<code>address_list</code> に指定された名前を <code>Bcc:</code> (秘密コピー) リストに挿入する。
<code>~caddress_list</code>	指定された名前を <code>Cc:</code> (コピー) リストに追加する。
<code>~C</code>	コアをダンプする。
<code>~d</code>	ホーム・ディレクトリにある <code>dead.letter</code> というファイルをメッセージに取り込む。異なるファイルをポイントするためにメール変数 <code>DEAD</code> が使用される。
<code>~e</code>	メール変数 <code>EDITOR</code> で指定されたエディタを起動して、メッセージを編集する。
<code>~f [msg_list]</code>	現在のメッセージまたは指定されたメッセージをユーザのメッセージに読み込む。
<code>~F[msg_list]</code>	<code>~f</code> コマンドと同様の機能であるが、 <code>discard</code> 、 <code>ignore</code> 、または <code>retain</code> などのコマンドに関係なく、すべてのヘッダを含めて取り込まれる。
<code>~h</code>	メッセージ・ヘッダ・フィールドを編集する。このコマンドは、一度に1つのフィールドを表示して、そのフィールドが変更できるようにする。フィールドの終わりにテキストを追加したり、削除キーを使用したり、あるいは、 <code>Ctrl/U</code> を押し、フィールド全体を消去して入力し直すことができる。このコマンドを使用する際には注意が必要。
<code>~i string</code>	指定された変数の値をメール・メッセージに挿入する。 たとえば、コマンド <code>~a</code> は、シグネチャをメッセージに挿入するコマンド <code>~i sign</code> と同じ。

表 E-1: メール用エスケープ・コマンド (続き)

コマンド	機能
<code>~m[msg_list]</code>	現在のメッセージまたは指定されたメッセージを、1 タブ分だけ右にシフトして挿入する。 <code>indentprefix</code> メール変数が設定されている場合、この値はタブ・ストップの前に挿入される。新しいメッセージに追加してメッセージを転送する場合に、区別できる。
<code>~M[msg_list]</code>	<code>~m</code> と同様の機能であるが、 <code>discard</code> 、 <code>ignore</code> 、または <code>retain</code> などのコマンドに関係なく、すべてのヘッダを含めて取り込まれる。
<code>~p</code>	作成中のメッセージを端末に表示する。メッセージが正しいか、または、表題や受信者リストが正しいかを確認する場合に有効。
<code>~q</code> <code>~Q</code>	<code>Ctrl/C</code> を 2 回押した場合のように、現在のメッセージを打ち切る。
<code>~rfile</code> <code>~<file</code> <code>~<!shell_cmd</code>	指定されたファイルをメール・メッセージに読み込む。引数が感嘆符 (!) で始まる場合、文字列の残りは、任意のシステム・コマンドとして認識され、標準出力がメール・メッセージに挿入される。
<code>~ssubject</code>	以前の表題の代わりに、 <code>subject</code> を新しい表題にする。
<code>~tname...</code>	名前をメッセージの <code>To:</code> リストに追加する。
<code>~v</code>	メール変数 <code>VISUAL</code> で指定されたエディタを起動して、メッセージを編集する。

表 E-1: メール用エスケープ・コマンド (続き)

コマンド	機能
<code>~wfile</code>	指定されたファイルにメッセージを書き込む。
<code>~ command</code> <code>~^command</code>	指定されたコマンドに、メッセージをパイプする。これは、メッセージにグローバル変更を行う場合に有効。たとえば、新しいメッセージにメッセージを取り込む場合、次のコマンドを使用すると、 <code>sed</code> エディタを使用して、各行の先頭に山カッコとスペースを付加できる。 <code>~ sed 's/^/> /'</code> その後にテキストが追加できる。結果は、次のとおり。 > This is the text of the message > you have included. > This is the text you add yourself.

F

mailx コマンドの使用

mailx プログラムにはたくさんのコマンドがあり，その一部については，付録 D および付録 E で説明しました。表 F-1 で説明するコマンドは，mailx 環境をより効果的に使用するためのものです。このほか，mailx(1) リファレンス・ページには，特別な場合にのみ有効なコマンドのリストがあります。

表 F-1: mailx プログラムのコマンド

コマンド	説明
=	現在のメッセージ番号をエコーする。
#	メール・スクリプト・ファイルにコメントを書く場合に使用する。
! <i>command</i>	ユーザが入力するシェル・コマンドを実行する。
-[<i>n</i>]	直前のメッセージまたは <i>n</i> 個前のメッセージを選択して表示する。たとえば，-4 は，4 つ前のメッセージを指定する。メールボックスにあるメッセージ数より前に戻ろうとすると，エラー・メッセージが表示される。
? help	ヘルプ情報が表示される。
alias alias <i>alias</i> alias <i>alias name...</i> group g	引数を指定しなければ，現在の別名をリストする。引数を 1 つ指定すると，その別名だけを表示する。2 つ以上の引数を指定すると，最初の引数に指定した別名で，2 番目以降の引数をそのメンバとして，別名を作成する。group コマンドは，alias コマンドと同じである。
alternates [<i>alt_list</i>]	<i>alt_list</i> にリストしたアドレスによってユーザを参照するように mailx に通知する。 <i>alt_list</i> を指定しない場合は，代替名の現在のリストを表示する。
chdir <i>path</i> cd <i>path</i>	cd シェル・コマンドを実行した場合と同様に，現在のディレクトリを指定されたパス名に変更する。ただし，chdir で指定したディレクトリは，ユーザが mailx 環境下にいる間だけ有効。

表 F-1: mailx プログラムのコマンド (続き)

コマンド	説明
<pre>copy [msg_list] file co [msg_list] file c [msg_list] file</pre>	現在のメッセージまたは指定されたメッセージをファイルにコピーする。 <i>file</i> が存在する場合、メッセージは追加される。このコマンドは、 <i>save</i> と同様に機能するが、コピー・メッセージにマークを付けて、 <i>mailx</i> セッションを終了する際に削除することはない。
<pre>Copy [msg_list] C [msg_list]</pre>	<i>msg_list</i> にある最初のメッセージの作成者から名前を取ったファイルに、指定されたメッセージを保存する。ただし、メッセージに保存のマーク付けはされない。この点を除けば、 <i>Save</i> コマンドと同じ。
<pre>delete [msg_list] d [msg_list]</pre>	現在のメッセージまたは指定されたメッセージを削除する。 <i>undelete</i> コマンドを使用すると、誤って削除したメッセージを回復できる。
<pre>discard [field_list]</pre>	<i>ignore</i> サブコマンドと同じ機能。
<pre>dp dt</pre>	現在のメッセージを削除して、次にアクティブなメッセージを表示する。
<pre>echo string</pre>	指定の文字列をエコーする。シェルの <i>echo</i> コマンドと同じ。
<pre>edit [msg_list] e [msg_list]</pre>	<i>EDITOR</i> で指定されたエディタを起動して、 <i>msg_list</i> をエディタへロードする。行われた変更は、メール・セッションを終了する際、 <i>msg_list</i> に保存される。
<pre>exit ex</pre>	システム・メールボックスを更新しないで、 <i>mailx</i> を終了する。
<pre>file [file] fi [file] folder [file] fo [file]</pre>	メール・ファイルまたはメール・フォルダを選択する。ファイルを指定しなければ、現在のパスとファイル名、および現在のファイルにあるメッセージ数を表示する。ファイルまたはフォルダを指定した場合には、現在のメール・ファイルに加えた変更が表示され、指定したファイルが読み込まれる。
<pre>folders</pre>	フォルダ・ディレクトリ内のフォルダ名をリストする。
<pre>followup message fo message</pre>	メッセージに応答し、そのメッセージの送信者から名前を取ったファイルに、その応答を保存する。 <i>record</i> オプションが設定されている場合、その設定は上書きされる。

表 F-1: mailx プログラムのコマンド (続き)

コマンド	説明
Followup [<i>msg_list</i>] F [<i>msg_list</i>]	<i>msg_list</i> の最初のメッセージに回答し、そのメッセージを <i>msg_list</i> にある各メッセージの作成者に送信する。表題は最初のメッセージから取られ、回答は、最初のメッセージの作成者から名前を取ったファイルに保存される。
from [<i>login</i>] f [<i>login</i>]	アクティブなメッセージのヘッダを表示する。ログイン名を指定すると、指定されたログイン名から送信されたすべてのアクティブなメッセージが表示される。
headers [<i>n</i>] h [<i>n</i>]	アクティブなメッセージのヘッダをリストする。 <i>screen</i> 変数の値が、表示するヘッダ数として使用される。 <i>screen</i> 変数の説明については、付録 D を参照。メッセージが 1 画面に収まらない場合には、 <i>z</i> コマンドを使用して、前方または後方に 1 画面ずつスクロールできる。メッセージ番号を指定した場合、 <i>headers</i> コマンドは、指定されたメッセージを 1 画面の範囲内で表示する。
hold [<i>msg_list</i>] ho [<i>msg_list</i>] preserve [<i>msg_list</i>] pre [<i>msg_list</i>]	現在のメッセージまたは指定されたメッセージを <i>mbox</i> ファイルに転送しないで、ユーザのシステム・メールボックスに保存する。
if <i>condition</i> i <i>condition</i> else e endif en	<i>mailx</i> サブコマンドを条件付きで実行するための構文。 <i>condition</i> が真の場合、 <i>if</i> の後に記述されているサブコマンドが実行される。 <i>condition</i> が偽の場合は、 <i>else</i> の後に記述されているサブコマンドが実行される。 <i>else</i> は必須ではないが、 <i>endif</i> は必須。 <i>condition</i> は、メールを送信する場合の <i>send</i> 、またはメールを受信する場合の <i>receive</i> のどちらか。
ignore [<i>field...</i>]	<i>print</i> コマンドまたは <i>type</i> コマンドを使用する際に、 <i>mailx</i> が、ヘッダの指定されたフィールドを省いてメッセージを表示するようにする。このコマンドは、付録 D で説明している <i>ignore</i> 変数とは異なる。引数を指定しないで <i>ignore</i> コマンドを入力すると、現在省略されているフィールドのリストが表示される。
list l	使用可能な <i>mailx</i> サブコマンドのリストを表示する。
local	ローカル・ホストの他の名称をリストする。
mail <i>user_name</i> m <i>user_name</i>	指定されたユーザにメッセージを送信する。

表 F-1: mailx プログラムのコマンド (続き)

コマンド	説明
<code>mbox [msg_list]</code>	現在のメッセージまたは指定されたメッセージが <code>mbox</code> に転送されるようにマークを付ける。 <code>.mailrc</code> ファイルで <code>hold</code> 変数を設定している場合に有効。
<code>more [msg_list]</code>	PAGER で定義したページャ・プログラムを使用して, <code>msg_list</code> のメッセージを表示する。 <code>page</code> サブコマンドと同機能。
<code>More [msg_list]</code>	<code>more</code> サブコマンドと同機能であるが, 省略されているヘッダ・フィールドも表示する。 <code>more</code> および <code>ignore</code> サブコマンドを参照。
<code>new [msg_list]</code> <code>New [msg_list]</code>	<code>msg_list</code> の各メッセージに, まだ参照されていないものとしてマークを付ける。 <code>unread</code> および <code>Unread</code> サブコマンドと同機能。
<code>page [msg_list]</code>	PAGER で定義したページャ・プログラムを使用して, <code>msg_list</code> のメッセージを表示する。 <code>more</code> サブコマンドと同機能。
<code>Page [msg_list]</code>	<code>page</code> と同機能であるが, 省略されているヘッダ・フィールドも表示する。 <code>More</code> サブコマンドと同機能。
<code>pipe [msg_list]</code> <code>[shell_command]</code> <code>pi [msg_list]</code> <code>[shell_command]</code> <code> [msg_list] [shell_command]</code>	<code>msg_list</code> を <code>shell_command</code> にパイプする。メッセージは, 読み込むものとして処理される。引数が指定されない場合は, 現在のメッセージが, <code>cmd</code> に指定されたコマンドへパイプされる。 <code>page</code> オプションが設定されていると, 各メッセージの後に改ページ文字が挿入される。
<code>next</code> <code>n</code> <code>+</code> <code>Return</code>	次のメッセージを表示する。
<code>Print [message]</code> <code>P [message]</code> <code>Type [message]</code> <code>T [message]</code>	<code>ignore</code> コマンドによって指定されたヘッダ・フィールドもすべて含めて, 現在のメッセージまたは指定されたメッセージを表示する。
<code>print [message]</code> <code>p [message]</code> <code>type [message]</code> <code>t [message]</code>	<code>ignore</code> コマンドで指定されたヘッダ・フィールドを省略して, 現在のメッセージまたは指定されたメッセージを表示する。

表 F-1: mailx プログラムのコマンド (続き)

コマンド	説明
quit q	mailx プログラムを終了して、システム・メールボックスを更新する。hold 変数を設定していない場合は、削除、保存、または保持されなかったメッセージはすべて、mbox ファイルに転送される。hold 変数を設定している場合は、これらのメッセージはすべてシステム・メールボックスに残され、既読としてマーク付けされる。メッセージを保存しないでセッションを終了するには、exit サブコマンドを使用する。
Reply R Respond	メッセージに回答する。そのメッセージがユーザ・グループ宛に送信された場合、Reply および Respond コマンドによる回答は、そのメッセージの発信者だけに送信される。
reply r respond	メッセージに回答する。そのメッセージがユーザ・グループ宛に送信された場合、reply コマンドおよび respond コマンドによる回答は、そのメッセージを受信したすべてのユーザに送信される。
retain [field_list]	print サブコマンドまたは type サブコマンドでメッセージを表示する際に保持されるヘッダのリストに、field_list で指定したヘッダ・フィールドを追加する。type および print を使用すると、保持されないフィールドを含め、メッセージ全体が表示される。引数を指定しないで retain を実行すると、現在保持されているフィールドがリストされる。
save [msg_list] file s [msg_list] file	現在のメッセージまたは指定されたメッセージをファイルに保存する。指定されたファイルの内容が削除されないように、メッセージは、そのファイルに追加される。
Save [msg_list] S [msg_list]	指定されたメッセージを、最初のメッセージの作成者から名前を取ったファイルに保存する。ファイル名は、作成者の名前からネットワーク・アドレス部分を取り除いたものになる。
set [variable] se [variable]	変数を指定しないで入力すると、設定しているオプションがすべて表示される。変数を指定すると、そのオプションが設定される。使用可能な変数については、付録 D を参照。
shell sh	対話式でシェルを呼び出す。
source file so file	ファイル (通常は .mailrc) からメール・コマンドを読み取る。

表 F-1: mailx プログラムのコマンド (続き)

コマンド	説明
<code>size [msg_list]</code> <code>si [msg_list]</code>	<code>msg_list</code> のメッセージの行数および文字数を表示する。
<code>top [msg_list]</code> <code>to [msg_list]</code>	現在のメッセージまたは指定された各メッセージの最初の数行を表示する。表示される行数は、 <code>toplines</code> 変数で指定される。省略時の値は5。
<code>touch [msg_list]</code>	<code>msg_list</code> のメッセージを、まだ読んでいない場合にも、mailx プログラムを終了すると、システム・メールボックスからユーザ個人の mbox に転送されるようにマーク付けする。それらのメッセージは、未読として mbox に保存される。 <code>touch</code> コマンドを使用すると、 <code>msg_list</code> の最後のメッセージが、現在のメッセージになる。
<code>unalias alias_list</code>	指定された別名を削除する。
<code>undelete msg_list</code> <code>u msg_list</code>	指定されたメッセージの削除を取り消す。
<code>unread [msg_list]</code> <code>Unread [msg_list]</code> <code>U [msg_list]</code>	<code>msg_list</code> の各メッセージを未読としてマーク付けする。 <code>new</code> および <code>New</code> サブコマンドと同機能。
<code>unset [variable]</code> <code>uns [variable]</code>	オプションの設定を取り消す (解除する)。たとえば、 <code>.mailrc</code> ファイルに <code>set hold</code> コマンドが記述されている場合、 <code>unset</code> コマンドを使用して、現在の mailx セッションの間、 <code>hold</code> 変数を無効できる。
<code>version</code> <code>ve</code>	mailx コマンドのバージョン・バーナーを表示する。
<code>visual</code> <code>v</code>	VISUAL メール変数で指定されたエディタを起動して、現在のメッセージを編集する。
<code>write</code> <code>[msg_list] file</code> <code>w [msg_list]</code> <code>file</code>	現在のメッセージまたは指定されたメッセージを、指定したファイルに保存する。 <code>save</code> コマンドと同機能であるが、 <code>write</code> は、各メッセージの本文のみを保存する。ヘッダは削除される。
<code>z[+]</code> <code>z[-]</code>	メッセージを前方または後方に1画面分ずつスクロールする。 <code>screen</code> 変数で1画面に表示するメッセージ数の指定が可能 (付録Dを参照)。前方に1画面分スクロールする場合は、 <code>z</code> または <code>z+</code> を入力し、後方にスクロールする場合は、 <code>z-</code> を入力する。

アクセス制御リスト (ACL)

ここでは、アクセス制御リスト (ACL) を使用して、特定のユーザおよびユーザのグループに対してファイルおよびディレクトリ・アクセス許可を設定する方法について説明します。

ACL は、ファイルおよびディレクトリに関する Tru64 UNIX のアクセス許可を拡張するものです。ACL を使用して、特定のユーザおよびグループにファイルおよびディレクトリに関するアクセス許可を設定することができます。ファイルやディレクトリに設定されている通常のアクセス許可に加え、ACL は各ユーザあるいはグループに対する特別なアクセス許可を設定します。

ファイルには、ACL が 1 つだけ対応付けられます。ディレクトリには、アクセス ACL、デフォルト・アクセス ACL、デフォルト・ディレクトリ ACL の 3 つの ACL を対応付けることができます。アクセス ACL は、誰がそのファイルあるいはディレクトリに対してアクセスできるかを決定します。デフォルト・アクセス ACL は、デフォルト・アクセス ACL を持つディレクトリでファイルあるいはディレクトリを作成した場合にどの ACL を継承するかを決定します。

ACL はいつでも設定できますが、ACL のアクセス・チェックと ACL の継承は ACL が有効になっている場合のみ行われます。システムで ACL が無効になっている場合は、ACL コマンドを実行すると警告メッセージが表示されます。ACL はすべてのファイル・システムでサポートされるわけではありません。ACL は、ACL をサポートするファイル・システム上のファイルあるいはディレクトリに対してのみ設定できます。

注意

ACL を作成するためには、システムで ACL サポートが有効になっている必要があります。使用しているシステムおよびファイル・システムで ACL が使用可能かどうかは、システム管理者に確認してください。

G.1 ACL の構造

アクセス制御リストは、それぞれが以下の 3 つのフィールドを含む多数の ACL エントリから構成されます。

- エントリ・タイプを識別するキーワード
- グループ ID, ユーザ ID または名前を含む修飾子フィールド
- 許可指定

ACL の外部 (出力可能) 表現は、カンマ (,) または改行で区切られたエントリからなります。ACL エントリ内のフィールドはコロン (:) で区切られます。次の例は、典型的な ACL エントリです。

```
user::rwx
user:juanita:r-w
user:sam:r-x
group::rwx
other::---
```

ACL エントリのキーワードと修飾子は以下のように定義されます。

`user::` 修飾子フィールドが空の `user` エントリは、ファイルを所有するユーザのアクセス許可を定義します。このエントリ (所有ユーザ・エントリと呼ぶ) は、常に `user` 許可ビットと同じです。ACL には、`user::` エントリが 1 つだけ含まれていなければなりません。

`user:` 修飾子フィールドが空でない `user` エントリは、修飾子フィールドで指定されたユーザのアクセス許可を定義します。修飾子フィールドには、ユーザ名と UID のどちらかが含まれていなければなりません。ACL には 0 個以上の `user:` エントリを含めることができます。

`group::` 修飾子フィールドが空の `group` エントリは、ファイルを所有するグループのメンバのアクセス許可を定義します。このエントリ (所有グループ・エントリと呼ぶ) は、常にグループ許可ビットと同じです。ACL には、`group::` エントリが 1 つだけ含まれていなければなりません。

- `group:` 修飾子フィールドが空でない `group` エントリは、修飾子フィールドで指定されたグループのメンバのアクセス許可を定義します。修飾子フィールドには、グループ名と GID のどちらかが含まれていなければなりません。ACL には、0 個以上の `group:` エントリを含めることができます。
- `other::` `other` エントリは、修飾子が空の場合にのみ有効です。このエントリは、ACL の他のエントリのどれとも一致しないユーザーすべてのアクセス許可を定義します。このエントリは、常に `other` 許可ビットと同じです。ACL には、`other::` エントリが 1 つだけ含まれていなければなりません。

許可フィールドの文字列は、`ls` コマンドが従来の許可ビットに対して表示する文字列と同じで、順序も同じです。`r` は読み取りアクセス、`w` は書き込みアクセス、そして `x` は実行または検索アクセスを示します。これらの文字の代わりにハイフン (`-`) 文字が使用されているときには、そのアクセスの拒否を示します。

ACL のユーザー、グループ、その他のエントリは、ファイルやディレクトリの従来の許可ビットに対応しています。ACL が有効で、`chmod` コマンドを使用してファイルやディレクトリの従来の許可ビットを変更した場合、`chmod` は所有ユーザー・エントリ、所有グループ・エントリ、および `other` エントリ用のアクセス ACL を適切に変更します。

表 G-1 では、典型的な ACL エントリを示して説明します。

表 G-1: ACL エントリの例

エントリ	照合基準
<code>group:acct:r--</code>	<code>acct</code> グループのすべてのユーザーに対し、読み取りを許可する。
<code>user:joe:rw-</code>	ユーザー <code>joe</code> に対し、読み取りと書き込みを許可する。
<code>user::rwx</code>	ファイルの作成後に所有者が変更されていないかどうかには関係なく、オブジェクトの所有者に対し、読み取り、書き込み、および実行を許可する。

表 G-1: ACL エントリの例 (続き)

<code>group::r--</code>	ファイルの作成後に所有グループが変更されていないかどうかには関係なく、オブジェクトの所有グループに対し、読み取りを許可する。
<code>other::r--</code>	所有ユーザおよび所有グループ、ACL エントリにリストされたユーザおよびグループを除いたすべてのユーザとすべてのグループに対し、読み取りを許可する。

G.2 アクセス決定のプロセス

プロセスがファイルまたはディレクトリへのアクセスを要求すると、以下の手順で、アクセス ACL がチェックされます。

1. プロセスにスーパーユーザ特権がある場合、ファイルやディレクトリへのアクセスが許可されます。アクセス ACL と許可ビットはチェックされません。
2. ACL が無効になっている場合、または ACL は有効になっているがファイルやディレクトリに対応するアクセス ACL がない場合、従来の UNIX 許可ビットでチェックされます。
3. ファイルやディレクトリのアクセス ACL は、以下のようにチェックされます。
 - a. プロセスがオブジェクトの所有者である場合、所有ユーザ `user::` エントリの許可が与えられます。他の ACL エントリはチェックされません。これは、`user` 許可ビットを使用するのと同じです。
 - b. プロセスの UID が `user:` エントリにリストされた UID に一致するか、または `user:` エントリにリストされたユーザ名に帰着する場合、そのエントリの許可が与えられます。他の残りの ACL エントリはチェックされません。
 - c. プロセスの GID がファイルの GID に一致する場合、またはプロセスの補助グループのいずれかがファイルの GID に一致する場合、プロセスは、`group:` エントリの許可と次の項目で説明しているような一致するすべての `group:` エントリの許可の論理和が与えられます。
 - d. プロセスの GID がいずれかの `group:` エントリの GID に一致するか、またはいずれかの `group:` エントリにリストされたグループ名に帰着する場合、またはプロセスのいずれかの補助グループの

GID またはグループ名が ACL のいずれかの `group:` エントリに一致する場合、一致するすべてのグループ・エントリの許可の論理和が与えられます。たとえば、グループ `sales` およびグループ `eng` に属するユーザがいるとき、ファイルのアクセス ACL がグループ `sales` に読み取りアクセスを与え、グループ `eng` に書き込みアクセスを与える場合、そのユーザにはファイルへの読み取りおよび書き込みアクセスが与えられます。

- e. `other:` エントリの許可が与えられます。これは、`other` 許可ビットを使用するのと同じです。

注意

従来の UNIX 許可ビットを持つファイルやディレクトリと、3 つの必須エントリ (`user::`、`group::`、および `other::`) だけを含むアクセス ACL を持つファイルやディレクトリは見分けがつきません。

G.3 ACL の継承

ファイルやディレクトリが作成される際には、親ディレクトリから ACL が継承されます。デフォルト ACL は、親ディレクトリ内に作成されるファイルおよびサブディレクトリにどのような ACL が継承されるかを、次のように決定します。

- 親ディレクトリにデフォルト ACL がない場合
 - そのディレクトリ内に作成される新しいファイルの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	なし

- そのディレクトリ内に作成される新しいサブディレクトリの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	なし
デフォルト・アクセス ACL	なし
デフォルト・ディレクトリ ACL	なし

許可ビットは、従来の UNIX 許可ビットの場合と同様に設定されます。

- 親ディレクトリにデフォルト・アクセス ACL はあるが、デフォルト・ディレクトリ ACL がない場合

- そのディレクトリ内に作成される新しいファイルの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	親のデフォルト・アクセス ACL

- そのディレクトリ内に作成される新しいサブディレクトリの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	親のデフォルト・アクセス ACL
デフォルト・アクセス ACL	親のデフォルト・アクセス ACL
デフォルト・ディレクトリ ACL	なし

- 親ディレクトリにデフォルト・アクセス ACL はないが、デフォルト・ディレクトリ ACL がある場合

- そのディレクトリ内に作成される新しいファイルの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	なし

- そのディレクトリ内に作成される新しいサブディレクトリの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	親のデフォルト・ディレクトリ ACL

ACL タイプ	ステータス
デフォルト・アクセス ACL	なし
デフォルト・ディレクトリ ACL	親のデフォルト・ディレクトリ ACL

- 親ディレクトリにデフォルト・アクセス ACL とデフォルト・ディレクトリ ACL の両方がある場合
 - そのディレクトリ内に作成される新しいファイルの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	親のデフォルト・アクセス ACL

- そのディレクトリ内に作成される新しいサブディレクトリの ACL は、次のように設定されます。

ACL タイプ	ステータス
アクセス ACL	親のデフォルト・ディレクトリ ACL
デフォルト・アクセス ACL	親のデフォルト・アクセス ACL
デフォルト・ディレクトリ ACL	親のデフォルト・ディレクトリ ACL

ディレクトリにデフォルト ACL を設定しても、そのディレクトリ内の既存のファイルやサブディレクトリの ACL は変更されません。

G.3.1 ACL 継承の例

ACL 継承の例をいくつか以下に示します。

- ディレクトリ `foo` にデフォルト ACL が無いときに、次のコマンドを実行して、`foo` にデフォルト・アクセス ACL を設定したとします。

```
% setacl -d -u user::rw-,group::r--,other::r--,user:jdoe:rw-\
foo
```

ディレクトリ `foo` 内に作成されるすべてのファイルやディレクトリは、アクセス ACL として次の ACL を継承します。

```
#
# file: foo
# owner: smith
# group: system
```

```
#
user::rw-
user:jdoe:rw-
group::r--
other::r--
```

- ディレクトリ `foo` にデフォルト ACL がないときに、次のコマンドを実行して `foo` にデフォルト・ディレクトリ ACL を設定したとします。

```
% setacl -D -u user::rw-,group::r--,other::r--, \
  user:jdoe:rw- foo
```

ディレクトリ `foo` 内に作成されるすべてのディレクトリは、アクセス ACL として次の ACL を継承します。デフォルト・ディレクトリ ACL も同様です。

```
#
# file: foo
# owner: smith
# group: system
#
user::rwx
user:jdoe:rwx
group::r--
other::r--
```

- ディレクトリ `foo` にデフォルト ACL がないときに、次のコマンドを実行して `foo` にデフォルト・アクセス ACL とデフォルト・ディレクトリ ACL を設定したとします。

```
% setacl -D -u user::rw-,group::r--,other::r--, \
  user:jdoe:rw- foo
```

```
% setacl -d -u user::rw-,group::r--,other::r--, \
  user:wilson:rwx foo
```

ディレクトリ `foo` 内に作成されるすべてのディレクトリは、デフォルト・ディレクトリ ACL とアクセス ACL として次の ACL を継承します。

```
#
# file: foo
# owner: smith
# group: system
#
user::rw-
user:jdoe:rw-
group::r--
other::r--
```

次の ACL は、デフォルト・アクセス ACL として継承されます。

```
#
# file: foo
# owner: smith
# group: system
#
user::rw-
user:wilson:rw-
group::r--
other::r--
```

ディレクトリ `foo` 内に作成されるすべてのファイルは、アクセス ACL として次の ACL を継承します。

```
#
# file: foo
# owner: smith
# group: system
#
user::rw-
user:wilson:rw-
group::r--
other::r--
```

G.4 ACL の管理

以下のコマンドを使用して、ACL の表示や変更を行います。

`dxsetacl` ファイルおよびディレクトリの ACL を一覧表示して変更するグラフィック・インタフェース。

`setacl` ファイルおよびディレクトリの ACL を変更する。

`getacl` ファイルおよびディレクトリの ACL をリストする。

注意

ACL の作成、変更、削除には、ファイルあるいはディレクトリの所有者アクセス許可 (あるいはスーパーユーザ特権) が必要になります。つまり、`ls -l` でファイル情報を表示した場合に、3 つ目のフィールドにあなたのユーザ名が表示されなければなりません。スーパーユーザ特権についての詳細は 5.7 節を参照してください。

G.4.1 dxsetacl インタフェースの使用

ACL の表示は、dxsetacl の GUI または getacl コマンドを使用して行います。dxsetacl インタフェースは、アプリケーション・マネージャからアクセスできる CDE デスクトップ・アプリケーション内に含まれています。またはコマンド行から次のように入力して起動することもできます。

```
% /usr/bin/X11/dxsetacl &
```

詳細は、dxsetacl(1) および dxsetacl のオンライン・ヘルプを参照してください。

G.4.2 getacl コマンドの使用

ファイルあるいはディレクトリの ACL を表示には getacl コマンドを使用します。例 G-1 では、getacl file.txt コマンドで file.txt ファイルの ACL を表示しています。

例 G-1: ファイルの ACL の表示

```
$ getacl file.txt
#
# file: file.txt
# owner: peter
# group: system
#
user::rw-
user:jdoe:rw-
group::r--
other::r--
```

getacl コマンドでファイルあるいはディレクトリのアクセス ACL を表示する場合、ファイルあるいはディレクトリにアクセス ACL が設定されていないと、Tru64 UNIX アクセス許可が ACL フォーマットで表示されます。

詳細は getacl(1) を参照してください。

G.4.3 setacl コマンドの使用

ファイルあるいはディレクトリの ACL の作成、変更、削除には、setacl コマンドを使用します。例 G-2 では、getacl file.txt コマンドで

G-10 アクセス制御リスト (ACL)

file.txt ファイルの ACL を表示しています。setacl コマンドは ACL を更新しています。

例 G-2: ファイルの ACL の設定

```
$ getacl file.txt
#
# file: file.txt
# owner: peter
# group: system
#
user::rw-
user:jdoe:rw-
group::r--
other::r--
$ setacl -u group::rw-,user:evan:rw- file.txt
$ getacl file.txt
#
# file: file.txt
# owner: peter
# group: system
#
user::rw-
user:jdoe:rw-
user:evan:rw-
group::rw-
other::r--
```

所有グループ・エントリが変更され、書き込み許可が含まれています。ユーザ evan のエントリは既存のエントリと一致せず、読み書き許可が追加されています。その他のエントリは変更ありません。

詳細については `setacl(1)` を参照してください。

G.5 コマンド，ユーティリティ，およびアプリケーションと ACL の相互作用

ACL は、POSIX P1003.6 ドラフト 13 に基づく、POSIX および System V 互換の UNIX 拡張です。すべての既存のコマンド、ユーティリティ、およびアプリケーションが ACL を適切に使用または伝達しているわけではありません。特に HP 以外から提供されたアプリケーションや POSIX 互換でないアプリケーションでは注意が必要です。コマンド、ユーティリティ、またはアプリケーションを使用して、ACL が設定されているファイル・シス

テム・オブジェクト (ファイルまたはディレクトリ) をアクセスまたは処理する場合は、処理の完了後に、ACL が削除も変更もされていないことを確認しなければなりません。

ファイルを修正する多くのプログラムは、以下の手順で処理します。

- ファイルの新しいバージョンを一時的な名前で作成する
- ファイルの既存のバージョンを削除する
- 新しいバージョンの名前を一時的な名前から本来の名前へ変更する

修正しているファイルに ACL があり、ファイルの一時的バージョンの作成時に、プログラムが ACL を複製しない場合、上記の手順ではファイルの ACL が削除されるか、または親ディレクトリのデフォルト・アクセス ACL (もしある場合) で ACL が置き換えられます。ACL のあるファイルに対してそのようなプログラムを使用した場合、後で ACL を復元しなければなりません。また、この手順では、ファイルからハード・リンクが削除されます。この方法でファイルを修正する一般的なコマンドには、以下のものがあります。

- `gzip`
- `compress`
- `emacs`

この問題を回避するには、元のファイルを一時ファイルへコピーし、一時ファイルに対して処理を行ってから、`-p` オプションを指定しないで `cp` を実行して、コピー・バックします。この手順では元の ACL を維持できます。

ACL のあるファイルをコピーするときには、必ず `cp -p` コマンドを使用します。このコマンドでは、ACL と他の拡張属性 (プロパティ・リスト) が適切にコピーされます。

G.5.1 pax および tar コマンド

`pax` および `tar` では、特に指定しない限り、アーカイブ作成時に、ファイルおよびディレクトリにある ACL および他の拡張属性がアーカイブされます。ただし、`pax` や `tar` ユーティリティを使用してアーカイブからファイルおよびディレクトリを抽出するときには、アーカイブされたファイルおよびディレクトリの ACL は、デフォルトではアーカイブから抽出されません。格納先ディレクトリにデフォルト ACL が定義されている場合、

アーカイブから抽出したファイルおよびディレクトリは、G.3 節で説明されているようにデフォルト ACL を継承します。

アーカイブから ACL 情報およびプロパティ・リスト情報を復元するには、アーカイブからファイルおよびディレクトリを抽出する際に、tar には -p オプションを指定し、pax には -p p または -p e オプションを指定します。pax および tar ユーティリティでは、ACL のユーザ情報およびグループ情報は UID および GID として保存されます。つまり、tar -p、pax -p p または pax -p e コマンドを使用して、システム上にアーカイブを復元し、そのシステムがアーカイブ元のシステムとユーザ情報およびグループ情報を共有していない場合、意図していないファイル・アクセスを許可することになる場合があります。

現時点では、ACL および拡張属性 (プロパティ・リスト) についての正式な業界標準はありません。このため、プロパティ・リストおよび ACL をサポートする pax および tar の拡張機能は、Tru64 UNIX 独自の機能です。他のベンダの pax および tar では、Tru64 UNIX 独自の拡張機能は無視されます。ただし、他のベンダのシステムとの相互運用性を確保するためには、マルチベンダ配布用にアーカイブを行うときには -v オプションを指定して、ACL および他の拡張属性がアーカイブされないようにします。



数字および記号

- @ (アットマーク)
 - メールでエコーされた Ctrl/c .. D-1
- ! (感嘆符)
 - エスケープ・コマンド(メール) E-1
 - シェル・コマンドでの実行 F-1
 - シェル・コマンドの実行 E-1
- ? (疑問符) 11-9
 - エスケープ・コマンド(メール) E-1
- : (コロン) E-1
- | (縦線)
 - エスケープ・コマンド(メール) E-1
 - コマンドへのメール・メッセージの
引き渡し E-1
- ~ (チルダ) E-1
 - パス名での表記 2-13
- + (プラス記号)
 - next コマンドとして使用(メー
ル) F-1
- (マイナス記号)
 - メール・メッセージの選択 F-1
- 1 つのファイルに対する複数のファイ
ル名 3-23
- 2 進数
 - 許可 5-15
- 8 進数
 - 許可の設定 5-13

A

- account** サブコマンド (**ftp**) 12-6
- ACL**
 - getacl コマンド G-10
 - アクセス許可 G-1
 - 維持 G-11
 - 決定プロセス G-4
 - 表示 G-10
 - ファイルのリスト G-10
 - フォーマット G-2
- ACL** エントリ
 - 削除 G-10
 - 修正 G-10
 - 追加 G-10
- add (a)** コマンド (**vi** テキスト・エ
ディタ) A-11
- alias**
 - C シェル 8-13
 - リスト (MH) 11-24
- alias** コマンド(メール) F-1
- ali** コマンド (**MH**) 11-24
- anno** コマンド (**MH**) 11-24
- append text (A)** コマンド (**vi** テキス
ト・エディタ) A-13
- append** サブコマンド (**ed** エディ
タ) B-3
- append** 変数(メール) D-1
- apropos** コマンド 1-17, 1-19

ASCII

- コード・セット C-2
- ascii** サブコマンド (**ftp**) 12-6
- askcc** 変数(メール) 11-20, D-1
- ask** 変数(メール) 11-20, D-1
- autoprint** 変数(メール) D-1

B

- Backspace** キー 1-8
- bg** コマンド 6-16
- bg** コマンド (**C** シェル) 8-13
- binary** サブコマンド (**ftp**) 12-6
- Bourne** シェル 7-1, 8-1, 8-13
 - .logout スクリプト 7-29
 - .profile ログイン・スクリプト 7-19, 8-14
 - エラーのリダイレクション 6-5
 - 組み込みコマンド 8-18
 - 組み込み変数 8-17
 - パターン照合 2-16
 - 変数値のクリア 7-26
 - 変数の設定 7-22
 - メタキャラクタ 8-15
 - ログイン・スクリプト 8-14
- burst** コマンド (**MH**) 11-24
- bye** サブコマンド (**ftp**) 12-6

C

- cat** コマンド 3-6, 3-8
- Cc:** リスト(メール) E-1
- cdup** サブコマンド (**ftp**) 12-8
- cd** コマンド 3-2, 4-4
 - Bourne シェル 8-19

- Korn シェル 8-34
- POSIX シェル 8-34
- cd** サブコマンド (**ftp**) 12-8
- change (c)** コマンド (**vi** テキスト・エディタ) A-15
- change (c)** サブコマンド (**ed** エディタ) B-22
- change word (cw)** コマンド (**vi** テキスト・エディタ) A-15
- chdir** コマンド(メール) F-1
- chgrp (change group)** コマンド 5-23
- chgrp** コマンド 5-23
- chmod (change mode)** コマンド 5-9, 5-12, 5-15
- chmod** コマンド 5-5
- chown (change owner)** コマンド 5-23
- chown** コマンド 5-23
- close** サブコマンド (**telnet**) 13-6
- codesets**
 - Unicode C-2
- comp** コマンド (**MH**) 11-24
- control shift**
 - Ctrl/Q 3-8
 - Ctrl/S 3-8
- copy** コマンド(メール) F-1
- cp** コマンド 3-22, 3-23
- CRT** 画面
 - talk コマンドの使用 11-28
- crt** 変数(メール) 11-21, D-1
- csh.login** システム・ログイン・スクリプト 7-19
- .cshrc** ログイン・スクリプト 7-20, 8-3

MH プログラムのための変更 11-21
 System V 環境 9-2
Ctrl/C
 メール・メッセージの打ち切
 り 11-5
Ctrl/D..... 5-20, 5-23
 メール・メッセージの終了. 11-20
 ログアウトのため 1-7
ct コマンド (UUCP)
 オプションの使用
 モデムを介したリモート・ホスト
 への接続 14-18
cu コマンド
 ローカル・コマンドの使用... 14-7
cu コマンド (UUCP)
 オプションの使用
 リモート・ホストへの接続 14-3
 ローカルからリモートへの接
 続 14-9
 ローカル・コマンドの使用... 14-3
C シェル 7-1, 8-1
 .cshrc ログイン・スクリプト 7-20,
 8-3
 .login スクリプト 7-20, 8-4
 .logout スクリプト 7-29
 環境変数の設定 7-24
 組み込みコマンド 8-12
 組み込み変数 8-11
 コマンド・ヒストリ..... 7-4, 8-7
 ファイル名補完 7-4, 8-9
 プロセスの停止と開始 6-15
 別のシェルへの変更..... 7-7
 別名 7-4, 8-9
 変数値のクリア 7-26

変数値の表示 7-26
 メタキャラクタ 8-5
 リダイレクション
 エラー..... 6-6
 ログイン・スクリプト 8-2
C ロケール C-8

D

date コマンド..... 1-8
dead.letter ファイル 11-5
 作成の防止..... D-1
 メール・メッセージ..... E-1
debug 変数(メール)..... D-1
delete (d) サブコマンド (**ed** エディ
 タ) B-19
delete character (x) コマンド (**vi** テ
 キスト・エディタ) A-15
delete line (dd) コマンド (**vi** テキ
 スト・エディタ) A-15
delete word (dw) コマンド (**vi** テキ
 スト・エディタ) A-14
delete コマンド(メール)..... F-1
delete サブコマンド (**ftp**) 12-8
df コマンド 3-17
diff コマンド..... 3-28
display サブコマンド (**telnet**) . 13-6
dist コマンド (**MH**) 11-24
dot オプション(メール)..... 11-20
dot 変数(メール) D-1
 ignoreeof 変数(メール)..... D-1
 設定しない際の注意..... D-1
 ビリオド
 メール・メッセージの終了 . D-1

dp コマンド(メール) F-1
dt コマンド(メール)..... F-1

E

echo コマンド 7-25
 Bourne シェル 8-19
 C シェル 8-13
 Korn シェル 8-34
 POSIX シェル..... 8-34
ed B-2, B-20
edit (e) サブコマンド (**ed** エディタ) B-7
edit (ed) コマンド (**ed** エディタ) B-8
EDITOR 変数(メール).... D-1, E-1
 VISUAL 変数(メール) D-1
ed エディタ B-1
 (append) サブコマンド B-3
 change (c) サブコマンド..... B-22
 delete (d) サブコマンド B-19
 edit (e) サブコマンド..... B-7
 edit (ed) コマンド B-8
 global (g) 演算子 B-17
 insert (i) サブコマンド B-24
 move (m) サブコマンド B-21
 print (p) サブコマンド B-4
 quit (q) サブコマンド B-6
 read (r) サブコマンド ... B-7, B-9
 substitute (s) サブコマンド . B-15
 t サブコマンド B-26
 write (w) サブコマンド.. B-5, B-7
 行のコピー..... B-26
 現在の行の削除 B-20
 現在の行の表示 B-10
 システム・コマンドの使用.. B-27

 タイプ・ミスの訂正..... B-3
 テキストの移動 B-21
 テキストの探索 B-13
 テキストの保存 B-5
 テキスト・ファイルの作成と保存 B-2
 ファイルの一部の保存 B-6
 複数行 B-21
 複数行での置換 B-17
 文脈探索..... B-13
 編集バッファ B-1
 文字の削除..... B-18
 文字列の置換 B-15
errorbells 環境変数 A-25
/etc/group ファイル 5-4
/etc/motd..... 1-5
/etc/passwd ファイル..... 5-2, 5-4, 7-15
exit コマンド 1-7, 5-20, 5-23
exit コマンド(メール)..... F-1
export コマンド 7-22
 Bourne シェル 8-19
 Korn シェル 8-34
 POSIX シェル..... 8-34
ex ライン・エディタ A-20

F

fc コマンド (**Korn** シェル)..... 8-34
fg コマンド 6-16
fg コマンド (**C** シェル)..... 8-13
file コマンド 3-34
file コマンド(メール) F-1
find コマンド 6-9
finger コマンド..... 10-3

folders コマンド
 MH 11-24
folders コマンド (MH) 11-24
folders コマンド(メール)..... F-1
folder コマンド
 MH 11-24
folder コマンド (MH) 11-24
folder コマンド(メール) 11-15, F-1
 現在のフォルダの確認 11-15
folder 変数(メール)..... 11-20
forw コマンド (MH) 11-24
from コマンド(メール) F-1
ftp サブコマンド . 12-6, 12-8, 12-9

G

getacl コマンド..... G-10
get サブコマンド (ftp)..... 12-6
global (g) 演算子 (ed エディタ) B-17
group コマンド(メール) F-1

H

headers コマンド(メール)..... F-1
help コマンド(メール) F-1
help サブコマンド (ftp) 12-9
history コマンド
 C シェル 8-13
 Korn シェル 8-34
 POSIX シェル..... 8-34
hold コマンド(メール)..... F-1
 hold 変数(メール) F-1
hold 変数(メール)..... D-1, F-1
 hold コマンド(メール) D-1

 quit コマンドの効果 F-1
HOME 環境変数 2-9, 7-18

I

I/O 6-2
ID
 変更 5-20
ignorecase 環境変数 A-25
ignoreeof 変数(メール) D-1
 dot 変数(メール) D-1
 設定の際の注意 D-1
ignore コマンド(メール) F-1
 print および type コマンドでの効果 F-1
 上書き F-1
ignore 変数(メール) D-1
inbox フォルダ (MH プログラム) 11-21
inc コマンド (MH) 11-21, 11-24
insert (i) サブコマンド (ed エディタ) B-24
insert text (i) コマンド (vi テキスト・エディタ)..... A-11
insert text (I) コマンド (vi テキスト・エディタ)..... A-13
ISO コード・セット..... C-2
i ノード番号..... 3-19
 シンボリック・リンク 3-20
 ファイルの移動と名前の変更 3-26

J

jobs コマンド 6-10, 6-12

C シェル 8-13
Korn シェル 8-34
POSIX シェル 8-34

K

keepsave 変数(メール) D-1
keep 変数(メール) D-1
kill コマンド 6-14
Korn シェル 7-1, 8-1, 8-19
 .kshrc ログイン・スクリプト 7-20, 8-21
 .logout スクリプト 7-29
 .profile ログイン・スクリプト 7-19, 8-20
 インライン編集 7-5
 エラーのリダイレクション 6-5
 組み込みコマンド 8-33
 組み込み変数 8-31
 コマンド行 8-26
 コマンド行の編集 8-26
 コマンド・ヒストリ 7-4, 8-24
 パターン照合 2-16
 ファイル名補完 7-4, 8-28
 別名 7-4, 8-29
 変数値のクリア 7-26
 変数の設定 7-22
 メタキャラクタ 8-23
 ログイン・スクリプト 8-19, 8-21
 .kshrc ログイン・スクリプト . 7-20, 8-21

L

LANG 環境変数 7-18, C-8

LC_ALL 環境変数 7-18, C-9
 設定の変更 C-11
LC_COLLATE 環境変数 . 7-18, C-9
LC_CTYPE 環境変数 7-18, C-9
LC_MESSAGES 環境変数 7-18, C-9
LC_MONETARY 環境変数 ... 7-18, C-9
LC_NUMERIC 環境変数 7-18, C-9
LC_TIME 環境変数 7-18, C-9
lcd サブコマンド (**ftp**) 12-8
ln コマンド 3-16, 3-18
locale コマンド C-6
login コマンド 1-2
 .login ファイル
 System V 環境の変更 9-2
 .login ログイン・スクリプト
 MH プログラムのための変更 11-21
 System V 環境 9-2
LOGNAME 環境変数 7-18
logout コマンド (C シェル) 8-13
 .logout スクリプト 7-29
lpq コマンド 3-14
lpr コマンド 3-12
lpstat コマンド 3-15
ls コマンド ... 2-14, 3-2, 3-21, 5-7
 ls サブコマンド (**ftp**) 12-8

M

Mail 11-1
 .mailrc ファイル 11-14
 カスタマイズの変更(メール) 11-18
mailx 11-1
 詳細モード, 例 D-6e

mailx コマンド
 他のメール環境に 11-8
MAIL 環境変数 7-18
mail コマンド(メール) F-1
man コマンド 1-8, 1-17
map コマンド (**vi** テキスト・エディタ) A-25
mark コマンド (**MH**) 11-24
mbox コマンド(メール) F-1
mbox ファイル 11-11, D-1
 quit コマンドでの更新 F-1
 メッセージの移動を防ぐ ... 11-11
 メッセージを保存しない場合 . F-1
metoo 変数(メール) D-1
mget サブコマンド (**ftp**) 12-6
mhl コマンド (**MH**) 11-24
mhmail コマンド (**MH**) 11-24
MH 内に入れる 11-21
MH プログラム 11-21, 11-25
 exploding digests in 11-24
 機能の調整 11-25
 シェル・プロンプトから使用するコマンド 11-21
 送信 11-24
 内容による選択 11-24
 パスの変更 11-21
 ファイリング 11-24
 ファイルの圧縮 11-24
 フォルダの削除 11-24
 フォルダの使用 11-21
 フォルダの設定またはリスト 11-24
 フォルダの選択 11-22
 フォルダのリスト 11-24
 別名のリスト 11-24
 ホストへのインストールの確認 11-21
 メッセージによるリスト ... 11-24
 メッセージの応答 11-24
 メッセージの検索 11-24
 メッセージの削除 .. 11-22, 11-24
 メッセージの作成 11-24
 メッセージの受信の報告 ... 11-24
 メッセージの消去 11-24
 メッセージの送信 11-24
 メッセージの挿入 11-24
 メッセージのソート 11-24
 メッセージの注釈付け 11-24
 メッセージの転送 11-24
 メッセージの配信 11-24
 メッセージのマーク付け ... 11-24
 メッセージの読み込み 11-24
 メッセージのリスト 11-24
 読み込み 11-24
mkdir コマンド 3-2, 4-2
mkdir サブコマンド (**ftp**) 12-8
more コマンド 3-6, 3-7, 3-9, 3-19e
 メッセージ表示用 11-10
 メールでの実行 D-1
 メールによる使用
 メッセージ表示用 11-21
move (**m**) サブコマンド (**ed** エディタ) B-21
mput サブコマンド (**ftp**) 12-6
msgcheck コマンド (**MH**) 11-24
mv コマンド 3-26, 3-27, 4-10

N

next コマンド (MH) 11-24
next コマンド(メール)..... F-1
noheader 変数(メール)..... D-1
noignore 環境変数 A-25
nonumber 環境変数 A-25
nosave 変数(メール)..... D-1
noshowmatch 環境変数 A-25
number 環境変数..... A-25

O

open line (o) コマンド (**vi** エディタ) A-11
open previous line (O) コマンド (**vi** エディタ)..... A-12
open サブコマンド (**ftp**)..... 12-6

P

packf コマンド (MH)..... 11-24
page コマンド 3-7
passwd コマンド 1-9, 1-15
PATH 環境変数 7-18
System V 9-2
PATH 変数..... 7-27
pg コマンド 3-6
pick コマンド (MH) 11-24
PID 番号..... 6-8
POSIX シェル 7-1
インライン編集 7-5
エラーのリダイレクション.... 6-5
組み込みコマンド..... 8-33
組み込み変数 8-31
コマンド行の編集 8-26

パターン照合 2-16
ファイル名補完 7-4, 8-28
別名 7-4, 8-29
変数値のクリア 7-26
メタキャラクタ 8-23
POSIX ロケール C-8
preserve コマンド(メール)..... F-1
prev コマンド (MH) 11-24
print (p) サブコマンド (**ed** エディタ) B-4
print コマンド(メール) F-1
Print コマンド(メール)
print コマンドとの相違 F-1
profile システム・ログイン・スクリプト 7-19
.profile ファイル
System V 環境 9-2
.profile ログイン・スクリプト 7-19, 8-14, 8-20
MH プログラムのための変更 11-21
System V 環境 9-2
prompter コマンド (MH) 11-24
prompt 変数(メール) D-1
pr コマンド 3-8
ps コマンド 6-10
put サブコマンド (**ftp**)..... 12-6
pwd コマンド..... 2-9
Bourne シェル 8-19
Korn シェル 8-34
POSIX シェル..... 8-34
pwd サブコマンド (**ftp**) 12-8

Q

quiet 変数(メール)..... D-1

quit (q) コマンド (**vi** エディタ) A-5, A-17
quit (q) サブコマンド (**ed** エディタ) B-6
quit コマンド(メール) F-1
quit サブコマンド (**ftp**)..... 12-6
quit サブコマンド (**telnet**) 13-6

R

r (read) 許可..... 5-11
rcvstore コマンド (**MH**)..... 11-24
read (r) サブコマンド (**ed** エディタ) B-7, B-9
record 変数(メール) 11-21, D-1
 ファイルへの直接送信 D-1
recv サブコマンド (**ftp**) 12-6
refile コマンド (**MH**) 11-24
rehash コマンド (**C** シェル).... 8-13
rename サブコマンド (**ftp**) 12-8
repeat コマンド (**C** シェル) 8-13
reply コマンド (**MH**) 11-24
reply コマンド(メール)..... F-1
Reply コマンド(メール)
 reply コマンドとの相違 F-1
respond コマンド(メール) F-1
Respond コマンド(メール)..... F-1
rlogin コマンド..... 13-1
rmdir コマンド 4-11, 4-13
rmdir サブコマンド (**ftp**)..... 12-8
rmf コマンド (**MH**) 11-24
rmm コマンド (**MH**) 11-24
rm コマンド 3-20, 3-22, 3-31
runique サブコマンド (**ftp**).... 12-6

ruptime コマンド 10-6
r コマンド(メール) 11-12
R コマンド(メール) 11-12

S

s (set) 許可..... 5-11
save コマンド(メール)..... F-1
scan コマンド (**MH**) 11-24
screen 変数(メール)
 headers コマンドでの使用 ... D-1, F-1
 z コマンドでの使用 F-1
sendmail 変数(メール) D-1
send コマンド (**MH**) 11-24
send サブコマンド (**ftp**)..... 12-6
setenv コマンド 7-24
setenv コマンド (**C** シェル) 8-13
set コマンド
 Bourne シェル 8-19
 C シェル 8-13
 Korn シェル 8-34
 POSIX シェル..... 8-34
set コマンド(メール)..... F-1
SHELL 環境変数..... 7-18
SHELL 変数(メール) D-1
showmatch 環境変数..... A-25
show コマンド (**MH**) 11-24
sortm コマンド (**MH**)..... 11-24
sort コマンド 3-30
source コマンド
 C シェルにおいて 8-13
 シェル・オプションの呼び出し
 MH 用..... 11-21

source コマンド(メール) F-1
status サブコマンド (**ftp**) 12-9
status サブコマンド (**telnet**)... 13-6
stderr..... 6-2
stdin 6-2
stdout 6-2
substitute (s) サブコマンド (**ed** エ
 ディタ) B-15
sunique サブコマンド (**ftp**).... 12-6
su コマンド 5-20
System V..... 9-1
 コマンドの概略 9-5
System V 環境
 .cshrc ファイルの更新..... 9-2
 .login ファイルの更新 9-2
 PATH の設定 9-2
 .profile ファイルの更新 9-2
 アクセス 9-2
 ロケーション 9-3

T

tabstop 環境変数 A-25
talk コマンド
 CRT 画面 11-28
telnet コマンド 13-3
 使用方法 13-4
TERM 環境変数..... 7-18
times コマンド
 Bourne シェル 8-19
 Korn シェル 8-34
 POSIX シェル..... 8-34
time コマンド (**C** シェル)..... 8-13
tip コマンド (**UUCP**)
 オプションの使用

 リモート・ホストへの接続 14-10
 ローカル・コマンドの使用. 14-11,
 14-14
To:リスト(メール)
 名前の追加..... E-1
toplines 変数(メール) D-1
top コマンド(メール) F-1
 toplines 変数(メール) F-1
touch コマンド 3-33
transfer (t) サブコマンド (**ed** エディ
 タ) B-26
trap コマンド
 Bourne シェル 8-19
 Korn シェル 8-34
 POSIX シェル..... 8-34
type コマンド(メール) F-1
Type コマンド(メール)
 type コマンドとの相違 F-1
TZ 環境変数..... 7-18

U

umask
 許可の組み合わせ 5-16
umask コマンド 5-15
 Bourne シェル 8-19
 C シェル 8-4
 Korn シェル 8-34
 POSIX シェル..... 8-34
unalias コマンド
 C シェル 8-13
 Korn シェル 8-34
 POSIX シェル..... 8-34
undelete コマンド(メール)..... F-1

undo (u) コマンド (vi エディタ) A-16

UNIX

 大文字と小文字の区別 2-7

UNIX 間コピー・プログラム

(UUCP) 14-1

unsetenv コマンド 7-26

unsetenv コマンド (C シェル) . 8-13

unset コマンド 7-26

 Bourne シェル 8-19

 C シェル 8-13

 Korn シェル 8-34

 POSIX シェル 8-34

unset コマンド (メール) F-1

/usr/spool/uucppublic ディレクトリ

(UUCP) 14-2

UUCP 14-1

 ファイル・アクセスのローカル・ホスト制御 14-28

 モニタ 14-37

 ローカル・コマンド 14-7

uucp コマンド 14-1

UUCP ジョブの終了

 uustat コマンドの使用 14-31

UUCP パス名規約 (UUCP) 14-1

UUCP ユーティティのログ表

 示 14-36

uulog コマンド (UUCP) 14-36

uumonitor コマンド (UUCP) 14-37

uustat コマンド (UUCP) 14-31

 UUCP ジョブの終了 14-31

uuto コマンド

 uupick とともに使用 14-28

 ファイルのコピー , ローカル・ホスト制御 , オプション 14-28

uux コマンド 14-21

 リモート・ホストからの実行 (UUCP) 14-21

uux コマンド (UUCP)

 オプション , リモート・コマンド実行のための使用 14-16

V

verbose サブコマンド (ftp) 12-9

verbose 変数 (メール) D-1

vi ... 11-4, A-2, A-12, A-14, A-15

visual コマンド (メール) F-1

VISUAL 変数 (メール) D-1, E-1

 EDITOR 変数 (メール) D-1

 visual コマンドでの使用 F-1

vi エディタ

 0 カーソル移動コマンド A-9

 b カーソル移動コマンド A-8

 Ctrl/B カーソル移動コマンド . A-9

 Ctrl/D カーソル移動コマンド . A-9

 Ctrl/F カーソル移動コマンド . A-9

 Ctrl/U カーソル移動コマンド . A-9

 cw コマンド A-15

 c コマンド A-15

 dw コマンド A-14

 ex 行編集コマンド A-20

 h カーソル移動コマンド A-7

 j カーソル移動コマンド A-7

 k カーソル移動コマンド A-7

 l カーソル移動コマンド A-7

 next (n) 検索コマンド A-18

W

X

- x (**execute**) 許可 5-11
- x コマンド(メール)..... F-1

Y

- yppasswd コマンド 1-10

Z

- z コマンド(メール) F-1
- z サブコマンド (**telnet**)..... 13-6

あ

- アクセス
 - ファイル 5-1
 - ヘルプ 1-16
- アクセス許可
 - ACL..... G-1
- アクセス制御リスト
 - (ACL を参照)
- アクティブなプロセス..... 6-18e
- アジア言語
 - サポートするコード・セット . C-2
- アットマーク (@)
 - メールでエコーされた Ctrl/c .. D-1
- アットマーク (@) , ロケール名の拡張子..... C-10
- アンパサンド (&) 演算子
 - バックグラウンド・プロセス . 6-8

い

- 一時停止
 - プロセス..... 6-15
- 一重引用符 7-14
- 移動
 - ファイル..... 3-25, 3-27
- 印刷
 - オプション..... 3-13
 - 作業ディレクトリ (pwd) 2-9
 - 省略時のプリンタ 3-12
 - 特定のプリンタ 3-12
 - ファイル..... 3-12
 - リファレンス・ページ 1-17
- 引用..... 7-13
 - 一重引用符..... 7-14
 - 二重 7-15
 - 二重引用符 (" ") 7-15
 - バックスラッシュ (\) 7-14
 - 変数名表示のため 7-14
- 引用符
 - 一重 7-14
- インライン編集
 - Korn または POSIX シェル.... 7-5

う

- ウイルス 5-25
- ウィンドウ・マネージャ 1-16
- 打ち切り
 - メール・メッセージ..... E-1

え

エスケープ・キー
 vi での使用 A-4
エスケープ・コマンド(メール) .. E-1
 リストの取得 E-1
エスケープ文字
 チルダ (~)(メール) D-1, E-1
 メールでの指定 D-1
 メール・メッセージ..... E-1
エディタ 2-1
 ed B-1
 vi..... A-1
 メール用に指定 D-1, E-1
エラー
 出力のファイルへのリダイレク
 ト 6-5
& 演算子 6-8
&& 演算子 7-9
|| 演算子 7-9
エンハンスド・セキュリティ・システ
ム..... 1-2

お

応答
 メール・メッセージ..... 11-11
大文字..... 2-7
大文字と小文字の区別..... 2-7
オプション
 コマンド..... 1-9
オプション(メール)
 .mailrc で設定したオプションの解
 除 F-1
 解除 F-1
 対話式での指定 D-1

変数 D-1
オペレーティング・システムのプロセ
ス..... 6-1
親ディレクトリ 2-11
オンライン・ドキュメント 1-17

か

開始
 vi エディタ A-5
解除
 オプション
 .mailrc ファイルで設定 F-1
書き込み許可..... 5-6, 5-11
拡張子..... 2-8
カスタマイズ
 ログイン・スクリプト 7-21
カッコ
 コマンドのグループ化のための使
 用 7-12
環境のカスタマイズ
 メール・プログラムのカスタマイ
 ズ 11-18
 メール変数の設定..... 11-20
環境変数 7-16, 7-22
 HOME..... 2-9, 7-18
 LANG 7-18
 LC_ALL 7-18
 LC_COLLATE 7-18
 LC_CTYPE..... 7-18
 LC_MESSAGES 7-18
 LC_MONETARY..... 7-18
 LC_NUMERIC..... 7-18
 LC_TIME..... 7-18
 LOGNAME..... 7-18

MAIL 7-18
 PATH 7-18
 SHELL 7-18
 TERM 7-18
 TZ 7-18
 ロケール
 LANG C-8
 ロケール・カテゴリに対応 C-9
 完全パス名 2-12
 感嘆符 (!)
 エスケープ・コマンド(メール) E-1
 シェル・コマンドでの実行 F-1
 シェル・コマンドの実行 E-1
 管理
 ディレクトリ 4-1

き

起動
 ed エディタ B-2
 疑問符 (?)
 エスケープ・コマンド(メール) E-1
 メール・プロンプト 11-9
 キャンセル
 コマンド 1-9
 キュー
 プリンタ 3-12
 行のコピー
 ed エディタ B-26
 今日のメッセージ (message of the
 day) 1-5
 許可 5-1
 2 進数 5-15
 8 進数での設定 5-13

umask を使用した指定 5-17
 組み合わせ 5-14
 絶対許可の設定 5-13
 ディレクトリ 5-6
 ファイルとディレクトリの設定 5-9
 読み取り 5-11
 許可の組み合わせ
 umask 5-16
 切り替え
 ルート・ユーザ 5-22

く

組み込み
 コマンド 8-12, 8-18, 8-33
 変数 8-11, 8-17, 8-31
 グラフィカル・ユーザ・インタフェー
 ス 1-16
 クリア
 変数値 7-26
 グループ 5-4
 グループ化
 カッコの使用 7-12
 中カッコの使用 7-13
 グループ・ファイル 5-2
 グローバル置換
 vi テキスト・エディタ A-21

け

言語
 ロケール C-2
 現在のディレクトリ 2-9
 現在のメッセージ 11-10

検索

vi エディタ

- コンテキスト A-17
- テキスト A-17
- コマンド名..... 1-19

こ

国際化..... C-1

- LANG 環境変数..... 7-18, C-8
- LC_ALL 環境変数..... 7-18, C-9
- LC_COLLATE 環境変数 7-18, C-9
- LC_CTYPE 環境変数... 7-18, C-9
- LC_MESSAGES 環境変数 .. 7-18, C-9
- LC_MONETARY 環境変数 .. 7-18, C-9
- LC_NUMERIC 環境変数 7-18, C-9
- LC_TIME 環境変数 7-18, C-9
- 言語 C-2
- 照合 C-3
- 数字と通貨のフォーマット.... C-5
- パターン照合 2-16
- 日付/時刻フォーマット..... C-4
- メッセージ..... C-6
- ロケール..... C-2

コピー

- ed エディタ B-26
- ファイル名の変更..... 3-25
- ファイルを1つのディレクトリから別のディレクトリへ 3-23
- ファイルを別のディレクトリ 3-24
- コピー(メール) E-1

プロンプトの表示..... 11-20

コマンド

- alias 8-13, 8-34
- apropos..... 1-17, 1-19
- bg..... 6-16, 8-13
- cat 3-6, 3-8
- cd 3-2, 4-4, 8-19, 8-34
- chgrp 5-23
- chmod 5-9
- chown..... 5-23
- cp 3-22, 3-23
- date..... 1-8
- df 3-17
- diff..... 3-28
- echo..... 7-25, 8-13, 8-19, 8-34
- exit 1-7, 5-20
- export..... 7-22, 8-19, 8-34
- fc..... 8-34
- fg 6-16, 8-13
- file 3-34
- find 6-9
- history..... 8-13, 8-34
- jobs 6-10, 6-12, 8-13, 8-34
- kill..... 6-14
- ln 3-16, 3-18
- login 1-2
- logout..... 8-13
- lpq 3-14, 3-15
- lpr 3-12, 3-13
- lprm (remove from print queue) 3-15
- lpstat 3-15
- ls..... 2-14, 3-2, 3-3, 3-21, 3-27, 5-7, 5-11
- man..... 1-8, 1-18

mkdir	3-2, 4-2	シェルへの組み込み.....	8-12,
more	3-6, 3-7, 3-9, 3-19e	8-18, 8-33	
mv	3-25, 3-27, 4-10	次行にわたる	11-19
page	3-8	実行の中断.....	1-9
passwd	1-10, 1-15	出力のリダイレクト.....	3-31
pg.....	3-6, 3-7e	順次に実行.....	7-8
pr	3-8	使用	1-7
ps	6-10	条件付きで実行	7-9
pwd.....	2-9, 8-19, 8-34	タイプ・ミスの修正.....	1-8
rehash.....	8-13	ドキュメント	1-17
repeat	8-13	特殊文字の使用	7-13
rm	3-20, 3-22, 3-31, 4-14	パイプによる接続.....	7-10
rmdir	4-11, 4-13	引数	1-8
set	8-13, 8-19, 8-34	複数のコマンドの実行	7-8
setenv	7-24, 8-13	フラグ	1-9
sort	3-30	コマンド行の編集	8-26
source	8-13	コマンドのグループ化	
su.....	5-20	カッコの使用	7-12
time.....	8-13	中カッコの使用	7-13
times	8-19, 8-34	コマンドの接続	
touch	3-33	パイプによる	7-10
trap.....	8-19, 8-34	コマンド引数	
umask	5-15, 8-19, 8-34	2ワード以上.....	1-19
unalias	8-13, 8-34	コマンド・ヒストリ	7-4
unset	7-26, 8-13, 8-19, 8-34	C シェル.....	8-7
unsetenv	7-26, 8-13	Korn シェル	8-24
vi.....	2-3	コマンド名	
w.....	6-18	思い出せない場合.....	1-19
wc.....	6-3	コマンド名の検索	1-19
who	6-17	コマンド・モード (vi エディタ) .	A-7
whoami.....	5-20	小文字.....	2-7
オプション.....	1-9	コロン	
コマンド名を思い出せない...	1-19	vi テキスト・エディタでの使	
		用	A-4, A-20

コロン (:)
 vi テキスト・エディタでの使用 2-3
 エスケープ・コマンド(メール) E-1
 メッセージ送信中のコマンドの開
 始 E-1
 コンピュータ・ウイルス 5-25
 コード・セット
 8 ビット C-2
 ASCII..... C-2
 ISO C-2
 アジア言語のサポート C-2
 ロケール C-2

さ

再開
 プロセス 6-15
 再実行
 コマンド 8-8
 再スタート
 プロセス 6-15
 再設定
 環境変数 7-24
 最大長
 ファイル名 2-8
 作業ディレクトリ 2-9
 削除
 clearing a line (D) コマンド . A-15
 ed エディタ
 特定の行 B-20
 複数行 B-21
 文字 B-18
 vi エディタ
 特定行 A-15
 複数行 A-15

一度に 1 文字 A-15
 キューに登録したプリント要
 求 3-15
 現在の行 B-20
 絶対許可 5-13e
 ディレクトリ 4-13
 ファイル 3-32, 3-33
 ファイルのリンク 3-20
 プリント・キューからジョブ 3-15
 メール・メッセージ F-1
 リンク 3-20
 削除キー 1-8
 タイプ・ミスの修正 1-8
 作成
 1 つのファイルに対する複数のファ
 イル名 3-23
 ed エディタ B-2
 vi エディタ A-5
 vi によるテキスト・ファイル . 2-2
 空のファイル 3-33
 シェル・スクリプト 7-31
 シェル・プロシージャ (例) ... 7-31
 シンボリック・リンク 3-18
 ディレクトリ 3-2
 ログアウト・スクリプト 7-28
 作成するメッセージの表示 E-1
 サブコマンド (**ftp**) 12-6, 12-8, 12-9
 !サブコマンド (**ftp**) 12-9
 ?サブコマンド (**ftp**) 12-9
 サブシェル 7-2, 7-12
 サブディレクトリ 2-8
 参照
 変数 7-23

し

シェル

- SHELL 変数(メール)..... F-1
- source コマンドの使用 11-21
- 一時的な変更 7-6
- 永久的な変更 7-7
- カスタム変数の定義..... 7-22
- 環境に省略時の値を割り当てる 7-16
- 環境変数の設定 7-24
- 機能 8-1
- 機能の比較..... 8-2
- 組み込みコマンド 8-12, 8-18, 8-33
- 組み込み変数 .. 8-11, 8-17, 8-31
- コマンド・インタプリタ 1-7
- コマンド行..... 8-26
- コマンド行の編集..... 8-26
- コマンドの実行 F-1
 - メール..... E-1
- コマンドの指定(メール) D-1
- コマンド・ヒストリ.... 8-7, 8-24
- サブシェル..... 7-2
- 省略時のシェル 7-1
- 省略時の実行シェル..... 7-32
- 省略時のプロンプト..... 7-6
- スクリプト..... 7-30
- 特殊文字 2-7, 7-13
- ファイルのメール..... 11-6
- ファイル名補完 8-9, 8-28
- プログラム..... 1-4
- プロンプト..... 1-3
- 別名 8-9, 8-29
- 変更 7-5
- メタキャラクタ .. 8-5, 8-15, 8-23
- メールからの呼び出し F-1
- ユーザの制限 7-5
- ルート・ユーザのための定義 5-22
- ログイン・スクリプト 8-2, 8-14, 8-19, 8-21
- シェル機能の比較 8-2
- シェル・コマンド(メール) F-1
- シェル・スクリプト 7-30
 - System V との互換性 9-5
- シェルの呼び出し
 - メールから..... F-1
- シェル変数 7-18
 - カスタムの定義 7-22
- 時刻フォーマット
 - 句読点 C-5
- システム管理者の責任..... 5-21
- システム・プロンプト..... 7-6
- システム・メールボックス
 - 空でも削除しない..... D-1
- 実行
 - コマンドを順次に..... 7-8
 - コマンドを条件付きで 7-9
 - シェル・プロシージャ 7-31
 - バックグラウンド・プロセス . 6-7
 - フォアグラウンド・プロセス . 6-8
- 実行許可 5-6, 5-11
- 実行シェル 7-32
- 指定
 - print コマンドでのプリンタの指定 3-12
 - 省略時の実行シェル..... 7-32

自動ログイン..... 1-2
従来のファイル保護メカニズム
 許可ビット..... G-3
 グループ..... G-3
 所有者..... G-3
終了
 ジョブまたはプロセス..... 6-14
 メール・メッセージ..... 11-3
 ローカル・メッセージ..... 11-27
受信 (**UUCP**)
 ファイル..... 14-25
受信者リスト(メール)..... E-1
使用
 Backspace キー, タイプ・ミス修正
 のため..... 1-8
 コマンド..... 1-7
 削除キー, タイプ・ミス修正のた
 め..... 1-8
 単一の文字を引用するためのバック
 スラッシュ..... 7-14
 パイプとフィルタの使用による複数
 のコマンドの実行..... 7-8
 パイプライン..... 7-10
 パス名でのチルダ..... 2-13
 パス名のチルダ..... 2-13
 ファイル名でのワイルドカー
 ド..... 2-14
 フィルタ..... 7-10
消去
 ファイル
 防止..... 3-23
条件付きコマンド
 実行..... 7-9
照合..... C-3
状態情報 (**UUCP**)..... 14-31

省略時
 シェルの値..... 7-16
 シェルのプロンプト..... 7-6
省略時の値の割り当て
 シェル..... 7-16
省略時の設定
 umask による許可の設定..... 5-15
 許可コード..... 5-9, 5-15
省略時のプリンタ..... 3-12
省略時のユーザ・マスク
 (**umask**)..... 5-20
所有者
 ファイルおよびディレクトリの所有
 者の変更..... 5-23
シンボリック・リンク..... 3-17
 i ノード番号..... 3-20

す

スイッチ
 ルート・ユーザ..... 5-22
数字と通貨のフォーマット..... C-5
スーパーユーザ..... 5-20
スーパーユーザ特権..... 5-6
 タスク..... 5-21

せ

制御
 アクセス..... 5-1
制限
 パスワード..... 1-15
 ファイル・アクセス..... 5-17
制限付き **Bourne** シェル... 7-1, 7-5
セキュリティ
 拡張システム..... 1-2

拒否されたログイン.....	1-5
グループ.....	5-2
考慮事項.....	5-24
制限付き Bourne シェル	7-5
ユーザ	5-2
セキュリティ・ファイル	5-2
接続の終了 (ローカル通信) ...	11-27
絶対許可	5-13
削除	5-13e
絶対パス名	2-12
設定	
8 進数を使用した許可.....	5-13
C シェルおよびKorn または POSIX シェルでの別名.....	7-4
C シェルにおける環境変数 ...	7-24
環境変数.....	7-16, 7-22
省略時のファイル許可	5-15
すべてのシェルにおける環境変 数	7-24
絶対許可	5-13
ファイル許可とディレクトリ許 可	5-9
ファイル保護	5-5
ユーザ・マスク	5-15
セミコロン	
コマンドの順次実行.....	7-8

そ

送信	
メール・メッセージ... 11-2, 11-3	
リモート・ホストへの転送 (UUCP)	14-16
ローカル・メッセージ	11-25

送信 (UUCP)	
ファイル.....	14-25
送信中のメール・メッセージの編 集.....	E-1
送信用メール・メッセージに挿入する メッセージ.....	E-1
送信用メール・メッセージに読み込む メッセージ.....	E-1
相対パス名.....	2-12
ソフト・リンク	3-17
ソート	
ファイルの内容	3-30
ソート規則	C-3

た

タイプ	
プログラム.....	6-1
タイプ・ミス	
vi エディタ	A-12
タイプ・ミス, 修正	
ed エディタ	B-3
タイプ・ミスの修正	
vi エディタ	A-12
コマンド.....	1-8
ログイン時.....	1-3
タスク	
ルート・ユーザによって実行され る	5-21
縦線 ()	
エスケープ・コマンド(メール) E-1	
コマンドへのメール・メッセージの 引き渡し	E-1

ち

置換

ed エディタ B-15

vi エディタ A-21

グローバル, vi テキスト・エディ
タ A-21

パラメータ 7-25

中カッコ

コマンドをグループ化するための使
用 7-13

中間ホスト (UUCP)

ファイル転送で使用 14-16

中止

コマンド 1-9

中止 (取り消し)

ジョブまたはプロセス 6-14

チルダ

パス名での表記 2-13

チルダ (~) 2-13

テキスト・ファイル内 2-3

チルダ (~)(メール)

エスケープ文字 E-1

つ

通貨フォーマット

小数桁の表現 C-6

ツリー構造 2-10

ツリー構造 (ファイル・システム) 2-6

て

定義

カスタム・シェル変数 7-22

ユーザ環境 7-15

ログイン・アカウント 5-2

停止

Ctrl/C によるプロセスの停止 6-13

停止と再スタート

プロセス 6-15

ディスク・パーティション 3-17

訂正, タイプ・ミス

ed エディタ B-3

ディレクトリ 2-8

親 2-11

空ディレクトリの削除 4-12

管理 4-1

許可 5-6

許可の変更 5-12

現在のディレクトリの削除... 4-13

現在のディレクトリの表示... 2-9

コピー 4-9

削除 4-11

作成 4-2

所有者の変更 5-23

探索パス 7-27

ツリー構造 2-10

定義 2-1

内容のリスト 3-2, 3-3

パス 2-11

表示 4-8

複数のディレクトリの削除... 4-13

別のディレクトリからのファイルの

コピー 3-23

変更 3-2, 4-4

ホーム 1-4

名称の変更 4-10

メールでの変更 F-1

ルート 2-11

ディレクトリ許可 5-12

ディレクトリ許可の設定 5-9
ディレクトリの移動 4-10
ディレクトリの構造 2-10
ディレクトリのコピー 4-9
ディレクトリの削除 4-11
 現在のディレクトリ 4-13
ディレクトリの作成 4-2
ディレクトリの内容のリスト ... 3-2,
 3-3
ディレクトリの表示 4-8
ディレクトリ名の変更... 3-26, 4-10
テキスト・エディタ 2-1
 vi A-1
 概要 2-1
テキストの移動
 ed エディタ B-21
 vi エディタ A-18
テキストの探索
 ed エディタ B-13
テキスト・ファイルの作成
 vi エディタ A-5
デバイス名
 cu コマンドによる指定
 (UUCP) 14-6
テリトリ
 ロケール C-2
転送
 メール・メッセージ
 mbox フォルダへ F-1
転送状態情報 (UUCP) 14-31
電話番号
 cu コマンドによる指定
 (UUCP) 14-6
データベース・セキュリティ 5-2

グループ 5-4

と

特殊文字 7-13
ドット記号 (.) 2-12
取り消し
 ジョブまたはプロセス 6-14
 プリント・ジョブ 3-15

な

名前の変更
 ファイル 3-27

に

二重引用符 7-15
入力
 読み取り 6-2
入力モード (vi エディタ) A-10

は

パイプ 7-10
パイプとフィルタ
 複数のコマンドの実行 7-8
パイプライン 7-10
パス
 MH プログラム使用のため . 11-21
パス名 2-10, 2-11
 完全 2-12
 絶対 2-12
 相対 2-12
 チルダの使用 2-13

ドット記号 (.)	2-12	プロセスを入れる	6-16
メール・フォルダ・ディレクト		バックグラウンド・プロセス	6-7
リ	D-1	状態の表示	6-11
パスワード		バックスラッシュ	7-14
一回	1-9	バックスラッシュ (\)	
一般	1-5	1 つの長いコマンドを複数行に入力	
エンハンスト・セキュリティ・シス		する	11-19
テム	1-2	パブリック・ディレクトリ	
システム生成の	1-13	(UUCP)	14-2
制限	1-15	パラメータ置換	7-25
セキュリティ制限	1-14	判断	
設定	1-9	ファイル・タイプ	3-34
ネットワーク・システムにおい		ハード・リンク	3-16
て	1-10		
ファイル	5-2		
有効期限	1-10		
要求されない	1-5		
ランダムで発音可能	1-13		
ランダムな英字	1-13		
ランダムな文字	1-13		
ログインのため	1-2		
忘れた場合	1-16		
パスワード			
期限切れ	1-4		
パスワードの設定			
エンハンスト・セキュリティ・シス			
テムにおいて	1-2		
パスワード・ファイル	5-2		
パターン照合			
国際化	2-16		
使用できる文字のリスト	2-15		
ファイル許可の変更での使用	5-12		
ファイル指定	2-14		
複数のファイルの削除	3-32		
バックグラウンド			

ひ

比較	
ファイル	3-28
引数	
コマンド	1-8
ヒストリ	
最近使用したコマンド .	8-7, 8-24
日付/時刻フォーマット	C-4
表示	
C シェルにおける変数の値 ...	7-26
アクティブなプロセス	6-18e
オンライン・リファレンス・ペー	
ジ	1-17
現在のディレクトリ名	2-9
コマンドの状態	6-10
すべてのファイルのパス名	6-9
ディスク・パーティション ...	3-17
ディレクトリ許可	5-7
ディレクトリの内容	3-2
非アクティブなユーザ	6-17

ファイル.....	3-6, 3-8	/etc/group.....	5-4
ファイル許可	5-7	/etc/passwd.....	5-2
ファイル・タイプ.....	3-34	i ノード番号	3-19
ファイルの i ノード番号	3-20	アクセスの制限	5-17
ファイルをフォーマットしない で	3-6	アクセスするための ID の変更	5-20
複数のファイルを同時に	3-7	移動	3-25, 3-27
プリント・キュー状態	3-14	印刷	3-12
プロセス情報	6-16	エラーのリダイレクト	6-5
プロセスの状態	6-11e	空のファイルの作成.....	3-33
変数値	7-25	記述子	6-5
変数名	7-14	許可の変更.....	5-11
ユーザ ID.....	5-21e	コピー	3-22, 3-23
ユーザ名	5-20	コピー (UUCP).....	14-26
ログインしているユーザ	1-18	削除	3-31
ログイン・ユーザの確認	6-17	シェルからのメール.....	11-6
標準エラー	6-2, 6-5	受信 (UUCP)	14-25
標準出力	6-2	出力のリダイレクト.....	6-4
標準入力	6-2	消去を防ぐための noclobber 変 数	3-23
フィルタリング	7-10	すべてのファイルのパス名の表 示	6-9
表題(メール)		セキュリティ	5-2
~s エスケープでの指定.....	E-1	セキュリティに関する考慮事 項	5-24
入力	D-1	絶対許可の設定	5-13
プロンプトの表示.....	11-20	説明	2-1
メール・メッセージのための入 力	11-3, 11-20	送信 (UUCP)	14-16, 14-25
ピリオド		タイプの判断	3-34
メール・メッセージの終了....	D-1	定義	2-6
		内容のソート	3-30
		名前の変更.....	3-25
		入力の読み取り	6-3
		比較	3-28

ふ

ファイル	
2 つのファイル間の相違の比 較	3-28

表示	3-6	ファイルの内容のソート	3-30
ファイル名で禁止されている文 字	2-7	ファイルの比較	3-28
ファイル名での使用が制限されてい る文字	2-7	ファイルの命名	2-8
フォーマット	3-8	ファイルのリンク	3-16, 3-18
複数の表示	3-7	ファイル・マネージャ	2-6
他のユーザにメール	11-6	ファイル名	2-6, 2-7
保護	5-5	拡張子	2-8
命名規則	2-8	最大長	2-8
メッセージの保存	11-15	使用が制限されている文字	2-7
メール	11-6, 11-7e	ファイル名で使用が制限されている文 字	2-7
読み取り許可	5-11	ファイル名における不正な文字 ..	2-7
リンク	3-16	ファイル名における文字数の制限	2-7
ワイルドカードの使用	2-14	ファイル名の変更	3-26
ファイル・アクセス	5-1	ファイル名補完	7-4, 8-9, 8-28
ファイル記述子	6-5	フィルタ	7-10
ファイル許可とディレクトリ許可の相 違	5-6	定義	7-10
ファイル許可の設定	5-9	フィルタリング	
ファイル・システム	2-1, 2-6	標準入力	7-10
ファイル指定		フォアグラウンド	
パターン照合	2-14	プロセスを入れる	6-16
ファイル・タイプ		フォアグラウンド・プロセス	6-8
判断	3-34	フォルダ(メール)	11-14, 11-15, 11-24
ファイル通し番号	3-19	MH プログラムでの指定 ...	11-22
ファイルの移動	3-25	MH プログラムによる使用 .	11-21
ファイルのコピー	3-22	削除 (MH)	11-24
ファイルのコピー (UUCP) ...	14-26	指定	D-1
ファイルのコピー , ローカル・ホスト 制御 (UUCP)	14-28	設定	11-14
ファイルの削除	3-31	リスト (MH)	11-22, 11-24
確認プロンプトの表示	3-33	フォルダの選択(メール)	11-15
ファイルの消去		複数バイト文字	
防止	3-23	コード・セットでのサポート .	C-2
		不正な文字	

ファイル名における..... 2-7
 フラグ..... 1-9
 プラス記号 (+)
 next コマンドとして使用(メー
 ル)..... F-1
 プリンタ
 print コマンドでの指定 3-12
 省略時 3-12
 命名 3-12
 プリンタ・キュー 3-12, 3-14
 プログラム
 タイプ 6-1
 プロセス 6-1
 アクティブなプロセスの表示 6-18e
 コマンドのグループ化 7-12, 7-13
 再開 6-15
 実行ユーザの表示..... 6-17
 状態の表示..... 6-10, 6-11e
 停止 6-13
 取り消し 6-14
 パイプを通しての実行 7-12
 プロセス識別子 6-1
 プロセス識別番号 (PID) 6-8
 プロセス情報..... 6-16
 プロンプト
 ?
 メール・プログラム内..... 11-9
 シェル 7-6
 文脈探索
 ed エディタ B-13

へ

別名..... 8-9, 8-29

C シェル 8-9
 C シェルおよびKorn または POSIX
 シェルでの設定..... 7-4
 Korn シェル 8-34
 mail の使用中に定義済の別名を確認
 する 11-20
 POSIX シェル..... 8-34
 送信するメッセージにユーザ自身を
 含む D-1
 メール 11-19
 リスト (MH)..... 11-24
 リストおよび指定(メール)..... F-1
 ヘルプ..... 1-16, 1-17, 11-17
 ヘルプ・コマンド
 メールの 11-17
 変更
 copy コマンド実行中のファイルの名
 前 3-25
 ID..... 5-20
 グループ..... 5-23
 シェル 7-5
 シェルを一時的に..... 7-6
 シェルを永久的に..... 7-7
 ディレクトリ 3-2, 4-4
 ディレクトリ許可..... 5-12
 パスワード..... 1-15
 ファイルおよびディレクトリの所有
 者 5-23
 ファイル許可 5-11
 保護 5-5
 編集
 POSIX シェルでのコマンド行 8-26
 コマンド行..... 8-26

送信中のメール・メッセージ . E-1
メール・メッセージ..... 11-4
メール・メッセージ・ヘッダ . E-1
リンクされたファイル 3-16
変数
値のクリア..... 7-26
値の表示..... 7-25
参照 7-23
シェルへの組み込み..... 8-11,
8-17, 8-31
メール 11-20
変数(メール)
オプション..... D-1
変数の参照 7-25
変数名
表示 7-14

ほ

防止
ファイルの消去 3-23
補完
ファイル名..... 7-4
保護
ファイル..... 5-5
変更 5-5
保存
ed エディタ
テキスト B-5
ファイルの一部..... B-6
vi エディタ
テキスト A-17
vi でのテキスト・ファイル.... 2-4
テキスト・ファイル..... B-2
ファイルの一部 A-23

編集中のメール・メッセージ . E-1

ま

マイナス記号 (-)
メール・メッセージの選択.... F-1
マルチタスキング 6-7
マン・ページ..... 1-17

み

未知のリモート・ホストへの接続
(UUCP)
モデムを介した 14-5, 14-12

め

メタキャラクタ
Bourne シェル 8-15
C シェル..... 8-5
Korn シェル 8-23
POSIX シェル..... 8-23
メッセージ C-6
exploding digests into in MH 11-24
応答 (MH) 11-24
検索 (MH) 11-24
削除 (MH) 11-22, 11-24
作成 (MH) 11-24
指定 (MH) 11-21
受信者の報告 (MH) 11-24
送信 (MH) 11-24
挿入 (MH) 11-24
ソート (MH)..... 11-24
注釈付け (MH) 11-24
転送 (MH) 11-24

内容による選択 (MH).....	11-24	mbox ファイルへのメッセージの移動を防ぐ	11-11
配信 (MH)	11-24	mbox フォルダへのメッセージの転送	F-1
表示 (MH)	11-24	MH による設定またはリスト	11-24
ファイリング (MH)	11-24	MH プログラム	11-21, 11-25
ファイルの圧縮 (MH).....	11-24	MH プログラムによる処理 .	11-21
フォーマットされたメッセージのリスト (MH).....	11-24	quit コマンドでの終了	F-1
マーク付け (MH).....	11-24	~s エスケープでの表題の指定 .	E-1
メッセージ (ローカル通信) .	11-25	To:リストへの名前の追加	E-1
読み込み (MH)	11-24	top コマンドで表示する行数の指定	D-1
リスト (MH).....	11-24	新しいメッセージの受信の表示	11-8
リストの表示 (MH)	11-22	打ち切られたメッセージの保存の防止	D-1
メッセージ (ローカル通信) 終了		エスケープ・コマンド	
ファイルの終端記号 (EOF) 11-27		要約の取得	E-1
終了, ファイルの終端記号 (EOF)	11-27	リスト.....	E-1
送信, write コマンド	11-25	エスケープ文字の指定	D-1
長い, ファイル内の.....	12-11	エディタ.....	D-1
メッセージの終端	11-27	指定	D-1
メッセージの終了	11-27	応答 (MH)	11-24
メール		オプションの解除.....	F-1
Cc:プロンプトの表示.....	D-1	オプションを対話式で指定....	D-1
Cc:リストへの名前の追加	E-1	カスタマイズ	11-18
Ctrl/c 割り込みの無視	D-1	現在の受信メッセージの編集 .	F-1
Ctrl/d の防止.....	D-1	現在のメッセージ	
exit コマンドでの終了	F-1	定義	11-10
exploding digests in MH....	11-24	コピー	11-3
folder コマンド		コマンド	
現在のフォルダの確認... 11-15		リスト.....	F-1
f オプション			
フォルダの選択.....	11-15		
Mail の使用	11-1		

! コマンドで使用するシェルの指 定	D-1	端末画面の表示行数の指定	D-1
コマンドへのメッセージの引き渡 し	E-1	次のメッセージの表示	F-1
削除 (MH)	11-24	ディレクトリの変更	F-1
削除したメッセージの回復	F-1	デバッグ機能	D-1
作成 (MH)	11-24	特定のユーザからのメッセージのリ スト	F-1
作成するメッセージの表示	E-1	内容による選択 (MH)	11-24
シェルの呼び出し	F-1	配信 (MH)	11-24
システム・メールボックス 空でも削除しない	D-1	発信者だけへの応答 Reply コマンドの使用	F-1
システム・メールボックスでのメー ルの受信	F-1	発信メッセージの自動保管	D-1
終了	11-17	バージョン番号の非表示	D-1
受信者としてユーザ自身を含む D-1		表題行への入力 of 要求	D-1
受信者の報告 (MH)	11-24	表題の入力	11-3, 11-20
詳細モード 説明	D-1	ファイリング (MH)	11-24
詳細モードでの実行	D-1	ファイルにメッセージを保存す る	11-15
設定されているオプションの表 示	F-1	ファイルの圧縮 (MH)	11-24
全受信者への応答 reply コマンドの使用	F-1	ファイルの送信	11-6
選択したヘッダ・フィールドの非表 示	F-1	ファイルまたはフォルダの指定 D-1	
選択したメッセージの最初の部分の 表示	F-1	ファイルまたはフォルダの選択 F-1	
前方または後方へのスクロール (ファイル内)	F-1	フォルダ	11-14, D-1
送信 (MH)	11-24	作成	11-15
送信するメールのコピーを入れ る	11-21	設定	11-14
送信中のメッセージの編集	E-1	フォルダ (MH)	11-24
送信用メッセージに挿入するメッ セージ	E-1	フォルダにメッセージを保存す る	11-15
		フォルダのディレクトリの指 定	11-20
		フォルダのリスト	F-1
		フォルダのリスト (MH)	11-22

フォーマットされたメッセージのリスト (MH).....	11-24	マイナス記号 (-) の使用.....	F-1
ヘッダおよびバージョンの非表示	D-1	メッセージの送信.....	11-2, 11-3, F-1
ヘッダおよびバージョンの表示 D-1		コピー.....	11-3, 11-20
別名	11-19	メッセージの送信 (MH)	11-24
リスト (MH)	11-24	メッセージの挿入 (MH)	11-24
リストおよび指定	F-1	メッセージのソート (MH) ..	11-24
別名にメッセージを送信する	11-19	メッセージの注釈付け (MH)	11-24
別名の展開		メッセージの転送.....	11-16
定義.....	D-1	メッセージの転送 (MH)	11-24
ヘルプ情報の取得.....	11-17	メッセージの到着の通知	11-8
ヘルプの参照	F-1	メッセージの表示.....	F-1
編集	E-1	削除時.....	D-1
変数	11-20	メッセージの表示 (MH)	11-24
変数の設定.....	F-1	メッセージの編集.....	11-4
保存されたメッセージの削除の防止	D-1	メッセージの保存.....	E-1
メッセージ送信中のコマンドの実行	E-1	ファイルの使用, ヘッダなし	F-1
メッセージ・テキストの表示 . D-1		フォルダまたはファイルの使用	F-1
メッセージの dead.letter ファイル	E-1	メッセージのマーク付け (MH)	11-24
メッセージの mbox ファイルへの移動	11-11	メッセージの読み込み	11-8, 11-9
メッセージの打ち切り	E-1	番号の指定	11-10
メッセージの検索 (MH)	11-24	メッセージの読み込み (MH)	11-24
メッセージのコピー.....	F-1	メッセージのリスト (MH) ..	11-24
メッセージの削除.....	F-1	メッセージ表示の長さの指定	11-21
メッセージの削除 (MH)	11-24	メッセージ・ヘッダの編集....	E-1
メッセージの作成 (MH)	11-24	メッセージ・ヘッダのリスト	11-10, F-1
メッセージの指定 (MH)	11-21	メッセージへの応答.....	11-11
メッセージの選択		メッセージを終了するピリオドの指定	D-1

メッセージを保存する順序の指 定	D-1	save.....	F-1
メール・コマンド・ファイルの実 行	F-1	top	F-1
メールをシステム・メールボックス に保持.....	D-1	toplines 変数	F-1
読み込み (MH)	11-24	type.....	F-1
メールのコマンド		Type	
chdir.....	F-1	type コマンドとの相違	F-1
copy.....	F-1	undelete	F-1
dp.....	F-1	visual.....	F-1
dt	F-1	z	F-1
exit	F-1	メールの挿入 (MH)	11-24
file	F-1	メールの変数	
from	F-1	append	D-1
headers.....	F-1	ask.....	D-1
help.....	F-1	askcc.....	D-1
hold.....	F-1	autoprint	D-1
hold 変数(メール)	F-1	crt	D-1
ignore.....	F-1	debug.....	D-1
print および type コマンドでの効 果.....	F-1	dot	D-1
上書き	F-1	ignoreeof 変数	D-1
mail.....	F-1	設定しない際の注意	D-1
mbox.....	F-1	ピリオド, メール・メッセージの 終了.....	D-1
next.....	F-1	EDITOR.....	D-1
preserve	F-1	VISUAL 変数.....	D-1
Print		hold.....	D-1, F-1
print コマンドとの相違.....	F-1	hold コマンド(メール).....	D-1
reply.....	F-1	quit コマンドの効果	F-1
Reply		ignore.....	D-1
reply コマンドとの相違.....	F-1	ignoreeof	D-1
respond.....	F-1	dot 変数	D-1
Respond	F-1	設定の際の注意.....	D-1
		keep	D-1
		keepsave	D-1

- metoo..... D-1
- noheader D-1
- nosave..... D-1
- prompt..... D-1
- quiet..... D-1
- record
 - ファイルへの直接送信..... D-1
- screen
 - headers コマンドによる使用 D-1
- sendmail D-1
- SHELL D-1
- toplines..... D-1
- verbose D-1
- VISUAL D-1
 - EDITOR 変数 D-1
 - visual コマンドでの使用.... F-1
- メール・フォルダの作成 11-15
- メール・プログラム
 - 終了 11-17
- メール・メッセージ
 - 転送 11-16
 - 保存
 - ファイルの使用..... 11-15
 - フォルダの使用..... 11-15
- メール・メッセージに対する応答
 - 応答時の注意 11-12
 - 他の受信者を含む 11-12
- メール・メッセージ・ヘッダ
 - 編集 E-1

も

- 文字

- 引用符で囲んでリテラルにする 7-13
- 大文字と小文字 2-7
- パターン照合のリスト 2-15
- 文字数
 - ファイル名の最大長..... 2-8
- 文字数の制限
 - ファイル名における..... 2-7
- 文字の削除
 - ed エディタ B-18
 - vi エディタ A-14
- 文字マップ
 - 1対2 C-3
 - 多対 1..... C-4

ゆ

- ユーザ
 - 非アクティブなユーザ 6-17
 - ログイン・ユーザの表示 6-17
- ユーザ/グループ **ID** 設定許可... 5-11
- ユーザ **ID**
 - 確認 5-21e
- ユーザ環境 7-15
 - 省略時の値を割り当てる 7-16
- ユーザ環境の制限
 - 制限付き Bourne シェル 7-5
- ユーザ・マスク 5-15
 - 省略時の設定 5-20
 - ログイン・スクリプトでの実行 5-18
- ユーザ名 1-2

よ

要約.....	11-24
読み取り	
入力	6-2
パイプからの入力.....	7-10
番号の指定.....	11-10
メッセージ (MH).....	11-22
メール・メッセージ.....	11-8
読み取り許可.....	5-6, 5-11

り

リスト	
パターン照合文字.....	2-15
メール・メッセージ.....	11-9
リダイレクション	
エラー	
Bourne シェル.....	6-5
C シェル.....	6-6
Korn シェル	6-5
標準エラーと出力の両方	6-6
リダイレクト	
エラー	6-5
出力	3-31, 6-2, 6-4
出力, バックグラウンド・プロセス からの.....	6-9
入出力	6-2
入力	6-2
リダイレクト・エラー.....	6-5
リテラル文字.....	7-13
一重引用符の使用.....	7-14
二重引用符の使用.....	7-15
バックスラッシュの使用	7-14

リネーム

ディレクトリ	3-25
ファイル.....	3-25
ファイルとディレクトリ	3-26
リファレンス・ページ.....	1-17
リモート tip コマンドの終了 (UUCP).....	14-16
リモート・コマンド	
実行 (UUCP)	14-16
リモート・コマンド, UUCP での実 行.....	14-16
リモート接続中のローカル・ホストへ の復帰(UUCP).....	14-9, 14-16
リモートの cu 接続の終了 (UUCP)	14-9
リモートの cu 接続の切断 (UUCP)	14-9
リモートの tip コマンドの切断 (UUCP)	14-16
リモート・ファイルの転送 (UUCP)	14-16
リモート・ホスト	
コマンドの実行 (UUCP)....	14-21
リモート・ホスト上でのコマンドの実 行 (uux).....	14-21
リモート・ホストとの通信 (UUCP)	14-1
リモート・ログイン	13-1, 13-3
リンク	
削除	3-20
ソフト	3-17
ハード	3-16

ファイル..... 3-16, 3-18

る

ルート・ディレクトリ..... 2-11

ルート・ユーザ

実行されるタスク..... 5-21

なる..... 5-22

のためのシェルの定義..... 5-22

れ

レコード・ファイル(メール)..... D-1

連結ファイル..... 3-8

ろ

ログアウト..... 1-7

スクリプト..... 7-28

ログアウト・スクリプト..... 7-28

ログイン..... 1-2

エンハンスド・セキュリティ・システムにおいて..... 1-2

拒否された..... 1-5

別のユーザ・アカウント..... 5-20

ルート・ユーザとして..... 5-22

ログイン・アカウント..... 5-2

ログイン・スクリプト

Bourne シェル..... 8-14

csh.login システム・スクリプト

..... 7-19

.cshrc スクリプト..... 7-20, 8-3

C シェル..... 8-2, 8-4

Korn シェル..... 8-19, 8-21

.kshrc スクリプト..... 7-20, 8-21

.login スクリプト..... 7-20, 8-4

profile システム・スクリプト 7-19

.profile スクリプト..... 7-19,
8-14, 8-20

umask の実行..... 5-18

カスタマイズ..... 7-21

ログイン・ディレクトリ..... 2-9

ログイン日時..... 1-5

ログイン・プログラム..... 7-15

ログイン・ユーザの表示..... 1-18

ロケール..... C-2

yes/no 応答文字列..... C-6

カテゴリ..... C-9

指定形式..... C-7

使用される環境変数..... C-7

数字と通貨のフォーマット..... C-5

設定..... C-8

推奨方法..... C-8

設定されているロケールの判断 C-6

設定時の制約..... C-11

ソフトウェアへの影響..... C-3

ソート順への影響..... C-3

名前..... C-8

アットマーク (@) 修飾子.. C-10

日付/時刻フォーマットへの影

響..... C-4

メッセージへの影響..... C-6

ローカル・コマンド (UUCP).. 14-3,

14-7, 14-11, 14-14

ローカル通信機能

メッセージ , 送信 , write コマン ド	11-25	ファイル許可の変更での使用	5-12
ローカル変数.....	7-18	ファイル削除での使用	3-33
ローカル・ホストのファイル・アクセ スの制御		複数のディレクトリの削除...	4-13
(UUCP を参照)		ワイルドカード [アスタリスク (*)].....	2-14
ローカル・メッセージ		割り込み	
終了	11-27	メールでの無視	D-1

わ

ワイルドカード

Tru64 UNIX ドキュメントの購入方法

Tru64 UNIX ドキュメントのご購入については、弊社担当営業または日本ヒューレット・パッカートの各営業所/代理店にお問い合わせください。

各ドキュメント・キットの注文番号は以下のとおりです。ドキュメント・キットに含まれるマニュアルの内容については『ドキュメント概要』を参照してください。

キット名	注文番号
Tru64 UNIX Documentation CD-ROM	QA-6ADAA-G8
Tru64 UNIX Documentation Kit	QA-6ADAA-GZ
End User Documentation Kit	QA-6ADAB-GZ
- Startup Documentation Kit	QA-6ADAC-GZ
- General User Documentation Kit	QA-6ADAD-GZ
- System and Network Management Documentation Kit	QA-6ADAE-GZ
Developer's Documentation Kit	QA-6ADAF-GZ
Reference Pages Documentation Kit	QA-6ADAG-GZ
TruCluster Server Documentation Kit	QA-6BRAA-GZ
Tru64 UNIX 日本語ドキュメント・キット	QA-6ADJB-GZ
スタートアップ・ドキュメント・キット	QA-6ADJC-GZ
一般ユーザ・ドキュメント・キット	QA-6ADJD-GZ
システム/ネットワーク管理ドキュメント・キット	QA-6ADJE-GZ
プログラミング・ドキュメント・キット	QA-6ADJF-GZ
CDE 翻訳ドキュメント・キット	QA-6ADJG-GZ
TruCluster Server 日本語ドキュメント・キット	QA-05SJA-GZ
Advanced Server for UNIX 日本語ドキュメント・キット	QA-5U2JA-GZ



マニュアルに対するご意見

Tru64 UNIX

ユーザーズ・ガイド

AA-RK3LC-TE

弊社のマニュアルに関して、ご意見、ご要望、または内容の不明確な部分など、お気づきの点がございましたら、下記にご記入の上、弊社社員にお渡しくださるようお願い申し上げます。

マニュアルの採点：

	大変良い	良い	普通	良くない
正確さ(説明どおりに動作するか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
情報量(十分か)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
分かり易さ	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
マニュアルの構成	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
図(役立つか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
例(役立つか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
索引(項目の検索性)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ページ・レイアウト(情報の検索性)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

内容の不明確な部分がありましたら、以下にご記入ください：

ペー ジ

その他お気づきの点がございましたら、以下にご記入ください：

ご使用のソフトウェアのバージョン： _____

貴社名/部課名 _____

御名前 _____

記入日 _____

(注) 当用紙を受け取った弊社社員は、すみやかに下記にお送りください。

ビジネスクリティカルシステム統括本部 **BCS** 技術本部 **Alpha** ソフトウェア技術部