はじめよう！日本語 OpenVMS

AA-RSTYA-TE.2

2009 年 11 月

本書は，日本語 OpenVMS の概要と基本操作について説明します。

日本ヒューレット・パッカード株式会社

本書は，日本語 VAX DOCUMENT V 2.1を用いて作成しています。

目次

10 ファイルを操作するためのコマンド

11 ファイルを保護する

15 シンボルを使用する

16 論理名を使用する

C ネットワークを利用したファイル操作

D JMAIL ユーティリティを使用する

図

表

まえがき

本書の目的と対象読者

本書は， OpenVMS についての概要と，日本語 OpenVMS の基本的な操作方法について説明します。

本書の対象読者

本書は，日本語 OpenVMS の概要や基本操作を知りたい方を対象にしています。

パーソナル・コンピュータ (PC) や UNIX 等のオペレーションをある程度経験されている方を対象に説明しています。

本書の構成

本書の構成は以下のとおりです。

第 1 部　概要

OpenVMS の特徴について簡単に説明します。

第 1 章　　OpenVMS の現在，過去，未来

第 2 部　基本操作

日本語 OpenVMS の開始や終了方法，コマンドの入力，エディタの使用方法やファイルの管理操作など，日本語 OpenVMS の基本操作について説明します。

第 2 章　　ログイン操作ログアウト操作
第 3 章　　DCL コマンドを使用する
第 4 章　　パスワードを変更する
第 5 章　　ヘルプとシステム・メッセージ
第 6 章　　キーボードを使用する
第 7 章　　エディタを使用する
第 8 章　　ファイルを指定する
第 9 章　　ディレクトリを作成する
第 10 章　　ファイルを操作するためのコマンド
第 11 章　　ファイルを保護する

第 3 部　日本語 OpenVMS の便利な機能を使う

日本語 OpenVMS の操作でよく使用される便利な機能について説明します。

第 12 章　ファイルの内容を印刷する
第 13 章　バッチ・キューを利用する
第 14 章　プロセスについて
第 15 章　シンボルを使用する
第 16 章　論理名を使用する
第 17 章　コマンド・プロシージャを使用する

付録

知っておくと便利な日本語 OpenVMS 関連の情報を説明します。クイック・リファレンスとしてもご利用いただけます。

付録 A　システムの起動と停止
付録 B　日本語環境を設定する
付録 C　ネットワークを利用したファイル操作
付録 D　JMAIL ユーティリティを使用する
付録 E　レキシカル関数一覧
付録 F　Linux または UNIX，MS–DOS，日本語 OpenVMS コマンド対応表
付録 G　関連アプリケーション
付録 H　PC 関連製品
付録 I　日本語 OpenVMS クイック・リファレンス
付録 J　日本語 OpenVMS マニュアル・クイック・リファレンス

本書の利用方法

本書は，最初から順に読み，例を試しながら日本語 OpenVMS の操作を習得していただけるように構成されています。

また，最初から順に読むだけでなく，必要な箇所だけを読んで利用することもできます。必要に応じて，目次または索引を利用して必要な操作の説明を調べてご利用ください。

関連資料

本書で説明した内容に関連する資料は，次のとおりです。詳しい説明をお知りになりたい場合は，該当する資料を参照してください。

日本語 OpenVMS の基本的な操作について

『OpenVMS DCL ディクショナリ』
『OpenVMS ユーザーズ・マニュアル』

システム管理関連について

『OpenVMS システム管理者マニュアル』
『OpenVMS システム管理ユーテリィティ・リファレンス・マニュアル』

日本語 EVE エディタについて

『日本語 EVE ユーザーズ・ガイド』
『日本語 EVE かな漢字変換入門』
『日本語 EVE リファレンス・マニュアル』
『漢字コード表』

日本語機能について

『日本語 OpenVMS 概説書』
『日本語 OpenVMS 日本語ユーティリティ 利用者の手引き』
『フォント管理ユーティリティ 利用者の手引き』

日本語 OpenVMS (CDE) の操作概要について

『New Desktop 使用概説書』

日本語 DECprint Supervisor for OpenVMS について

『日本語 DECprint Supervisor for OpenVMS ユーザーズ・ガイド』

これらのドキュメントは，下記の URL の OpenVMS の web サイトで参照することができます。

http://www.hp.com/jp/openvms/

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。

表記法	意味
Ctrl + x	Ctrl + xという表記は， Ctrl キーを押しながら別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
PF1 + x	PF1 + xという表記は， PF1 に定義されたキーを押してから，別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
Return	例の中で，キー名が四角で囲まれている場合には，キーボード上でそのキーを押すことを示します。テキストの中では，キー名は四角で囲まれていません。 HTML 形式のドキュメントでは，キー名は四角ではなく，括弧で囲まれています。

表記法	意味
. . .	例の中の水平方向の反復記号は，次のいずれかを示します。 • 文中のオプションの引数が省略されている。 • 前出の 1 つまたは複数の項目を繰り返すことができる。 • パラメータや値などの情報をさらに入力できる。

第1部 概要

第 1 部では， OpenVMS の概要を説明します。

OpenVMS の現在，過去，未来について見ながら， OpenVMS の特徴がわかります。

項目	参照箇所
OpenVMS の現在	第 1.1 節
OpenVMS の歴史	第 1.2 節
OpenVMS の今後	第 1.3 節

1

OpenVMS の現在，過去，未来

注意

この章の内容は， OpenVMS **が** Integrity **サーバにポーティングされる前の** 2002 **年時点の情報になります。**

Integrity **サーバへのポーティングや** OpenVMS 30 **周年の話題などを紹介した** OpenVMS **の歴史のページを下記の** URL **の** OpenVMS **の** web **サイトで公開していますので，合わせてご参照ください。**

http://www.hp.com/jp/openvms/

1.1 さあ，始めよう！ OpenVMS

春もうららのある日，新入社員の A さんが，学校の先輩 Z 氏のオフィスへやってきます。

「やあ， A さん。入社おめでとう。」

「Z さん，入社にあたっては色々とお世話になりました。おかげさまで配属部署も希望どおりになりました。」

「それはよかった。でも，それにしては少し元気がないね。」

「実は，それで相談にうかがったんですけど， 1 つシステムの管理を任せていただけることになったんです。」

「君の実力が評価されたんだね。チャレンジしがいがあるじゃないか。」

「でも，そのシステム， OpenVMS というんです。 UNIX なら経験があるんですけど。まったく知らなくて。それで，学生時代， VMS の仙人っていわれてた Z さんからコツを教えていただこうとうかがったんです。」

「ハハ，仙人はまいったな。まあ， OpenVMS といって何も特殊なことはないんだ。 UNIX の経験があれば， OpenVMS のオペレーションを理解することはそんなに難しいことじゃない。コマンド体系なんかは，いろんな機能を寄せ集めた UNIX よりも， 1 社で開発した OpenVMS の方がずっと体系的だよ。 OpenVMS のオペレーションそのものについては，第 2 章以降で具体的に説明するとして，その前に OpenVMS について概要を紹介しておこう。」

「おねがいします。」

1.1.1 OpenVMS の定義

「まず， OpenVMS とは何だろうか？」

「オペレーティング・システムですよね。 UNIX や Windows なんかと同じだと思いますけど。」

「そう，ではオペレーティング・システムとして， OpenVMS の特徴はなんだろう？」

「えーと。」

「第 1 の特徴は，シングル・アーキテクチャということなんだ。」

「シングル・アーキテクチャ？」

「シングル・アーキテクチャとは，小規模から大規模まですべてのコンピュータ・システムで，同じ環境を提供するというコンセプトのことなんだ。 OpenVMS では， 1 CPU から 32 CPU までのシステムを，すべて同じ OpenVMS でサポートしているんだ。これは，ハードウェアとしての Alpha アーキテクチャと Alpha システム，オペレーティング・システムとしての OpenVMS が密接に協力し合うことで初めて実現できていることなんだ。」

「でも，すべてのコンピュータ・システムと言っても，モバイル環境やデスクトップ環境まではサポートできていないと思うんですけど。」

「確かにその通りだね。後でもう少し詳しく説明するけれど，シングル・アーキテクチャというコンセプトは，今から 20 年以上も前， OpenVMS の初期設計の段階から構想されていたことなんだ。そういう意味では，現在の PDA やノート PC，デスクトップ PC などなど，多様なクライアント環境をサポートすることは想定されていなかったんだね。」

「ということは， OpenVMS はシングル・アーキテクチャを実現するオペレーティング・システムということですね。」

「そう，そしてこのことは OpenVMS の第 2 の特徴でもあるんだ。つまり， OpenVMS はサーバ・システム向けのオペレーティング・システムであるということだ。」

「サーバ・システム向けってどういう意味ですか？」

「言い換えると，サーバ・システムとして必要な特性を備えたオペレーティング・システムということだ。サーバ・システムとして必要な特性に関しては，いろんな考え方があるだろうが，僕は 2 つの重要な特性があると思う。

拡張性
信頼性

この 2 つだね。」

「それぞれの特性をもう少し詳しく説明してください。」

「まず，拡張性だけれど，サーバ・システムは，より大量のデータをより短時間で処理することを常に目標にしていると考えていいだろう。拡張性は，この目標を実現するための特性なんだ。もちろん， Alpha チップをはじめとして，サーバ・システムで利用される CPU はどれも高速性を競っているし，メモリにしても，ストレージにしても大容量なものがサポートできる。しかしシングル CPU システムではやはり限界があるわけだ。」

「それで対称型マルチプロセッシング SMP が登場するわけですね。」

「1 CPU よりは 2 CPU， 2 CPU よりは 4 CPU と CPU を増やしていけば，処理能力は増加することになる。同時に，メモリ容量や接続できる IO スロット数も増加することになる。たとえば，最も大規模な Alpha サーバ， GS320 は， 32 CPU，メモリ容量 256 GB， 224 個の PCI スロットまでサポートできる。もちろん OpenVMS は，この大規模な SMP システムが最高の性能を実現できるようにサポートしているわけだ。」

「でも CPU の数が増えてくると，単純な拡張は難しいと聞いたのですが。」

「そう，実際のところ GS320 のシステム・アーキテクチャも，単純な SMP ではなくなっている。しかし，アプリケーションがそれを意識するようでは，シングル・アーキテクチャとは呼べない。このギャップを解消することも OpenVMS の役割なんだ。」

「32 CPU も搭載できれば，どんなアプリケーションでも対応できますね。」

「ところがそうはいかない。実際， GS320 でも処理しきれないほどの処理量を抱えたシステムがいくつも存在するんだ。また，アプリケーションの性質によっては，SMP が有効でない場合もある。」

「そういう場合は，どうするんですか？」

「最近，クラスタリングやクラスタ・システムという言葉をよく聞くだろう。これは，単一システムを超えた拡張性を実現するために， OpenVMS がはじめて提供した機能なんだ。クラスタリングとは，複数のマシンを接続して，全体を 1 つのシステムに見せる技術だ。クラスタリングによって，ほとんど制約なしに，無限の拡張性を実現することが可能になるんだ。」

「でも，最近話題になるクラスタリングは， UNIX や Windows のものですけれど。」

「OpenVMS は既に 15 年以上も前からクラスタリング機能を提供している。 UNIX や Windows はこれに追いつこうとしているわけだ。けれど，クラスタリングの機能はオペレーティング・システムの核の部分と密接に関わっている。クラスタリング・ソフトウェアをオペレーティング・システムの上にかぶせただけのやり方では，十分な拡張性が提供できないだけでなく，システムとしての品質を維持できなくなる。」

「つまり， OpenVMS のような完成されたクラスタリング機能でなければならないということですね。」

「だんだん，わかってきたじゃないか。次に，信頼性についても簡単に説明しておこう。信頼性の定義も議論のあるところだが，サーバ・システムとして一番重要な信頼性の目標がサービスを停止させないこと，もし一時的に停止したとしてもできるだけ短期間でサービスを再開できることにあるということは誰でも合意できるだろう。」

「当然，ハードウェアやソフトウェアの品質や信頼性が問題になるわけですね。」

「Alpha システムでは，ハードウェアとしての信頼性を高めるさまざまな機能を提供している。同時に， OpenVMS もソフトウェアとしての品質を維持しながら，信頼性を高める機能を提供している。しかし．．．」

「しかし 1 台のシステムでは，信頼性の向上は限界がある。それでクラスタリングというわけですね。」

「うむ，クラスタを構成する複数のマシンは接続しているとはいえ，それぞれが独立したシステムだ。 1 台に障害が発生したとしても，他のマシンに直接影響するわけではない。このことを利用して， 1 台のシステムで実現できない信頼性を提供することもクラスタリングの重要な機能なんだ。」

「具体的にはどうするんですか？」

「単純に言えば，クラスタ・システムの 1 台のマシンが障害を起こしたとき，そのマシンでサービスを提供していたアプリケーションを他のマシンが引き継いで実行し，サービスを継続させる。これをフェールオーバと一般的に呼んでいる。」

「それは， UNIX や Windows のクラスタ・システムでも提供していますよね。」

「重要な点は，特別なプログラミングや特別な操作を必要とせず，できるだけ制約事項なしに，フェールオーバを実現することだ。 OpenVMS のクラスタリング機能は，クラスタ・システムの各マシンから，システム全体がまったく同じように見えるシングル・システム・イメージを完全に実現している。このため， OpenVMS では，ほとんどのアプリケーションで，手を加える必要なしにフェールオーバが可能なんだ。」

「なるほど。 OpenVMS の第 3 の特徴は，完全なクラスタリング機能を提供するオペレーティング・システムということですね。」

「そして，これが僕の説明したかった最後の特徴になるわけだ。」

1.1.2 OpenVMS とユーザ

「OpenVMS のすばらしさはわかってきた気がするんですけど，残念ながらあまり使われている場面を見たことがないんですけど。」

「うんうん，確かに日常的にお目にかかるというわけにはいかないだろうけど，聞いたところでは，全世界で 50 万台，約 1 千万人のユーザがいるそうだ。」

「どんな分野で利用されているんですか？」

「これも聞いた話では，全世界の十大証券取引所の 5 つで OpenVMS が取引業務に利用されているそうだ。また，全世界の携帯電話の課金処理の 50%以上が OpenVMS で処理されている。そして全世界の CPU チップ生産の 90%以上は OpenVMS が管理している。」

「日本ではどうなんですか？」

「製造業やテレコム業界で，よく利用されているようだね。特に半導体や液晶の生産管理システムで採用されているのは，よく話題になっている。」

「なんだか裏方的な役割が多いですね。」

「そうだね。しかし，取引業務や課金処理，生産管理など，その事業の根幹に関わるサービスを提供する役割なんだ。クライアント・システムのように，直接利用するシステムではないけれど，僕らの社会を支える基盤サービスを提供しているなくてはならないオペレーティング・システムといえるだろう。」

1.2 OpenVMS 年代記

「ところで，さきほど OpenVMS が誕生して 20 年以上になるっておっしゃってましたけど，そんなに古いオペレーティング・システムなんですか？」

「ハハ，古いとは人聞きが悪いね。 OpenVMS のすばらしさは，その 20 年以上にわたって，常にその時代の先端になる機能を提供しつづけているということなんだ。それでは， OpenVMS の栄光の歴史を年代記風に辿ってみるとするかな。」

1.2.1 VMS の誕生

「OpenVMS はいつ誕生したんですか？」

「VMS V1.0 が発表されたのは 1977 年だ。同時に最初の VAX システム，VAX–11 /780 が発表されている。これは完全な 32 ビット・アドレッシングのシステムとしては，最初のものの 1 つだろうね。当時は， OpenVMS ではなくて， VMS と呼んでいたんだ。ちなみに， VMS を開発したのは DEC，ディジタル・イクイップメント社だ。」

「VMS ってどういう意味なんですか？」

「仮想メモリ・システム Virtual Memory System の頭文字をとって命名されたんだ。 32 ビット・アドレッシングという当時としては使い切れないほどの大規模なメモリ空間をサポートする一番重要な機能，仮想メモリ・システムにあやかったんだろうね。」

「ところでさっき，シングル・アーキテクチャが最初から構想されていたという話がありましたけど。」

「実は， DEC は 16 ビット・アドレッシングのシステム， PDP の時代に重大な問題を抱えていた。 DEC では PDP のために用途ごとに異なったオペレーティング・システムを提供していたんだ。この方式では，アプリケーションの互換性がなくなってしまうし，開発や管理のコストが増大してしまう。 OpenVMS と VAX の開発では，この問題に対する深刻な反省の結果として，シングル・アーキテクチャというコンセプトが採用されたんだ。つまり， 1 つのプラットフォーム， 1 つのオペレーティング・システムというキャッチフレーズだ。」

1.2.2 ネットワーキング

「ところで， OpenVMS ではインターネットの他に， DECnet というネットワーク・システムを持っていると聞いたんですけど。」

「実は，シングル・アーキテクチャのキャッチフレーズには，続きがあって， 1 つのネットワークで締めくくられる。それほど，ネットワーキングを重要視していたんだね。そして，ネットワーキングでシングル・アーキテクチャを締めくくる製品が DECnet だったわけだ。実際， DECnet は VMS V1.0 から同時に提供されているんだ。」

「インターネットに関しては，どうでしょうか？」

「DEC は，インターネットの開拓者の 1 つといえるだろうね。たとえば，イーサネットの開発/標準化をゼロックス，インテルと共同で行っている。その他にもさまざまなインターネット技術が DEC の技術者の手で開発されている。初期のファイアウォールも DEC の技術者によって開発されたんだ。 OpenVMS 上のインターネット製品に関しては， DECnet が成功しすぎたこともあって，いささか出遅れていたけれど，現在では IPv6 や IPsec など先端の技術が提供されている。」

1.2.3 クラスタリング

「クラスタリング機能は 15 年以上前に登場したとおっしゃってましたね。」

「正確には， 1983 年のことになる。当時は， VAXcluster と呼んでいたんだ。」

「UNIX や Windows のクラスタリングが登場する 10 年以上前なんですね。」

「驚くべきことに，この最初の発表当時に，現在のクラスタリング技術の基本コンセプトが全て提供できていたということだ。特に重要なことは，クラスタ・システムを 1 つのマシン環境のように見せるシングル・システム・イメージが最初から実現されていたと言うことだ。これは，現在でも， UNIX や Windows のクラスタ・システムでは，ほとんど実現できていない。」

「登場時点で完成されたクラスタ・システムだったわけですね。そして，その後の発展はどうなんでしょうか？」

「クラスタ・システムで一番重要なコンポーネントは，クラスタ・メンバ間を接続するインターコネクトなんだ。発表当初はインターコネクトとして，専用のハードウェアが必要だったんだ。クラスタリング技術を普及させるために，このインターコネクトの選択肢として，より安価で標準的なハードウェアを提供する必要があると考えたんだね。そこで，以降のバージョンでは，イーサネットや FDDI などが順次サポートされていったんだ。」

「OpenVMS は，災害にも対応できるって聞いたんですけど。」

「そう，それもクラスタリングの機能を利用している。対災害性を強調するために，ディザスタトレランスと呼んでいる。ディザスタトレランスというのは，ビル全体や地域全体に被害が及ぶような，火災や地震といった災害に対しても，サービスの継続を保証する技術なんだ。具体的には一種のクラスタ・システムなんだが，クラスタのメンバが，数十キロから数百キロ離れた場所に配置される。」

「それでどちらか一方のメンバが被害を受けても，のこりのメンバがサービスを継続できるんですね。クラスタ・システムでそんなことまで実現できるんですね。」

「もちろん，現在のクラスタ・システムで，ここまでサポートできるのは，OpenVMS だけなんだ。クラスタ・システムの頂点に立っていると言えるだろうね。」

1.2.4 Alpha と 64 ビット・オペレーティング・システム

「Alpha システムはいつ頃から登場したんですか？」

「ようやく僕の時代に入ってきたね。 Alpha チップと Alpha システムが発表されたのは， 1992 年のことになる。」

「えー，上司からは VAX 世代の Z さんって聞いたんですけど。」

「むむむ，それはさておき， Alpha と聞いて何を思いつくかな？」

「ともかく早いということでしょうか。」

「そうだね，しかしソフトウェアの側から見ると，世界最初の 64 ビット・アドレッシングということが重要なんだ。 1992 年当時では，まだまだ 32 ビット・アドレッシングで十分という意見が多かったが，最近ではデスクトップ PC でも数百 MB のメモリが必要になっている。まして，サーバ・システムでは数 GB 単位でメモリを持つことが常識になっている。また，それに対応してアプリケーションの側でも大規模なメモリ空間を利用した高速化が当然のこととして行われている。この大容量の実メモリと大規模なメモリ空間を有効に使い切るためには， 64 ビット・アドレッシングは必須の機能になっている。」

「つまり， Alpha は時代を先取りして，現在の 64 ビット・アドレッシングによる高速サーバ・システムの時代を切り開いたんですね。」

「もちろん， OpenVMS もこの 64 ビット・アドレッシングにいち早く対応し，その可能性を追求している。たとえば，大容量の実メモリを利用して，データベースの高速化を図る VLM， Very Large Memory などが既に提供されている。しかし， 64 ビットというほとんど無限大のアドレス空間は，まだまだ無限の可能性を秘めていると言えるだろうね。その可能性を極めることが， OpenVMS に課せられた 21 世紀にまたがる課題と言えるだろう。」

「ところで年代記で重要な点にまだ触れていただいていないんですけれど。」

「なんだい？」

「いつ VMS が OpenVMS になったんでしょう？」

「これはうっかりしてた。 OpenVMS に正式に名称を変更したのは， 1991 年で Alpha の発表とほぼ同期しているね。 OpenVMS は 1 つの企業が設計/開発しているオペレーティング・システムだけど，そのインタフェースとしては POSIX や OSF/Motif などオープンな規格が採用されている。オペレーティング・システムとしてのオープン性を強調するネーミングと言えるだろう。」

1.3 21 世紀の OpenVMS

「さきほど， 21 世紀にまたがる課題とおっしゃってましたけど， OpenVMS の将来はどうなんでしょうか？」

「進歩しないものに未来はない。年代記でも説明したように， OpenVMS は常にその時代の最先端の技術を提供しつづけている。これは今後とも変わりないことだと思うよ。問題はどの分野にフォーカスするかということだね。」

「具体的にどの分野にフォーカスしていくんでしょうか？」

1.3.1 インターネットと Web アプリケーション

「まず，これからのアプリケーション開発を考えた場合，インターネットの利用は避けて通れないことだろう。当然， OpenVMS でもインターネットへの取り組みは重要だね。 OpenVMS とインターネットの関わりを考えると， IPv6 など基礎的な分野に対しては十分に対応してきたと言えるだろう。」

「最近はやりの WWW や eCommerce への取り組みはどうでしょうか？」

「残念ながら，この分野は UNIX や Windows に一日の長があると言わざるを得ないね。しかし，状況は変わりつつある。 Java や XML のようにプラットフォームに依存しない開発環境が， Web アプリケーションの主流になってきているんだ。これは， OpenVMS にとってはチャンスと言えるだろうね。」

「具体的にはどのようなツールが提供されているんでしょうか？」

「たとえば， Web サーバとしては Apache ベースの SWS (Secure Web Server) を提供している。このようにオープン系のツールをうまく利用することを考えているようだね。オープン系のツールとしては， Tomcat や XML 関連のツールが既に提供されているし，今後 Java の IDE や Web サービス関連のツールなどが出てくるだろう。もちろん並行して，商用のツールのサポートも進めている。」

1.3.2 クラスタリングの未来と Galaxy

「クラスタリング機能はどうなんでしょうか？既に完成済みですか？」

「クラスタリング機能そのものは，更に完成度を高め，より高速で安定したクラスタ・システムを提供することを目標にしている。新しい展開としては，新しいインターコネクトのサポートがやはり重要だろうね。専用のインターコネクトとしてはメモリ・チャネル，標準化されたものとしてはギガビット・イーサネット，ファイバー・チャネルなどがサポートされてきている。ところで拡張性・信頼性といった特性に対するまったく新しい動向としてはパーティショニングがある。」

「パーティショニングってどういう技術なんですか？」

「クラスタリングが複数のマシンを 1 つのシステムにまとめる技術なのに対して，パーティショニングは 1 つのマシンを複数のシステムに分割する技術だ。」

「今度は分割しちゃうんですか？」

「そう， GS320 に代表されるように，現在のハイエンドのサーバ・マシンは 1 台で数十 CPU をサポートするようになっている。しかしながら，現実のアプリケーションで数十 CPU を完全な並列処理を実現できるものがそんなに存在するわけではない。一方，複数のアプリケーションを 1 つのオペレーティング・システム上で並列に処理するマルチ・プログラミングにしても，数十 CPU という環境ではシステム・リ

ソースの競合などで，必ずしも期待したパフォーマンスが提供できない場合が多くなる。これを解決するのが，パーティショニングだ。」

「1 つのマシンで複数のオペレーティング・システムを稼動させるんですか？」

「パーティショニングにはいくつか方式がある。1 つは，ハードウェアの機能として，複数のオペレーティング・システムを稼動できるようにするハードウェア・パーティショニングだ。この場合は，ハードウェア・レベルでマシンのリソースを分割することになるので，リソースの割り当てが固定的になってしまうが，分割された 1 つの部分，パーティションの障害が，他のパーティションに波及することはない。また，ハードウェア・レベルで分割されているので，複数の種類のオペレーティング・システムを稼動させることも可能だ。」

「その他の方式って，ハードウェアに対するソフトウェア・パーティショニングですか？」

「そう，ソフトウェア・パーティショニングは，実際にはオペレーティング・システムのリソース管理機能と考えることができる。つまり，特定のアプリケーションやアプリケーションのグループに対して，ポリシーに基づいてシステム・リソースを正確に割り当てていく機能なんだ。この場合は，ポリシーを動的に変更したり，直接指示を与えることによって，リソースの割り当てを非常に柔軟に行うことができるが，全体が 1 つのオペレーティング・システムで実行されているので，障害がシステム全体に波及することになる。このように，ハードウェア・パーティショニングとソフトウェア・パーティショニングでは，リソース割り当ての柔軟性とシステム全体の信頼性の間でトレード・オフがあるんだ。」

「それで OpenVMS の場合はどうなんですか？」

「このトレード・オフを解消するために， OpenVMS が提供しているのが，Galaxy[1]だ。 Galaxy では，ハードウェアやソフトウェアのレベルではなく，ファームウェア・レベルでマシンを分割する。従って， Galaxy の場合は， 1 つのパーティションに割り当てられたシステム・リソースを，他のパーティションへアプリケーション実行中に再割り当てすることができる。しかも，各パーティションでは，独立したオペレーティング・システムが稼動しているので， 1 つのパーティションの障害が，他のパーティションに波及することを避けることができる。」

「つまり Galaxy は，ハードウェア・パーティショニングとソフトウェア・パーティショニングの良いとこ取りをしているわけですね。」

「そのとおりだね。更に面白いのは， Galaxy で分割したパーティション同士をクラスタ・メンバとして， 1 台のマシンでクラスタ・システムを構築することができる。しかも，そのときには， Galaxy が提供する共有メモリをインターコネクトに利用できるということだ。つまりインターコネクトのパフォーマンスがシステム・バスのレベルになるわけで，非常に高速なクラスタ通信が実現できる。」

[1] OpenVMS Integrity では Galaxy はサポートしていません。

「Galaxy とクラスタで，先進的な大規模システムの構築が可能になるわけですね。」

1.3.3 Itanium サポート

「さて， 21 世紀の OpenVMS を締めくくるためには， Itanium サポートに触れないわけにはいかないだろう。」

「確か Alpha からインテルの Itanium アーキテクチャへ鞍替えするんですよね。」

「そういわれると心外だね。まず， Alpha がすぐになくなるわけじゃない。次世代の Alpha チップ， Alpha 21364 は既に学会でも発表され，システムもまもなくリリースされることになっている。それに， Alpha の設計チームはインテルへ移籍して，Alpha が培ったさまざまな高速化技術は Itanium へ引き継がれることになる。」

「Alpha がすぐなくなるわけではないんですね。」

「もちろんだよ，特にハイエンドの大規模サーバとして， Alpha システムに匹敵するものを提供するには，それなりの時間が必要だろう。今，ここにあるシステムとしては，やはり Alpha システムが最高のシステムの 1 つと言えるだろう。」

「それじゃ， Itanium 版の OpenVMS は，まだまだ先の話なんですね。」

「システムやアプリケーションの移行には，相当な時間がかかる。そのため，OpenVMS の Itanium ポーティングは既に開始されていて，着々と進行しているようだ。少なくとも評価用の Itanium 版 OpenVMS はそんなに遠くない将来に提供されるだろう。」

「アプリケーションの開発環境やシステム管理ツールのポーティングはどうですか？」

「うん，オペレーティング・システムと平行して，ポーティングのプロジェクトが進められているようだ。ほとんどのツールは， Itanium でも利用できると考えていいだろう。その他，アプリケーションのマイグレーションに必要なさまざまなツールも提供されることになっている。」

「つまり， Itanium への対応は万全ということですね。」

「そういうことだね。しかし，最も重要なことは， Itanium のサポートによって，OpenVMS の利用環境が一気に拡大される可能性があるということだ。デスクトップやノート・サイズのマシンで， OpenVMS が自由に利用できるようになれば，OpenVMS も学びがいがあるってもんだろ。」

「早くそうなるといいですね。」

「その日のためにも，今から勉強しておくことだね。それではいよいよ，第 2 章から OpenVMS を拡張して，日本語機能を追加した，日本語 OpenVMS のオペレーションについて紹介していこう。」

第2部 基本操作

第 2 部では，日本語 OpenVMS を使用する場合に必ず知っておく必要のある基本的な操作について説明します。

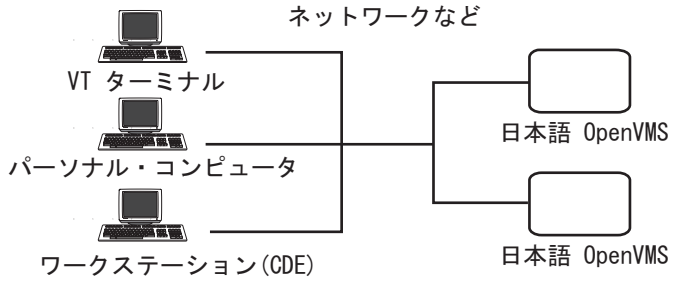
項目	参照箇所
ログイン操作 ログアウト操作	第 2 章
DCL コマンドを使用する	第 3 章
パスワードを変更する	第 4 章
ヘルプとシステム・メッセージ	第 5 章
キーボードを使用する	第 6 章
エディタを使用する	第 7 章
ファイルを指定する	第 8 章
ディレクトリを作成する	第 9 章
ファイルを操作するためのコマンド	第 10 章
ファイルを保護する	第 11 章

2

ログイン操作 ログアウト操作

この章では

日本語 OpenVMS の使用を開始することをログインするといい，使用を終了することをログアウトするといいます。日本語 OpenVMS へのログイン操作，ログアウト操作は弊社のターミナル・デバイス (VT) やパーソナル・コンピュータ，ワークステーション上の CDE (Common Desktop Environment) などから行うことができます。（日本語 OpenVMS 上で実装された CDE を New Desktop と呼ぶこともあります。）



ここでは，日本語 OpenVMS の開始と終了の操作について説明します。

関連資料

- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS ユーザーズ・マニュアル』
- 『New Desktop 使用概説書』

2.1 ログインからログアウトまでの流れ

日本語 OpenVMS にログインしてからログアウトするまでの基本的な流れを追ってみます。

2.1.1 ログインする

日本語 OpenVMS の使用を開始することを，**ログインする**といいます。正しくログインすることにより，日本語 OpenVMS システムに接続され，使用することができます。

ログインするには，ユーザ名とパスワードが必要です。ユーザ名と最初のパスワードは，システム管理者が日本語 OpenVMS システムのユーザ登録簿に登録します。

```
Username: ユーザ名
Password: パスワード
```

パスワードの変更については，第 4 章をご覧ください。

日本語 OpenVMS へのログイン操作の説明は，次の箇所をご覧ください。

VT ターミナル (弊社のターミナル・デバイス) でのログイン操作について	第 2.2 節
CDE (Common Desktop Environment) でのログイン操作について	第 2.3 節
PC (パーソナル・コンピュータ) でのログイン操作について	第 2.4 節

2.1.2 DCL コマンドを使用する

日本語 OpenVMS での作業は， DCL (Digital Command Language) と呼ばれる**コマンド**を利用して，会話形式で行います。 DCL コマンドについては，第 3 章で説明します。

2.1.3 ユーティリティを使用する

ユーティリティとは

ユーティリティは，DCL コマンドより高度な機能を提供するソフトウェアです。ユーティリティを起動すると，通常，ユーティリティ専用のプロンプトが表示されるので，ユーティリティ専用のコマンド (サブコマンドという) を入力します。

詳しくは各ユーティリティの説明書を参照してください。

2.1.4 ログアウトする

日本語 OpenVMS の使用を終了することを，**ログアウトする**といいます。正しくログアウトすることにより，日本語 OpenVMS システムへの接続が終了されます。

ログアウトするには， LOGOUT コマンドを使用します。次のように入力します。

```
$ LOGOUT
```

ログアウトすると，次のようにディスプレイに表示されます。

```
$ LOGOUT
  YAMADA       logged out at 12-FEB-2002 17:05:07.06
    1                          2
```

1 ログアウトしたユーザ名

2 ログアウトした日付と時刻

注意

ログインしたまま放置しておくと，ユーザ本人ではない他の人がそのターミナルを使用して，コンピュータを使用することができてしまうので，コンピュータを，それ以上使用しない場合 (一日の作業の終わりなど) やコンピュータを，長時間使用しない場合 (ターミナルから離れるときなど) には，必ずログアウトしてください。

2.2 VT でのログイン/ログアウト操作

VT ターミナル（弊社のターミナル・デバイス）でログインする場合は，次の手順で行います。

1. Username: プロンプトの後に，ユーザ名を入力します。

   30 秒以内に入力しないと，時間切れになるので注意してください。

2. Password: プロンプトの後に，パスワードを入力します。

   入力したパスワードは，機密保護のためディスプレイに表示されません。 30 秒以内に入力しないと，時間切れになるので注意してください。

ログインに成功すると，日本語 OpenVMS が**プロンプト**を表示します。プロンプトは次の入力を促す記号で，通常， ドル記号 ($) です。

ユーザ名を入力します。

```
Username: YAMADA
```

パスワードを入力します。(パスワードはディスプレイに表示されません。)

```
Password:
    Last interactive login on Tuesday,  12-FEB-2002 10:00
```

DCL コマンドを入力するためのプロンプトが表示されます。

```
$ _
```

ログアウトするには，次のようにします。

```
$ Logout
```

2.3 CDE でのログイン/ログアウト操作

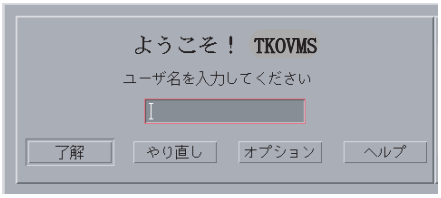
CDE (Common Desktop Environment) は X Window System 上で一貫したユーザ環境と開発環境を提供する業界標準の共通デスクトップ環境です。CDE はアプリケーションを表示するワークスペース，アプリケーション・ウィンドウを管理するウィンドウ・マネージャ，各種アプリケーションを起動するフロントパネル，ファイルを表示管理するファイル・マネージャなどを提供します。

2.3.1 CDE にログインする

CDE のログイン画面から日本語 OpenVMS へログインするには，次のようにします。

1. CDE のログイン画面の入力領域にユーザ名を入力し Return キーを押します。
2. 次にパスワードを入力します。

図 2–1 CDE のログイン画面



ログインが許可されると，CDE のワークスペースとフロントパネル，日本語入力サーバが表示されます (図 2–2)。

図 2–2　CDE セッションの開始画面

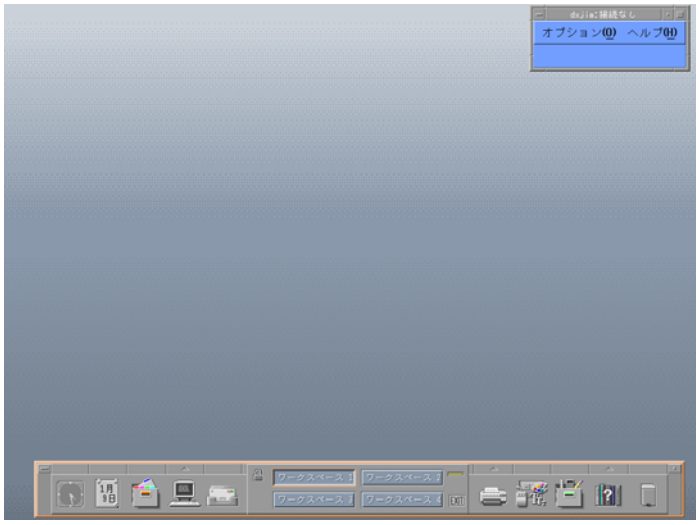
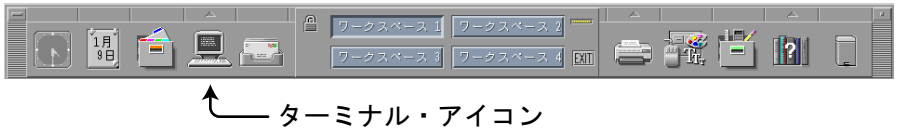


図 2–3　フロントパネル



CDE の言語設定

CDE のログイン画面の言語設定メニューで，起動する CDE セッションの言語およびコードセットを選択できます。通常はデフォルトの日本語 (sdeckanji) を使用しますが，英語を選択することもできます。

2.3.2　漢字端末エミュレータを使用する

CDE でログインした場合，日本語 OpenVMS 上での作業を行うには，「漢字端末エミュレータ」を使用します。

「漢字端末エミュレータ」を起動するにはフロントパネルのターミナル・アイコンをクリックします。

メモ

アプリケーションを起動するには，フロントパネル上の各アイコンをクリックします。

アプリケーションを多数起動して 1 画面に表示しづらい場合は，アプリケーションをいくつかのワークスペース画面に分けて表示することができます。ワークスペースの切り替えはフロントパネルの中央のワークスペースボタン

で行います。また，各ワークスペースボタンにフォーカスを与えると，ワークスペースに名前をつけることができます。

2.3.3 CDE からログアウトする

フロントパネルの「EXIT」ボタンを押して，アプリケーションを終了し， CDE からログアウトします。

この時，ログイン中に行った CDE の設定を保存するか，保存しないかを選択できます。

2.4 PC でのターミナル操作について

Windows XP などの動作する PC (パーソナル・コンピュータ) から日本語 OpenVMS にログインするには， Telnet ターミナル・エミュレータが必要です。各社からさまざまな Telnet ターミナル・エミュレータが提供されていますが，日本語 OpenVMS にログインするためには，弊社のターミナル・デバイスである VT ターミナルのエミュレーション機能を持っている必要があります。

ここでは，次の Telnet ターミナル・エミュレータ・ソフトウェアを使用して日本語 OpenVMS にログインする方法について説明します。

Tera Term Pro (フリー・ウェア) 第 2.5 節
Reflection for UNIX and OpenVMS (サイバネットシステム株式会社) 第 2.6 節

注意

Windows XP などでは，標準で telnet ユーティリティが提供されていますが，VT ターミナルのエミュレーション機能がなく，また文字コードが日本語 OpenVMS とは異なるため使用できません。

日本語 OpenVMS システムに接続するにはいくつかの方法がありますが，ここではネットワークを使って telnet 接続する場合について説明します。ターミナル・エミュレータからシリアルポート (RS-232C) を使用して日本語 OpenVMS システムに接続する方法については説明しません。

2.5 Tera term Pro を使う

Tera term Pro はフリーの Telnet ターミナル・エミュレータです。
(http://hp.vector.co.jp/)

ここでは， Tera term Pro V2.3 での実行例を示します。

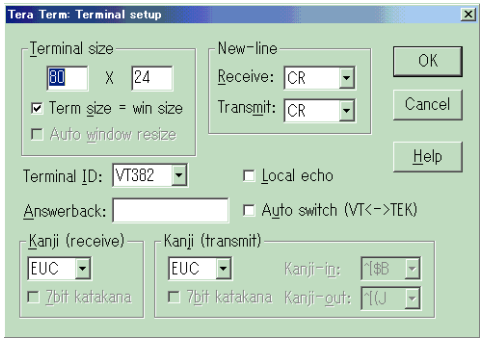
2.5.1 初期設定をする

Tera term Pro を使って日本語 OpenVMS にログインするには，初めに「Setup」メニューでいくつかの設定を行います。

1. 「Terminal setup」ダイアログ・ボックスの設定

   「Kanji (receive)」と「Kanji (transmit)」の設定を両方とも EUC にします。また，「Terminal ID」を VT382 に設定します (図 2–4)。

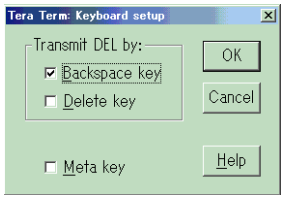
図 2–4 Terminal Setup ダイアログ・ボックス



2. 「Keyboard setup」ダイアログ・ボックスの設定

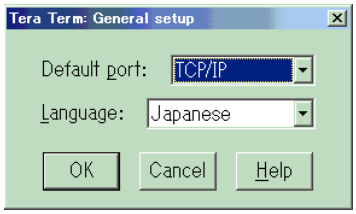
   「Transmit DEL by」の設定を Backspace Key にします (図 2–5)。

図 2–5 Keyboard Setup ダイアログ・ボックス



3. 「General setup」ダイアログ・ボックスの設定

図 2–6　General Setup **ダイアログ・ボックス**



「Default port」を TCP/IP に，「Language」を Japanese に設定します (図 2–6)。

以上の設定が完了したら，「Setup」メニューの「Save setup」項目を選択し，行った設定をファイルに保存しておくと，次回から Tera Term Pro を使用する時に便利です。

2.5.2　**日本語** OpenVMS **にログインする**

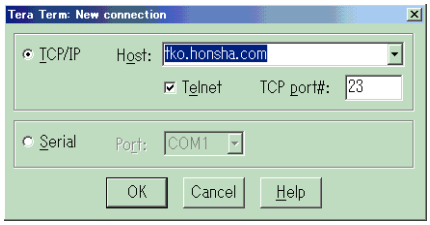
日本語 OpenVMS にログインするには，次の操作を行います。

1. 「File」メニューから「New connection」を選びます。
2. 「New connection」ダイアログ・ボックス内の「TCP/IP」チェック・ボックスが有効になっているか確認します。(有効になっていない場合は，「General setup」ダイアログ・ボックスで正しく設定を行ってください。)

   ここで「Host」テキスト・ボックスに日本語 OpenVMS システムの名前もしくは TCP/IP アドレスを入力して「OK」ボタンをクリックします。

   システムに接続したことを示すメッセージが表示され，Username: プロンプトが表示されます。

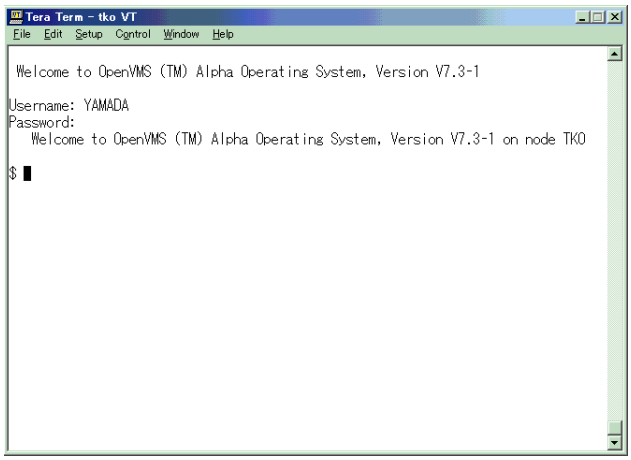
図 2–7　**「**New connection**」ダイアログ・ボックス**



3. Username: プロンプトの後にユーザ名を入力します。

   次にPassword: プロンプトの後に，パスワードを入力します。ユーザ名とパスワードの入力が完了すると，日本語 OpenVMS システムへのログインが行なわれ，プロンプトとして$が表示されます (図 2–8)。

図 2–8 Tera Term Pro を使った日本語 OpenVMS システムへのログイン



これで日本語 OpenVMS へログインできました。

ログアウトするには，次のようにします。

```
$ Logout
```

2.6 Reflection を使う

Reflection for UNIX and OpenVMS (以下 Reflection と略す) は，弊社のターミナル・デバイスである VT ターミナルと高い互換性を持つ Telnet ターミナル・エミュレータ・ソフトウエアです。(Reflection は，サイバネットシステム株式会社の製品です。 (http://www.cybernet.co.jp/))

（Reflection がインストールされている Windows XP システムの場合は「すべてのプログラム」→「アクセサリ」→「Reflection」→「ホスト - UNIX および Digital」で起動できます。）

2.6.1 初期設定をする

Reflection を使用して日本語 OpenVMS システムにログインするには，初めにキーボードの設定を行います。

1. 「設定」メニューの「ターミナル」項目を選択し，「ターミナルの設定」ダイアログ・ボックスを表示します。

   次にダイアログ・ボックスの上部にある「キーボード」タブを選択します (図 2–9)。

2. ダイアログ・ボックスの右半分にある「端末キー」の設定項目の「VT Backspace キー」項目を「削除」に設定します。

図 2–9 「ターミナルの設定」ダイアログ・ボックス

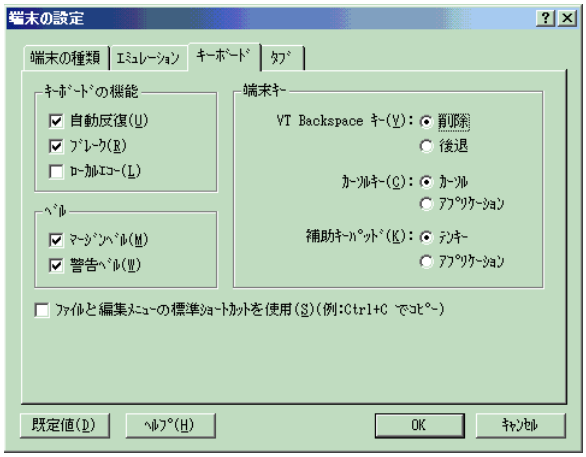


図 2–10 「ホスト/サービスに接続」ダイアログ・ボックス

設定が完了したら，「ファイル」メニューの「保存」または「名前を付けて保存」項目を選択し，行った設定をファイルに保存しておくと，次回から Reflection を使用する時に便利です。

2.6.2 日本語 OpenVMS にログインする

日本語 OpenVMS システムにログインするには，次の操作を行います。

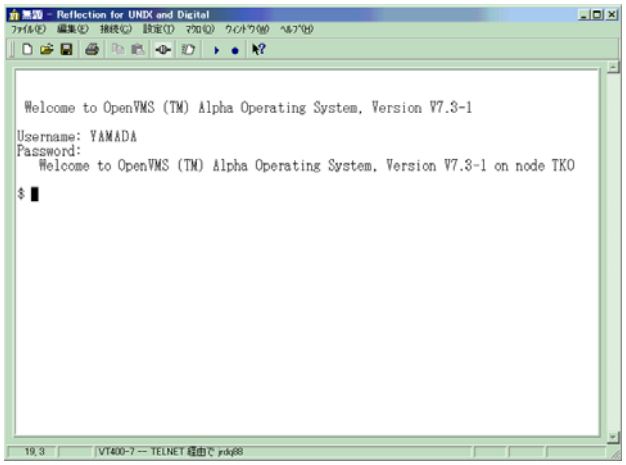
1. 「接続」メニューから「最適ネットワークに接続」を選びます。

   表示された「ホスト/サービスに接続」ダイアログ・ボックスで，日本語 OpenVMS システムの名前もしくは TCP/IP アドレスを入力して「OK」ボタンをクリックします (図 2–10)。

   日本語 OpenVMS システムに接続したことを示すメッセージが表示され，Username: プロンプトが表示されます。

2. Username: プロンプトの後に，ユーザ名を入力します。次にPassword: プロンプトの後に，パスワードを入力します。ユーザ名とパスワードの入力が完了すると，日本語 OpenVMS へのログインが行なわれ，プロンプトとして$が表示されます (図 2–11)。

図 2–11　Reflection を使った日本語 OpenVMS システムへのログイン



これで日本語 OpenVMS へログインできました。

ログアウトするには，次のようにします。

```
$ Logout
```

2.7　コラム：電源はいきなり切らない

日本語 OpenVMS を使用している際に，電源をいきなり切らないでください。日本語 OpenVMS はマルチタスクで，複数のユーザが同時に使用できるオペレーション・システムです。プログラムが何かの処理をしている最中の場合もありますし，他のユーザが同時に使用している場合もあります。また，ハード・ディスクが破損する可能性もあります。

電源を切るには，状況を確かめてから，正しいシャットダウンの手順をおこなってください。

シャットダウンの手順は，付録 A や『システム管理者マニュアル』を参照してください。

3

DCL コマンドを使用する

この章では

日本語 OpenVMS 上での作業は DCL (Digital Command Language) と呼ばれるコマンドを利用して会話形式で行います。 DCL コマンドを使用して，システムについての情報を入手したり，ファイルの操作，プログラムの開発等を行うことができます。

ここでは， DCL コマンドの基本的な使用方法，パスワードの変更方法について説明します。また，日本語の環境を設定する方法についても説明します。

よく使用される DCL コマンドの一覧は，付録 I もご覧ください。

関連資料

- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS ユーザーズ・マニュアル』

3.1 DCL コマンドを使用する

日本語 OpenVMS 上での作業は，DCL (Digital Command Language) **コマンド**と呼ばれるコマンドを利用して会話形式で行います。 DCL コマンドはプロンプト (通常は，ドル記号 "$") が表示されているときに入力できます。

3.2 DCL コマンドの形式

DCL コマンドの形式は次のようになっています。

```
$ コマンド名［/修飾子］［パラメータ...］［/修飾子］
      1          2           3            2
```

1 コマンド名

DCL コマンドの名前。 DCL コマンドの機能を示します。

2 修飾子

/で始まる単語。 DCL コマンドやパラメータに付加的な指示を与える場合に付けます。指定できる修飾子は， DCL コマンドによって異なります。

修飾子を指定する場合は修飾子とパラメータはスペースで区切ります。また，修飾子の前はスペースがあってもなくても構いません。

3 パラメータ

DCL コマンドを実行する対象を示します。指定できる対象は， DCL コマンドによって異なります。コマンド名とパラメータはスペースで区切ります。

DCL と英語

DCL は，英語に似せて作られたコマンド体系です。コマンド名にはたいてい英語の動詞が使われています。 DCL は先頭に動詞があるので英語の命令文にあたります。パラメータは目的語に，修飾子は，副詞や形容詞などに相当します。

3.3 一般的に使用される DCL コマンド

次の表に，一般的なシステム操作を実行するときに使用される DCL コマンドをいくつか示します。

コマンド	操作
COPY	指定されたファイルをコピーする。

コマンド	操作
CREATE	ファイルまたはディレクトリを作成する。
DELETE	指定されたファイルをディレクトリから削除する。
DIRECTORY	ディレクトリの内容 (ファイルのリスト) を表示する。
EDIT	テキスト・ファイルの内容を調べて変更する。
LOGOUT	セッションを終了する。
PRINT	指定されたファイルをプリンタに送って印刷する。
RENAME	指定されたファイルの名前または位置を変更する。
SET	画面上でのシステムの表示のしかたを制御する。
SHOW	システムの状態を表示する。
TYPE	指定されたファイルの内容を画面に表示する。

3.4 DCL コマンドの実行方法

DCL コマンドを実行するには，プロンプト (通常 "$") が表示されているときに，プロンプトに続けてコマンド名を入力し，Return キーまたは Enter キーを押します。

表 3–1 に，さまざまな形式の DCL コマンドの例を示します。

表 3–1 DCL コマンドの入力例

コマンド形式	例
$ コマンド	$ LOGOUT
$ コマンド パラメータ	$ TYPE FILE.TXT
$ コマンド/修飾子 パラメータ	$ DIRECTORY/FULL FILE.TXT
$ コマンド パラメータ/修飾子	$ PRINT FILE.TXT/COPIES=2
$ コマンド パラメータ, パラメータ	$ PRINT FILE.TXT, TEST.LIS
$ コマンド パラメータ, パラメータ/修飾子	$ PRINT A.TXT, B.LIS/COPIES=3

3.5 DCL コマンドを短縮して使う

DCL コマンドや修飾子は文字を短縮できます。通常，最初の 4 文字まで入れれば，コマンドは実行されます。他のコマンドと区別できる場合は， 4 文字以下でも実行されます。 (先頭が NO で始まる修飾子の場合は， NO を除いて 4 文字)

```
$ LOGOUT
$ LOGO
$ LO
```

ただし，コマンド・プロシージャ (第 17 章を参照) 内では，明確さを維持するためにコマンドを短縮しないでください。

3.6 DCL コマンドを複数行にわたって入力する

DCL コマンド行を複数行にわたって継続するには，次のように行末にハイフン (-) を入力して Return キーを押します。すると '_$' というプロンプトが表示されるので，コマンドの続きを入力します。

```
$ DIRECTORY/DATE/SINCE=YESTERDAY -
  _$ A.TXT,B.TXT,C.TXT
```

3.7 DCL コマンド行を編集する

DCL コマンドを入力しているときに誤りに気づいた場合，⌫ キー (または Back Space キー) を使用して修正することができます。その際，文字列の途中の誤りを修正するために，文字列の最後から誤りの文字まで ⌫ キーを使って削除してしまうと入力した文字が無駄になってしまいます。

日本語 OpenVMS では，入力中の DCL コマンド行内でカーソルを移動して，任意の文字を挿入，変更，削除することができます。コマンド行内でカーソルを移動するには，次のキーを使います。

キー	説明
Ctrl + H	カーソルを行の先頭に移動
Ctrl + E	カーソルを行の末尾に移動
←	カーソルを 1 文字分だけ左に移動
→	カーソルを 1 文字分だけ右に移動
Ctrl + A	入力の上書きモードと挿入モードの切り替え (トグルになっている)

コマンド行から文字を削除するには，次のキーを使います。

キー	説明
⌫	カーソルの左の 1 文字を削除
Ctrl + J	カーソルの左の 1 語を削除
Ctrl + U	行の先頭からカーソルの左までの文字列を削除

3.7.1 編集した DCL コマンドの実行

編集した DCL コマンド行を実行するには，カーソルの位置に関係なく，そのまま Return キーまたは Enter キーを押します。カーソルをコマンド行の最後まで移動する必要はありません。

3.7.2 DCL コマンド行の編集例

たとえば，$ SHOW USERS と入力するつもりが，まちがえて $ SHOW ISERS と入力して，Return キーを押す前に気が付きました。この場合は，次の手順で修正します。(この例では，入力モードは上書きモードになっています。)

1. カーソルは文字列の後に表示されています。

```
$ SHOW ISERS _
```

2. ← キーを 5 回押して，カーソルを I の上に移動します。

```
$ SHOW ISERS
```

3. U を入力します。

```
$ SHOW USERS
```

4. カーソルは S の上に移動したままですが，そのまま Return キーを押せば，コマンドが実行できます。

```
$ SHOW USERS
```

3.8 DCL コマンドの再呼び出し

すでに実行したコマンドを呼び出して，そのまま，あるいは修正して実行することができます。

日本語 OpenVMS では，ユーザが実行したコマンドを新しいものからさかのぼって最大 254 個まで記憶できます。記憶されたコマンドを呼び出すには，次のキーあるいはコマンドを使います。

キーまたはコマンド	説明
↑	直前に実行したコマンドを呼び出して表示する。さらにこのキーを押すと，実行したコマンドを新しいものからさかのぼって順番に呼び出せる。
↓	↑ でコマンドの呼び出しを行っているとき，そのコマンドの次に実行したコマンドが呼び出せる。
RECALL/ALL	実行したコマンドを新しいものからさかのぼって最大 254 個まで一覧表にして出力する。
RECALL 番号	前記の RECALL/ALL コマンドで表示される各コマンドに付いている番号でコマンドを呼び出せる。
Ctrl + B	直前に実行したコマンドを呼び出して表示する。さらにこのキーを押すと，実行したコマンドを新しいものからさかのぼって順番に呼び出せる。

RECALL コマンドを使用して，コマンドを呼び出してみましょう。

1. RECALL/ALL コマンドで最近実行したコマンドの一覧を表示します。

```
$ RECALL/ALL

 1 TYPE MEIBO.TXT
 2 EDIT/XTPU MEIBO.TXT
 3 PRINT/QUEUE=PRINTER1/NOTIFY MEIBO.TXT
 4 EDIT/XTPU MEIBO.TXT
 5 TYPE MEIBO_TMP.TXT
 6 TYPE MEIBO.TXT
 7 APPEND MEIBO_TMP.TXT MEIBO.TXT
 8 EDIT/XTPU MEIBO_TMP.TXT
 9 DIRECTORY
10 SET DEFAULT [.MEMO]
11 DIRECTORY
```

2. 3 番目のコマンドを呼び出します。

```
$ RECALL 3
$ PRINT/QUEUE=PRINTER1/NOTIFY MEIBO.TXT_
```

3. Return キーを押すと，コマンドが実行されます。

```
$ PRINT/QUEUE=PRINTER1/NOTIFY MEIBO.TXT
```

3.9 DCL のオンライン・ヘルプとコマンド・リファレンス

DCL コマンドについての詳細は，DCL オンライン・ヘルプおよび『OpenVMS DCL ディクショナリ』を参照してください。これらのドキュメントでは，OpenVMS で提供するすべての DCL コマンドについてその使用方法を説明しています。

なお，オンライン・ヘルプについては本書の第 5 章で説明します。

4

パスワードを変更する

この章では

パスワードは，使おうとしている人がユーザ本人であるかを日本語 OpenVMS が識別するための情報で，機密保護のための重要な情報です。

日本語 OpenVMS でパスワードを変更するするためのコマンド，パスワードの規則，有効期間，変更する時期について説明します。

関連資料

- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS ユーザーズ・マニュアル』

4.1 パスワード変更コマンド — SET PASSWORD

最初はシステム管理者がパスワードを用意します。その後は，ユーザが必要に応じて自由に変更することができます。

パスワードの変更には，次のコマンドを使います。

```
$ SET PASSWORD
```

4.2 パスワードの規則

パスワードの最小文字数は，システム管理者によって決められており，通常は 6 〜 8 文字程度です。また，最大文字数は 31 文字です。英数字，ドル記号 ($)，アンダースコア (_) を使うことができます。英字を使用する場合には，大文字と小文字は区別しません。

4.3 パスワードの有効期間

パスワードには，有効期間があります。有効期間の設定は，システム管理者が行います。有効期間内であれば，いつでもユーザ自身がパスワードの変更を行うことができます。有効期間を過ぎると，最初のログイン時にパスワードの変更を促すプロンプトが表示されます。ここでパスワードを変更しなければシステムにログインできません。

パスワード変更プロンプトについては，詳細は，第 4.4.3 項を参照してください。

4.4 パスワードを変更する時期

パスワードを変更する時期には，次の 3 つの場合があります。

4.4.1 初めて日本語 OpenVMS にログインした場合

初めて日本語 OpenVMS にログインしたときは，次のように表示されます。

```
Username: YAMADA
Password:

Your password has expired; you must set a new password to log in

New password: _
```

この場合，すぐに新しいパスワードの設定を行わなければなりません。次のようにします。

1. 新しいパスワードの入力を促すプロンプト New password: の後に，新しいパスワードを入力し，Return キーを押します。(パスワードは，ディスプレイに表示されません。)

```
New password:
```

2. 新しいパスワードの確認入力を促すプロンプト Verification: の後に，確認のため同じパスワードをもう一度入力します。

```
Verification:
```

3. DCL コマンドを入力するためのプロンプト $ が表示されると，パスワードの設定が成功したことを意味します。必要に応じて DCL コマンドを入力します。

```
$
```

4.4.2 ユーザが必要に応じてパスワード変更する場合

パスワードの有効期間内に，ユーザが必要に応じて変更するには，次のようにします。

1. SET PASSWORD コマンドを入力します。

```
$ SET PASSWORD
```

2. 現在のパスワードを Old password: の後に入力し，Return キーを押します。(パスワードはディスプレイに表示されません。)

```
Old password:
```

3. 新しいパスワードを New password: の後に入力し，Return キーを押します。

```
New password:
```

4. 確認のため同じパスワードを Verification: の後に，もう一度入力します。

```
Verification:
```

5. パスワードの変更が成功するとプロンプトに $ が表示されます。

```
$ _
```

4.4.3 日本語 OpenVMS がパスワードの変更を促す場合

パスワードの有効期間が過ぎると，ログインしたときに日本語 OpenVMS が自動的にパスワードの変更を促します。次のようなメッセージが表示されます。

```
  Your password has expired; you must set a new password to log in

Username: YAMADA
Password:

    Last interactive login on Wednesday,  16-FEB-2002 10:15

Your password has expired; you must set a new password to log in

New password: _
```

パスワードを変更するには，次のようにします。

1. 新しいパスワードを New password: の後に入力し，Return キーを押します。(パスワードはディスプレイに表示されません。)

   ```
   New password:
   ```

2. 確認のため同じパスワードを Verification: の後に，もう一度入力します。

   ```
   Verification:
   ```

3. パスワードの変更が成功するとプロンプトに $ が表示されます。

   ```
   $
   ```

4.5 パスワードと機密保護

パスワードは，ユーザ名ごとに設定します。ユーザ名とパスワードを対にして使用することにより，使用を許可されていない人がコンピュータを不正に使用することを防ぐことができます。日本語 OpenVMS では，パスワードは機密保護のため入力してもディスプレイに表示されません。

重要

機密保護のため，次の事項を守ってください。

- **パスワードは暗記しておく。**

  パスワードを紙などに書いて第三者が容易に確認できるような場所に放置しない。

- **パスワードを他の人に教えない。**
- **パスワードは，第三者が類推しにくいものにする。**

  自分の氏名，電話番号，ユーザ名，個人的なデータ (誕生日など) は，避けてください。

5
ヘルプとシステム・メッセージ

この章では

日本語 OpenVMS では，DCL コマンドの種類や使い方を調べるための機能として，**オンライン・ヘルプ**が用意されています。また，ユーザの行った操作について，**システム・メッセージ**を表示して操作の結果を報告します。操作の結果がエラーであった場合にメッセージを読んでエラーの原因と解決方法を知ることができます。

ここでは，日本語 OpenVMS のオンライン・ヘルプの使い方とシステム・メッセージについて説明します。

関連資料

- 『日本語 OpenVMS 概説書』
- 『日本語 OpenVMS 日本語ユーティリティ 利用者の手引き』
- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』

5.1 ヘルプを使う — HELP

日本語 OpenVMS では，日本語と英語のヘルプを用意しています。ここでは，ヘルプの言語の切り替え方，ヘルプの使い方について説明します。

5.1.1 ヘルプの言語の切り替え — @JSY$SYSTEM:JSY$SWITCH

日本語 OpenVMS では日本語のオンライン・ヘルプが用意されていますが，標準版(英語版)OpenVMS の英語のオンライン・ヘルプを使用することもできます。

現在のヘルプの設定が英語か日本語かを調べるには，次のようにします。

```
$ @JSY$SYSTEM:JSY$SWITCH  SHOW
```

ヘルプの日本語と英語の切り替えは，次のように行います。

◆日本語から英語へ切り替える (英語のヘルプを表示)

```
$ @JSY$SYSTEM:JSY$SWITCH  ENGLISH
```

◆英語から日本語へ切り替える (日本語のヘルプを表示)

```
$ @JSY$SYSTEM:JSY$SWITCH  JAPANESE
```

5.1.2 ヘルプを開始する — コマンドがわからない場合

コマンドがわからない場合には，コマンドを指定しないでヘルプを開始します。次のように入力します。

```
$ HELP
```

コマンドを指定しないでヘルプを開始すると，コマンド一覧が表示されるので，一覧を見てコマンドを探すことができます。

コマンド一覧は，複数の画面に分かれていて， 1 画面ずつ順に表示されます。次の画面に進むときは，Return キーを押してください。

```
HELP


    HELP コマンドは，コマンドあるいはトピックについての情報を表示する
    HELP 機能を起動します。 "Topic?" というプロンプトに応答することで，
    次の指示ができます。

      ・ 知りたいコマンドあるいはトピックの名前をタイプします。

      ・ HELP の使用方法についてのより詳細な情報が欲しいときは，
        INSTRUCTIONS とタイプしてください。

      ・ 知りたいコマンドあるいはトピックの名前がはっきり分からない場合は
        HINTS とタイプしてください。

      ・ HELP/MESSAGE コマンドについて知りたい場合は，HELP/MESSAGE とタイ
        プします。

      ・ 一番最後に要求したテキストをもう一度見たいときは，疑問符 (?) を
        タイプしてください。

      ・ HELP 機能から抜けたいときには，RETURN キーを１回あるいは数回押し
        てください。

継続する場合には，RETURN を押してください ...
```

```
  INITIALIZE INPUT      INQUIRE    INSTALL    JAVA       JOB        LATCP
  Lexicals   LIBRARY    LICENSE    Line_editing          LINK       LMCP
  LOCALE     LOGIN      LOGOUT     LPQ        LPRM       MAIL
  MAIL_Command          MERGE      MIBCOMP    MONITOR    MOUNT      ON
  OPEN       PASSWORD   PATCH      PHONE      PIPE       PRINT
  PrintServers          PRINT_Parameter       PSWRAP     PURGE      RCP
  READ       RECALL     RENAME     REPLY      REQUEST    RETURN     REXEC
  RLOGIN     RPCGEN     RSH        RUN        RUNOFF     SEARCH     SET
  SHOW       SORT       SPAWN      START      STOP       SUBMIT
  SUBROUTINE Symbol_Assign         SYNCHRONIZE           SYSGEN     SYSMAN
  TCPIP_Services        TCPTRACE   TELNET     TN3270     TYPE       UNLOCK
  V62_Features          V70_Features          V71_Features
  V72_Features          VIEW       WAIT       WRITE

  Additional help libraries available (type @name for topics):

  HELPLIB  JSYHELP

Topic? 
```

調べたいコマンドがわかったら，そのコマンドの説明を表示することができます。最後の画面の "Topic?"の後か，または，各画面の Return キーを押すところに説明を表示したいコマンド名を入力します。

たとえば，TYPE コマンドを調べたいときは，次の例のように TYPE と入力します。

```
  INITIALIZE INPUT      INQUIRE    INSTALL    JAVA       JOB        LATCP
  Lexicals   LIBRARY    LICENSE    Line_editing          LINK       LMCP
  LOCALE     LOGIN      LOGOUT     LPQ        LPRM       MAIL
  MAIL_Command          MERGE      MIBCOMP    MONITOR    MOUNT      ON
  OPEN       PASSWORD   PATCH      PHONE      PIPE       PRINT
  PrintServers          PRINT_Parameter       PSWRAP     PURGE      RCP
  READ       RECALL     RENAME     REPLY      REQUEST    RETURN     REXEC
  RLOGIN     RPCGEN     RSH        RUN        RUNOFF     SEARCH     SET
  SHOW       SORT       SPAWN      START      STOP       SUBMIT
  SUBROUTINE Symbol_Assign         SYNCHRONIZE           SYSGEN     SYSMAN
  TCPIP_Services        TCPTRACE   TELNET     TN3270     TYPE       UNLOCK
  V62_Features          V70_Features          V71_Features
  V72_Features          VIEW       WAIT       WRITE

  Additional help libraries available (type @name for topics):

  HELPLIB  JSYHELP

Topic? TYPE
```

TYPE コマンドの説明が表示されます。

```
TYPE

     １つまたは複数のファイルの内容を，現在の出力装置に表示します。

     Format

        TYPE    ファイル指定[,...]



  Additional information available:

  Parameter  Qualifier
  /BACKUP    /BEFORE    /BY_OWNER  /CONFIRM   /CONTINUOUS           /CREATED
  /EXACT     /EXCLUDE   /EXPIRED   /HEADER    /HIGHLIGHT /MODIFIED  /OUTPUT
  /PAGE      /SEARCH    /SINCE     /STYLE     /TAIL      /WRAP
  Examples

TYPE Subtopic?
```

5.1.3 ヘルプを開始する — コマンドがわかっている場合

調べたいコマンドやその修飾子がわかっている場合には，コマンドを指定してヘルプを開始します。次のように入力します。

```
$ HELP コマンド[/修飾子]
       1        2
```

1 調べたいコマンド。

2 調べたいコマンドの調べたい修飾子。

コマンドを指定しないでヘルプを開始すると，コマンド一覧が最初に表示されます。これに対して，コマンドを指定してヘルプを開始すると，コマンド一覧は表示されずに，指定したコマンドの説明が最初に表示されます。

たとえば，TYPE コマンドを調べたいときは，DCL コマンドラインで次のように入力します。

```
$ HELP TYPE
```

次のように，TYPE コマンドの説明が表示されます。

```
TYPE

     １つまたは複数のファイルの内容を，現在の出力装置に表示します。

     Format

        TYPE    ファイル指定[,...]



  Additional information available:

  Parameter  Qualifier
  /BACKUP    /BEFORE    /BY_OWNER  /CONFIRM   /CONTINUOUS           /CREATED
  /EXACT     /EXCLUDE   /EXPIRED   /HEADER    /HIGHLIGHT /MODIFIED  /OUTPUT
  /PAGE      /SEARCH    /SINCE     /STYLE     /TAIL      /WRAP
  Examples

TYPE Subtopic? _
```

さらに，そのコマンドに対して指定できる修飾子の使い方や働きを調べる場合は，"コマンド名 Subtopic?"に対して，調べたい修飾子を入力します。

たとえば，TYPE コマンドを調べているときに/PAGE 修飾子を調べたい場合は，/PAGE と入力します。

```
TYPE

     １つまたは複数のファイルの内容を，現在の出力装置に表示します。

     Format

        TYPE    ファイル指定[,...]



  Additional information available:

  Parameter  Qualifier
  /BACKUP    /BEFORE    /BY_OWNER  /CONFIRM   /CONTINUOUS           /CREATED
  /EXACT     /EXCLUDE   /EXPIRED   /HEADER    /HIGHLIGHT /MODIFIED  /OUTPUT
  /PAGE      /SEARCH    /SINCE     /STYLE     /TAIL      /WRAP
  Examples

TYPE Subtopic? /PAGE_
```

次のように， TYPE/PAGE の説明が表示されます。

```
TYPE

  /PAGE

        /PAGE[=キーワード]
        /NOPAGE（省略時の設定）

     画面上の情報の表示を制御します。

     /PAGE修飾子では，次のキーワードを使用できます。

     CLEAR_SCREEN      ページモードで表示（毎回画面を消去する）

     SCROLL            スクロールモードで表示（毎回画面を消去しない）


     SAVE[=n]          nページ分の履歴を保持する（前ページに戻ること等が
                       可能）

     /PAGE=SAVE修飾子を指定すると最大5画面（最大255カラムまで）分の履歴
     を保存できます。ページャ内では以下のキーを使って画面の移動などがで

継続する場合には，RETURN を押してください ...
```

5.1.4 コマンドの説明の再表示

コマンドの説明をもう一度表示させたい場合は， "コマンド名 Subtopic?"に対して， "?"を入力します。この場合，Return キーを押す必要はありません。

次のように， "?"と入力します。

```
     F10，Ctrl/Z                    終了(他のコマンドでは違うものもあり
                                    ます)

     Help (F15)                     ヘル・テキストを表示

     Do (F16)                       最新（現在）画面と（履歴内で）最古画
                                    面の入れ換え

     Ctrl/W                         再表示

     N                              次ファイル(複数ファイルを指定した場
                                    合)

     Q                              終了

     /PAGE修飾子は/OUTPUT修飾子と同時に指定することはできません。


TYPE Subtopic? ?
```

"?"と入力すると， TYPE コマンドの説明がもう一度表示されます。

```
TYPE

     １つまたは複数のファイルの内容を，現在の出力装置に表示します。

     Format

        TYPE    ファイル指定[,...]



  Additional information available:

  Parameter  Qualifier
  /BACKUP    /BEFORE    /BY_OWNER  /CONFIRM   /CONTINUOUS           /CREATED
  /EXACT     /EXCLUDE   /EXPIRED   /HEADER    /HIGHLIGHT /MODIFIED  /OUTPUT
  /PAGE      /SEARCH    /SINCE     /STYLE     /TAIL      /WRAP
  Examples

TYPE Subtopic? _
```

5.1.5 Topic? **に戻る**

修飾子を調べているレベル ("コマンド名 Subtopic?") からコマンドを調べるレベル ("Topic?") へ戻るには，"コマンド名 Subtopic?"に対して，何も入力せずに Return キーを押します。

たとえば，次の例のカーソルの位置で，Return キーを押します。

```
     F10, Ctrl/Z                    終了(他のコマンドでは違うものもあり
                                    ます)

     Help (F15)                     ヘル・テキストを表示

     Do (F16)                       最新（現在）画面と（履歴内で）最古画
                                    面の入れ換え

     Ctrl/W                         再表示

     N                              次ファイル(複数ファイルを指定した場
                                    合)

     Q                              終了

     /PAGE修飾子は/OUTPUT修飾子と同時に指定することはできません。


TYPE Subtopic? _
```

Return キーを押すと，次の図のように "Topic?"に戻ります。

```
     F10, Ctrl/Z                    終了(他のコマンドでは違うものもあり
                                    ます)

     Help (F15)                     ヘル・テキストを表示

     Do (F16)                       最新（現在）画面と（履歴内で）最古画
                                    面の入れ換え

     Ctrl/W                         再表示

     N                              次ファイル(複数ファイルを指定した場
                                    合)

     Q                              終了

     /PAGE修飾子は/OUTPUT修飾子と同時に指定することはできません。


TYPE Subtopic?
Topic?
```

5.1.6 コマンド一覧の表示

コマンド一覧を表示させたい場合は，"Topic?"に対して，"?"を入力します。この場合，Return キーを押す必要はありません。次のように，"?"と入力します。

```
TYPE

     １つまたは複数のファイルの内容を，現在の出力装置に表示します。

     Format

        TYPE    ファイル指定[,...]



  Additional information available:

  Parameter  Qualifier
  /BACKUP    /BEFORE    /BY_OWNER  /CONFIRM   /CONTINUOUS           /CREATED
  /EXACT     /EXCLUDE   /EXPIRED   /HEADER    /HIGHLIGHT /MODIFIED  /OUTPUT
  /PAGE      /SEARCH    /SINCE     /STYLE     /TAIL      /WRAP
  Examples

TYPE Subtopic?
Topic? ?
```

"?"と入力すると，コマンド一覧が表示されます。

```
HELP

    HELP コマンドは，コマンドあるいはトピックについての情報を表示する
    HELP 機能を起動します。 "Topic?" というプロンプトに応答することで，
    次の指示ができます。

      ・ 知りたいコマンドあるいはトピックの名前をタイプします。

      ・ HELP の使用方法についてのより詳細な情報が欲しいときは，
        INSTRUCTIONS とタイプしてください。

      ・ 知りたいコマンドあるいはトピックの名前がはっきり分からない場合は
        HINTS とタイプしてください。

      ・ HELP/MESSAGE コマンドについて知りたい場合は，HELP/MESSAGE とタイ
        プします。

      ・ 一番最後に要求したテキストをもう一度見たいときは，疑問符 (?) を
        タイプしてください。

      ・ HELP 機能から抜けたいときには，RETURN キーを１回あるいは数回押し
        てください。

継続する場合には，RETURN を押してください ...
```

5.1.7 ヘルプを終了する

ヘルプは，次のいずれかの方法を使用して終了します。

- "Topic?"に対して，何も入力せずに Return キーを押す。
- 任意の箇所で，Ctrl + Z を入力する。

5.2 システム・メッセージとは

日本語 OpenVMS では， DCL コマンドを使用して作業を行います。 DCL コマンドを入力すると，日本語 OpenVMS は処理を実行し，ユーザが行った操作についてシステム・メッセージを表示して，処理の実行結果を報告します。処理が正常終了すると通常はプロンプトのみを表示します。 (ユーザが行った操作について，正常終了後にメッセージを表示して，その結果を報告する場合もあります。) しかし，エラーが発生した場合は，メッセージを表示した後にプロンプトを表示します。

5.2.1 メッセージの形式と種類

日本語 OpenVMS のメッセージのほとんどのものは，次の形式をしています。

```
%FACILITY-L-IDENT,  MESSAGE TEXT
   1     2  3        4
```

1 ファシリティ・コード

この実行結果を通知したユーティリティ・プログラム名等を短縮形で示します。

2 重大度レベル・コード

重大度に応じて 5 つに分類されます。

3 メッセージ識別コード

メッセージの短縮形を示します。

4 メッセージ本文

DCL コマンドの実行結果に関するメッセージを示します。

メッセージは，重大度レベルに応じて 5 つの種類に分けられています。 (表 5–1 を参照。) メッセージが表示されたときに，そのメッセージが 5 つのうちのどれかを知るには，重大度レベル・コードを見ます。

表 5–1 システム・メッセージの種類

重大度レベル・コード	内容	説明
S (Success)	正常終了	処理が正しく実行されたことを示す。
I (Information)	情報	処理が正しく実行されたことを示す。 日本語 OpenVMS では，原則として処理が正常終了した場合に，システム・メッセージは表示しない。ただし，結果をユーザに通知することが適当である処理については，重大度レベルに S または I を使用してシステム・メッセージを表示する。
W (Warning)	警告	ユーザが要求した処理のうち，一部は実行されたが，すべては実行されなかった，という状態や軽微なエラーが発生したことを示す。結果の確認が必要。
E (Error)	エラー	処理が正しく実行されなかったことを示す。
F (Fatal)	回復不可能	回復不可能なエラーが発生したことを示す。

5.2.2 エラー・メッセージの例

処理が正しく実行されなかった場合には，次のようなメッセージが表示されます。

```
$ COPY GROUP_JAN.TXT GROUP_FEB.TXT
%COPY-E-OPENIN, error opening DKA0:[YAMADA.GIJIROKU]GROUP_JAN.TXT; as input
-RMS-E-FNF, file not found
```

5.3 エラーを解決する

システム・メッセージを読み，エラーを解決する方法を例をあげて説明します。

COPY コマンドを実行したときに，メッセージが表示されました。

```
$ COPY GROUP_JAN.TXT GROUP_FEB.TXT
%COPY-E-OPENIN, error opening DKA0:[YAMADA.GIJIROKU]GROUP_JAN.TXT; as input
-RMS-E-FNF, file not found
```

次の手順で解決します。

1. 最初にメッセージを分析します。

   この例では，重大度レベル・コードが "E"なので，エラーが発生したことがわかります。また，メッセージ本文から， COPY コマンドの入力ファイルが開けないことがわかります。

2. 2 行目のメッセージから存在しないファイルを指定したことがエラーの原因であるとわかります。

3. ファイル名をもう 1 度確認します。

4. 正しいファイル名を確認したら，そのファイルを指定して COPY コマンドを実行します。

5.4 システム・メッセージのオンライン・ヘルプ — HELP/MESSAGE

システム・メッセージについてのオンライン・ヘルプを表示するには，次のようにします。 (最後に入力したコマンドがどのように実行されたかがわかります。)

```
$  HELP/MESSAGE
```

メッセージ識別子やメッセージ・テキストの中の単語を指定すれば，特定のメッセージについての情報も表示できます。たとえば，次のように入力します。

```
$  HELP/MESSAGE BADACP
```

メッセージ状態コードを入力すると，メッセージとその説明も表示できます。

```
$  HELP/MESSAGE/STATUS=%X00038090
```

メッセージ状態コードがわからない場合には， SHOW SYMBOL コマンドと$STATUS グローバル・シンボルを入力することにより表示できます。次の例を参照してください。

```
$  SHOW SYMBOL $STATUS
  $STATUS == "%X00038090"
```

5.5 コラム：画面出力をファイルに落とす

画面出力 (画面に表示されている文字列) をファイルに落として保存することができます。後で参照できるので便利です。

5.5.1 ヘルプのテキストをファイルに落とすには

```
$ LIBRARY/HELP SYS$HELP:HELPLIB.HLB/EXTRACT=出力したい項目/OUTPUT=ファイル名
```

（ヘルプを全部出力するには，/EXTRACT=*を指定します。）

5.5.2 システム・メッセージのテキストをファイルに落とすには

```
$ HELP/MESSAGE/OUTPUT=ファイル名　項目
```

項目を指定しない場合は，エラー等が起きた直後のシステム・メッセージが出力されます。

5.5.3 コマンドの実行結果をファイルに落とすには

いくつかの方法があります。

使用するコマンドの/OUTPUT 修飾子を使う

DCL コマンドの多くには，画面に表示される文字列を指定ファイル (任意の名前を付けられる) に出力するための/OUTPUT 修飾子があります。

```
$ DIRECTORY /OUTPUT=FILES.LIS
```

PIPE コマンドを使う

PIPE コマンドは UNIX や DOS のリダイレクト機能を提供します。/OUTPUT 修飾子のないコマンドに対して，画面出力をファイルに保存することができます。

```
$ PIPE DIRECTORY > FILES.LIS
```

SYS$OUTPUT をファイルに再定義する

出力デバイスを示す論理名 SYS$OUTPUT を変更し，ファイルに定義してしまう方法です。

```
$ DEFINE /USER_MODE SYS$OUTPUT FILES.LIS
$ DIRECTORY
```

DEFINE コマンドに/USER_MODE を指定すると，次に実行するコマンドに対してだけ論理名が有効になります。そのため DIR コマンドの実行が終わると，SYS$OUTPUT は元に戻り，正常に画面出力されるようになります。

TELNET コマンドなどのログ機能を使う

TELNET， RLOGIN， SET HOST などのネットワークコマンドは，画面に表示される文字をファイルに保存する機能を持っています。再ログインが必要になりますが，この方法はどのコマンドの実行結果もファイルに保存することができます。

```
$ TELNET /LOG_FILE=FILES.LIS TKO.HONSHA.COM
```

ターミナル・エミュレータのログ機能を使う

PC 等のターミナル・エミュレータから日本語 OpenVMS にログインしている場合は，ターミナル・エミュレータの持つログ機能を使うことができます。使い方は各ターミナル・エミュレータのマニュアルをお読みください。

6

キーボードを使用する

この章では

キーの割り当ては，使用するキーボードやソフトウェアによって異なります。ここでは，ワークステーション用の標準的なキーボード，パーソナル・コンピュータ (PC) のキーボードで Reflection や Teraterm Pro を使用する場合について説明します。

また，日本語 OpenVMS では，コマンド列をキーボードの 1 つのキーに割り当て，そのキーを押すだけでコマンド列を入力したようにさせることができます。これを，**キー定義**といいます。日本語 OpenVMS がデフォルトで割り当てているキーもありますが，ユーザがキーを定義することもできます。よく使用するコマンドを特定のキーに割り当てておくと便利です。

関連資料

- 『OpenVMS ユーザーズ・マニュアル』
- 『日本語 EVE リファレンス・マニュアル』
- 『日本語 EVE ユーザーズ・ガイド』
- 『日本語 OpenVMS ユーザ・キー定義 利用者の手引き』
- 『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』

6.1 キーボードとキーの割り当て

使用するキーボードによってキーの配列が異なります。また，キーの割り当てはキーボードや使用するソフトウェアなどによって異なります。

ここでは弊社のワークステーションの標準的なキーボードとパーソナル・コンピュータ (PC) の標準的なキーボードでの役割，割り当てについて説明します。

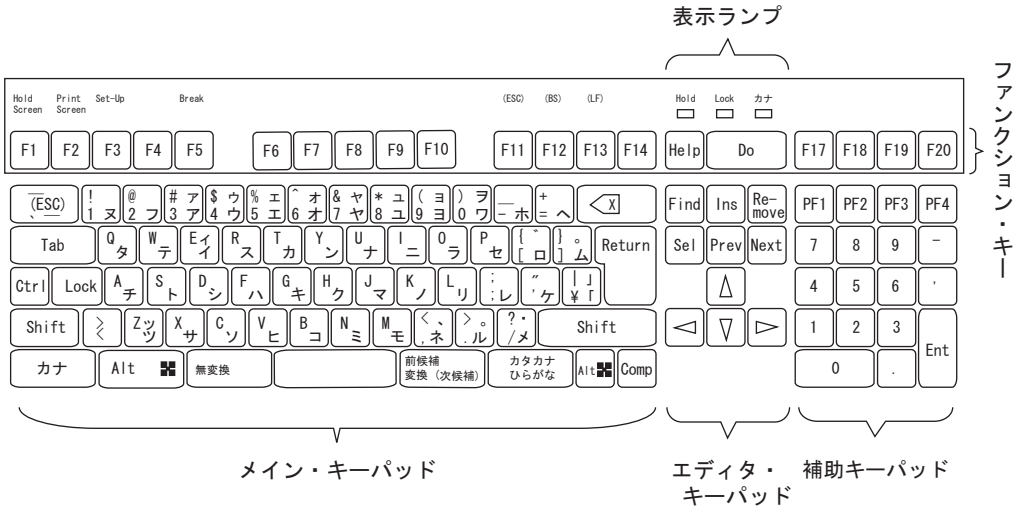
6.1.1 キーの表記

本書では，次のようにキーを表しています。

キー	説明
X	キーを表す。
Ctrl + X	コントロール・キーを押しながら，別のキーを押す。
PF1 → X	PF1 を押してから，別のキーを押す。
<X	デリート (1 文字削除) キーを表す。

6.1.2 キーボードの配列

図 6–1 キーボードの配列例



キーボードには，4つのキー・グループがあります。キー・グループは機能によって，次のように分かれています。

◆メイン・キーパッド

キーボードの左側の部分。英数字と記号，カタカナと句読点が含まれています。

◆エディタ・キーパッド

メイン・キーパッドのとなりにあるキーパッド。矢印キー (↑, ↓, →, ←) と Find, Select キーなどのエディタ・キーが含まれています。

◆補助キーパッド（テンキー）

キーボードの右側にある数値キーパッド。

◆ファンクション・キー

キーボードの上部に並んでいるキー。

6.1.3 日本語 OpenVMS (CDE) のキー割り当て

ワークステーション用の標準的なキーボードと日本語 OpenVMS (CDE) のキーの割り当ては，次のようになっています。日本語 EVE エディタを使用する際に選ぶ変換キーパッドや日本語 OpenVMS のさまざまなユーティリティによってキー割り当ては異なってきます。(詳しくは，日本語 EVE エディタやユーティリティの説明書を参照してください。)

図 6–2 日本語 OpenVMS (CDE) のキー割り当て

表 6–1 日本語 OpenVMS (CDE) の一般的なキー割り当て

キー	説明
ファンクション・キー (Fx)	特別な機能が割り当てられているキー。
F1	F1 キーを押すとキーボードの Hold インディケータが点燈し，画面表示が一時停止されていることを示す。もう一度 F1 キーを押すとインディケータは消え，画面表示は再開さる。
Shift	キーの上段にかかれた文字 (アルファベットの大文字，特殊記号など) を入力するときは，Shift キーを押しながら，該当するキーを押す。
Return または Enter	改行するときに押す。また，ある特定の動作が終了したことをシステムに知らせる役割もある。
Ctrl	かな漢字変換など，特定のキーに割り当てられた機能を実行させるときに使用。
Lock	このキーを押すと，キーボードの上部にある Lock 表示ランプが点燈する。この状態でアルファベット・キーを押すと，その大文字が表示される。もう 1 度 Lock キーを押すと， Lock 表示ランプが消え，アルファベット・キーを押すと小文字が表示される。
<X	カーソルの左側の文字を 1 文字消去する。
Tab	カーソルを次の (指定された) タブ位置まで移動。

(次ページに続く)

表 6–1 (続き) 日本語 OpenVMS (CDE) の一般的なキー割り当て

キー	説明
Do	日本語 EVE エディタでコマンドを入力する Command: プロンプトを表示。
Help	日本語 EVE エディタでキーについてのヘルプを表示。 Do キーを押して Command: プロンプト上に HELP と入力した場合は，日本語 EVE のコマンドのヘルプが表示される。
F14 または Ctrl + A	入力モードを切り替える。
PF1 (Gold ともいう)	かな漢字変換などで使われる。
Find	文字列を探す。
Insert Here	文字列を挿入。
Remove	文字列を削除。
Select	文字列を選択。
Prev	現在の画面の前に移動。
Next	現在の画面の後に移動。

DCL コマンド・ラインでよく使用されるキーの割り当ての一覧は表 I–7 を，日本語 EVE エディタで JVMS キーパッドを選択した場合のよく使用されるキー割り当ての一覧は表 7–2 をご覧ください。

6.1.4 Reflection のキー割り当て

パーソナル・コンピュータ (PC) 用の標準的なキーボードでの Reflection のキーの割り当ては，次のようになっています。

図 6–3 Reflection のキー割り当て

表 6–2 Reflection と日本語 OpenVMS (CDE) のキー割り当て対照表

日本語 OpenVMS (CDE) で用いるキー	対応する PC キーボードのキー	説明
ファンクション・キー	ファンクション・キー	上段はシフト・キーを押しながら該当するキーを押した時のキー割り当てを示す。下段はそのままキーを押した時のキー割り当てを示す。日本語 OpenVMS で用いる F3 ～ F4 キーは， Reflection では割り当てられていない。
F1	Scroll Lock	Scroll Lock キーを押すとキーボードの Scroll Lock インディケータが点燈し，画面表示が一時停止されていることを示す。もう一度 Scroll Lock キーを押すとインディケータは消燈し，画面表示は再開される。
Shift	Shift	キーの上段にかかれた文字 (アルファベットの大文字，特殊記号など) を入力するときは，Shift キーを押しながら，該当するキーを押す。
Return	Enter	改行するときに押す。また，ある特定の動作が終了したことをシステムに知らせる役割もある。
Ctrl	Ctrl	かな漢字変換など，特定のキーに割り当てられた機能を実行させるときに使用。
Lock	Shift+ Caps Lock	このキーを押すと，キーボードの上部にある表示ランプが点燈する。この状態でアルファベット・キーを押すと，その大文字が入力される。もう 1 度 Shift + Caps Lock キーを押すと，表示ランプが消え，アルファベット・キーを押すと小文字が入力される。
<X	Back Space	1 文字削除する。このキー割り当てを有効にするには， Reflection 側での設定が必要。Back Space キーを 1 文字削除キーに割り当てるためには，「端末の設定」ダイアログ・ボックス (図 2–9) で「VT Backspace キー」の項目を「削除」に設定。
Tab	Tab	カーソルを次の (指定された) タブ位置まで移動。
Do	Shift + F6	日本語 EVE エディタでコマンドを入力する Command: プロンプトを表示。

(次ページに続く)

表 6–2 (続き) Reflection と日本語 OpenVMS (CDE) のキー割り当て対照表

日本語 OpenVMS (CDE) で用いるキー	対応する PC キーボードのキー	説明
Help	Shift + F5	日本語 EVE エディタでキーについてのヘルプを表示。 Do キーを押して Command: プロンプト上に HELP と入力した場合は，日本語 EVE のコマンドのヘルプが表示される。
F14 または Ctrl + A	Shift + F4	入力モードを切り替える。
PF1 (Gold ともいう)	Num Lock	かな漢字変換などで使われる。 Reflection の場合， Num Lock キーを押している間はキーボードの Num Lock インディケータが点燈するが，これは無視する。
	F2	現在 Reflection ウインドウに表示されている端末画面が， PC に接続されているプリンタに印刷される。
Insert Here	Insert	文字列を挿入。
Find	Home	文字列を探す。
Prev	Page Up	現在の画面の前に移動。
Remove	Delete	文字列を削除。
Select	End	文字列を選択。
Next	Page Down	現在の画面の後に移動。

DCL コマンド・ラインでよく使用されるキーの割り当ての一覧は表 I–7 を，日本語 EVE エディタで JVMS キーパッドを選択した場合のよく使用されるキー割り当ての一覧は表 7–2 をご覧ください。

6.1.5 Tera Term Pro **のキー割り当て**

パーソナル・コンピュータ (PC) 用の標準的なキーボードでの Tera Term Pro のキー割り当ては，次のようになっています。

図 6–4 Tera Term Pro **のキー割り当て**

表 6–3 Tera Term Pro **と日本語** OpenVMS (CDE) **のキー割り当て対照表**

日本語 OpenVMS (CDE) **で用いるキー**	**対応する** PC **キーボードのキー**	**説明**
ファンクション・キー	ファンクション・キー	上段はシフト・キーを押しながら該当するキーを押した時のキー割り当てを示す。下段はそのままキーを押した時のキー割り当てを示す。
Shift	Shift	キーの上段にかかれた文字 (アルファベットの大文字，特殊記号など) を入力するときは，Shift キーを押しながら，該当するキーを押す。
Return	Enter	改行するときに押す。また，ある特定の動作が終了したことをシステムに知らせる役割もある。
Ctrl	Ctrl	かな漢字変換など，特定のキーに割り当てられた機能を実行させるときに使用。
Lock	Shift+ Caps Lock	このキーを押すと，キーボードの上部にある表示ランプが点燈する。この状態でアルファベット・キーを押すと，その大文字が入力される。もう 1 度 Shift + Caps Lock キーを押すと，表示ランプが消え，アルファベット・キーを押すと小文字が入力される。
<X	Back Space	1 文字削除する。このキー割り当てを有効にするには， Tera Term Pro 側での設定が必要。Back Space キーを 1 文字削除キーに割り当てるためには，「Keyboard Setup」ダイアログ・ボックス (図 2–5) で「Transmit DEL by」の項目を「Backspace Key」に設定する。
Tab	Tab	カーソルを次の (指定された) タブ位置まで移動。
Do	Shift + F6	日本語 EVE エディタでコマンドを入力する Command: プロンプトを表示する。
Help	Shift + F5	日本語 EVE エディタでキーについてのヘルプを表示。 Do キーを押して Command: プロンプト上に HELP と入力した場合は，日本語 EVE のコマンドのヘルプが表示される。
F14 または Ctrl + A	Shift + F4	入力モードを切り替える。

(**次ページに続く**)

表 6–3 (続き) Tera Term Pro と日本語 OpenVMS (CDE) のキー割り当て対照表

日本語 OpenVMS (CDE) で用いるキー	対応する PC キーボードのキー	説明
Gold または PF1	Num Lock	かな漢字変換などで使われる。 Tera Term Pro の場合， Num Lock キーを押すたびにキーボードの Num Lock インディケータが点いたり消えたりしますが，これは無視する。
Find	Insert	文字列を探す。
Insert Here	Home	文字列を挿入。
Remove	Page Up	文字列を削除。
Select	Delete	文字列を選択。
Prev	End	現在の画面の前に移動。
Next	Page Down	現在の画面の後に移動。

DCL コマンド・ラインでよく使用されるキーの割り当ての一覧は表 I–7 を，日本語 EVE エディタで JVMS キーパッドを選択した場合のよく使用されるキー割り当ての一覧は表 7–2 をご覧ください。

6.2 コマンド列をキーに割り当てる (キーを定義する) — DEFINE/KEY

コマンド列をキーボードの 1 つのキーに定義するには，次のようにします。

```
$ DEFINE/KEY  キー名  コマンド列
              1       2
```

1 キー名

コマンド列を割り当てるキーの名前

2 コマンド列

割り当てるコマンド列

実際のコマンド列と同じ形式で指定する。コマンド列の中に空白文字が含まれる場合，コマンド列を二重引用符 (") で囲む

修飾子には次のようなものがあります。

/IF_STATE=(状態名,・・・)	キーを使用したときに設定されている状態名が，指定したリストにあれば，この定義を使用する。
/NOLOG	キー定義したときのシステム・メッセージの出力をさけることができる。
/SET_STATE=状態名	キーを使用したときに，指定した状態名を設定する。
/TERMINATE	キーを入力するだけですぐにコマンドを実行する。

次にキーを定義した例を示します。

PF1 キーに， "SHOW DEFAULT"を割り当てます。

```
$ DEFINE/KEY PF1 "SHOW DEFAULT"
%DCL-I-DEFKEY, DEFAULT 状態の PF1 キーが定義されました。
```

PF1 キーの定義を確認します。

```
$ SHOW KEY PF1
DEFAULT keypad definitions:
  PF1 = "SHOW DEFAULT"
```

PF1 キーを押します。

```
$ PF1
```

PF1 キーを押した行に "SHOW DEFAULT"と表示されます。このコマンドを実行する場合は，Return キーを押します。

```
$ SHOW DEFAULT
```

6.3 定義したキーを参照する — SHOW KEY

DEFINE/KEY コマンドを使用してキー定義ができたかどうかを確認する場合は，SHOW KEY コマンドを使用します。

すべてのキー定義を表示する場合は，次のように入力します。

```
$ SHOW KEY/ALL
```

特定のキー定義を表示する場合は，次のように入力します。

```
$ SHOW KEY キー名
```

修飾子には次のものがあります。

/DIRECTORY	存在する状態名の一覧を表示する。
/FULL	詳細な情報付きで表示する。
/STATE=[状態名,...]	指定した状態で使われる定義が参照できる。

6.4 キー定義を無効にする — DELETE/KEY

一度定義したキーを無効にするには，次のようにします。

```
$ DELETE/KEY [/修飾子] [キー名]
```

キー名を指定しないで/ALL 修飾子を付けると，現在の状態名のすべてのキー定義を削除できます。

修飾子には次のようなものがあります。

/STATE=[状態名,...]	指定した状態名のキー定義を削除できる

```
$ SHOW KEY/ALL
DEFAULT keypad definitions:
  PF1 = "SHOW "
$ SHOW KEY/STATE=GOLD/ALL
GOLD keypad definions:
  PF1 = "PROCESS"
$
```

デフォルト状態でのキー定義をすべて削除します。

```
$ DELETE/KEY/ALL
%DCL-I-DELKEY, DEFAULT key PF1 has been deleted
```

GOLD 状態でのキー定義をすべて削除します。

```
$ DELETE/KEY/STATE=GOLD/ALL
%DCL-I-DELKEY, GOLD key PF1 has been deleted
$
```

デフォルト状態でのキー定義がすべて削除されています。

```
$ SHOW KEY/ALL
DEFAULT keypad definitions:
```

GOLD 状態でのキー定義がすべて削除されます。

```
$ SHOW KEY/STATE=GOLD/ALL
GOLD keypad definitions:
$
```

一般に，キー定義はログアウトすると自動的に削除されます。

7

エディタを使用する

この章では

日本語 OpenVMS 上で動く対話型のエディタを使用したり，コマンドを用いることによって，新しいファイルの作成や既存のファイルを編集することができます。

日本語 OpenVMS では，日本語を使用できるエディタとして**日本語** EVE (Extended Extensible Versatile Editor) (JEVE と呼ぶこともあります) を提供しています。

ここでは，基本的なエディタの使用方法と，日本語 EVE を使用してファイルを作成する方法について説明します。また， DCL コマンドを使用して，複数のファイルの内容を連結，追加したり，ファイルの内容を並べかえる方法についても説明します。

関連資料

- 『日本語 EVE ユーザーズ・ガイド』
- 『日本語 EVE かな漢字変換入門』
- 『日本語 EVE リファレンス・マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』

7.1 日本語 EVE エディタを起動する — EDIT/XTPU

エディタを起動するには，$プロンプトに続けて次のように入力します。

```
$ EDIT/XTPU［ファイル名］
```

指定された名前のファイルが存在しない場合には[End of file] という文字のみが表示されます。ここでファイルに書き込む内容を入力してください。すでに存在するファイル名を指定すると，その内容が表示されます。

7.2 日本語 EVE エディタを使用する

初めて日本語 EVE を起動したときには，次のような初期画面があらわれます。

注意

この画面は，日本語 EVE **を初めて起動したときにのみ表示されます。** 2 **回目以降はこの画面は表示されず，起動後すぐに編集画面があらわれます。**

7.2.1 かな漢字変換キーパッドを設定する

日本語 EVE エディタの初期画面で，日本語を入力するための変換キーパッドを選びます。

たとえば JVMS キーパッドを選ぶには，"**変換キーパッド名を入力してください**" の後に次のように JVMS と入力します。

```
変換キーパッド名を入力してください（EVEJ/JVMS/TARO/JEDI/LEIA): JVMS
```

このとき選択したキーパッドは，2 度目以降の起動にも反映されます。次回に日本語 EVE を起動するときには，すぐに編集画面が表示され，変換キーパッド名を聞かれません。

7.2.2 変換キーパッドを変更する — SET KEYPAD

変換キーパッドを変更するには，SET KEYPADコマンドを使用します。

1. 編集画面で Do キーを押してCommand: プロンプトを表示させます。

注意

Doキーは，PC ではShift + F6を使用します。

2. Command: プロンプトに次のように使用したいキーパッド名を入力します。

   たとえば EVEJ キーパッドに変更したい場合は次のようにします。

```
Command:  SET KEYPAD EVEJ
```

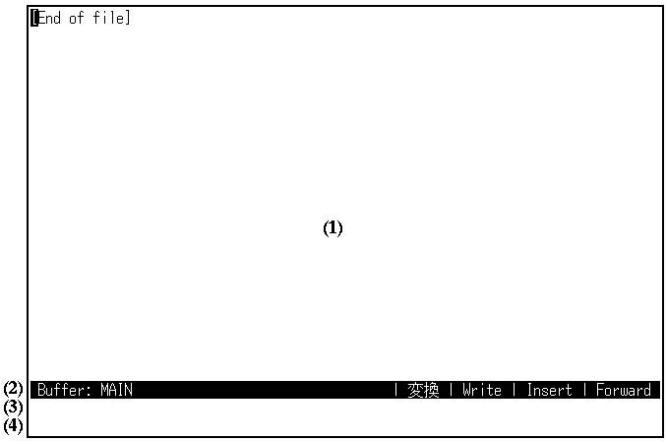
3. Return キーを押します。

本書では，JVMS キーパッドを選択した場合についてのみ説明します。その他のキーパッドを選択した場合については『日本語 EVE ユーザーズ・ガイド』をお読みください。

7.2.3 日本語 EVE エディタの編集画面

変換キーパッド名を入力すると，図 7–1 のような編集画面があらわれます。

図 7–1 日本語 EVE の編集画面



(1)	テキスト・ウィンドウ	テキストを入力する部分です。 画面では，[End of file] という文字列が左上に表示されます。この文字列は，"ファイルの終わり "という意味です。テキストを入力すると，[End of file] は画面の下の方へ自動的に移動して行きます。
(2)	ステータス・ライン	編集中のモードをあらわします。
(3)	コマンド・ウィンドウ プロンプト・ウィンドウ	日本語 EVE のコマンドを入力する部分です。日本語 EVE が入力を要求するときに，プロンプトが表示されます。
(4)	メッセージ・ウィンドウ	日本語 EVE からのメッセージが表示されます。

7.3 日本語 EVE エディタを終了する

エディタを終了するには，次のようにします。

ファイルを保存する場合

Ctrl + Z キーを押します。(または，Do キーを押し，Command: プロンプトで EXIT を入力し，Return キーを押します。)

注意

Do キーは，PC では Shift + F6 を使用します。

ファイルを保存しない場合，または編集内容を保存しない場合

Do キーを押し，Command: プロンプトで QUIT を入力し，Return キーを押します。

ファイルが変更されている場合には，以下のようなメッセージが表示されます。

```
セーブされていないバッファがあります，QUIT しますか [Yes]?
```

ここで Yes を入力するとファイルの内容を保存せずに日本語 EVE を終了し， No を入力すると編集画面に戻ります。

ファイル指定を省略して日本語 EVE を起動したときの終了方法

日本語 EVE を起動するときにファイル指定を省略することができます。この場合，日本語 EVE を正常終了させる際に次のメッセージが表示されます。

```
MAIN バッファのファイル名を入力してください（キャンセル→RETURN）：
```

ここでファイル名を入力するとその名前で内容が保存され，ファイル名を入力せず Return キーのみの場合は，内容は保存されずに日本語 EVE が終了します。

7.4 文字を入力する

文字を入力するには，カナで入力する方法（**カナ入力**）とローマ字で入力する方法（**ローマ字入力**）の 2 種類があります。

◆カナ入力

キーボードに表示されているカタカナに従って入力します。

(カナ入力の方法については第 B.4 節を参照。)

◆ローマ字入力

キーボードに表示されているアルファベットに従ってローマ字の読みで入力します。

たとえば，弊社の標準的なワークステーション用のキーボードでローマ字入力をするには，カナ表示ランプを確認し，点灯していたら，Ctrl + **カナ** キーを押して解除してください。次にキーボードのアルファベット・キーを押すと，画面上にアルファベットが表示されます (ただし， TARO キーパッドを選択した場合は，ローマ字入力でもひらがなが表示されますので，ご注意ください)。

まちがえた場合は，⌫ キー (または Back Space キー) で消して，入力し直します。

7.4.1 PC の日本語入力機能を使って文字を入力する場合

パーソナル・コンピュータ (PC) の日本語入力機能 (IME など) を用いて日本語を入力する場合，標準では日本語 EVE は入力された日本語を正しく受け取ることができません。日本語 EVE が入力された日本語を正しく受け取るためには，次の 2 つの設定が必要です。

（あらかじめ日本語環境設定ユーティリティまたは KANJIGEN ユーティリティでターミナルの漢字入力を有効にしている場合は，日本語 EVE を起動した時点で PC の日本語入力機能が有効になっているので，下記 1. の設定は必要ありません。詳しくは付録 B，または『日本語ユーティリティ利用者の手引き』，『フォント管理ユーティリティ利用者の手引き』を参照してください。）

1. ターミナルからの漢字入力を有効にする

   Do キーを押し， Command: プロンプトの後に以下のコマンドを入力します。

   注意

   Do キーは，PC では Shift + F6 を使用します。

```
Command: SET INPUT MODE KANJI
```

2. 日本語 EVE のかな漢字変換を無効にする

   Do キーを押し， Command: プロンプトの後に以下のコマンドを入力します。

```
Command: HENKAN MODE OFF
```

   日本語 EVE のステータス・ラインから「変換」の文字が消え，日本語 EVE のかな漢字変換機能が無効になったことがわかります。これで PC の日本語入力機能によるかな漢字変換を行うことができます。

   日本語 EVE のかな漢字変換を有効に戻すには以下のコマンドを入力します。

```
Command: HENKAN MODE ON
```

7.5 入力モードの切り替え

文字を入力するとき，**挿入モード**と**重ね書きモード**のどちらかを選択できます。

日本語 EVE を起動したときは，自動的に挿入モードに設定されています。モードを切り替えるには，F14 キー (PC では Shift + F4) か Ctrl +A キーを押してください。

◆挿入モード

挿入モードのときは，入力した文字がカーソルの位置に挿入されます。すでに入力済みの文字の上にカーソルがある場合には，入力した文字がその位置に挿入され，入力済みの文字が右にずれていきます。

◆重ね書きモード

重ね書きモードのときは，入力した文字がカーソルの位置に上書きされます。すでに入力済みの文字の上にカーソルがある場合には，入力した文字が入力済みの文字の上に重ね書きされ，入力済みの文字が消えていきます。

現在，どちらのモードに設定されているのか知りたいときは，ステータス・ライン (図 7–1 を参照) の表示を確認します。ステータス・ライン上に Insert と表示されている場合は挿入モードに， Overstrike と表示されている場合は重ね書きモードに設定されています。

7.6 文字を変換する

日本語 EVE エディタでは，選択したキーパッドにより，文字の変換方法が少し違います。ここでは JVMS キーパッドを選び，ローマ字入力をする場合で説明します。その他のキーパッドについての説明は『日本語 EVE ユーザーズ・ガイド』をご覧ください。また，本書での説明は，おもに弊社の標準的なワークステーション (CDE) のキーボードを使用した場合を想定しています。

7.6.1 JVMS キーパッドの変換キー

表 7–1 は，入力した文字列を変換するための JVMS キーパッドの変換キーを示しています。

表 7–1 JVMS キーパッドの変換キー

変換の種類	変換キー
漢字変換	Ctrl + スペース
ひらがな変換	Ctrl + L
カタカナ変換	Ctrl + K
半角カタカナ変換	Ctrl + G → Ctrl + K
全角変換	Ctrl + F
半角変換	Ctrl + G → Ctrl + F
無変換	Ctrl + N

7.7 入力文字変換操作

SAMPLE.TXT というファイルを例に説明します。 (ここではローマ字入力で文字を入力しています。)

日本語 EVE を起動して， SAMPLE.TXT というファイルを作成します。

```
$ EDIT/XTPU SAMPLE.TXT
```

日本語 EVE の編集画面があらわれます。

7.7.1 ひらがな変換 Ctrl + L

まず最初に， konnichiha と入力します。

入力文字列は，高輝度表示されています。この状態で，Ctrl + L (Ctrl キーを押しながら，L を押す) を押します。 konnichiha がひらがなに変換されます。

7.7.2 カタカナ変換 Ctrl + K

ki-paddo と入力します。

入力文字列は，高輝度表示されています。この状態で，Ctrl + K を押します。
ki-paddo がカタカナに変換されます。

7.7.3 全角変換 Ctrl + F

DIGITAL と入力します (大文字を入力するときは Shift キーを押しながらまたは Lock キーを押して，文字を入力します)。

入力文字列は，高輝度表示されています。この状態で，Ctrl + F を押します。
DIGITAL が全角に変換されます。

■ 高輝度表示 ＝ 変換可能な状態 ■

ここで，変換キーを押したあとも，入力文字列が高輝度表示されたままなのに注目してください。入力文字列は，変換が確定されるまで (高輝度表示されている間) は，何

度でも変換可能です。 DIGITAL が高輝度表示された状態で，Ctrl + L, Ctrl + K, Ctrl + G → Ctrl + K, Ctrl + F) と順番に押してみて，変換されることを確認してください。

7.7.4 半角変換 Ctrl + G → Ctrl + F

全角のＤＩＧＩＴＡＬの文字が高輝度表示された状態 (変換を確定していない状態) で，Ctrl + G を押し，続けて Ctrl + F を押してください。 DIGITAL が半角に変換されます。

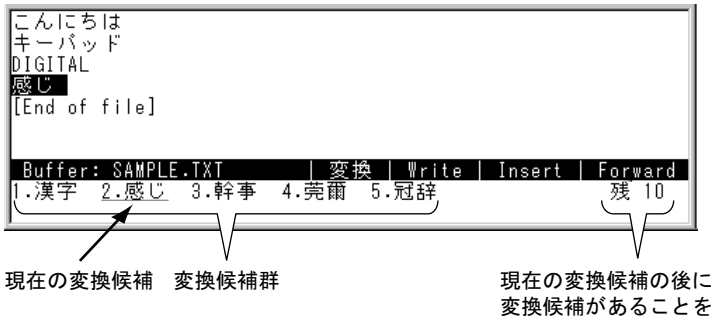
7.7.5 漢字変換 Ctrl + スペース

kanji と入力して，Ctrl + スペース を押します。 kanji が "漢字 "に変換されます。

注意

漢字変換の結果は，ユーザの個人辞書によって，若干異なります。必ずしも，変換結果が例のとおりになるとは限りませんので，ご注意ください。

ここでもう 1 度，変換キー (Ctrl + スペース) を押してください。変換結果が，"感じ "に変わり，画面の下のほう (ステータス・ラインの 1 行下) に，変換候補群が表示されます。



■ 変換候補の選び方 ■

変換キー (Ctrl + **スペース**) を連続して押すと，表示された候補の順にしたがって，変換結果が変化していきます。

1 つ前の変換結果に戻すときは，Ctrl + G を押してから，Ctrl + **スペース** を押します。

変換候補を数字キーで選ぶこともできます。たとえば，"監事 "に変換したいときは，数字キーの7 を押します。

変換候補群の中に期待する変換結果が見つからないときは，Next キーまたは Prev キーで他の変換候補群を表示して探してください。

7.7.6 日本語 EVE の終了 Ctrl + Z

日本語 EVE を終了するときは，Ctrl + Z を押します。これで日本語 EVE エディタの使用は終了し， SAMPLE.TXT というファイルが作成されます。

ファイルを保存しない場合は，Do キーを押して，画面下のコマンド・ライン にQUITコマンドを入力してください。次に Return を押すと，ファイルを作成せずに編集セッションを終了します。

注意

Do キーは， PC では Shift + F6 を使用します。

7.8 文字 (行) を削除/挿入する

文字や行の削除，挿入操作に使用するキーには，次のものがあります。

キー	機能
<X	カーソルの左側の文字が削除される。 (PC では Back Space キー)
Select	文章中のある領域を選択できる。選択を取り消すときは，もう 1 度 Select キーを押す。 (Reflection では End キー， Tera Term Pro では，Delete キー)
Remove	選択した部分を 1 度に削除できまる。 (Reflection では Delete キー， Tera Term Pro では，Page Up キー)
Insert Here	Remove キーによって削除した部分を，文章中の任意の場所にそのまま挿入できる。 (Reflection では Insert キー， Tera Term Pro では，Home キー)

7.9 日本語 EVE のコマンドを入力してテキストを編集する

日本語 EVE には，テキストの編集やカーソルの移動などの操作を行うために，日本語 EVE コマンドが用意されています。日本語 EVE コマンドの入力には，次の 2 つの方法があります。

- 定義済みのキーを押す方法
- Command: プロンプトに対してコマンドを入力する方法

7.9.1 日本語 EVE コマンドを入力するための定義済みキーを使用する

日本語 EVE は省略時の設定として，一部のキーに日本語 EVE コマンドを定義しています。定義済みキーは，それぞれ 1 つの編集コマンドを実行します。

定義済みキーは，キーパッドによって異なります。表 7–2 は編集でよく使用される日本語 EVE コマンドと JVMS キーパッドで定義されているキーの一覧です。 JVMS キーパッド以外のキーパッドでの定義済みキーについては，『日本語 EVE ユーザーズ・ガイド』をご覧ください。

表 7–2 JVMS キーパッドの便利な定義済みキー

	定義済みキー		
日本語 EVE コマンド	日本語 OpenVMS(CDE)	Reflection	Tera Term Pro
TOP	Ctrl + G → ↑	Ctrl + G → ↑	Ctrl + G → ↑
BOTTOM	Ctrl + G → ↓	Ctrl + G → ↓	Ctrl + G → ↓
START OF LINE	Ctrl + H または Ctrl + G → ←	Ctrl + H または Ctrl + G → ←	Ctrl + H または Ctrl + G → ←

(次ページに続く)

表 7–2 (続き) JVMS キーパッドの便利な定義済みキー

日本語 EVE コマンド	定義済みキー 日本語 OpenVMS(CDE)	Reflection	Tera Term Pro
END OF LINE	Ctrl + E または Ctrl + G → →	Ctrl + E または Ctrl + G → →	Ctrl + E または Ctrl + G → →
CODE	Ctrl + G → X	Ctrl + G → X	Ctrl + G → X
KIGOU	Ctrl + G → Z	Ctrl + G → Z	Ctrl + G → Z
DO	Do または PF4	Shift + F6	Shift + F6
EXIT	Ctrl + Zまたは F10	Ctrl + Z または F10	Ctrl + Z または F10
HELP KEYPAD	HELP	Ctrl + F5	Ctrl + F5
TAB	TAB	TAB	TAB
DELETE	<x	Back Space	Back Space
RETURN	Return または Enter	Enter	Enter
FIND	Find	Home	Home
INSERT HERE	Insert Here	Insert	Insert
REMOVE	Remove	Delete	Page Up
SELECT	Select	End	Delete
NEXT SCREEN	Next	Page Down	Page Down
PREVIOUS SCREEN	Prev	Page Up	End
NEXT WINDOW	Ctrl + G → Next	Ctrl + G → Page Down	Ctrl + G → Page Down
PREVIOUS WINDOW	Ctrl + G → Prev	Ctrl + G → Page Up	Ctrl + G → End
RECALL	Ctrl + B	Ctrl + B	Ctrl + B
REMEMBER	Ctrl + R	Ctrl + R	Ctrl + R
ERASE START OF LINE	Ctrl + U	Ctrl + U	Ctrl + U
ERASE WORD	Ctrl + J または F13	Ctrl + J または F13	Ctrl + J または F13
RESTORE WORD	Ctrl + G → F13	Ctrl + G → Shift + F3	Ctrl + G → Shift + F3

7.9.2 Command: プロンプトで日本語 EVE コマンドを入力する

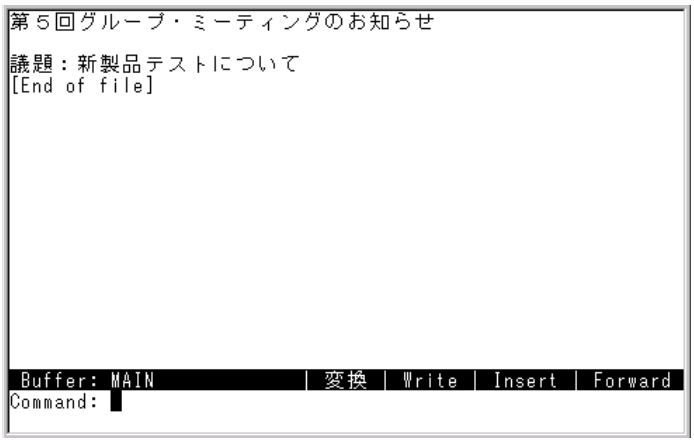
Command: プロンプトに対して日本語 EVE コマンドを入力するには，次のようにします。

1. **Do キーを押す**

   Do キーを押すと，日本語 EVE は，ステータス・ラインのすぐ下のコマンド・ウィンドウにCommand: というプロンプトを表示します。

注意

Doキーは，PC ではShift + F6を使用します。



2. Command: **プロンプトに対して，日本語** EVE **コマンドを入力する**

   Command: プロンプトに続いて，日本語 EVE コマンドを入力します。コマンドは大文字でも小文字でもかまいません。

   次の例では，EXITコマンドを入力しています。

   ```
   Command: EXIT
   ```

   このとき，さらにパラメータの指定が必要な場合は，コマンド・ウィンドウに入力を要求するプロンプトが表れます。

   次の例では， 15 行目に移動するためにLINEコマンドを入力し，次に表示されるLine Number: プロンプトに対して 15 を入力しています。

   ```
   Command: LINE
   Line Number: 15
   ```

3. Return **キー (または** Do **キー) を押す**

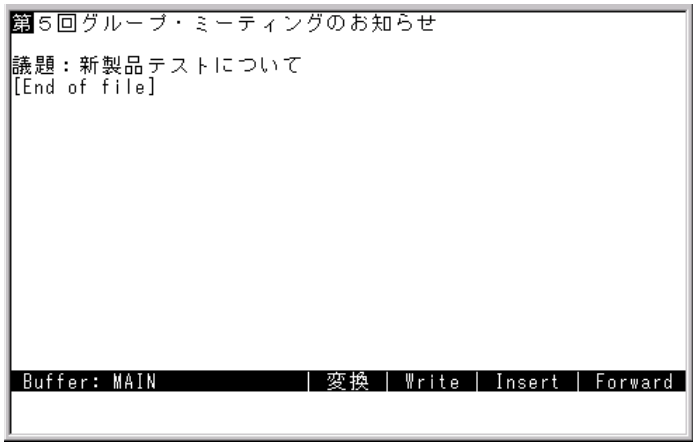
   注意

   Do キーは， PC **では** Shift + F6 **を使用します。**

   これで日本語 EVE は，コマンドを実行します。

7.9.3 日本語 EVE コマンドの実行例

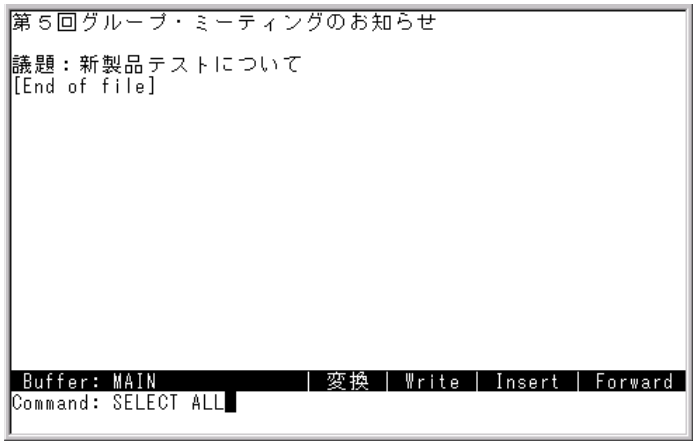
現在の編集画面が次のようになっているとします。



Do キーを押します。次にCommand: プロンプトに続いて， SELECT ALL と入力します。

注意

Do キーは，PC では Shift + F6 を使用します。



Return キーを押すと，コマンドが実行されます。この例ではテキストがすべて選択された状態になります。(Select キーを押すと選択が解除され元に戻ります。)

7.10 カーソルを移動する

日本語 EVE エディタでは，選択したキーパッドによりカーソルを移動する方法が少し違います。ここでは， JVMS キーパッドを選んだ場合で説明します。

表 7–3 は， JVMS キーパッドでカーソルを移動するための定義済みキーと，対応する日本語 EVE コマンドを示しています。

表 7–3 JVMS キーパッドのカーソル移動に関する定義済みキー

編集キー	日本語 EVE コマンド	機能
↑	MOVE UP	カーソルを 1 行上に移動
↓	MOVE DOWN	カーソルを 1 行下に移動
←	MOVE LEFT	カーソルを 1 文字，あるいは 1 カラム左に移動
→	MOVE RIGHT	カーソルを 1 文字，あるいは 1 カラム右に移動
Ctrl + H または Ctrl + G → ←	START OF LINE	カーソルを現在行の先頭に移動
Ctrl + E または Ctrl + G → →	END OF LINE	カーソルを現在行の最後に移動
Ctrl + G → ↑	TOP	カーソルを現在のバッファの先頭に移動
Ctrl + G → ↓	BOTTOM	カーソルを現在のバッファの最後に移動
F11	CHANGE DIRECTION	現在のバッファの方向 (カーソルの移動方向) を変更する。バッファの方向はステータス・ラインに表示 (Forward または Reverse) される。
F14 または Ctrl + A	CHANGE MODE	現在のバッファの入力モードを切り替える。入力モード (Insert または Overstrike) はステータス・ラインに表示される。

カーソル移動の設定

カーソルの移動の設定には，次の 2 があります。日本語 EVE コマンド・ラインで好みの設定にすることができます。

SET CURSOR BOUND

バッファの使用されていない部分（余白）に移動できない設定です。たとえば，カーソルが行の最後にある時に→キーを押すと，カーソルは行の先頭に移動します。カーソルが右マージンより右に移動することはありません。

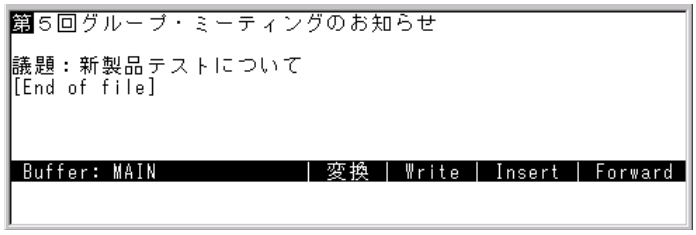
SET CURSOR FREE

デフォルトの設定です。文字がすでにあるかどうかとは無関係に，カーソルはバッファ内のどこにでも移動できます。

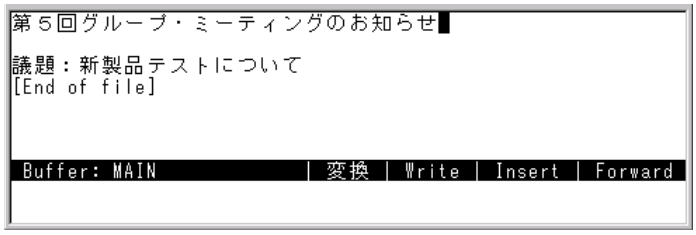
7.10.1 カーソル移動の例

現在の編集画面が次のようになっているとします。

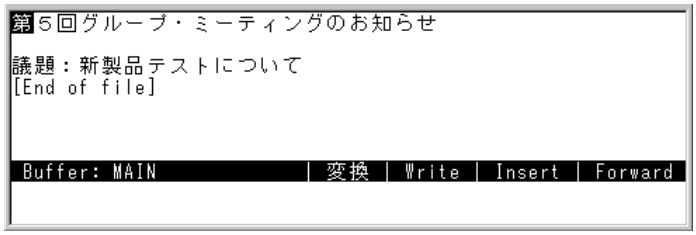
カーソルは，ユーザが挿入したテキストの最後に移動します。カーソルをファイルの先頭に移動するには，Ctrl + G を押したあと，↑ キーを押します。



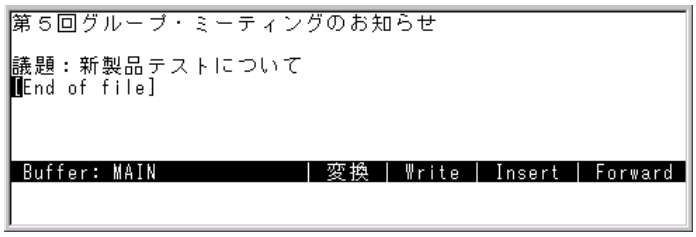
カーソルを行の最後に移動するには，Ctrl + E を押します。



カーソルを行の最初に移動するには，Ctrl + H を押します。



カーソルをバッファの最後に移動するには，Ctrl + G を押したあと，↓ キーを押します。



7.11 日本語 EVE コマンド名の短縮

コマンド名は短縮することもできます。他のコマンド名と区別できない短縮形を入力すると，対応するコマンドのリストが表示され，適切なコマンドを選択するように促すプロンプトが表示されます。もう 1 度，他のコマンド名と区別できるように入力し直してください。

7.12 文字列を探す — FIND

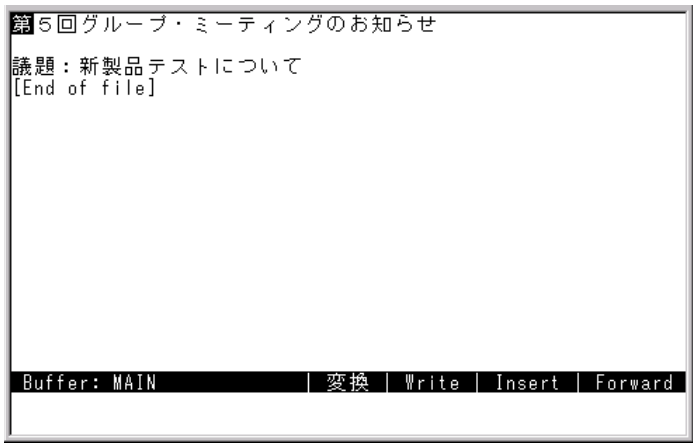
文章中から特定の文字列を探し出すには，FIND コマンドを使用します。

Do キーを押して FIND と入力し，Return キーを押します。プロンプトが表示されたら探したい文字列を入力します。カーソルが検索された文字列に移動します。なお，FIND コマンドは，Find キー (Reflection では Home キー，Tera Term Pro では Insert キー) を押すことによって同様に実行できます。

注意

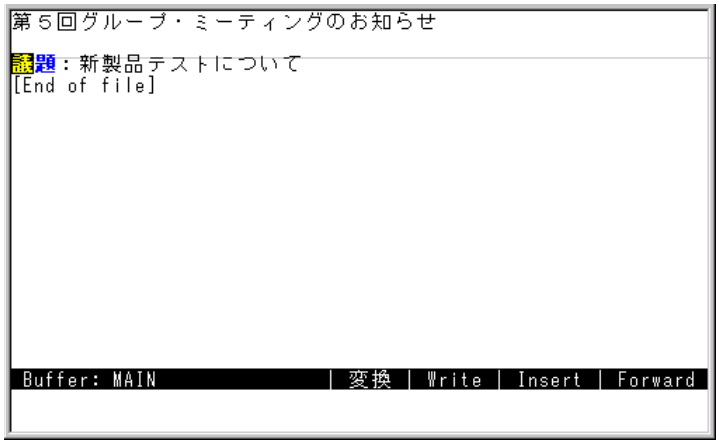
Do キーは，PC では Shift + F6 を使用します。

現在の編集画面が次のようになっているとします。



FIND コマンドを入力します。Find (**順方向**): プロンプトが表示されるので，このプロンプトに続けて，"議題 "と入力してください。

Return キーを押すと，文章中の "議題 "が検索され，高輝度表示になります。



7.12.1 アルファベットの文字列を探す

アルファベットの文字列を探したい場合，Find (**順方向**): プロンプトに続けてすべて小文字で入力すると，日本語 EVE は大文字と小文字の区別を無視して，一致するすべての文字列を検索します。

入力した文字列に 1 文字でも大文字が含まれていると，日本語 EVE は大文字と小文字を正確に区別して検索します。たとえば，プロンプトに続けて tHis と入力した場合には， tHis のみが検索され， This や THIS などは検索されません。

7.13 文字列を置き換える — REPLACE

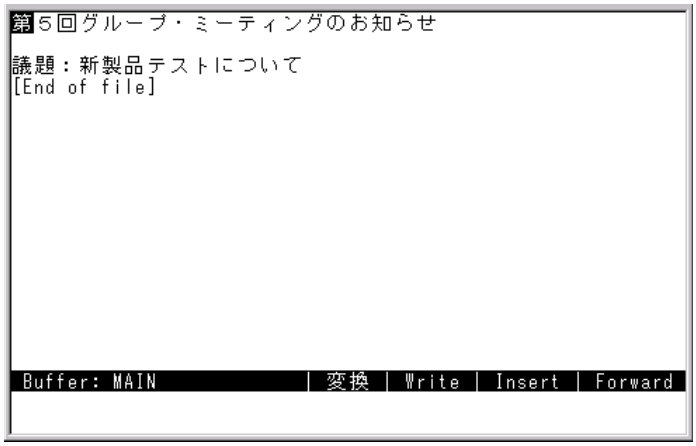
文章中から特定の文字列を選び，指定した別の文字列に置き換えるには， REPLACE コマンドを使用します。

Do キーを押して， REPLACE と入力し，Return キーを押します。プロンプトが表示されたら，順番に置換される文字列，置換する文字列をそれぞれ入力します。

注意

Do キーは， PC では Shift + F6 を使用します。

現在の編集画面が次のようになっているとします。

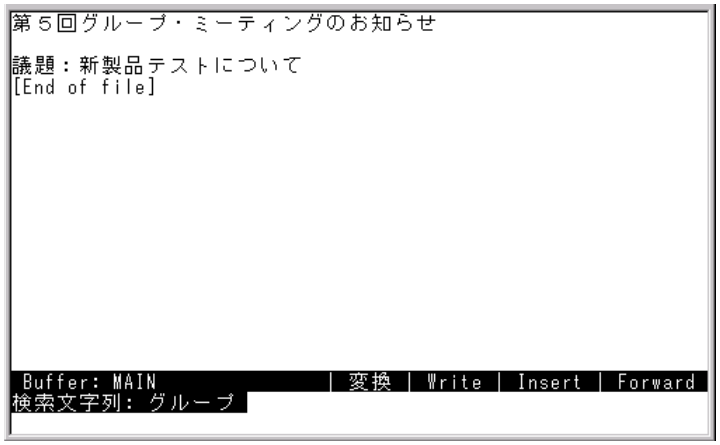


"グループ"という文字列を"ぐるーぷ"に置き換えてみましょう。Do キーを押して，REPLACE と入力します。

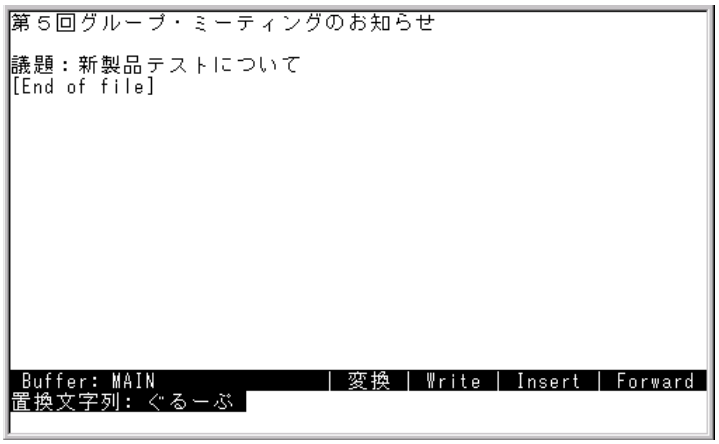
注意

Do キーは，PC では Shift + F6 を使用します。

Return キーを押すと，**検索文字列**: というプロンプトが表示されます。プロンプトに続けて，"グループ"と入力します。



Return キーを押すと，今度は，**置換文字列**: というプロンプトが表示されます。プロンプトに続けて，"ぐるーぷ"と入力します。

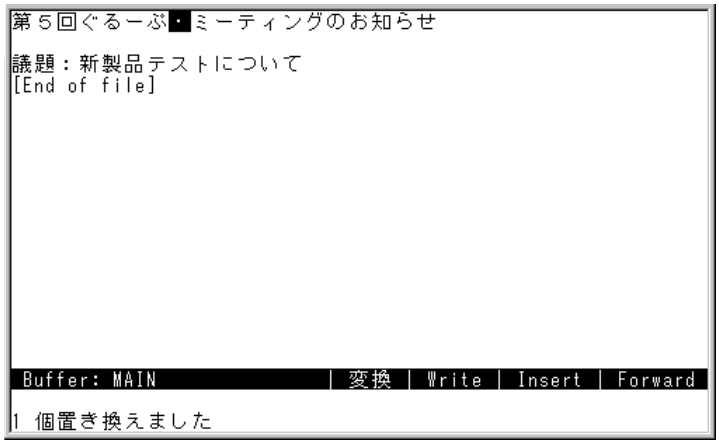


Return キーを押すと，文章中の対応する文字列が反転表示され，**置き換えますか？**というプロンプトが表示されます。

ステータス・ラインに表示された (Yes, No, All, Last, Quit) という選択肢から， 1 つを選んで入力します。選択肢は，それぞれ次のように動作します。

応答	結果
Yes	反転表示された文字列を置き換え，次に対応する文字列を検索する。
No	反転表示された文字列を置き換えずに，次に対応する文字列を検索する。
All	ファイル中の対応するすべての文字列を置き換える。
Last	反転表示された文字列を置き換え，ここで検索を中止する。
Quit	反転表示された文字列を置き換えずに，ここで検索を中止する。

ここでは，プロンプトに続けて， YES と入力して，Return キーを押してください。文字列が置き換えられます。



7.13.1 アルファベットの文字列を置き換える

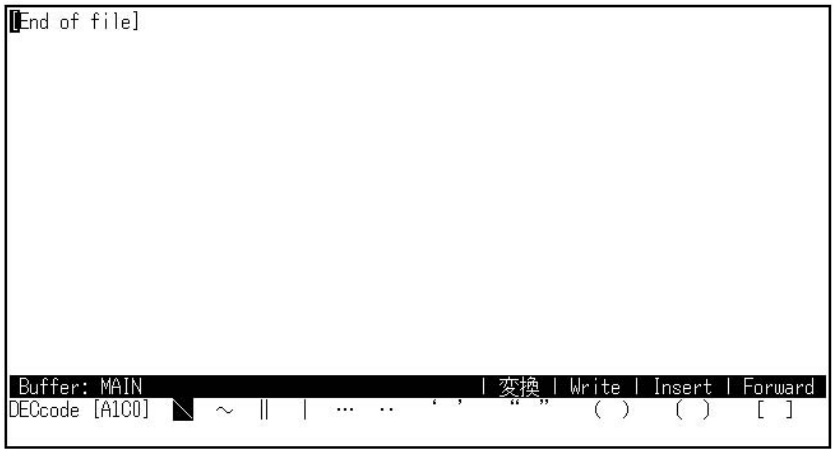
アルファベットの文字列を置き換えたい場合，プロンプトに続けて，すべて小文字で入力すると，日本語 EVE は大文字と小文字の区別を無視して，一致するすべての文字列を置き換えます。

入力した文字列に 1 文字でも大文字が含まれていると，日本語 EVE は大文字と小文字を正確に区別して置き換えます。

7.14 記号を入力する — KIGOU

文字入力モードの記号またはコードを使用して， DEC 漢字コード表に含まれている記号 (たとえば，〇や★など) を入力することができます。記号を入力するには，KIGOU コマンドを使用します。

KIGOUコマンドを実行すると，画面の 1 番下 (ステータス・ラインの下) に DEC 漢字コード一覧表が 1 行ずつ表示されます。



矢印キー (↑, ↓, →, ←) で上下左右にカーソルを移動できます。Return キーを押すと，カーソルが置かれている記号を入力して，コマンドを終了します。Selectキーを使えば，コマンドを終了せずに記号を何文字でも入力することができます。記号を入力しないでコマンドを終了するときはCtrl + Z を押してください。

7.14.1 **コード番号で記号を入力する** — KIGOU BY CODE

入力したい記号のコード番号がわかっている場合には，KIGOU BY CODEコマンドが便利です。

KIGOU BY CODEコマンドを実行すると，ステータス・ラインのDECcode [ ] 内にコード番号を入力できるようになります。コード番号は『漢字コード表』で調べることができます。

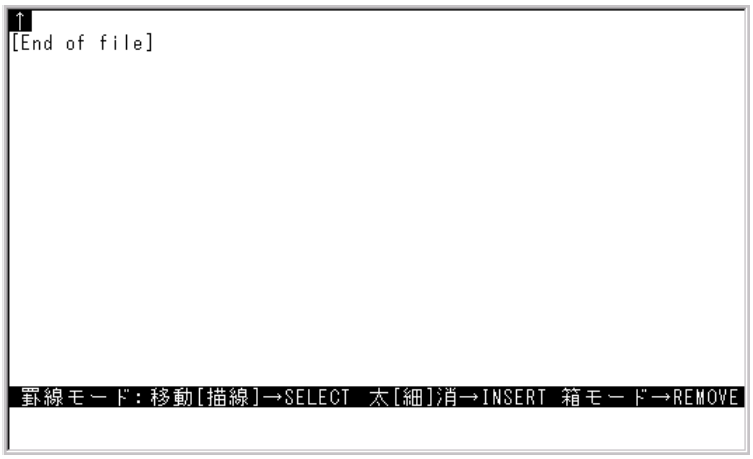
7.15 **罫線モードを使用する** — DRAW KEISEN

DRAW KEISEN コマンドを実行すると，罫線機能を使用できます。図や表を作成するのに便利です。

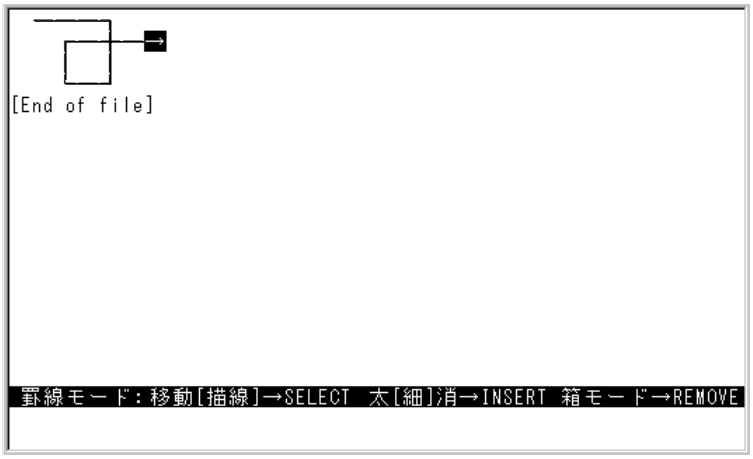
Do キーを押して， DRAW KEISEN と入力し，Return キーを押してください。ステータス・ラインの表示が次のようになり，カーソル位置に矢印が表示されます。

注意

Doキーは，PC ではShift + F6を使用します。



矢印キー ↑ ↓ ← → を使用して罫線を引きます。



移動/[描線]→ SELECT

Select キーを使用して，移動/描線を切り替えます。 "移動 "を選択すると，罫線を引かずにカーソルを移動できます。 "描線 "を選択すると，カーソルを移動して罫線を引くことができます。

太[細]消→ INSERT

Insert Here キーを使用して，罫線の種類を指定できます。 "太 "を選択すると太い罫線， "細 "を選択すると細い罫線， "消 "を選択すると，罫線消去モードになります。

箱→ REMOVE

Remove キーを使用して，箱モードに切り替えます。Remove キーを押すと，カーソル位置に小さい箱が表示されます。矢印キーを使用して，この箱を拡大/縮小することができます。

箱モードにはいると，ステータス・ラインが次のように変わります。

Select キーを押すと，箱が確定され， "移動 "モードになります。Remove キーを押すと，箱がリセットされ， "移動 "モードになります。

7.15.1 罫線モードの終了

罫線モードを終了し，通常の編集画面に戻るには，Return キーを押してください。

7.16 日本語 EVE のオンライン・ヘルプを使用する

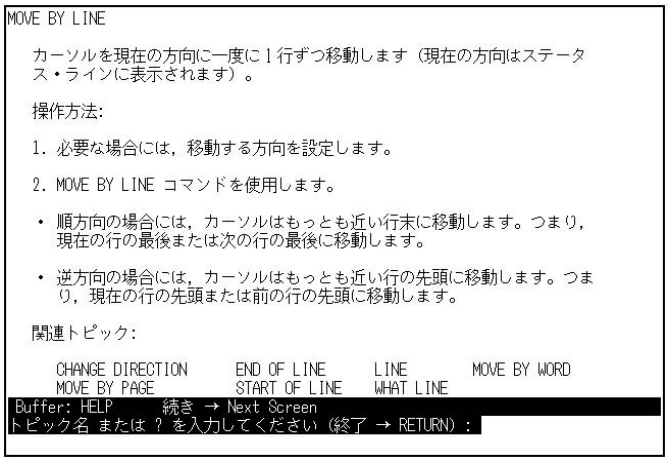
コマンドの使い方がわからないときなどに，日本語 EVE の HELP コマンドを使用してコマンドに関する情報を表示することができます。

Do キーを押して， HELP と入力し，Return キーを押します。画面に日本語 EVE コマンドの一覧が表示されます。Prev キーと Next キーで一覧全体を見ることができます。

注意

Do キーは，PC では Shift + F6 を使用します。

特定のコマンドに関する情報を表示したいときは，プロンプトに対して，そのコマンド名を入力してください。たとえば， MOVE BY LINE コマンドの説明が見たいときは， MOVE BY LINE と入力して，Return キーを押すと，次のようなヘルプ・テキストが画面に表示されます。



ヘルプから抜けて編集画面に戻るときは，Return キーを押します。

7.17 編集によく使用される日本語 EVE コマンド

テキスト編集によく使われるコマンドをまとめます。

表 7–4 編集によく使用される日本語 EVE コマンド

日本語 EVE コマンド	機能
おもな編集コマンド	
TOP	カーソルをテキストの先頭へ移動
BOTTOM	カーソルをテキストの最後へ移動
LINE 行番号	カーソルを指定した行へ移動
ERASE CHARACTER	カーソルのある 1 文字を削除
ERASE WORD	カーソルのある 1 語を削除
ERASE LINE	カーソルのある 1 行を削除
RESTORE	削除した 1 行（1 語）を復元する
REPLACE "文字列 1" "文字列 2"	文字列 1 を文字列 2 に置き換える
COPY (または STORE TEXT)	インサート・バッファへの取り込み
REPEAT 回数	指定回数コマンドを繰り返す

(次ページに続く)

表 7–4 (続き) 編集によく使用される日本語 EVE コマンド

日本語 EVE コマンド	機能
ファイルおよびウィンドウ (画面分割) に関するコマンド	
BUFFER バッファ名	バッファを切り替える
NEXT/PREVIOUS BUFFER	次 (前) のバッファを表示する
SHOW BUFFER	作成したバッファのリストを表示する
DELETE WINDOW	現在のウィンドウを削除
GET FILE ファイル名	バッファを作りそこにファイルの内容を読み込む
INCLUDE ファイル名	ファイルの内容をカーソルの位置に挿入
NEXT/PREVIOUS WINDOW	カーソルを次 (前) のウィンドウに移動
ONE WINDOW	ウィンドウを 1 つにする
SPLIT WINDOW [数]	ウィンドウを分割する
WRITE FILE ファイル名	バッファの内容をファイルへ書き出す
その他のコマンド	
HELP	コマンドについての HELP を表示する
HELP KEYPAD	キーの現在の定義状況を表示する
DCL [DCL コマンド]	エディタの中から DCL コマンドを使う
SPAWN	サブプロセスを作り制御を移す
DEFINE KEY EVE コマンド	キーを定義する

7.18 コラム：エディタについて

日本語 OpenVMS にはさまざまなエディタがありますが，日本語が使用できるエディタは日本語 EVE のみです。

日本語をサポートしていない汎用エディタ (英語が使用できます) としては， EVE があり，日本語 EVE と基本操作は共通です。

EVE を起動するには，次のようにします。

```
$ EDIT/TPU［ファイル名］
```

この他に EDT エディタがあります。

プログラム作成用のエディタとしては， LSE があります。 LSE はテンプレート機能とコンパイラ連動機能とを持った高度なエディタで，弊社のプログラミング支援ツールである DECSET の一部として提供されています。

UNIX 系のエディタでは， emacs などが英語版 OpenVMS フリーウェアとして提供されています。 OpenVMS フリーウェアについては下記の URL をご覧ください。

```
http://www.hp.com/go/openvms/freeware/
```

8

ファイルを指定する

この章では

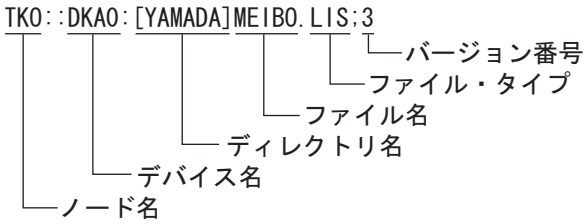
ファイルを管理するためには，ファイルやデバイスを認識するための情報をシステムに与える必要があります。ここでは，ファイルを指定する方法について説明します。

関連資料

- 『OpenVMS DCL ディレクトリ』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』
- 『日本語 OpenVMS 概説書』

8.1 ファイル指定の要素と形式

日本語 OpenVMS での完全なファイルの指定は，6 つの要素を次の形式で指定します。



◆ノード名

コンピュータ・システムの名前。ノード名の後には :: を付けます。

ネットワークを使って他のコンピュータにアクセスする場合に指定します。同じコンピュータ内での指定には必要ありません。

◆デバイス名

ファイルが存在するデバイスの名前。デバイス名の後には : を付けます。

◆ディレクトリ名

ファイルの名前などの情報が登録されているファイル。ファイルを使用目的などによって分類する場合に使用します。ディレクトリ名は [ ] で囲みます。

◆ファイル名

ファイルの固有名。(通常，ファイルの内容を示す名前。)

◆ファイル・タイプ

ファイルの性質を示します。ファイル・タイプの前にはピリオド (.) を付けます。

ファイル・タイプは任意の名前を付けることができますが，ファイルの性質または用途によって日本語 OpenVMS が自動的に付ける場合があります。たとえば，ディレクトリ・ファイルにはファイル・タイプとして DIR が付けられます。また，ファイル・タイプは省略できます。ファイル・タイプを省略した場合には，あらかじめ決まっているファイル・タイプを指定したとみなされます。

◆バージョン番号

ファイルのバージョンを示します。バージョン番号の前にはセミコロン (;) を付けます。

バージョンは 10 進数で表され，最初に作成したファイルは，バージョン番号が 1 です。

ファイルを更新すると，更新後のファイルにはファイル名，ファイル・タイプが同じで，1 が加算されたバージョン番号が付けられます。ユーザが指定しない場合にはシステムが自動的にバージョン番号を割り当てます。日本語 OpenVMS では，ファイルの変更前の内容と，変更後の内容を同じファイル名とファイル・タイプで別々に保存することができます。

バージョン番号を省略した場合，最大のバージョン番号を指定したとみなされます。

8.2 ファイル指定の規則

ファイル指定には，次の規則があります。

- 各要素の間には区切り記号 (: や ; など) が必要。
- 各要素の指定の中にスペースは使用できない。
- ノード名およびデバイス名は英数字で指定する。
- ディレクトリ名，ファイル名，ファイル・タイプに使用できる文字 :
  - 大文字または小文字の英字 (A ～ Z, a ～ z)，数字 (0 ～ 9)，アンダスコア (_)，ハイフン (-)，ドル記号 ($)

    ただし，ハイフン (-) は，最初の文字としては使用できない。
  - 日本語ファイル名が有効な場合 (第 8.9 節を参照) は， JIS 第 1 ～第 2 水準文字
- 各要素の指定に使用できる文字数は表 8–1 のとおり。

表 8–1 ファイル指定の各要素に使用できる文字数

	使用できる文字数		
要素	**英数字 (1 バイト)**	**日本語 (2 バイト)**	**例**
ノード名	1 ～ 6 文字	—	MOON
デバイス名	4 ～ 15 文字	—	DKA300
ディレクトリ名	1 ～ 39 文字	1 ～ 118 文字	REPORT，開発課
ファイル名	0 ～ 39 文字	118 文字 (ファイル名とファイル・タイプ，区切りのピリオド (.) 合わせて)	JAN，山田
ファイル・タイプ	0 ～ 39 文字	118 文字 (ファイル名とファイル・タイプ，区切りのピリオド (.) 合わせて)	TXT，書類

(次ページに続く)

表 8–1 (続き) ファイル指定の各要素に使用できる文字数

	使用できる文字数		
要素	**英数字 (1 バイト)**	**日本語 (2 バイト)**	**例**
バージョン番号	整数の 1 ～ 32767	—	3

ファイル指定全体の長さは 252 バイト以内でなければなりません。日本語ファイル名が使える場合は 4096 バイト以内です。

8.3 ファイル指定のデフォルト

ファイル指定の各要素にはデフォルト (省略時の値) が決まっています。ユーザがファイルの指定の一部を省略すると，デフォルトが使用されます。ファイル指定の各要素とデフォルトの内容を表 8–2 にまとめます。

表 8–2 ファイル指定の各要素とデフォルト

要素	デフォルト
ノード名	ローカル・ノード，つまり使用しているコンピュータ・システム名
デバイス名	ログイン時，または SET DEFAULT コマンドによって指定されたデバイス
ディレクトリ名	ログイン時，または SET DEFAULT コマンドによって指定されたディレクトリ名
ファイル名	デフォルトは適用されない
ファイル・タイプ	DCL コマンドごとに異なる
バージョン番号	一番大きい番号のバージョン

8.4 現在のデフォルトを参照する — SHOW DEFAULT

現在のデフォルトを参照するには，次のコマンドを入力します。

```
$ SHOW DEFAULT
```

8.5 デフォルトを指定する — SET DEFAULT

デフォルトを指定するには，デバイス名:[ディレクトリ] または[ディレクトリ] のみを入力します。

```
$ SET DEFAULT デバイス名:[ディレクトリ]
```

8.6 デフォルトのファイル指定の例

現在のデフォルトのデバイス名とディレクトリ名を調べます。

```
$ SHOW DEFAULT
```

デフォルトはデバイス名 DKA0，ディレクトリ名 [YAMADA] です。

```
DKA0:[YAMADA]
```

デフォルトを DKB0:[PROJECT] に変更します。

```
$ SET DEFAULT DKB0:[PROJECT]
```

デフォルト指定を調べます。

```
$ SHOW DEFAULT
```

デフォルトが DKB0:[PROJECT] に変更されました。

```
DKB0:[PROJECT]
```

8.7 ワイルドカードを使用する

ワイルドカードは，ファイルやディレクトリの名前に指定できる特別な文字です。ワイルドカードを指定すると，名前のすべての文字列を指定しなくてもファイルやディレクトリを指定することができます。また，複数のファイルやディレクトリを指定する場合，ワイルドカードを使用して簡単に指定することができます。

8.7.1 ワイルドカード文字の種類

ファイルやディレクトリの名前に使用できるワイルドカード文字は，次のとおりです。

文字	機能
*	文字数や文字の種類に関係なく，任意の 0 文字以上の文字を指定したとみなされる。ディレクトリ名，ファイル名，ファイル・タイプ，バージョン番号に使用できる。
%	1 文字を指定したとみなされる。ディレクトリ名，ファイル名，ファイル・タイプに使用できる。バージョン番号には使用できない。
?	日本語ファイル名の使用が有効な時 (第 8.9 節を参照) のみに使用できる。 1 文字を指定したとみなされ，ディレクトリ名，ファイル名，ファイル・タイプに使用できる。バージョン番号には使用できない。

8.8 ワイルドカード文字の使用例

たとえば，ディレクトリ[YAMADA.REPORT]に以下のファイルがあるとします。

```
Directory DKA0:[YAMADA.REPORT]

DAY.TXT;2           DAY.TXT;1           DAY1.TXT;1          DAY2.TXT;15
DAYOFF.TXT;3        PRODUCT.TXT;6       REPORT.LIS;8        REPORT.MEM;8

Total of 8 files.
```

ワイルドカードは，次のように使用できます。

DAY.TXT のすべてのバージョンを表示する場合

```
$ DIRECTORY  DAY.TXT;*

Directory DKA0:[YAMADA.REPORT]

DAY.TXT;2      DAY.TXT;1

Total of 2 files.
```

バージョン番号が 1 であるファイルをすべて表示する場合

```
$ DIRECTORY  *.*;1

Directory DKA0:[YAMADA.REPORT]

DAY.TXT;1      DAY1.TXT;1

Total of 2 files.
```

すべてのファイルを表示する場合

```
$ DIRECTORY  *.*;*

Directory DKA0:[YAMADA.REPORT]

DAY.TXT;2           DAY.TXT;1           DAY1.TXT;1          DAY2.TXT;15
DAYOFF.TXT;3        PRODUCT.TXT;6       REPORT.LIS;8        REPORT.MEM;8

Total of 8 files.
```

ファイル・タイプが TXT であり，ファイル名が DAY で始まり，その後に 1 文字が続くファイルのすべてのバージョンを表示する場合

```
$ DIRECTORY  DAY%.TXT;*

Directory DKA0:[YAMADA.REPORT]

DAY1.TXT;1      DAY2.TXT;15

Total of 2 files.
```

最初の 3 文字が DAY であるすべてのファイルを表示する場合

```
$ DIRECTORY  DAY*.*;*

Directory DKA0:[YAMADA.REPORT]

DAY.TXT;2       DAY.TXT;1       DAY1.TXT;1       DAY2.TXT;15
DAYOFF.TXT;3

Total of 5 files.
```

8.9 日本語ファイル名について

ファイル名に日本語 (かな漢字) を使用するには，次の 2 つの条件を満たしている必要があります。

- デバイスがディスクで形式が ODS 5 である
- ユーザの環境で日本語ファイル名が有効になっている

ディスクの形式が ODS-5 かどうかは，次のコマンドで調べることができます。

```
$ SHOW DEVICE <ディスク> /FULL
```

```
Disk TOKYO$DKA100:, device type RZ26, is online, mounted, file-oriented device,
    shareable, available to cluster, error logging is enabled.

    Error count                    0    Operations completed                16
    Owner process                 ""    Owner UIC                       [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Total blocks             2050860    Sectors per track                   57
    Total cylinders             2570    Tracks per cylinder                 14

    Volume label               "USER"    Relative volume number               0
    Cluster size                    4    Transaction count                    1
    Free blocks                 39104    Maximum files allowed           410172
    Extend quantity                 5    Mount count                          1
    Mount status               System    Cache name       "_TOKYO$DKA0:XQPCACHE"
    Extent cache size              64    Maximum blocks in extent cache     3910
    File ID cache size             64    Blocks in extent cache                0
    Quota cache size                0    Maximum buffers in FCP cache       2894
    Volume owner UIC          [SYSTEM]    Vol Prot    S:RWCD,O:RWCD,G:RWCD,W:RWCD

  Volume Status:  ODS-5, subject to mount verification, write-back caching enabled.
```

最後の行の Volume Status: の部分を確認します。 ODS-5 と記述されている場合，ディスクの形式が ODS-5 であることを示しています。

ユーザの環境で日本語ファイル名を有効にするには，以下のコマンドを入力します。

```
$ JSY$CONTROL:==$SYS$SYSTEM:JSY$CONTROL.EXE
$ JSY$CONTROL SET RMS/FILENAME=SDECKANJI
```

これにより， DCL コマンドや日本語ユーティリティ等でファイル名に日本語を使用できるようになります。(日本語ファイル名を使用できるのは，日本語 OpenVMS バージョン 7.2 からです。)

日本語ファイルに関する詳細は『日本語 OpenVMS 概説書』を参照してください。

9

ディレクトリを作成する

この章では

ディレクトリとは，ファイルの登録簿です。日本語 OpenVMS のもとで，ディスク上に作成されるファイルは，必ずディレクトリに登録されます。

ここでは，ディレクトリの階層構造や，ディレクトリの作成方法等について説明します。

関連資料

- 『OpenVMS DCL ディレクトリ』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』
- 『日本語 OpenVMS 概説書』

9.1 ディレクトリとは

日本語 OpenVMS のもとで，ディスク上に作成されるファイルは，必ずディレクトリに登録されます。つまり，ディレクトリはファイルの登録簿です。ディレクトリの内容は，登録されているファイルのファイル名，ファイル・タイプ，バージョン番号，ファイルの位置に関する情報です。

ディレクトリ自身も，ディスク上の情報であり 1 つのファイルです (ディレクトリ自身のことをディレクトリ・ファイルということもあります)。ファイル名=ディレクトリ名で，ファイル・タイプは.DIR，バージョン番号は 1 です。

9.2 ディレクトリの種類と階層構造

ディレクトリは，ツリー状の階層構造になっています。そのためディレクトリの名前を指定する場合には，階層構造のどこにあるかを明示する必要があります。ディレクトリには，次の 3 種類があります。

◆ルート・ディレクトリ

すべてのユーザファイル・ディレクトリを登録してあるディレクトリ。ルート・ディレクトリは 1 個のディスクに必ず 1 個だけ存在します。ルート・ディレクトリのファイル名は 000000，ファイル・タイプは .DIR です。

```
[000000]
```

◆ユーザファイル・ディレクトリ

ユーザの持つディレクトリ。ユーザファイル・ディレクトリはディレクトリ名を[ ] で囲んで指定します。

ユーザファイル・ディレクトリを指定する際にルート・ディレクトリの指定は省略できます。たとえば DKA0:[000000.YAMADA]と指定すると，ディスク DKA0 の中にある[YAMADA]というユーザファイル・ディレクトリを指定することになります。

```
[YAMADA]
```

◆サブファイル・ディレクトリ (サブ・ディレクトリとも呼ぶ)

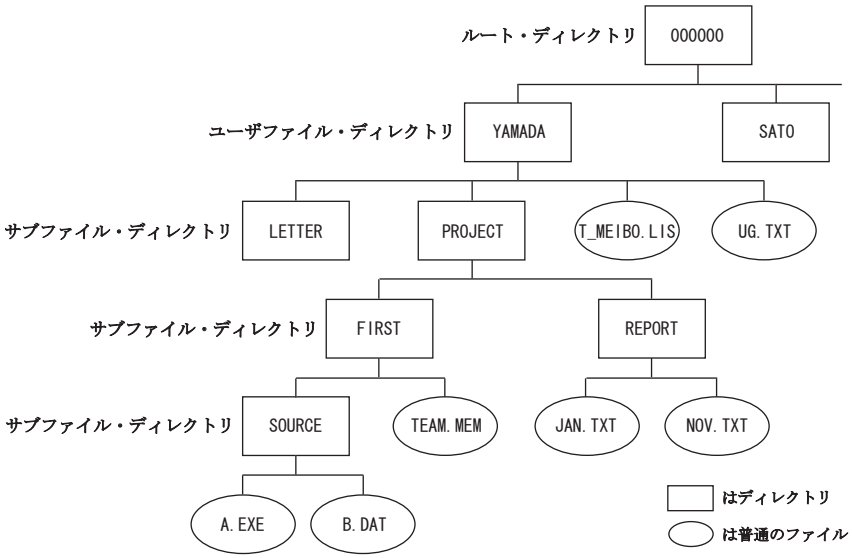
ユーザファイル・ディレクトリの下に作るディレクトリ。ディレクトリ名をピリオドで区切って指定します。

サブファイル・ディレクトリは，ユーザファイル・ディレクトリを含めて 255 階層まで作成することができます。 (日本語 OpenVMS のバージョン 7.2 より前のバージョンでは 8 階層まで。)

```
[YAMADA.PROJECT.REPORT]
```

ディレクトリ・ファイルやファイルの関係を図 9–1 に示します。

図 9–1 ディレクトリ階層構造



9.3 ディレクトリ名の規則

ディレクトリ名は，次の規則に従って付けます。

- 使用できる文字
  - 大文字または小文字の英字 (A ～ Z, a ～ z)，数字 (0 ～ 9)，アンダスコア (_)，ハイフン (-)，ドル記号 ($)

    ただし，ハイフン (-) は，最初の文字としては使用できない。
  - 日本語 (日本語ファイル名を使用できる場合のみ)
- 長さ
  - 英数字のみの場合は，1 ～ 39 文字
  - 日本語ファイル名が使用できる場合は，.DIR を含めて 118 文字

9.4 ディレクトリを作成する — CREATE/DIRECTORY

ルート・ディレクトリとユーザファイル・ディレクトリは通常，システム管理者が作成します。ユーザがサブファイル・ディレクトリを作成するには，次のようにします。

```
$ CREATE/DIRECTORY［ディレクトリ名］
```

ディレクトリ名は次のように指定します。

```
[ユーザファイル・ディレクトリ名.サブファイル・ディレクトリ名]
```

ディレクトリを指定する場合には，ディレクトリの名前を，[ ]で囲む点に注意してください。また，複数のディレクトリ名の指定をすることもできます。複数のディレクトリを指定する場合は，ディレクトリの名前をコンマで区切ります。

現在のディレクトリを調べます。

```
$ SHOW DEFAULT
  DKA0:[YAMADA]
```

ユーザファイル・ディレクトリ[YAMADA]の下に新しくサブファイル・ディレクトリ[YAMADA.SUB]を作成します。 (現在のディレクトリの下のサブディレクトリを作成する場合は，現在のディレクトリ名の代わりにピリオドを指定できます。
$ CREATE/DIRECTORY [.SUB])

```
$ CREATE/DIRECTORY [YAMADA.SUB]
```

ユーザファイル・ディレクトリ[YAMADA]の下にディレクトリ・ファイル SUB.DIR が作成されたことを確かめます。

```
$ DIRECTORY

Directory DKA0:[YAMADA]

LETTER.DIR;1   PROJECT.DIR;1   SUB.DIR;1   T_MEIBO.LIS;1
UG.TXT

Total of 5 files.
```

新しく作成されたサブファイル・ディレクトリ[YAMADA.SUB]にデフォルト・ディレクトリを変更します。 ($ SET DEFAULT [.SUB]でもよい。)

```
$ SET DEFAULT [YAMADA.SUB]
```

サブファイル・ディレクトリ[YAMADA.SUB]の中を確認すると，まだ何もファイルはありません。

```
$ DIRECTORY
%DIRECT-W-NOFILES, no files found
$
```

9.5 現在のディレクトリを調べる — SHOW DEFAULT

現在のディレクトリがわからない場合は， SHOW DEFAULT コマンドで調べることができます。また，現在のディレクトリを変更した場合に， SHOW DEFAULT コマンドで確認することができます。

```
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU]
```

9.6 現在のディレクトリを変更する — SET DEFAULT

SET DEFAULT コマンドを使用すると，ユーザが管理するディレクトリであれば，自由に現在のディレクトリを変更することができます。ファイルを探す場合だけでなく，ファイルを置くディレクトリを変更したい場合にも， SET DEFAULT コマンドを使用して，現在のディレクトリを変更します。

```
$ SET DEFAULT [ディレクトリ名]
```

注意

SET DEFAULT コマンドを使用する際に存在しないディレクトリを指定しても，日本語 OpenVMS はエラー・メッセージを表示しません。したがって，SET DEFAULT コマンドを実行したら，現在のディレクトリが正しく変更できたかどうかを SHOW DEFAULT コマンドで確認してください。

9.7 ディレクトリ指定の簡略化

ディレクトリ指定を簡単にするために，表 9–1 にある特殊なシンボルを使うことができます。

表 9–1 ディレクトリの指定で使用するシンボル

シンボル	意味
-（ハイフン）	ディレクトリ階層構造の中の 1 レベル "上 "を指定 (ディレクトリ指定の最初に書ける)
.（ピリオド）	ディレクトリ階層構造の中の区切りを表す (この後にサブファイル・ディレクトリ名または特殊シンボルである * や % を指定する)
...（ピリオド 3 つ）	現在のディレクトリおよびその下にあるすべてのレベルのサブファイル・ディレクトリを指定
*（アスタリスク）	ディレクトリ名を構成する 0 から 39 文字までと一致
%（パーセント）	ディレクトリ名の 1 文字と 1 対 1 に対応
?（感嘆符）	日本語ファイル名の使用が有効な時のみに使用できる。ディレクトリ名の 1 文字と 1 対 1 に対応

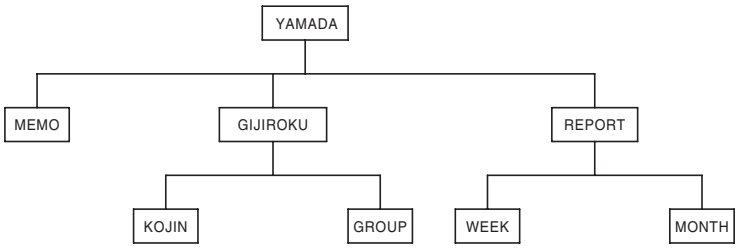
サブファイル・ディレクトリを指定する場合は，次の 2 つの方法があります。

- ユーザファイル・ディレクトリ名から指定する方法
- 先頭に . または - を指定する方法

先頭に．または - を使用して指定すると，その前に現在のデフォルト・ディレクトリを追加して指定したのと同じ意味になります。また，単に，[] とだけ指定した場合は，現在のデフォルト・ディレクトリ指定そのものを表します。

9.8 現在のディレクトリの変更例

SET DEFAULT コマンドを実行したときの実行例を示します。これらの例は，下記の図のディレクトリ構造で実行した場合の実行結果です。図の中で，□で囲まれている名前が，ディレクトリの名前です。また，現在のディレクトリは，[YAMADA.GIJIROKU]です。



ディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU]
$ SET DEFAULT [YAMADA.REPORT.GROUP]
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT.GROUP]
```

1 つ上のレベルのディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU]
$ SET DEFAULT [-]
$ SHOW DEFAULT
  DKA0:[YAMADA]
```

1 つ上のディレクトリを基準にしてディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU]
$ SET DEFAULT [-.REPORT.WEEK]
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT.WEEK]
```

現在のディレクトリの下にあるディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU]
$ SET DEFAULT [.KOJIN]
$ SHOW DEFAULT
  DKA0:[YAMADA.GIJIROKU.KOJIN]
```

9.9 現在のディレクトリからファイルを探す — DIRECTORY

ファイルの一覧を表示したり，現在のディレクトリにどのようなファイルがあるか，また，ファイルの名前を知りたい場合に， DIRECTORY コマンドを使用します。また，ファイルの名前だけでなく個々のファイルの大きさやファイルの個数などの情報を知ることもできます。

```
$ DIRECTORY ファイル名
```

複数のファイルを指定することもできます。複数のファイルを指定する場合は，ファイルの名前をコンマで区切ります。ファイルの名前にワイルドカード文字も使用でき，またファイルの名前を省略することもできます。省略した場合，現在のディレクトリのすべてのファイルが表示されます。

9.10 現在のディレクトリからファイルを探す例

DIRECTORY コマンドを実行すると，次の実行例のようにファイルの一覧が表示されます。

1 つのファイルを指定した場合

```
$ DIRECTORY GROUP_AUG.TXT

Directory DKA0:[YAMADA.GIJIROKU]

GROUP_AUG.TXT;1

Total of 1 file.
```

複数のファイルを指定した場合

```
$ DIRECTORY GROUP_AUG.TXT,GROUP_SEP.TXT

Directory DKA0:[YAMADA.GIJIROKU]

GROUP_AUG.TXT;1     GROUP_SEP.TXT;5     GROUP_SEP.TXT;4
GROUP_SEP.TXT;3     GROUP_SEP.TXT;2     GROUP_SEP.TXT;1

Total of 6 files.
```

ファイルの名前にワイルドカード文字 (*) を指定した場合

```
$ DIRECTORY GROUP_S*.TXT

Directory DKA0:[YAMADA.GIJIROKU]

GROUP_SEP.TXT;5     GROUP_SEP.TXT;4     GROUP_SEP.TXT;3
GROUP_SEP.TXT;2     GROUP_SEP.TXT;1

Total of 8 files.
```

ファイルを省略した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.GIJIROKU]

GROUP_AUG.TXT;1     GROUP_OCT.TXT;2     GROUP_OCT.TXT;1
GROUP_SEP.TXT;5     GROUP_SEP.TXT;4     GROUP_SEP.TXT;3
GROUP_SEP.TXT;2     GROUP_SEP.TXT;1

Total of 8 files.
```

9.11 指定したディレクトリからファイルを探す – DIRECTORY

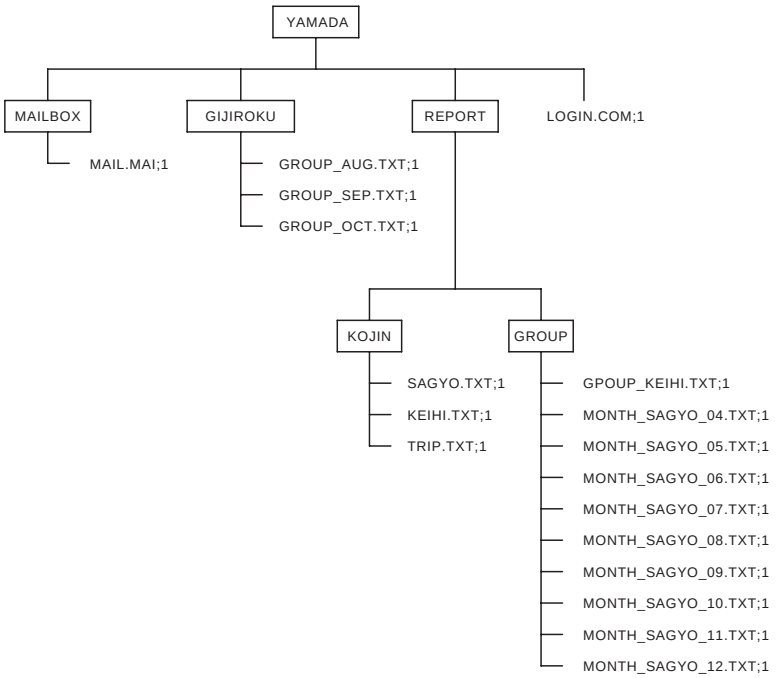
DIRECTORY コマンドにディレクトリを指定すると，指定したディレクトリのファイルの一覧を表示することができます。現在のディレクトリを変更しないで，他のディレクトリに探したいファイルがあるかどうかを調べることができます。

```
$ DIRECTORY  [ディレクトリ名]ファイル名
```

複数のディレクトリを指定することもできます。複数のディレクトリを指定する場合は，ディレクトリの名前をコンマで区切ります。ディレクトリの名前にワイルドカード文字も使用できます。

9.12 指定ディレクトリからファイルを探す例

DIRECTORY コマンドを実行したときの実行例を示します。これらの例は，下記の図のディレクトリ構造で実行した場合の実行結果です。図中，ディレクトリを[ ]で囲んで示しています。それ以外は通常のファイルです。



1 つのディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT.KOJIN]


$ DIRECTORY [YAMADA.REPORT]

Directory DKA0:[YAMADA.REPORT]

GROUP.DIR;1         KOJIN.DIR;1

Total of 2 files.
```

複数のディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT]


$ DIRECTORY [YAMADA.REPORT.KOJIN],[YAMADA.REPORT.GROUP]

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1

Total of 3 files.

Directory DKA0:[YAMADA.REPORT.GROUP]
```

```
GROUP_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1

Total of 10 files.

Grand total of 2 directories, 13 files.
```

1 つ上のレベルのディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT.KOJIN]



$ DIRECTORY [-]

Directory DKA0:[YAMADA.REPORT]

GROUP.DIR;1         KOJIN.DIR;1

Total of 2 files.
```

現在のディレクトリの下にあるディレクトリを指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT]



$ DIRECTORY [.KOJIN]

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1

Total of 3 files.
```

現在のディレクトリの下にあるディレクトリをすべて指定した場合

```
$ SHOW DEFAULT
  DKA0:[YAMADA.REPORT]



$ DIRECTORY [...]

Directory DKA0:[YAMADA.REPORT]

GROUP.DIR;1         KOJIN.DIR;1

Total of 2 files.

Directory DKA0:[YAMADA.REPORT.GROUP]

GROUP_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1
```

```
Total of 10 files.

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1

Total of 3 files.

Grand total of 3 directories, 15 files.
```

ディレクトリにワイルドカード文字 (*) を指定した場合

```
$ DIRECTORY [YAMADA.REPORT.*]

Directory DKA0:[YAMADA.REPORT.GROUP]

GROUP_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1

Total of 10 files.

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1

Total of 3 files.

Grand total of 2 directories, 13 files.
```

9.13 ファイルの大きさを表示する — DIRECTORY/SIZE

ファイルの大きさを表示するには，次のように入力します。

```
$ DIRECTORY/SIZE  ディレクトリ指定
```

ここでディレクトリ指定をしない場合は現在のディレクトリ内にあるファイルの大きさが表示されます。

9.14 作成日を表示する — DIRECTORY/DATE

ファイルの作成日を表示するには，次のように入力します。

```
$ DIRECTORY/DATE  ディレクトリ指定
```

ここでディレクトリ指定をしない場合は現在のディレクトリ内にあるファイルの作成日が表示されます。

9.15 ファイルの個数を表示する — DIRECTORY/TOTAL

ファイルの個数を表示するには，次の例のように入力します。

```
$ DIRECTORY/TOTAL  ディレクトリ指定
```

ここでディレクトリ指定をしない場合は現在のディレクトリ内にあるファイルの個数が表示されます。

9.16 結果をファイルに出力する — DIRECTORY/OUTPUT

ファイルの一覧をディスプレイに表示せずにファイルに出力するには，次のように入力します。

```
$ DIRECTORY/OUTPUT=出力ファイル  ディレクトリ指定
```

ここでディレクトリ指定をしない場合は現在のディレクトリ内にあるファイルの一覧が出力されます。

(修飾子 /COLUMNS=1 を付けると，出力表示のカラム数が 1 になり見易いリストができます。)

10
ファイルを操作するためのコマンド

この章では

ここでは，ファイルを操作するためによく使用される DCL コマンドについて説明します。

TYPE
DIFFERENCES
SEARCH
RENAME
COPY
PURGE
DELETE
APPEND
SORT

関連資料

- 『OpenVMS DCL ディレクトリ』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』
- 『日本語 OpenVMS 概説書』

10.1 ファイルの内容を表示する — TYPE

ファイルの内容をそのままディスプレイに表示するには，TYPE コマンドを使用します。

```
$ TYPE ファイル指定
```

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合は，ファイルの名前をコンマで区切ります。

```
$ TYPE 1番目のファイル,2番目のファイル, . . .  ,n番目のファイル
```

ファイル・タイプは省略できます。省略した場合，.LIS を指定したものとみなされます。バージョン番号も省略でき，最新のバージョンを指定したものとみなされます。ファイル名，ファイル・タイプ，バージョン番号にはワイルドカード文字を指定できます。

TYPE コマンドは，ファイルの内容を最初から最後まで停止せずに表示します。適当な箇所で表示を停止，再開，強制終了させたい場合は，次のキーを使用します。

表示	キー操作	結果
停止	Hold Screen または Ctrl + S （または F1, Pause）	表示がその箇所で停止する。(Hold Screen ランプが点灯する。)
再開	Hold Screen または Ctrl + Q （または F1, Pause）	表示がその箇所から再開される。(Hold Screen ランプが消灯する。)
強制終了	Ctrl + Y	表示がその箇所で終了する。再び$プロンプトが表示される。

表示するファイルに，複数のファイルまたはワイルドカード文字を指定したときの表示の順序は，次のようになります。

複数のファイルを指定した場合

コマンドに指定した順番

ワイルドカード文字を指定した場合

ファイル名の数字，アルファベット順にファイルを並べた順番

10.2 ファイルの内容を 1 画面ごとに表示する — TYPE/PAGE

1 画面ごとに停止させて表示するには，次のように入力します。

```
$ TYPE/PAGE ファイル指定
```

表示が停止した場合は，次のようなキー操作を行います。

次の画面に進みたいとき

Return を入力する。

表示を強制終了したいとき

Ctrl + Y を入力する。

10.3 ファイルの内容を比較する — DIFFERENCES

DIFFERENCES コマンドを使用すると， 2 つのファイルの内容を比較し，その相違箇所をディスプレイに表示またはファイルに出力して保存することができます。

```
$ DIFFERENCES  第1ファイル  [第2ファイル]
```

第 2 ファイルの名前全体を省略することもできます。省略した場合，第 1 ファイルの 1 つ前のバージョンのファイルを指定したものとみなされます。

DIFFERENCES コマンドには，次のような特徴があります。

◆比較する単位

DIFFERENCES コマンドは， 2 つのファイルの内容を行単位で比較します。つまり，文字列と文字列が一致するかをひとつひとつ表示するのではなく，行全体が一致するかどうかを表示します。

◆得られる結果

DIFFERENCES コマンドを使用すると，次のことを知ることができます。

- 相違する行の内容
- 相違する行の行番号
- 相違する行の総数

10.4 比較結果をファイルに出力する — DIFFERENCES/OUTPUT

比較した結果をディスプレイに表示せずに，ファイルに出力するには，次のように入力します。出力ファイルは，結果を出力するファイルです。

```
$ DIFFERENCES/OUTPUT=出力ファイル 第1ファイル［第2ファイル］
```

10.5 ファイルの内容を比較する例

FILE1.TXT の内容を表示します。

```
$ TYPE FILE1.TXT

このファイルのファイル名は
FILE 1 です。

★部門名簿（TKO)★
§東京本社

- 以上 -
```

FILE2.TXT の内容を表示します。

```
$ TYPE FILE2.TXT

このファイルのファイル名は
FILE 2 です。

★部門名簿（OSK)★
§大阪本社

- 以上 -
```

FILE1.TXT と FILE2.TXT を比較して，結果を SABUN.TXT に出力します。

```
$ DIFFERENCES FILE1.TXT FILE2.TXT/OUT=SABUN.TXT
```

SABUN.TXT の内容を表示します。

```
$ TYPE SABUN.TXT
************
File DKA0:[YAMADA.MEIBO]FILE1.TXT;1
    2   FILE 1 です。
    3
    4   ★部門名簿 (TKO)★
    5   §東京本社
    6
******
File DKA0:[YAMADA.MEIBO]FILE2.TXT;1
    2   FILE 2 です。
    3
    4   ★部門名簿 (OSK)★
    5   §大阪本社
    6
************

Number of difference sections found: 1
Number of difference records found: 4

DIFFERENCES /IGNORE=()/MERGED=1/OUTPUT=DKA0:[YAMADA.MEIBO]SABUN.TXT;1-
    DKA0:[YAMADA.MEIBO]FILE1.TXT;1-
    DKA0:[YAMADA.MEIBO]FILE2.TXT;1
```

10.6 比較結果の表示方法を変更する — DIFFERENCES/PARALLEL

比較した結果を，左右に分割して表示するには，次のように入力します。

```
$ DIFFERENCES/PARALLEL 第1ファイル [第2ファイル]
```

```
--------------------------------------------------------------------------------
File DKA0:[YAMADA.MEIBO]FILE1.TXT;1  |  File DKA0:[YAMADA.MEIBO]FILE2.TXT;1
---------------- 2 ----------------------------- 2 ----------------------
FILE 1 です。                        |  FILE 2 です。
                                     |
★部門名簿 (TKO)★                    |  ★部門名簿 (OSK)★
§東京本社                            |  §大阪本社
--------------------------------------------------------------------------------

Number of difference sections found: 1
Number of difference records found: 4

DIFFERENCES /IGNORE=()/OUTPUT=DKA0:[YAMADA.MEIBO]SABUN-PARA.TXT;1/PARALLEL-
    DKA0:[YAMADA.MEIBO]FILE1.TXT;1-
    DKA0:[YAMADA.MEIBO]FILE2.TXT;1
```

10.7 ファイルの内容を検索する — SEARCH

指定したファイル内で指定した文字列を検索するには， SEARCH コマンドを使用します。 SEARCH コマンドで検索する文字列には，任意の文字列を指定できます。また，/EXACT 修飾子を使用しないかぎり，英小文字と英大文字は区別しません。(日本語文字列を DCL コマンド・ラインに入力する方法は，第 B.3 節を参照。)

SEARCH コマンドを使用すると，指定した文字列が含まれる行の内容を知ることができます。また修飾子を追加することにより，指定した文字列が含まれる行の行番号や検索したファイルの数なども知ることができます。

```
$ SEARCH 検索ファイル[, . . . ]  "検索文字列[, . . . ]"
           1                          2
```

1 検索ファイル

検索するファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。バージョン番号は省略でき，最新のバージョンを指定したものとみなされます。ファイル名，ファイル・タイプ，バージョン番号にはワイルドカード文字を指定できます。

2 検索文字列

検索する文字列。

文字列の中に空白が含まれる場合や日本語の文字列の場合は，文字列を二重引用符 (") で囲みます。 1 つまたは複数の文字列を指定できます。複数の文字列を指定する場合，文字列をコンマで区切ります。

次の例では現在のディレクトリ内にあるファイル・タイプが TXT のファイルを対象に，文字列 tko が含まれる行を検索しています。

```
$ SEARCH *.TXT tko
```

検索結果は次のようになります。

```
DKA0:[YAMADA.MEIBO]FILE1.TXT;1
★部門名簿（TKO)★
```

10.8 検索結果をファイルに出力する — SEARCH/OUTPUT

検索した結果をディスプレイに表示せずに，ファイルに出力するには，次のように入力します。

```
$ SEARCH/OUTPUT=出力ファイル　検索ファイル　検索文字列
```

出力ファイルは，結果を出力するファイルです。検索ファイルと検索文字列は複数指定できます。

10.9 完全に一致するものを検索する — SEARCH/EXACT

SEARCH コマンドは，ファイルを検索する場合に英小文字と英大文字を区別しないので，区別して検索したい場合は，次のように入力します。

```
$ SEARCH/EXACT 検索ファイル　検索文字列
```

検索ファイルと検索文字列は複数指定できます。

10.10 検索結果の統計情報を表示する — SEARCH/STATISTICS

検索した結果の統計情報を表示するには，次のように入力します。

```
$ SEARCH/STATISTICS 検索ファイル　検索文字列
```

検索ファイルと検索文字列は複数指定できます。

10.11 検索文字列の行番号を表示する — SEARCH/NUMBERS

検索文字列が含まれる行の行番号を表示するには，次のように入力します。

```
$ SEARCH/NUMBERS 検索ファイル　検索文字列
```

検索ファイルと検索文字列は複数指定できます。

10.12 検索結果の表示方法を変更する — SEARCH/HIGHLIGHT

検索文字列を強調して表示する方法を変更するには，次のようにします。

検索文字列を点滅させる

```
$ SEARCH/HIGHLIGHT=BLINK 検索ファイル　検索文字列
```

検索文字列にアンダーラインを引く

```
$ SEARCH/HIGHLIGHT=UNDERLINE 検索ファイル　検索文字列
```

検索ファイルと検索文字列は複数指定できます。

10.13 ファイルの名前を変更する — RENAME

ファイルの名前を変更するには， RENAME コマンドを使用します。ファイルの名前を変更する場合，ファイル名，ファイル・タイプ，バージョン番号をすべて変更する

ことも，ファイル名，ファイル・タイプ，バージョン番号のいずれかだけを変更することもできます。

```
$ RENAME 既存のファイル[,...] 新しいファイル
         1                    2
```

1 既存のファイル

名前を変更したいファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。

2 新しいファイル

ファイルの新しい名前。

ファイル名，ファイル・タイプは省略できます。省略した場合，既存のファイルに指定したファイル名，ファイル・タイプを指定したとみなされます。

バージョン番号も省略できます。省略した場合，バージョン番号は自動的に付けられます。

注意

既存のファイルに複数のファイルを指定した場合，指定したすべてのファイルの名前が新しいファイルの名前に変更されるので注意してください。

また，新しいファイルの名前を指定するときに，ファイル名 (またはファイル・タイプ) を省略すると，既存ファイルのファイル名 (またはファイル・タイプ) を指定したとみなされます。したがって，指定した複数のファイルのそれぞれのファイル名 (またはファイル・タイプ) が新しいファイルに付けられます。

10.14 ファイルの名前を変更する例

ファイル名とファイル・タイプを指定した場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1          SAGYO.TXT;1          TRIP.TXT;1
Total of 3 files.
```

SAGYO.TXT を MONTH_SAGYO.TXT に変更します。

```
$ RENAME SAGYO.TXT MONTH_SAGYO.TXT;1
```

確認します。

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         MONTH_SAGYO.TXT;1   TRIP.TXT;1
Total of 3 files.
```

新しいファイルのファイル・タイプを省略した場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1
Total of 3 files.
```

SAGYO.TXT を MONTH_SAGYO.TXT に変更します。

```
$ RENAME SAGYO.TXT MONTH_SAGYO
```

確認します。

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         MONTH_SAGYO.TXT;1   TRIP.TXT;1
Total of 3 files.
```

新しいファイル・タイプが省略されたため，既存のファイル・タイプと同じファイル・タイプ TXT が付けられています。

新しいファイルのファイル名を省略した場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1
Total of 3 files.
```

SAGYO.TXT を SAGYO.LIS に変更します。

```
$ RENAME SAGYO.TXT .LIS
```

確認します。

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.LIS;1         TRIP.TXT;1
Total of 3 files.
```

新しいファイル名が省略されたため，既存のファイル名と同じファイル名が付けられています。

10.15 ファイルを移動する — RENAME

RENAME コマンドを使用して，別のディレクトリにファイルを置くこともできます。

```
$ RENAME 既存のファイル[,...] ディレクトリ
         1                    2
```

1 既存のファイル

ディレクトリを変更したいファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。ファイルの名前にワイルドカード文字も使用できます。

2 ディレクトリ

ファイルを置くディレクトリの名前。

ディレクトリと共にファイルの名前を指定できます。ファイルの名前を指定すると，その名前がファイルに付けられます。

10.16 ファイルを移動する例

1 つのファイルを指定した場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1
Total of 3 files.
```

SAGYO.TXT を[YAMADA.REPORT.KOJIN]から[YAMADA.REPORT.GROUP]に移します。

```
$ RENAME SAGYO.TXT [YAMADA.REPORT.GROUP]
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]

Directory DKA0:[YAMADA.REPORT.GROUP]

MONTH_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1
SAGYO.TXT;1

Total of 11 files.


$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         TRIP.TXT;1

Total of 2 files.
```

複数のファイルを指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1

Total of 4 files.
```

SAGYO.TXT のすべてのバージョンのファイルを[YAMADA.REPORT.KOJIN]から[YAMADA.REPORT.GROUP]に移します。

```
$ RENAME SAGYO.TXT;* [YAMADA.REPORT.GROUP]
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]

Directory DKA0:[YAMADA.REPORT.GROUP]

MONTH_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1
SAGYO.TXT;2         SAGYO.TXT;1

Total of 12 files.
```

新しいファイルの名前を指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1

Total of 3 files.
```

SAGYO.TXT を[YAMADA.REPORT.KOJIN]から[YAMADA.REPORT.GROUP]に移し名前を MONTH_SAGYO_01.TXT;1 に変更します。

```
$ RENAME SAGYO.TXT [YAMADA.REPORT.GROUP]MONTH_SAGYO_01.TXT;1
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]

Directory DKA0:[YAMADA.REPORT.GROUP]

MONTH_KEIHI.TXT;1   MONTH_SAGYO_01.TXT;1
MONTH_SAGYO_04.TXT;1                    MONTH_SAGYO_05.TXT;1
MONTH_SAGYO_06.TXT;1                    MONTH_SAGYO_07.TXT;1
MONTH_SAGYO_08.TXT;1                    MONTH_SAGYO_09.TXT;1
MONTH_SAGYO_10.TXT;1                    MONTH_SAGYO_11.TXT;1
MONTH_SAGYO_12.TXT;1

Total of 11 files.
```

10.17 ファイルをコピーする — COPY

COPY コマンドを使用すると，既存のファイルと同じ内容のファイルを現在のディレクトリや別のディレクトリに作成することができます。その場合，ファイルの名前を変えることもできます。 COPY コマンドを使用した場合，もとのファイルはそのまま残ります。

```
$ COPY 既存のファイル[,...] 新しいファイル
            1                2
```

1 既存のファイル

コピーしたいファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。

2 新しいファイル

新しく作成するファイルのディレクトリの名前およびファイルの名前。

ディレクトリの名前だけを指定した場合は，既存のファイルと同じ名前のファイルが作成されます。

10.18 ファイルをコピーする例

現在のディレクトリ内で新しいファイルの名前を指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1

Total of 4 files.
```

TRIP.TXT をコピーし， TRIP_2.TXT という名前にします。

```
$ COPY TRIP.TXT TRIP_2.TXT
```

確認します。

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1
TRIP_2.TXT;1

Total of 5 files.
```

他のディレクトリに新しい名前でファイルを置く場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1

Total of 4 files.
```

[YAMADA.REPORT.KOJIN]にある SAGYO.TXT と同じ内容で， MONTH_SAGYO_01.TXT;1 という名前のファイルを[YAMADA.REPORT.GROUP]に作成します。

```
$ COPY SAGYO.TXT [YAMADA.REPORT.GROUP]MONTH_SAGYO_01.TXT;1
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]

Directory DKA0:[YAMADA.REPORT.GROUP]

MONTH_KEIHI.TXT;1   MONTH_SAGYO_01.TXT;1
MONTH_SAGYO_04.TXT;1                    MONTH_SAGYO_05.TXT;1
MONTH_SAGYO_06.TXT;1                    MONTH_SAGYO_07.TXT;1
MONTH_SAGYO_08.TXT;1                    MONTH_SAGYO_09.TXT;1
MONTH_SAGYO_10.TXT;1                    MONTH_SAGYO_11.TXT;1
MONTH_SAGYO_12.TXT;1
```

```
Total of 11 files.
```

新しいファイルにディレクトリの名前だけを指定した場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.TXT;1         TRIP.TXT;1
Total of 3 files.
```

SAGYO.TXT と同じ内容のファイルを同じ名前で[YAMADA.REPORT.GROUP]に作成します。

```
$ COPY SAGYO.TXT [YAMADA.REPORT.GROUP]
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]
Directory DKA0:[YAMADA.REPORT.GROUP]
MONTH_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1
SAGYO.TXT;1
Total of 11 files.
```

複数のファイルをコピーする場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1
Total of 4 files.
```

SAGYO.TXT のすべてのバージョンのファイルと同じ内容のファイルを同じ名前で[YAMADA.REPORT.GROUP]に作成します。

```
$ COPY SAGYO.TXT;* [YAMADA.REPORT.GROUP]
```

確認します。

```
$ DIRECTORY [YAMADA.REPORT.GROUP]
Directory DKA0:[YAMADA.REPORT.GROUP]
```

```
MONTH_KEIHI.TXT;1   MONTH_SAGYO_04.TXT;1
MONTH_SAGYO_05.TXT;1                    MONTH_SAGYO_06.TXT;1
MONTH_SAGYO_07.TXT;1                    MONTH_SAGYO_08.TXT;1
MONTH_SAGYO_09.TXT;1                    MONTH_SAGYO_10.TXT;1
MONTH_SAGYO_11.TXT;1                    MONTH_SAGYO_12.TXT;1
SAGYO.TXT;2         SAGYO.TXT;1

Total of 12 files.
```

10.19 古いバージョンのファイルを消去する — PURGE

日本語 OpenVMS では，同じファイル名，ファイル・タイプのファイルを複数持つことができます。これは，日本語 OpenVMS が古いファイルを削除せずに新しいファイルを作成するためです。ファイルの違いはバージョン番号で区別されます。

最新バージョン (バージョン番号が最も大きいファイル) 以外のファイルが必要なくなった場合， PURGE コマンドを使用して，最新バージョン以外のファイルを削除することができます。このように最新バージョン以外のファイルを削除することをファイルを**パージする**といいます。

```
$ PURGE ファイル[,...]
        1
```

1 ファイル

パージしたいファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。ファイルは省略できます。省略した場合，現在のディレクトリにあるすべてのファイルを指定したとみなされます。バージョン番号は，指定できません。

メモ

エディタを使用してファイルを編集すると，編集前の内容のファイルが残ります。

このようにファイルを編集しても変更前の内容が残されているため，誤って内容を変更してしまった場合でも，もとの内容を利用することができます。

編集前の内容を保存する必要がなければ，PURGE コマンドを使用して，古いファイルを削除してください。

10.20 古いバージョンのファイルを消去 (パージ) する例

ファイルを指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;2         KEIHI.TXT;1         SAGYO.TXT;3         SAGYO.TXT;2
SAGYO.TXT;1         TRIP.TXT;1

Total of 6 files.
```

SAGYO.TXT の最新バージョン以外のファイルをパージします。

```
$ PURGE SAGYO.TXT
```

確認します。

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;2         KEIHI.TXT;1         SAGYO.TXT;3         TRIP.TXT;1

Total of 4 files.
```

ファイルを指定しなかった場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;2         KEIHI.TXT;1         SAGYO.TXT;5         SAGYO.TXT;4
SAGYO.TXT;3         TRIP.TXT;1

Total of 6 files.
```

[YAMADA.REPORT.KOJIN]にあるすべてのファイルの最新バージョン以外のファイルをパージします。

```
$ PURGE
```

確認します。

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;2         SAGYO.TXT;5         TRIP.TXT;1

Total of 3 files.
```

10.21 ファイルを削除する — DELETE

DELETE コマンドを使用すると，すべてのバージョンのファイルを削除することができます。また，特定のバージョンのファイルだけを削除することもできます。

```
$ DELETE ファイル[,...]
         1
```

1 ファイル

削除するファイルの名前。

1 つまたは複数のファイルを指定できます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。

バージョン番号は必ず指定しなければなりません。異なるバージョンのファイルをすべて削除する場合は，バージョン番号にアスタリスク・ワイルドカード文字 (*) を指定します。

注意

ファイルは削除すると，その後は編集したり内容を見ることはできません。したがって，ファイルを削除する場合は削除してよいかどうかを必ず確認してください。

10.22 ファイルを削除する例

特定のバージョン番号を指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;3         SAGYO.TXT;2         SAGYO.TXT;1
TRIP.TXT;1

Total of 5 files.
```

[YAMADA.REPORT.KOJIN]にある SAGYO.TXT;3 だけを削除します。

```
$ DELETE SAGYO.TXT;3
```

確認します。

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1         SAGYO.TXT;2         SAGYO.TXT;1         TRIP.TXT;1

Total of 4 files.
```

バージョン番号にアスタリスク・ワイルドカード文字 (*) を指定した場合

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1          SAGYO.TXT;3          SAGYO.TXT;2          SAGYO.TXT;1
TRIP.TXT;1

Total of 5 files.
```

[YAMADA.REPORT.KOJIN]にある SAGYO.TXT のすべてのバージョンのファイルを削除します。

```
$ DELETE SAGYO.TXT;*
```

確認します。

```
$ DIRECTORY

Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1          TRIP.TXT;1

Total of 2 files.
```

10.23 削除するかどうかを確認する — DELETE/CONFIRM

1. 次のように DELETE コマンドを実行します。

   ```
   $ DELETE/CONFIRM MONTH_SAGYO_*.TXT;*
   ```

2. ファイルを削除するかどうかを確認するメッセージが表示されます。

   ```
   DKA0:[YAMADA.REPORT.GROUP]MONTH_SAGYO_04.TXT;1 を
   削除しますか ? [N]:
   ```

3. ファイルを削除する場合は， Y を入力します。

   ファイルを削除しない場合は， N を入力します。

10.24 ディレクトリを削除する — DELETE

ディレクトリを削除する場合は，**ディレクトリ・ファイル**を削除します。

ディレクトリ・ファイルは，普通のファイルと同様に削除することができますが，次の 2 つの条件が必要です。

- ディレクトリ・ファイルの中にサブディレクトリ・ファイルや普通のファイルが登録されていると削除できません。したがって，ディレクトリ・ファイルを削除する場合は，そこに登録されているファイルやディレクトリ・ファイルをまず削除します。

  あるファイルだけを残したい場合は，そのファイルを別のディレクトリに移して保存します。

- ディレクトリの保護コードに D (DELETE) が必要です。

  (保護コードについては，第 11.1 節および第 11.5 節を参照。)

10.25 ディレクトリを削除する例

ディレクトリにあるファイルをすべて削除する場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT.KOJIN]
KEIHI.TXT;1         TRIP.TXT;1
Total of 2 files.
```

ディレクトリにあるすべてのファイルを削除します。

```
$ DELETE *.*;*
```

確認します。

```
$ DIRECTORY
%DIRECT-W-NOFILES, no files found
```

ディレクトリ・ファイルを削除する場合

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT]
GROUP.DIR;1         KOJIN.DIR;1
Total of 2 files.
```

ディレクトリ・ファイル KOJIN.DIR を削除します。 KOJIN.DIR には登録されているファイルが 1 つもないことが必要です。 (登録されているファイルがある場合は，そのファイルを削除してから， KOJIN.DIR を削除します。)

```
$ DELETE KOJIN.DIR;1
```

確認します。

```
$ DIRECTORY
Directory DKA0:[YAMADA.REPORT]
GROUP.DIR;1
Total of 1 files.
```

10.26 複数のファイルの内容を 1 つのファイルにまとめる — COPY

COPY コマンドを使用して，複数のファイルの内容を連結し，1 つの新しいファイルにまとめることができます。

```
$ COPY 入力ファイル[,...] 出力ファイル
          1                2
```

1 入力ファイル

連結する入力ファイルの名前。

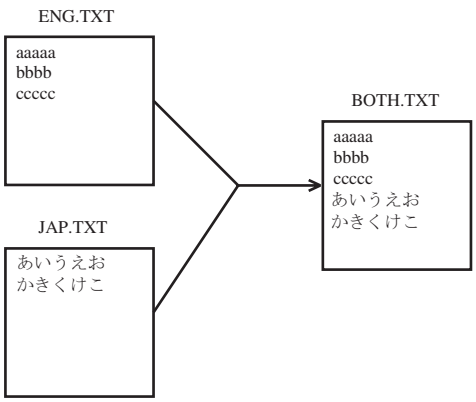
複数のファイルを指定する場合は，ファイルの名前をコンマで区切ります。入力ファイルは，すべて指定された順番に連結されます。

2 出力ファイル

入力ファイルがコピーされる出力ファイルの名前。

たとえば，ENG.TXT という名前のファイルと，JAP.TXT という名前のファイルの内容を連結して，新しく BOTH.TXT というファイルを作成します。

```
$ COPY ENG.TXT,JAP.TXT BOTH.TXT
```



この例のように，既存のファイルはそのまま残して，複数のファイルの内容を 1 つにまとめた新しいファイルを作成したいときは，COPY コマンドを使用します。

10.27 複数のファイルの内容を 1 つのファイルに追加する —APPEND

APPEND コマンドを使用して，指定したファイルに複数のファイルの内容を追加することができます。

```
$ APPEND 入力ファイル[,...] 出力ファイル
           1                2
```

1 入力ファイル

追加する入力ファイルの名前。

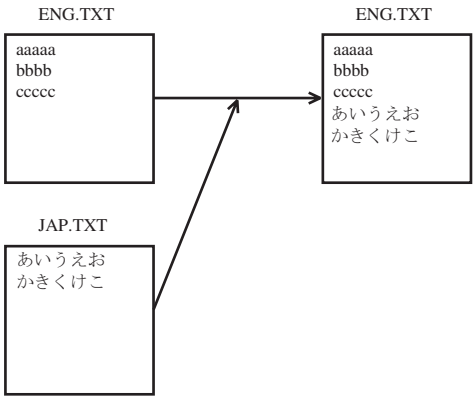
複数のファイルを指定する場合は，ファイル名をコンマで区切ます。入力ファイルは，すべて指定された順番に出力ファイルの最後に追加されます。

2 出力ファイル

入力ファイルを追加する出力ファイルの名前。

たとえば， JAP.TXT という名前のファイルの内容を， ENG.TXT という名前のファイルの最後に追加します。

```
$ APPEND JAP.TXT ENG.TXT
```



この例のように， APPEND コマンドを使うと，複数のファイルの内容を追加した新しいファイルが作成されるのではなく，指定した出力ファイル (この例では, ENG.TXT) の内容が変更されることに注意してください。

10.28 ファイルの内容を並べかえる —SORT

SORT コマンドは，指定したファイルの内容を，音読み，訓読みなどの順序に従って，並べかえます。

```
$ SORT 入力ファイル[,...] 出力ファイル
          1                2
```

1 入力ファイル

内容を並べかえる入力ファイルの名前。

複数のファイルを指定する場合は，ファイル名をコンマで区切ます。

2 出力ファイル

内容を並べかえた結果を書き込む出力ファイルの名前。

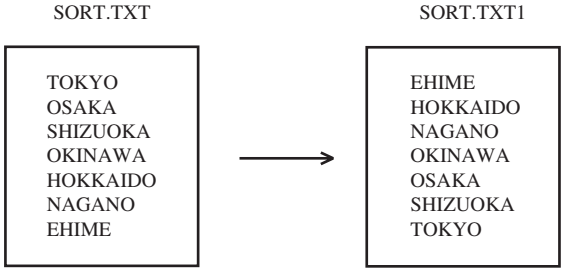
修飾子で並べかえる順序を指定しないと，英文の場合はアルファベット順に，日本語文の場合は JIS コード・テーブル順に並べかえます。

並べかえる順序には，次のものがあります。

- アルファベット順
- 音読み
- 訓読み
- 部首コード
- 総画数
- 国語辞典方式，全角振り仮名
- 国語辞典方式，半角振り仮名
- JIS コード・テーブル

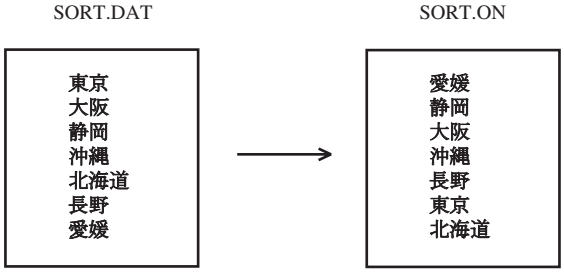
アルファベット順の例

```
$ SORT SORT.TXT SORT.TXT1
```



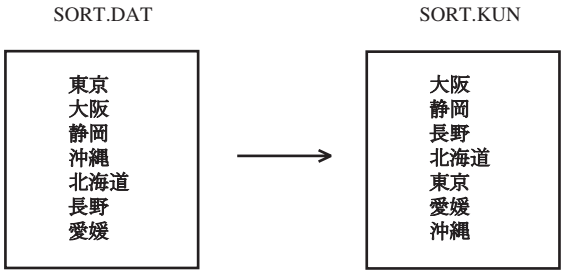
音読み順の例

```
$ SORT/KEY=(POSITION:1, SIZE:2, ONYOMI) SORT.DAT SORT.ON
```



訓読み順の例

```
$ SORT/KEY=(POSITION:1, SIZE:2, KUNYOMI) SORT.DAT SORT.KUN
```



10.29 コラム：複数バージョンのファイルを作らない方法

日本語 OpenVMS では，ファイル名，ファイル・タイプが同じファイルをバージョン番号の違いで管理できるのは便利ですが， PC や UNIX のユーザにとっては初めての概念です。うっかりたくさんのバージョンのファイルを作ってしまうことも珍しくありません。

そのようなことを避けるために，バージョンの数を制限する方法があります。バージョンの数が制限されたファイルは，新しいバージョンのファイルが作成される時に，自動的に最も古いバージョンのファイルを削除するので，同じファイル名のファイルが多数作成されてしまうのを避けることができます。

バージョンの数を制限するには，次のようにします。

```
$ SET FILE /VERSION_LIMIT=3 LIMITED.TXT;*
```

上記例では， LIMITED.TXT のバージョンの数を 3 つまでに制限します。バージョン番号には必ずアスタリスク (*) を指定してください。アスタリスクを指定しないと，一番新しいバージョンのファイルに対してだけバージョンの数の制限が適用されます。

この場合， LIMITED.TXT のファイル数が 3 つ未満の場合には，通常通りにファイルが作成されます。 LIMITED.TXT が既に 3 つ存在する状態で，新しいバージョンの LIMITED.TXT を作成しようとすると，自動的に一番古いバージョンの LIMITED.TXT が削除されます。

SET FILE /VERSION_LIMIT コマンド自身はファイルの削除を行わないので，バージョンの数を設定する前に既に設定したい数よりも多くのバージョンのファイルがある場合には， PURGE コマンドを使って古いバージョンのファイルを削除してください。

★ディレクトリにバージョン数の制限を設定する方法★

特定のディレクトリにバージョン数の制限を設定して，その中に登録されるすべてのファイルのバージョン数の制限をする方法もあります。

ディレクトリにはデフォルト・バージョン・リミットがあり，ディレクトリに新しくファイルが作成されるときに，その制限数が自動的にファイルに適用されます。

ディレクトリのデフォルト・バージョン・リミットを設定するには，次のようにします。

```
$ SET DIRECTORY /VERSION_LIMIT=3 TMP.DIR
```

上記例では， TMP ディレクトリのデフォルト・バージョン・リミットを 3 に設定します。設定後，この TMP ディレクトリ上で新たにファイルを作成すると，そのファイルのバージョンの数は自動的に 3 までに制限されます。他のディレクトリからファイルをコピーしてきた場合も，自動的に 3 に制限されます。

11

ファイルを保護する

この章では

通常，ユーザが作成したファイルはユーザが自由に操作できますが，日本語 OpenVMS が作成した特殊なファイルやディレクトリ・ファイルは誤って削除すると問題が生じることがあります。また，他のユーザのファイルを勝手に操作できても困ります。そこで，日本語 OpenVMS はファイルを保護するためのしくみを用意しています。

ここでは，ファイルの保護について説明します。

関連資料

- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS ユーザーズ・マニュアル』
- 『日本語 OpenVMS 概説書』

11.1 ファイルの保護コード

日本語 OpenVMS では，個々のファイルに対して，許可された操作と許可されない操作を設定することができます。設定できる操作を表 11–1 に示します。

表 11–1 ファイル保護のために設定できる操作

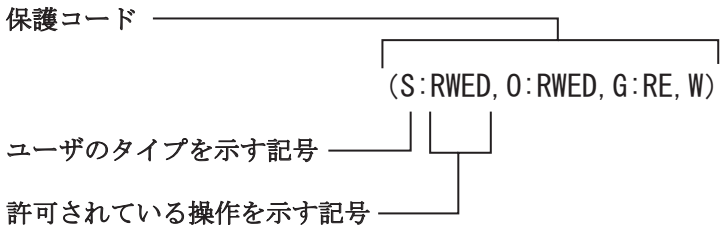
ファイルへのアクセス権	ファイルに対して許可される操作
READ	ファイルの読み込み ファイルの内容を読むことができる。この操作が許可されていないと，ファイルの内容を表示したり，印刷したりすることができない。
WRITE	ファイルへの書き込み ファイルにデータを書き込むことができる。この操作が許可されていないと，データを追加することができない。
EXECUTE	ファイルの実行/作成 ファイルがイメージ・ファイルであれば実行できる。
DELETE	ファイルの削除 ファイルを削除することができる。この操作が許可されていないと，ファイルを削除することができない。

また，これらの操作ができるかどうかは，ユーザごとに設定できます。ユーザは，次の 4 つの種類に分類されます。

ユーザのタイプ	説明
SYSTEM (システム)	特別な権利をもつユーザ。通常は，システム管理者。
OWNER (所有者)	ファイルやディレクトリの所有者。つまり，ファイルやディレクトリを作成したユーザ。
GROUP (グループ)	所有者と同じグループに登録されているユーザ。システム管理者が登録する。
WORLD (ワールド)	すべてのユーザ。

次に保護コードの指定例を示します。ユーザのタイプやアクセス権は英単語の頭一文字を使用して設定することができます。

図 11–1 保護コードの指定例



この例では，次のように設定されています。

ユーザのタイプ	許可される操作
SYSTEM	読み込み (READ)，書き込み (WRITE)，実行 (EXECUTE)，削除 (DELETE)
OWNER	読み込み (READ)，書き込み (WRITE)，実行 (EXECUTE)，削除 (DELETE)
GROUP	読み込み (READ)，実行 (EXECUTE)
WORLD	なし

11.2 ファイルの保護コードを表示する — DIRECTORY/PROTECTION

ファイルの保護コードを表示するには，次のようにします。

```
$ DIRECTORY/PROTECTION ファイル指定
```

11.3 ファイルの保護コードを変更する — SET PROTECTION

ファイルの保護コードを変更するには，次のようにします。

```
$ SET SECURITY/PROTECTION=(保護コード) ファイル指定
```

11.4 デフォルトの保護コード

日本語 OpenVMS はシステム全体に共通なデフォルトの保護コードを持っています。このデフォルトの保護コードはプロセス単位で変更することができ，変更後に作成するファイルには，変更した保護コードが与えられます。

たとえば，新しくファイルを作成すると，そのファイルにはデフォルトの保護コードが与えられます。また，すでに存在しているファイルの新しいバージョンを作成する

場合（エディタを使ってファイルを編集しなおす場合など）は，もとのファイルの保護コードが与えられます。

11.4.1 デフォルトの保護コードを表示する — SHOW PROTECTION

デフォルトの保護コードを表示するには次のようにします。

```
$ SHOW PROTECTION
```

11.4.2 デフォルトの保護コードを変更する — SET PROTECTION

デフォルトの保護コードを変更するには，次のようにします。

```
$ SET PROTECTION=(保護コード)/DEFAULT
```

```
$ SET PROTECTION=(S:RWED,O:RWED,G:R,W:R)/DEFAULT

$ SHOW PROTECTION
SYSTEM=RWED,OWNER=RWED,GROUP=R,WORLD=R
```

11.5 ディレクトリを保護する

ディレクトリも 1 つのファイルであり，保護コードが与えられます。ディレクトリの保護はファイルの保護に優先します。つまり，個別ファイルの保護コードである操作が認められていても，ディレクトリの保護コードでその操作が禁止されていればそのファイルの操作はできません。

11.5.1 ディレクトリの保護コード

ディレクトリの保護コードは，1 階層上のディレクトリの保護コードから D(DELETE) を除いたものになっています。ディレクトリの保護コードはディレクトリの作成時に指定することもできます。

```
$ CREATE/DIRECTORY/PROTECTION=(保護コード) [ディレクトリ指定]
```

また，ディレクトリの保護コードの参照や変更は一般のファイルと同様に行います。

ディレクトリへのアクセス権	ディレクトリに対して許可される操作
READ	ディレクトリ・ファイルの内容を見る ファイルの一覧を見ることができる

ディレクトリへのアクセス権	ディレクトリに対して許可される操作
WRITE	ディレクトリ・ファイルの内容の変更 新しいファイルを登録したり，削除できる
EXECUTE	ディレクトリの下のファイルを特定してアクセス READ アクセス権がない時だけ意味を持ち，ディレクトリの下のファイルの名前を，ワイルド・カードを使わずにきちんと指定した場合だけ，実行や READ アクセスができる
DELETE	ディレクトリ・ファイル自身を削除 ディレクトリ・ファイル自身を削除したり RENAME できる

11.6 コラム：ファイルの保護 - UIC -

ファイルは，保護コードが与えられて保護されるだけでなく， UIC (User Identification Code) が与えられることによっても保護されます。

UIC はユーザを識別する番号であり，システム管理者によってユーザ単位に与えられます。たとえば，[JVMS,YAMADA]のようになります。ユーザは，システムにログインして利用している間，システム管理者に与えられた UIC を持ちます。

ファイルが作成されると，作成を行ったユーザの持っている UIC がそのファイルに付けられます。これをファイルの**オーナ** UIC を呼びます。

次のようにすると， UIC を見ることができます。

```
$ DIRECTORY /SECURITY ファイル指定
```

UIC は，次のような形式になっています。

表 11–2 UIC の形式

[g,m]	g : グループ番号 0 ～ 37776 の 8 進数 m : メンバ番号 0 ～ 177776 の 8 進数
[member]	member : メンバ名 1 ～ 31 文字の文字列
[group,member]	group : グループ名 1 ～ 31 文字の文字列

数字形式の UIC [g,m]と文字形式の UIC [member]は 1 対 1 に対応します。グループ名 (group) は，システム管理者がグループ番号に対応づけて付与します。

ファイルの作成者の UIC と，そのファイルにアクセスするユーザの UIC との比較により，ユーザは次の 4 つのタイプに分類できます。

表 11–3 ファイルにアクセスするユーザのタイプ

ユーザのタイプ	条件
SYSTEM	1 ～ 10 （8 進数）のグループをもつユーザ（メンバ番号は無視）
OWNER	ファイルの UIC と同じ UIC をもつユーザ（グループ番号，メンバ番号ともに一致）
GROUP	ファイルの UIC のグループ番号と同じグループ番号の UIC をもつユーザ（メンバ番号は無視）
WORLD	すべてのユーザ

ユーザは，必要に応じてファイルに ACL （Access Control List）という保護情報を付加することができます。 ACL を利用すると， UIC ベースの保護より，さらに細かい保護設定ができます。

UIC や ACL についての説明は『OpenVMS システム管理者マニュアル』，『OpenVMS ユーザーズ・マニュアル』等を参照してください。

第3部
日本語OpenVMSの便利な機能を使う

第 3 部では，日本語 OpenVMS のさらに便利な機能の操作方法を説明します。

項目	参照箇所
ファイルの内容を印刷する	第 12 章
バッチ・キューを利用する	第 13 章
プロセスについて	第 14 章
シンボルを使用する	第 15 章
論理名を使用する	第 16 章
コマンド・プロシージャを使用する	第 17 章

ここでは，各機能についての説明は簡単に記していますので，詳しい説明は関連資料をお読みください。

12

ファイルの内容を印刷する

この章では

多くのプリンタは，複数のユーザにより共有して使用されます。しかし，ユーザが，1 台のプリンタで同時に印刷することはできませんので，なんらかの制御を行って，順番に印刷していく必要があります。日本語 OpenVMS は，その制御を行います。

ここでは，印刷の流れ，印刷の方法，状態の確認方法について説明します。

関連資料

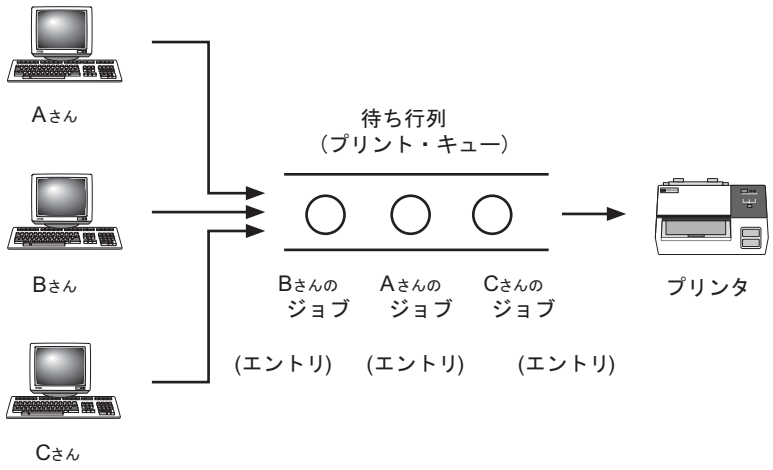
- 『OpenVMS ユーザーズ・マニュアル』
- 『OpenVMS DCL ディクショナリ』
- 『日本語 DECprint Supervisor for OpenVMS ユーザーズ・ガイド』

12.1 印刷の流れ

ファイルを印刷するには， PRINT コマンドを使用します。ユーザが PRINT コマンドを発行すると**プリント・ジョブ**が作成されます。プリント・ジョブは，システムがプリント処理を行う単位です。

ユーザがジョブを依頼しても，すぐには印刷されません。日本語 OpenVMS は，ジョブを待ち行列の中に入れ，ジョブが依頼された時刻の順に 1 つずつ印刷していきます。このとき，日本語 OpenVMS は，複数のユーザのジョブを**エントリ**という呼び名で取り扱います。エントリには，依頼された時刻の順に通番で番号が付けられます。

ユーザ，ユーザのジョブ，エントリ，プリンタの関係を図で示すと，次のようになります。



DCL コマンドでプリンタを指定する際の擬似的な名前を，**プリント・キュー**といいます。プリント・キューは任意の名前で，システム管理者が付けます。プリント・キューはシステムが管理する出力待ち行列のことです。

プリント・ジョブは，プリント・キューに登録されプリンタへ出力されます。

12.2 印刷する — PRINT

ファイルの内容をプリンタで印刷するには， PRINT コマンドを使用します。

PRINT コマンドの入力形式は，次のとおりです。

```
$ PRINT/QUEUE=プリント・キュー    印刷するファイル[, . . . ]
                  1                        2
```

1 プリント・キュー

プリント・キューの名前

2 印刷するファイル

印刷するファイルの名前

複数のファイルを指定することもできます。複数のファイルを指定する場合，ファイルの名前をコンマで区切ります。

ファイル・タイプは省略できます。省略した場合， LIS を指定したものと見なされます。バージョン番号も省略でき，最新のバージョンを指定したものと見なされます。

ファイル名，ファイル・タイプ，バージョン番号にはワイルドカード文字を指定できます。

PRINT コマンドを実行すると，次のように表示されます。

```
$ PRINT/QUEUE=PRINTER1 MONTH_KEIHI.TXT
Job MONTH_KEIHI (queue PRINTER1, entry 355) started on PRINTER1
     1                2             3               4
```

1 ジョブの名前。 (通常，ファイル名になる。)

2 プリント・キューの名前

3 エントリ番号

4 プリント・キューの名前

12.3 印刷状態の通知を指定する — PRINT/NOTIFY

印刷状態をディスプレイに通知することを指定できます。次のように入力します。ファイルは複数指定できます。

```
$ PRINT/QUEUE=プリント・キュー/NOTIFY 印刷するファイル
```

次のような印刷状態が通知されます。

- started — 印刷開始
- completed — 印刷終了
- Retained on error — 印刷中に障害が発生

12.4 印刷パラメータを指定する — PRINT/PARAMETERS

PRINT コマンドに/PARAMETERS 修飾子を使用すると，印刷のデータ・タイプや部数の指定などができます。

```
$ PRINT/QUEUE=プリント・キュー/PARAMETERS=指定パラメータ ファイル名
```

PRINT コマンドのパラメータには次のようなものがあります。(ここに記載してあるキーワードは一部のみです。)

表 12–1 PRINT **コマンドのパラメーター覧**

印刷するデータ・タイプを指定	
パラメータ	DATA_TYPE
キーワード	KANJI, POSTSCRIPT
例	PRINT/QUEUE=**プリント・キュー**/PARAM=DATA=KANJI **ファイル名**
1 枚の用紙片面に複数ページ分を印刷する	
パラメータ	NUMBER_UP
キーワード	0 - 100
例	PRINT/QUEUE=**プリント・キュー**/PARAM=NUMBER_UP=4 **ファイル名**
印刷する範囲の最初と最後のページ	
パラメータ	PAGE_LIMIT
キーワード	1 - 10000
例	PRINT/QUEUE=**プリント・キュー**/PARAM="PAGE_LIMIT=(2,8)"**ファイル名**
用紙の方向の指定	
パラメータ	PAGE_ORIENTATION
キーワード	PROTRAIT (縦), LANDSCAPE (横)
例	PRINT/QUEUE=**プリント・キュー**/PARAM=PAGE_ORIENT=LAND **ファイル名**
印刷する部数	
パラメータ	SHEET_COUNT
キーワード	1 - 10000
例	PRINT/QUEUE=**プリント・キュー**/PARAM=SHEET_COUNT=3 **ファイル名**
用紙サイズの指定	
パラメータ	SHEET_SIZE
キーワード	LETTER, LEGAL, A3, A4, B5
例	PRINT/QUEUE=**プリント・キュー**/PARAM=SHEET_SIZE=A4 **ファイル名**
片面，両面印刷の指定	
パラメータ	SIDES
キーワード	1, 2
例	PRINT/QUEUE=**プリント・キュー**/PARAM=SIDES=2 **ファイル名**

パラメータは次のように複数指定 (8 個まで) することもできます。

```
PRINT/QUEUE=プリント・キュー/PARAM=(SIDE=2,NUMBER_UP=2,SHEET_SIZE=A) ファイル名
```

図 12–1 プリント・パラメーター覧表

日本語 DECprint Supervisor for OpenVMS プリント・パラメータ 一覧表
ユーザ・リファレンス・カード

概要	主なパラメータ	主なキーワード	例
印刷するデータ・タイプを指定	DATA_TYPE	ANSI KANJI LA_KANJI POSTSCRIPT	PRINT/QUEUE=*queue-name*/PARAM=DATA=KANJI *filename*
給紙トレイの設定 *	INPUT_TRAY	TOP BOTTOM TRAY_1 TRAY_2... MANUAL_FEED	PRINT/QUEUE=*queue-name*/PARAM=INPUT_TRAY=TOP *filename*
排紙トレイの設定 *	OUTPUT_TRAY	UPPER LOWER FACE_UP TOP SIDE BIN_1	PRINT/QUEUE=*queue-name*/PARAM=OUTPUT_TRAY=SIDE *filename*
用紙に印刷するページ数（図 1 参照）	NUMBER_UP	0 - 100	PRINT/QUEUE=*queue-name*/PARAM=NUMBER_UP=4 *filename*
印刷する範囲の最初と最後のページ	PAGE_LIMIT	1 - 10000	PRINT/QUEUE=*queue-name*/PARAM="PAGE_LIMIT=(2,8)" *filename*
ポートレートかランドスケープの指定（図 2,3,6 参照）	PAGE_ORIEN-TATION	PORTRAIT LANDSCAPE	PRINT/QUEUE=*queue-name*/PARAM=PAGE_ORIENT=LAND *filename*
用紙を印刷する枚数	SHEET_COUNT	1 - 10000	PRINT/QUEUE=*queue-name*/PARAM=SHEET_COUNT=3 *filename*
用紙サイズの指定 *	SHEET_SIZE	LETTER LEDGER LEGAL EXECUTIVE A A3 A4 A5 B B4 B5	PRINT/QUEUE=*queue-name*/PARAM=SHEET_SIZE=A4 *filename*
片面、両面印刷の指定（図 4,5,6 参照）	SIDES	1 2 TUMBLE	PRINT/QUEUE=*queue-name*/PARAM=SIDES=2 *filename*
パラメータを複数指定する場合	パラメータの指定は 8 つまで	PRINT/QUEUE=*queue-name*/PARAM=(SIDES=2,NUMBER_UP=4,PAGE_O=LAND,SHEET_S=A) *filename*	

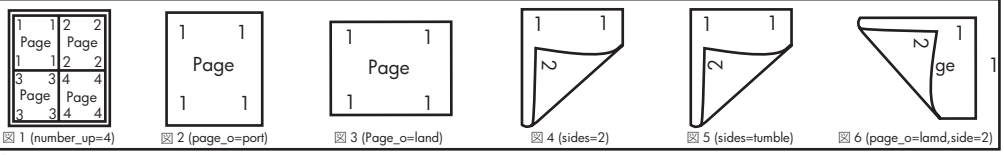


図 1 (number_up=4) 図 2 (page_o=port) 図 3 (Page_o=land) 図 4 (sides=2) 図 5 (sides=tumble) 図 6 (page_o=lamd,side=2)

* プリンタによって使用できるキーワードが異なります。
詳細は，オンライン・ヘルプまたは『日本語 DECprint Supervisor for OpenVMS ユーザーズ・ガイド』を参照してください。

OpenVMS の web サイト (http://www.hp.com/jp/openvms/) で図 12–1 のようなプリント・パラメータ一覧表 (PDF) を提供していますので合わせてご参照ください。

12.5 印刷の状態を確認する

PRINT コマンドを実行しても，他のユーザのジョブが終了していない場合は，ジョブはプリント・キューに登録された状態で印刷の順番を待ち，すぐには印刷されません。また，なんらかの理由で障害が起こる場合もあるので，印刷の状態を確認することは大切です。

印刷には，次のような状態があります。

- Printing — 印刷中
- Pending — 印刷を待っている状態
- Retained on error — 印刷中に障害が発生

12.6 ユーザの登録したジョブの状態を確認する — SHOW ENTRY

ユーザのプリント・ジョブの状態を確認するには， SHOW ENTRY コマンドを使用します。

```
$ SHOW ENTRY
```

SHOW ENTRY コマンドを実行すると，次のように画面に表示されます。

```
  Entry  Jobname         Username     Blocks  Status
  -----  -------         --------     ------  ------
    355  MONTH_KEIHI     YAMADA            3  Pending
    1       2               3            4    5
         On Terminal queue PRINTER1
```

1 エントリ番号

2 ジョブの名前

3 ユーザ名

4 ファイルの大きさ。単位はブロック (1 ブロックは 512 バイト)

5 印刷の状態

12.7 キューの状態を確認する — SHOW QUEUE

キューの状態やプリント・ジョブの状態を確認するには， SHOW QUEUE コマンドを使用します。

```
$ SHOW QUEUE [/修飾子] [キュー名]
```

12.8 印刷を中止する — DELETE/ENTRY

印刷の障害が発生した場合や，印刷を中止したい場合には， DELETE/ENTRY コマンドを使用します。 SHOW ENTRY コマンドを使用して，中止したいジョブのエントリ番号を調べておく必要があります。

```
$ DELETE/ENTRY=エントリ番号
```

エントリ番号は，中止したいジョブのエントリ番号です。

DELETE/ENTRY コマンドを実行して確実に印刷を中止できたことを確認するには，SHOW ENTRY コマンドを使用します。

12.9 印刷を中止する例

印刷を実行します。

```
$ PRINT/QUEUE=PRINTER1 MONTH_KEIHI.TXT
Job MONTH_KEIHI (queue PRINTER1, entry 355) pending
    pending status caused by queue busy
```

ジョブのエントリ番号を調べます。

```
$ SHOW ENTRY
  Entry  Jobname         Username     Blocks  Status
  -----  -------         --------     ------  ------
    355  MONTH_KEIHI     YAMADA            3  Pending
         On Terminal queue PRINTER1
```

印刷を中止します。

```
$ DELETE/ENTRY=355
```

印刷を中止できたことを確認します。

```
$ SHOW ENTRY
%JBC-E-NOSUCHENT, このようなエントリは存在しません。
```

13

バッチ・キューを利用する

この章では

会話型の処理では，通常，ある 1 つのコマンドの実行が終了するまで，次のコマンドは入力できません。そのため実行に長い時間が必要なプログラムのコンパイルやリンクなどの操作に対応するため，日本語 OpenVMS では，バッチ・キューという待ち行列を用意しています。

ここでは，非会話型のプロセスであるバッチ・ジョブとバッチ・キューの利用について説明します。

関連資料

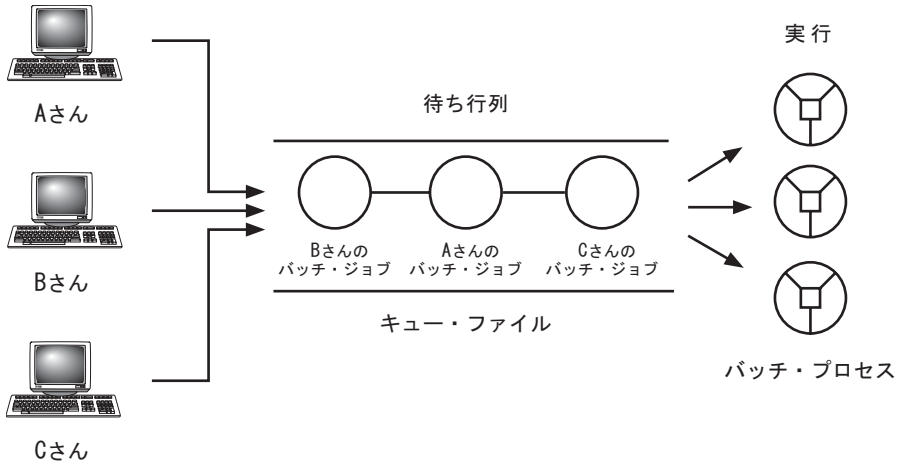
- 『OpenVMS ユーザーズ・マニュアル』
- 『OpenVMS DCL ディレクトリ』

13.1 バッチ・ジョブとバッチ・キュー

バッチ・ジョブは，非会話型の独立プロセスで実行される仕事の単位です。固有のプロセスで実行されるため，同時に異なる処理を行うプロセスを複数指定することができます。また，プログラムの実行開始時間を指定したり，必要なリソースを細かく指定することもできます。(プロセスについては第 14 章をご覧ください。)

ユーザが SUBMIT コマンドを発行するとバッチ・ジョブが作成され，バッチ・キューに登録されます。**バッチ・キュー**は，バッチ・ジョブの待ち行列です。図 13–1 にバッチ・ジョブの流れを示します。

図 13–1 バッチ・ジョブの流れ



13.2 バッチ・ジョブを登録する — SUBMIT

バッチ・キューにバッチ・ジョブを登録するには，次のようにします。

```
$ SUBMIT [/修飾子] コマンド・プロシージャ名 [/修飾子]
```

実行するファイルはコマンド・プロシージャ・ファイルです。(コマンド・プロシージャについては第 17 章を参照してください。)

修飾子には次のようなものがあります。

/AFTER=時刻指定	ジョブの実行開始時刻を指定
/HOLD	ジョブ実行の休止指定
/NOTIFY	ジョブの実行終了を通知

/QUEUE=キュー名	実行するバッチ・キューを指定
/RESTART	ジョブの再スタート指定

13.3 バッチ・ジョブの状態を表示する — SHOW ENTRY

ユーザの登録したバッチ・ジョブ状態を表示するには，次のようにします。

```
$ SHOW ENTRY [エントリ番号,...]
```

エントリ番号は複数指定することもできます。複数のエントリ番号を指定する場合，エントリ番号はコンマで区切ります。

バッチ・ジョブの状態には，次の 4 つの状態があります。

状態	説明
Executing	実行中
Pending	実行待ちの状態
Holding until time	指定時刻を待っている状態
Holding	保留中

13.4 バッチ・キューの状態を表示する — SHOW QUEUE

バッチ・ジョブを登録したバッチ・キューの状態を表示するには，次のようにします。

```
$ SHOW QUEUE/BATCH [/修飾子] [キュー名]
```

13.5 バッチ・ジョブの属性を変更する — SET ENTRY

バッチ・キューに登録した後，ジョブの属性を変更することができます。ジョブの属性を変更するには次のようにします。

```
$ SET ENTRY ジョブ番号 /修飾子
```

13.6 バッチ・ジョブを削除する — DELETE/ENTRY

バッチ・ジョブを削除するには，次のようにします。

```
$ DELETE/ENTRY=(エントリ番号[,...]) [キュー名]
```

エントリ番号は複数指定することもできます。エントリ番号を 1 つだけ指定する場合は ( ) を省略できます。

13.7 バッチ・ジョブを再起動する

システム障害等が発生した後に，ジョブを再始動させるには，次のようにします。

```
$ SUBMIT/RESTART コマンド・プロシージャ名
```

14
プロセスについて

この章では

日本語 OpenVMS は，同時に複数の利用者が作業をしたり，複数のプログラムを実行できるマルチタスクのオペレーティング・システムです。しかし，利用者からは自分が 1 台のコンピュータを専有して使用しているように見えるので，他の利用者が何をしているのかを気にする必要はありません。これは，日本語 OpenVMS が利用者一人一人にプロセスというものを与えているからです。

この章では，プロセスについて説明します。

関連資料

- 『OpenVMS ユーザーズ・マニュアル』
- 『OpenVMS DCL ディクショナリ』

14.1 プロセスとは

プロセスは，システムとの会話を可能にするために日本語 OpenVMS で作成される "ユーザの作業環境 "あるいは "プログラムの実行環境 "です。 DCL コマンドやプログラムは，必ずこのプロセスという環境の中で実行されます。

ユーザが次のいずれかのタスクを実行すると，システムはプロセスを作成します。

- ログイン

  システムはユーザがログインするたびにそのユーザ用にプロセスを作成します。

- バッチ・ジョブの登録

  システムはバッチ・ジョブを受け取るとそのジョブ用にプロセスを作成します。バッチ・ジョブが完了すると，システムはプロセスを削除します。

- サブプロセスの生成

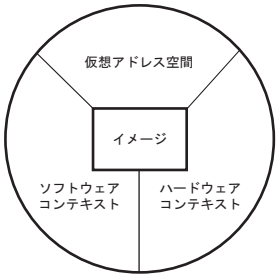
  ユーザが SPAWN コマンドを使用すると，システムはプロセスを作成します。

- プログラムの実行

  /DETACHED 修飾子または/UIC=uic 修飾子を使用してプログラムを実行すると，システムはプロセスを作成します。

図 14–1 は，プロセスの構成要素を概念的に表しています。

図 14–1 プロセスの構成要素



プロセスの構成要素は次のとおりです。

◆イメージ

実行可能なプログラムのことです。イメージは必ずプロセスの環境の中で実行されます。この環境を規定するのが他の 3 つの構成要素です。

◆仮想アドレス空間

イメージが参照している仮想アドレスの情報を持つ空間です。

◆ハードウェア・コンテキスト

CPU のことです。

◆ソフトウェア・コンテキスト

プロセスのソフトウェア状態を規定するもので，次のような項目を含んでいます。

- プロセスの優先順位 (Priority)

  プロセスの切り替えを行う際の優先順位
- 特権 (Privilege)

  どのような作業を行うことができるかを規定する権限
- クォータ (Quota)

  システムのいろいろな資源をどれだけ使用できるかを規定する制限値
- ユーザ識別コード (UIC)

これらの情報は，システム管理者がユーザ登録簿にユーザ名，パスワードを登録する際に設定します。

14.2 プロセスの種類

日本語 OpenVMS では，おもに次の 3 のプロセスの種類があります。

◆会話型プロセス

ユーザがログインした時に作られる会話型ログイン

ネット経由の COPY や FTP (ネットワーク・プロセス)

◆サブプロセス

SPAWN コマンドなどで作られる ($ SPAWN)

◆独立プロセス

バッチ処理をする時などに使われるバッチ・プロセス ($ SUBMIT)

指定されたイメージを実行するプロセスを作成する ($ RUN/DETACH)

14.3 自分のプロセスの状態を表示する — SHOW PROCESS

自分のプロセスの状態を確認するには次のようにします。

```
$ SHOW PROCESS
```

修飾子には次のようなものがあります。

/ALL	詳細な情報もつけて表示する
/SUBPROCESS	プロセスの親子関係を字下げして表示する

14.4 プロセス名を変更する — SET PROCESS

プロセス名を変更するには次のようにします。

```
$ SET PROCESS/NAME="プロセス名"
```

プロセス名を二重引用符 (") で囲った文字列に変更します。 15 字以内の文字列が指定できます。

14.5 サブプロセスを作成する — SPAWN

サブプロセスを作成し，制御をサブプロセスに移すには次のようにします。

```
$ SPAWN
```

修飾子には次のようなものがあります。

/NOWAIT	制御は親プロセスに置く

サブプロセスを終了するには，次のようにします。

```
$ LOGOUT
```

14.6 指定したプロセスに制御を移す — ATTACH

指定したプロセスに制御を移すには次のようにします。

```
$ ATTACH [/IDENTIFICATION=pid] [プロセス名]
```

14.7 指定したプロセスを強制終了する — STOP

プロセスを強制的に終了させるには次のようにします。

```
$ STOP [/IDENTIFICATION=pid] [プロセス名]
```

プロセス名，または PID (Process IDentification) の一方を指定します。

14.8 各種プロセスに関する情報をユーザ名で得る — SHOW USERS

各種プロセスに関する情報をユーザ名で得るには，次のようにします。

```
$ SHOW USERS [ユーザ名]
```

ユーザ名を指定するとそのユーザに関する情報のみが得られます。

修飾子には次のようなものがあります。

/FULL	各ユーザが持っているプロセスの情報を表示する
/INTERACTIVE	会話型プロセスのみを表示する
/SUBPROCESS	サブプロセスのみを表示する

14.9 システム上にある全プロセスを表示する — SHOW SYSTEM

システム上にあるすべてのプロセスを表示するには，次のようにします。

```
$ SHOW SYSTEM
```

修飾子には次のようなものがあります。

/BATCH	バッチ・プロセスのみを表示する
/INTERACTIVE	会話型プロセスのみを表示する
/NETWORK	ネットワーク・プロセスのみを表示する
/SUBPROCESS	サブプロセスのみを表示する

15
シンボルを使用する

この章では

日本語 OpenVMS では，コマンド列を別の文字列で置き換えることができます。この文字列のことをシンボルといいます。シンボルを使用すると長いコマンド列を短くしたり，操作時間を短縮したり，変数として使用することができます。

ここでは，シンボルの利用について説明します。

関連資料

- 『OpenVMS DCL ディクショナリ』
- 『OpenVMS ユーザーズ・マニュアル』

15.1 シンボルとは

コマンド列を別の任意の文字列で置き換えることができます。この文字列を，**シンボル**といいます。つまり，シンボルとはコマンド列の全体または一部につける別名と言えます。

DCL コマンドの中には，多くのパラメータや修飾子を必要とするために，コマンド列が非常に長くなるものがあります。そこで，シンボルを使用して長いコマンド列を別の文字列でおきかえて短くしたり，コマンド列内で変数として使用したりします。

いったんシンボルを定義すると，その後はシンボルを DCL コマンドと同様に使用することができます。定義したシンボルは，ログアウトするまで有効です。

15.2 シンボルを定義する

シンボルを定義するには次のようにします。

```
$ シンボル =[=] "DCL コマンド列の全体または一部"
    1                          2
```

1 シンボル

コマンド列を置き換える文字列。

シンボルは 1 ～ 255 文字の英数字，ドル記号 ($)，アンダーライン (_) の文字列を使用します。先頭は英字，$，_のいずれかにしなければいけません。 (数字は使用できません。)

2 コマンド列

置き換えるコマンド列。

実際にコマンド列を入力するように指定します。

15.3 シンボルの使用例

シンボル DSD に，ディレクトリ内のファイルをサイズと作成日時の情報も含めて一覧表示するコマンド列全体を割当てます。

```
$ DSD == "DIR/SIZE/DATE"
```

DCL コマンドのかわりに DSD と入力するとサイズと作成日時情報も含めたファイルの一覧が表示されます。

```
$ DSD

Directory DKA0:[YAMADA]

LOGIN.COM;1                1   9-MAY-2001 15:59:46.53
MAIL.MAI;1                27  22-JAN-2002 10:41:56.21
```

```
Total of 2 files, 28 blocks.
$
```

コマンドの短縮形としてのシンボルを使う際には，必要に応じて追加の修飾子やパラメータを指定することもできます。

次の例では，/OWNER 修飾子を追加し，/SIZE 修飾子のかわりに/SIZE=ALL 修飾子を指定しています。またパラメータとして *.COM を指定しています。

```
$ DSD/OWNER/SIZE=ALL *.COM

Directory DKA0:[YAMADA]

LOGIN.COM;1               1/9         9-MAY-2001 15:59:46.53  [VMS,YAMADA]

Total of 1 file, 1/9 blocks.
$
```

また，あるコマンドに対し，特定のパラメータを頻繁に指定する場合，そのパラメータもシンボルに含めてしまうことができます。次に例を示します。

```
$ DIRCOM == "DIR/SIZE/DATE *.COM"
$ DIRCOM

Directory DKA0:[YAMADA]

LOGIN.COM;1               1   9-MAY-2001 15:59:46.53

Total of 1 file, 1 blocks.
$
```

15.4 シンボルを確認する — SHOW SYMBOL

シンボルを確認するには， SHOW SYMBOL コマンドを使用します。

```
$ SHOW SYMBOL シンボル
```

すべてのシンボルを表示する時は，シンボル名のかわりに/ALL 修飾子を使います。

```
$ SHOW SYMBOL/ALL
```

15.5 シンボルを削除する — DELETE/SYMBOL

シンボルを削除するには次のようにします。

```
$ DELETE/SYMBOL シンボル
```

すべてのシンボルを削除する時は，シンボルのかわりに/ALL 修飾子を使います。また，シンボルはログアウトすると削除されます。

```
$ DELETE/SYMBOL/ALL
```

15.6 シンボルの省略形

DCL コマンドと同じように，シンボルも省略形を使うことができます。シンボルを定義するときに，シンボル名の中にアスタリスク (*) 文字を入れると，* 以後の文字は省略可能になります。

省略形を指定したシンボル TODAY を定義します。

```
$ TOD*AY =  "SHOW TIME"
```

省略形を使ってシンボル名を入力します。

```
$ TODAY
   14-FEB-2002  17:54:49
$ TODA
   14-FEB-2002  17:54:54
$ TOD
   14-FEB-2002  17:54:57
```

TO は省略形として指定していないため，エラーとなります。

```
$ TO
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\TO\
$
```

15.7 ローカル・シンボルとグローバル・シンボル

シンボルには有効範囲の違いで，ローカル・シンボルとグローバル・シンボルの 2 種類があります。一般に，ローカル・シンボルは，計算結果を一時的に格納しておくための変数に使います。グローバル・シンボルは，ユーザの環境を設定するために使います。

イコール記号 1 つ (=) で定義したシンボルのことをローカル・シンボルと言い，イコール記号 2 つ (= =) で定義したシンボルのことをグローバル・シンボルと言います。

同じ名前のローカル・シンボルとグローバル・シンボルを定義した場合は，ローカル・シンボルが優先されます。ローカル・シンボルを削除するとグローバル・シンボルを参照できるようになります。

グローバル・シンボルを定義するには，次のようにします。

```
$ シンボル == "DCL コマンド列の全体または一部"
```

15.8 グローバル・シンボルを参照する — SHOW SYMBOL/GLOBAL

グローバル・シンボルを参照するには，次のようにします。

```
$ SHOW SYMBOL/GLOBAL シンボル
```

すべてのグローバル・シンボルを表示する時は，シンボル名のかわりに/ALL 修飾子を使います。

```
$ SHOW SYMBOL/GLOBAL/ALL
```

15.9 グローバル・シンボルを削除する — DELELTE/SYMBOL/GLOBAL

グローバル・シンボルを削除するには，次のようにします。

```
$ DELETE/SYMBOL/GLOBAL シンボル
```

すべてのグローバル・シンボルを削除する時は，シンボルのかわりに/ALL 修飾子を使います。

```
$ DELETE/SYMBOL/GLOBAL/ALL
```

15.10 コマンド・プロシージャの中でシンボルを使う

コマンド・プロシージャとは DCL コマンドと使用されるデータ行を 1 つのテキスト・ファイルにまとめたものです。(詳しくは，第 17 章をご覧ください。)

通常，コマンド・プロシージャの中では，ローカル・シンボルを使います。これはコマンド・プロシージャ内で定義されたローカル・シンボルの有効範囲が，そのコマンド・プロシージャの中だけに限られるからです。コマンド・プロシージャの中で定義されたローカル・シンボルは，コマンド・プロシージャの実行を終了すると消去されます。

一方グローバル・シンボルは， DCL コマンド・ライン上でもコマンド・プロシージャの中でも定義，参照，変更，削除をすることができます。そのため，何らかの値を DCL プロンプト・レベルに返す場合や，ユーザの環境を設定する場合に使うことができます。

たとえば，ログイン・コマンド・プロシージャの中で，ユーザがよく使うコマンドの短縮形を定義する場合には，必ずグローバル・シンボルを使う必要があります。次に例を示します。

```
$ cd == "set default"
$ pwd == "show default"
$ who == "show user"
$ less == "type /page=save"
```

ログイン・コマンド・プロシージャ LOGIN.COM は，ログイン時に一連の DCL コマンドを自動的に実行するためのものです。(詳しくは，第 17 章をご覧ください。)

15.11 変数としてのシンボルを使用する

シンボルは，計算結果を一時的に格納しておくための変数として使うことができます。この機能は主にコマンド・プロシージャの中で使われますが， DCL プロンプト上でも使うことができます。次に例を示します。

```
$ x = (1 + 2) * 3
$ show symbol x
  X = 9   Hex = 00000009  Octal = 00000000011
$ surname = "山田"
$ show symbol surname
  SURNAME = "山田"
$
```

最初の例のようにシンボルに計算結果を代入すると，シンボルは数値変数になります。 2 番目の例のように文字列を代入すると，シンボルは文字列変数になります。シンボルがどちらの型になるかは代入時に決定されるので，あらかじめ定義しておく必要はありません。

15.12 特別な意味を持つシンボル

いくつかのシンボルは特別な目的のために使用されます。ユーザは必要に応じて，それらのシンボルを参照または上書きすることができます。

15.12.1 $STATUS (グローバル・シンボル)

直前に実行したコマンドやアプリケーションが，正常に実行されたかどうかを示します。正常に実行された場合は奇数が代入されます。異常終了した場合は偶数が代入されます。コマンド・プロシージャなどでアプリケーションを自動実行する場合のエラー処理に使うことができます。

$STATUS の値を調べることで，異常終了に関する情報や対処方法を得られる場合があります。(詳しくは，第 5 章をご覧ください。)

15.12.2 $SEVERITY (グローバル・シンボル)

常に$STATUS の下位 3 ビットの値と等しくなります。この部分は重大度レベルと呼ばれ，アプリケーションが異常終了した際の異常の重大度を示します。

値	重大度
0	警告 (Warning)
1	正常終了 (Success)
2	エラー (Error)
3	情報 (Information)
4	致命的なエラー (Fatal)

15.12.3 P1 ～ P8 (ローカル・シンボル)

コマンド・プロシージャ実行時に，コマンド・ライン上に指定されたパラメータです。コマンド・プロシージャの中では最大 8 個まで受け取ることができます。パラメータが指定されなかった場合や 8 個未満だった場合には，空文字列が代入されます。(詳しくは，第 17 章をご覧ください。)

15.12.4 $RESTART (グローバル・シンボル)

バッチ・ジョブを実行するためにコマンド・プロシージャの中で，障害によるバッチ・ジョブの中断から回復する時に使うシンボルです。

15.13 シンボルの便利な使い方

シンボルには， DCL コマンドを置き換える以外に，次のような使い方があります。

15.13.1 レキシカル関数の結果を受け取る

レキシカル関数とは，コマンド・プロシージャ中で任意のファイルやシステム等の情報を得るための関数です。レキシカル関数の実行結果は，数値または文字列として返されるので，それをシンボルに代入して保存することができます。（付録 E に，よく使用されるレキシカル関数の一覧をまとめてあります。)

たとえば，次の例ではレキシカル関数を使って現在のプロセスのユーザ名を表示しています。

```
$ u = f$getjpi("","username")
$ show symbol u
```

15.13.2 文字列中にシンボルを埋め込む

引用符 (') を 2 重に使って，あらかじめ決められた文字列にシンボルの内容を埋め込むことができます。

たとえば，次の例では文字列中に自分のユーザ名を埋め込んでいます。

```
$ u = f$edit(f$getjpi("","username"),"trim")
$ write sys$output "現在のユーザ名は ''u' です。"
```

引用符 (') を使わずに，同様のことを行うには，次のようにします。

```
$ write sys$output "現在のユーザ名は " + u + " です。"
```

15.13.3 DCL コマンドのパラメータとして使う

DCL コマンドに対するパラメータ，またはパラメータの一部としてシンボルを使うには，シンボルを引用符 (') で囲みます。

次の例は，今年作成されたファイルの一覧を表示します。

```
$ thisyear = f$cvtime("today",,"year")
$ dir /since=1-jan-'thisyear'
```

15.13.4 自作ユーティリティをコマンド定義する

任意のディレクトリにある任意の実行イメージを， DCL コマンドであるかのように定義できます。この場合，定義される実行イメージは DCL の RUN コマンドで実行できることが必要です。

次のようにして定義します。

```
$ mytool == "$ sys$login:mytool.exe"
```

この場合，グローバル・シンボルを使うのが一般的です。一度シンボルを定義すると，シンボルを削除するか，ログアウトするまで有効です。定義された実行イメージを起動するには， DCL プロンプトで次のように入力します。

```
$ mytool
```

16

論理名を使用する

この章では

論理名は，ファイルやディレクトリ，あるいはデバイスの別名を定義するために使用したり，ユーザの環境設定のために使用します。

定義された論理名は，日本語 OpenVMS システムのさまざまなアプリケーションから参照することができます。

関連資料

- 『OpenVMS ユーザーズ・マニュアル』
- 『OpenVMS システム管理者マニュアル』

16.1 論理名とは

ファイル指定で，各要素の一部または全部を**論理名**と呼ぶ別の名前で置き換えることができます。論理名とは，ファイル指定，装置指定，ディレクトリ指定等にあたえた"別名 "です。

論理名を使うと入力するコマンド列を短くできたり，覚えやすい名前を付けることができます。

16.2 論理名を定義する — DEFINE

論理名を定義するには次のようにします。(どちらを使用してもかまいません。)

```
$ DEFINE [/修飾子] 論理名   等価名[,...]
                          1       2
$ ASSIGN [/修飾子] 等価名[,...]  論理名
```

1 論理名

1 ～ 255 文字までの英数字，ドル記号 ($)，アンダーライン (_) を使用して，わかりやすい名称を付けます。

2 等価名

論理名で置き換える装置名，ディレクトリ名，ファイル名を指定します。

16.3 論理名定義の例

デバイス名 DKA0 のディレクトリ名[YAMADA.REPORT.KOJIN]に論理名 MINE を割り当てます。

```
$ DEFINE MINE DKA0:[YAMADA.REPORT.KOJIN]
```

この後，次のようにファイル指定ができます。

```
$ DIRECTORY MINE:
```

以下のように表示されます。

```
Directory DKA0:[YAMADA.REPORT.KOJIN]

KEIHI.TXT;1  TRIP.TXT;1

Total of 2 files.
```

一般に論理名はログアウトすると自動的に削除（DEASSIGN）されてしまいます。したがって，再びログインしたあと，同じ論理名を使用する場合は，その都度，新しく割り当てる必要があります。そこで，ログイン時に自動的に環境設定を行うログイ

ン・コマンド・プロシージャというファイルを作り，ログインの都度，新しく論理名を割り当てる煩雑さを避けることができます。(第 17 章をご覧ください。)

16.4 論理名を参照する — SHOW LOGICAL

使用している論理名に割り当てられている等価名を参照するには，次のようにします。

```
$ SHOW LOGICAL [/修飾子] [論理名]
```

論理名を省略すると，定義してある論理名がすべて表示されます。

16.5 論理名を削除する — DEASSIGN

論理名を削除するには，次のようにします。

```
$ DEASSIGN  論理名
```

16.6 論理名の属性と種類

論理名には様々な属性や，階層構造があります。一般のユーザが定義できる論理名は，自分以外のユーザに影響を与えない範囲に限定されています。以下にそれらを簡単に説明します。

◆属性

論理名には CONCEALED と TERMINAL という 2 つの属性があり，論理名を等価名に翻訳する時の動作を設定できます。

◆論理名の有効範囲

論理名は，そのプロセスの中でのみ有効なものや，システム全体にわたり有効なものなどがあり，それらは論理名テーブルにより管理されます。プロセス，ジョブ，グループ，システム，クラスタの 5 種類があります。

◆論理名の検索順序

論理名を等価名に変換する際に，論理名を検索する検索順序は通常，プロセス，ジョブ，グループ，システム，クラスタの順です。

◆アクセス・モード

論理名が一般のアプリケーションのために設定されたのか，特権を持った特別なアプリケーションのために設定されたのかを示します。ユーザ，スーパーバイザ，エグゼクティブ，カーネルの 4 種類があります。

16.7 知っておくと便利な論理名

日本語 OpenVMS では，さまざまな論理名が自動的に定義されています。いくつかの論理名はユーザが日本語 OpenVMS を使う上でたいへん役に立ちます。それらの論理名の主なもを以下に示します。

16.7.1 SYS$LOGIN

ユーザのログイン・ディレクトリを示す論理名。

16.7.2 SYS$INPUT

ユーザの入力デバイスを示す論理名。

通常はユーザの使用しているターミナルのキーボードを示します。コマンド・プロシージャ実行中は，そのコマンド・プロシージャ自身を示します。

(C 言語の stdin に相当します。)

16.7.3 SYS$OUTPUT

ユーザの出力デバイスを示す論理名。

通常はユーザの使用しているターミナルのディスプレイを示します。

(C 言語の stdout に相当します。)

16.7.4 SYS$ERROR

ユーザのエラーメッセージ出力デバイスを示す論理名。

通常は SYS$OUTPUT と同じデバイスを示します。

(C 言語の stderr に相当します。)

16.7.5 SYS$SCRATCH

一時的に使用するファイルを置くディレクトリを示す論理名。

エディタや SORT ユーティリティなどが作業中のファイルを一時的に保管するために使います。

16.7.6 SYS$DISK

デフォルト・ディレクトリのあるディスクを示す論理名。

DCL の SET DEFAULT コマンドで変更されます。

16.7.7 SYS$SYSDEVICE

システム・ディスクを示す論理名。

注意

上記の論理名はすべて SYS$で始まっています。これは，これらの論理名が日本語 OpenVMS によって定義された論理名であることを示しています。ユーザは特別な理由がない限り，SYS$で始まる論理名を削除したり変更することは禁止されています。

16.8 コラム：論理名とシンボルの違い

論理名もシンボルも，コマンドやファイルの別名を定義できるという意味では似ていますが，用途は大きく異なります。

★論理名★

デバイス，ディレクトリ，ファイル，キューなどを指定できます。 DCL コマンドのパラメータとして使用します。

論理名は，日本語 OpenVMS システム上のほぼあらゆるアプリケーションから参照することができます。

★シンボル★

コマンドまたは，コマンド文字列の一部を指定できます。コマンド文字列の代わりに使用します。

たとえば，次の例では，論理名 MYTOOL を使用して， MYSORT というシンボルに自作のユーティリティを定義しています。

```
$ DEFINE MYTOOL DISK$USER:[YAMADA.TOOL]
$ MYSORT == "$ MYTOOL:MYSORT.EXE"
```

17

コマンド・プロシージャを使用する

この章では

DCL コマンドを一定の順序で繰り返し実行することがしばしばあります。このように，決まった手順のコマンドを繰り返し実行する場合，コマンド・プロシージャを使うと便利です。**コマンド・プロシージャ**とは，DCL コマンドと DCL コマンドで使用されるデータ行を 1 つのファイルにしたものです。コマンド・プロシージャを実行することは，その中のコマンド列を実行するのと同じことです。

関連資料

- 『OpenVMS ユーザーズ・マニュアル』
- 『OpenVMS システム管理者マニュアル』
- 『OpenVMS システム管理ユーテリィティ・リファレンス・マニュアル』

17.1 コマンド・プロシージャを作成する

コマンド・プロシージャはエディタを使用して作成します。コマンド・プロシージャの作成規則は次のとおりです。

- ファイル名は，コマンド・プロシージャで実現しようとする機能にふさわしい名前を付けます。ただし，日本語ファイル名 (かな漢字) は使用しないでください。
- ファイル・タイプは.COM を使います。
- 各行の先頭は，$で開始します。 (空白だけの行にも先頭に$が必要)
- 注釈（コメント）は，！の後に記述します。

17.2 コマンド・プロシージャを実行する — @

作成したコマンド・プロシージャを実行するには次のように入力します。

```
$ @コマンド・プロシージャ・ファイル指定［パラメータ1］［パラメータ2］…
```

コマンド・プロシージャ・ファイルがデフォルト・ディレクトリにない場合は，ディレクトリ指定も必要です。

コマンド・プロシージャは，コマンド・プロシージャ・ファイルのコマンドを先頭から順番に実行します。また， 1つのコマンドの実行が完全に終わってから次のコマンドに進みます。

17.3 実行コマンドを表示する — SET VERIFY

デフォルトでは，コマンド・プロシージャ実行中に，コマンドそのものは画面に表示されません。実行コマンドを表示するには，次のようにします。

```
$ SET VERIFY
```

実行コマンドの表示をしない場合は，次のようにします。

```
$ SET NOVERIFY
```

これらのコマンドは，ターミナルからコマンドとして入力できますが，コマンド・プロシージャの中に含めることもできます。

17.4 コマンド・プロシージャの実行例

ABC.FOR というプログラムをコンパイル，リンク，実行するコマンド・プロシージャ (TEST.COM) の例を次に示します。

TEST.COM

```
-----------------------------------------
    $ FORTRAN ABC
    $ LINK ABC
    $ RUN ABC
-----------------------------------------
```

コマンド実行の状態を確認できるように， SET VERIFY を実行します。

```
$ SET VERIFY
```

TEST.COM 実行します。

```
$ @TEST
```

各コマンドを表示しながら自動的に実行します。

```
$ FORTRAN ABC
$ LINK ABC
$ RUN ABC
```

17.5 パラメータの使用

ファイル指定が異なるためにプログラムごとに別々のコマンド・プロシージャを作るのは不便です。そこで，コマンド・プロシージャにパラメータを使用して，実行時にファイル名を与えることができます。

パラメータの使用には次の規則があります。

- コマンド・プロシージャ内では，パラメータは 'P1'， 'P2'，……， 'P8' と 8 個まで使えます。 P1 ～ P8 はそのコマンド・プロシージャ内のみで有効なローカル・シンボルです。これらのシンボルをコマンド・プロシージャ内で使う場合は引用符 (') で囲むことが必要です。
- 実行時には，コマンド・プロシージャ・ファイル名のあとにスペースを 1 つ以上あけて第 1 パラメータから順に入力します。パラメータの区切りにはスペースを使います。
- パラメータは自動的に大文字扱いにされます。大文字扱いにしたくない場合は，パラメータを二重引用符 (") で囲みます。パラメータとして漢字やスペースを指定する場合も二重引用符 (") で囲みます。

17.6 パラメータの使用例

ソース・プログラム名をパラメータとして，コンパイル，リンク，実行をするコマンド・プロシージャ (TEST1.COM) の例を次に示します。

TEST1.COM

```
-------------------------------------------
    $ FORTRAN 'P1'
    $ LINK 'P1'
    $ RUN 'P1'
-------------------------------------------
```

コマンド実行の状態を確認できるように SET VERIFY を実行します。

```
$ SET VERIFY
```

パラメータとして ABC を指定して TEST1.COM を実行します。

```
$ @TEST ABC
```

各コマンドを表示しながら自動的に実行します。

```
$ FORTRAN ABC
$ LINK ABC
$ RUN ABC
```

17.7 ログイン・プロシージャを使用する

ログインすると自動的に実行されるプロシージャを，**ログイン・プロシージャ**といいます。

キーの定義やコマンド列の置き換えは，ログアウトすると無効になってしまいます。同じ定義や置き換えを使いたい場合は，ログイン時に再定義しなければなりません。キーの定義やコマンド列の置き換えをログイン・プロシージャに入れておけば，自動的に定義されるので，ログインするたびに定義し直す手間が省けます。

ログイン・コマンド・プロシージャには，通常は次のようなコマンドを盛り込みます。

コマンド	参照箇所
論理名の定義 (ASSIGN， DEFINE コマンド)	第 16 章
グローバル・シンボルの定義 (= =)	第 15 章
キーの定義 (DEFINE/KEY コマンド)	第 6 章
デフォルト・ディレクトリの設定や表示 (SET DEFAULT コマンド， SHOW DEFAULT コマンド)	第 8 章

コマンド	参照箇所
環境設定 (JSY$CONTROL ユーティリティ)	第 3 章

17.8 ログイン・プロシージャを作成する

ログイン・プロシージャのファイル名とファイル・タイプは，通常は， LOGIN.COM です。このファイルは，ログイン・ディレクトリに置かなければなりません。エディタを使用して作成します。

ログイン・プロシージャを作成する際は，次の規則に従ってください。

- 1 行に 1 コマンド列のみを入力する。
- 行の先頭は，$で開始する。
- 実際のコマンド入力と同じ形式で入力する。
- コメントは ! の後に記述する。

17.9 ログイン・プロシージャの例

ログイン・プロシージャの例 (LOGIN.COM) を示します。

LOGIN.COM の内容を表示します。

```
$ TYPE LOGIN.COM
```

LOGIN.COM

```
------------------------------------------
   $! LOGIN.COM (YAMADA)
   $ DEFINE/KEY PF1 "SHOW DEFAULT"
   $ DEFINE/KEY PF2 "SET DEFAULT"
   $ PR == "PRINT/QUEUE=PRINTER1"
   $ EXIT
------------------------------------------
```

この例では，キーを定義 (SHOW DEFAULT, SET DEFAULT) しています。また，シンボルも定義 (PR = = "PRINT/QUEUE=PRINTER1") しています。そして，処理の実行を終了するために EXIT としています。 (感嘆符 (!) に続く文字列は注釈 (コメント) です。)

17.10 プロンプト・メッセージを出力するコマンド・プロシージャ — INQUIRE

画面にプロンプト・メッセージを出力し，ユーザが入力した文字列をシンボルに割り当てるには次のようにします。(小文字は大文字に変換されます。)

```
$ INQUIRE[/GLOBAL] シンボル  "プロンプト・メッセージ"
```

次の例 HELP.COM では， HELP で画面に出される内容をファイルに出力します。その際，作ったファイルをプリンタへ出力するか尋ね，ユーザが Y と入力した場合はプリンタ出力します。(感嘆符 (!) に続く文字列は注釈 (コメント) です。)

HELP.COM

```
-----------------------------------------------------
    $
    $! HELP FILE を出力するコマンド・プロシージャ
    $
    $ HELP/NOPAGE/NOPROMPT/OUTPUT='P1'.HLP 'P1'
    $ INQUIRE ANS " PRINT OUT? [N] "
    $ IF ANS THEN PRINT 'P1'.HLP
    $ EXIT
-----------------------------------------------------
```

17.11 シンボルと文字列をコマンド・プロシージャに組み込む

DCL コマンド列以外の文字列をシンボルに定義してシンボルを使用することができます。たとえば，シンボルにディレクトリ名を定義し，コマンド列の一部としてそのシンボルを用いることができます。

ただし，シンボルを DCL コマンド行の先頭以外で使う場合には，次のように引用符 (') で囲むことが必要です。

```
$ DIR_NAME = "[YAMADA.WORK]"
$ SET DEF 'DIR_NAME'
```

次の例では，実行時にパラメータが指定されなかった時は， P1 を入力するように促します。(コマンド行の先頭以外でシンボルを使う場合には，引用符 (') で囲みます。)

DEFAULT.COM

```
-----------------------------------------------------------
    $ IF P1 .EQS. "" THEN INQUIRE P1 "INPUT P1"
    $ SET DEFAULT [YAMADA.'P1']
    $ SHOW DEFAULT
    $ EXIT
-----------------------------------------------------------
```

SET VERIFY を実行します。

```
$ SET VERIFY
```

DEFAULT.COM を実行し， P1 として WORK を指定します。

```
$ @DEFAULT WORK
$ IF P1 .EQS. "" THEN INQUIRE P1 "INPUT P1"
```

P1 が WORK に置き変わって実行されます。

```
$ SET DEFAULT [YAMADA.WORK]
$ SHOW DEFAULT
  DKA0:[YAMADA.WORK]
$ EXIT
```

17.12 シンボルとコマンド・プロシージャを組み合わせる

シンボルとコマンド・プロシージャを組み合わせることもできます。

次の例では，表示を整えるために空白行をターミナルに出力し，It was fun working with you. Have a nice day ! というメッセージをターミナルに出力します。次に表示を整えるために空白行をターミナルに出力し，ログアウトします。

LOGOUT.COM

```
-------------------------------------------------------------------------------
    $ WRITE SYS$OUTPUT ""
    $ WRITE SYS$OUTPUT "It was fun working with you. Have a nice day !"
    $ WRITE SYS$OUTPUT ""
    $ LOGOUT
-------------------------------------------------------------------------------
```

BYE というシンボル名に，コマンド・プロシージャを起動するコマンド文字列を定義します。

```
$ BYE == "@SYS$LOGIN:LOGOUT.COM"
```

シンボル BYE を実行します。コマンド・プロシージャ LOGOUT.COM が起動し，ターミナルに空白行が出力されます。

```
$ BYE
```

ターミナルにメッセージが出力され，次に空白行が出力されます。

```
It was fun working with you.  Have a nice day  !
```

コマンド・プロシージャ LOGOUT.COM の最終行が実行され，ログアウトします。

```
YAMADA   logged out at 14-FEB-2002 18:30:45.10
```

17.13 ログイン・コマンド・プロシージャを起動させずにログインする

ログイン・コマンド・プロシージャを起動しないでログインするには，ユーザ名を入力するときに/NOCOMMAND という修飾子をつけます。

```
        Welcome to OpenVMS/Alpha V7.3-1
Username: YAMADA/NOCOMMAND
Password:
      Welcome to OpenVMS/Alpha version V7.3-1 on node ABC
   Last interactive login on Tuseday, 12-FEB-2002 10:26
$
```

また，次のようにして，別のコマンド・プロシージャ・ファイルをログイン・コマンド・プロシージャとして指定することもできます。

```
/COMMAND = ファイル指定
```

17.14 条件式を使用する

IF コマンドを使うことで，条件によって処理を変えることができます。次のコマンド・プロシージャは，入力した数値が負数の場合に警告を表示します。

```
$ INQUIRE VALUE
$ IF VALUE .LT. 0 THEN WRITE SYS$OUTPUT "負数です。"
```

IF コマンドの形式は次の通りです。

```
$ IF 条件式 THEN コマンド
```

条件式が成り立つ場合 (真の場合) に THEN の後のコマンドが実行されます。 THEN の後に書くことのできるコマンドは 1 行につき 1 つだけです。

複数のコマンドを書く場合には次のように書きます。条件式が成り立つ場合は，THEN と ENDIF で挟まれたコマンドが実行されます。

```
$ IF 条件式
$ THEN
$   コマンド1
$   コマンド2
$       :
$   コマンドn
$ ENDIF
```

条件式が成り立たない時にコマンドを実行させたい場合には， ELSE を使います。条件式が成り立たなかったとき， ELSE と ENDIF ではさまれたコマンドが実行されます。

```
$ IF 条件式
$ ELSE
$   コマンド1
$   コマンド2
$       :
$   コマンドn
$ ENDIF
```

THEN と ELSE の両方を使うこともできます。条件式が成り立つ場合には， THEN と ELSE に挟まれたコマンドが実行されます。条件式が成り立たなかった場合には ELSE と ENDIF に挟まれたコマンドが実行されます。

```
$ IF 条件式
$ THEN
$   コマンド1a
$   コマンド2a
$       :
$   コマンドna
$ ELSE
$   コマンド1b
$   コマンド2b
$       :
$   コマンドnb
$ ENDIF
```

コマンドが 1 つしかない場合は，次のように短縮して書くこともできます。

```
$ IF 条件式
$ THEN コマンドa
$ ELSE コマンドb
$ ENDIF
```

17.15 条件式の書き方

条件式の代表的な書き方は， 2 つの数値や文字列の比較によるものです。 2 つの数値や文字列 (またはそれらを含んだシンボル) を比較演算子で結びます。

たとえば，I .LT. 0 は，「シンボル I が 0 より小さいとき」という意味です。.LT. はその左辺と右辺を数値として比較します。左辺が右辺より小さければ真，そうでなければ偽の値を持ちます。 DCL の数値比較演算子には次のようなものがあります。

表 17–1 数値比較演算子

演算子	意味
.EQ.	等しい
.GE.	左辺が右辺より大きいまたは等しい
.GT.	左辺が右辺より大きい
.LE.	左辺が右辺より小さいまたは等しい
.LT.	左辺が右辺より小さい
.NE.	等しくない

文字列の比較は文字列用の比較演算子を使用します。文字列の大小の比較は ASCII コードで比較されます。

表 17–2 文字列比較演算子

演算子	意味
.EQS.	等しい
.GES.	左辺が右辺より大きいまたは等しい
.GTS.	左辺が右辺より大きい
.LES.	左辺が右辺より小さいまたは等しい
.LTS.	左辺が右辺より小さい
.NES.	等しくない

条件式を.AND. または.OR. でつなぐと複数の条件式を組み合わせることもできます。次のようにします。

```
$ IF 条件式1 .AND. 条件式2 THEN コマンド
```

IF-THEN-ELSE-ENDIF の形式でも同じように書くことができます。たとえば，次のコマンド・プロシージャでは， 1 ～ 9 以外の数字を入力した時に警告を出します。

```
$ INQUIRE VALUE
$ IF VALUE .GE. 1 .OR. VALUE .LE. 9
$ THEN
$   WRITE SYS$OUTPUT "正しい数値です"
$ ELSE
$   WRITE SYS$OUTPUT "範囲外の数値です"
$ ENDIF
```

条件式が複雑でわかりにくい時は，次のように括弧 ( ) で括って整理しましょう。

```
$ IF (VALUE .GE. 1) .OR. (VALUE .LE. 9)
```

17.16 ラベルが付いた行へジャンプする

GOTO コマンドを使って特定のラベルが付いた所へジャンプさせることができます。たとえば次のコマンド・プロシージャは，ラベル LEAP_YEAR にジャンプさせるものです。

```
$ GOTO LEAP_YEAR
    :
    :
$ LEAP_YEAR:
$ WRITE SYS$OUTPUT "ここがLEAP_YEARです。"
```

ラベルは，行頭の$の後に書き，最後にコロンを付けます。ラベルには空白文字は使えません。

次のように，ラベルのすぐあとにコマンドを書くこともできます。

```
$ LEAP_YEAR: WRITE SYS$OUTPUT "ここがLEAP_YEARです。"
```

17.17 ループを作る

IF コマンドと GOTO コマンドを組み合わせることでループを作ることができます。たとえば，次のコマンド・プロシージャは 10 回回るループです。

```
$ I = 1
$ LOOP: IF I .GE. 11 THEN GOTO EXIT_LOOP
$   WRITE SYS$OUTPUT "ループ''I'回目"
$   I = I + 1
$ GOTO LOOP
$ EXIT_LOOP:
```

17.18 エラー処理を組み入れる

コマンドの実行中にエラーが起こるとコマンド・プロシージャはその実行を中断します。エラー処理を組み入れることで実行を継続したり後処理をするなどの制御ができるようになります。(エラー・メッセージの種類，形式については，第 5 章をご覧ください。)

17.18.1 エラー処理を規定する

エラーが起こった場合の処理を ON コマンドであらかじめ規定しておくことができます。

```
$ ON WARNING THEN コマンド
$ ON ERROR THEN コマンド
$ ON SEVERE_ERROR THEN コマンド
```

ON WARNING は警告 (-W-) が起きた場合の処理を規定します。 ON ERROR はエラー (-E-) が起きた場合の処理を規定します。 ON SEVERE_ERROR は致命的なエラー (-F-) が起きた場合の処理を規定します。

17.18.2 エラー処理を組み込れたコマンド・プロシージャの例

たとえば，エラーが起きた時にメッセージを表示して終了するには，次のようなコマンド・プロシージャを書きます。

```
$ ON ERROR THEN GOTO ERRPROC
      :
$ ERRPROC:
$ WRITE SYS$OUTPUT "エラーが発生しました。中断します。"
$ EXIT
```

上記の例では，エラーが起きた時にラベル ERRPROC へジャンプします。

17.19 Ctrl + Y による強制終了時の処理

コマンド・プロシージャの実行中にキーボードからCtrl + Y (コントロール Y) を入力すると実行を中断してコマンド待ちの状態に戻りますが，Ctrl + Y が入力されたときの処理を ON コマンドであらかじめ規定しておくことができます。

```
$ ON CONTROL_Y THEN コマンド
```

次の例では，コマンド・プロシージャの実行中にCtrl + Y が入力された場合，ファイルが開いていれば，それを閉じてから終了します。

```
$ ON CONTROL_Y ERROR THEN GOTO INTERRUPT
$ OPEN/READ DATAFILE FOO.DAT
      :
$ INTERRUPT:
$ IF F$TRNLNM("DATAFILE",,,,,"MAX_INDEX") .NES. "" THEN CLOSE DATAFILE
$ WRITE SYS$OUTPUT "強制中断されました。"
$ EXIT
```

また，この ON コマンドによるCtrl + Y の制御は，Ctrl + C が入力された場合でも同じように働きます。

付録

項目	参照箇所
システムの起動と停止	付録 A
日本語環境を設定する	付録 B
ネットワークを利用したファイル操作	付録 C
JMAIL ユーティリティを使用する	付録 D
レキシカル関数一覧	付録 E
Linux または UNIX，MS–DOS，日本語 OpenVMS コマンド対応表	付録 F
関連アプリケーション	付録 G
PC 関連製品	付録 H
日本語 OpenVMS クイック・リファレンス	付録 I
日本語 OpenVMS マニュアル・クイック・リファレンス	付録 J

A

システムの起動と停止

ここでは，日本語 OpenVMS システムの起動と停止について簡単にまとめます。

A.1 システムを起動する

システムを起動するには，次の方法があります。(ここで ddcu はデバイス名を指定します。)

通常のブート・コマンドで起動

- Alpha システムの場合　>>> BOOT [-FLAG 0,0 ddcu]
- VAX システムの場合　　>>> BOOT [ddcu]

A.2 シャットダウン・コマンド・プロシージャでシステムを停止する

シャットダウン・コマンド・プロシージャでシステムを停止するには，次の手順を行います。

1. シャットダウン・コマンド・プロシージャを実行します。

```
$ @SYS$SYSTEM:SHUTDOWN.COM
```

次のようなシャットダウン時の質問事項に答えると，システムがシャット・ダウンされます。

```
How many minutes until final shutdown [0]:
Reason for shutdown [Standalon]:
Do you want to spin down the disk volumes [NO]?
Do you want to invoke the site-specific shutdown procedure [YES]?
Should an automatic system reboot be performed [NO]?
When will the system be rebooted [later]:
Shutdown options (enter as a comma-separated list):
    REBOOT_CHECK       Check existence of basic system files
    SAVE_FEEDBACK      Save AUTOGEN feedback information from this boot
    DISABLE_AUTOSTART  Disable autostart queues
Shutdown options [NONE]:
```

2. メッセージが出たら，シャットダウンは完了です。この後，必要であれば電源を切ります。

```
SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

B

日本語環境を設定する

日本語 OpenVMS で日本語機能を使用するには，日本語機能を有効にする設定を行う必要があります。(通常はシステム管理者が設定します。) この設定によって，日本語のヘルプやファイル名，日本語ユーティリティ等が有効になります。ここでは，日本語 OpenVMS 上で日本語の環境を設定する方法について説明します。

B.1 日本語環境が有効かどうかを調べる

日本語環境が有効になっているかどうかを調べるには，日本語環境設定ユーティリティ (JSY$CONTROL) を使用します。以下の 2 行のコマンドを入力してください。

```
$ JSYCP == "$ SYS$SYSTEM:JSY$CONTROL.EXE"
$ JSYCP SHOW ALL
```

次のようなメッセージが表示された場合は，日本語機能が有効になっています。

```
現在のロケールは JA_JP_SDECKANJI です。
論理名 SYS$HELP の値は SYS$SYSROOT:[SYSHLP.JA_JP] です。
論理名 HLP$LIBRARY の値は SYS$SYSROOT:[SYSHLP]HELPLIB.HLB です。
論理名 SYS$MESSAGE の値は SYS$SYSROOT:[SYSMSG.JA_JP] です。
論理名 UTIL$SMGSHR の値は JSY$SMGSHR です。
論理名 SMG$DEFAULT_CHARACTER_SET の値は SDK です。
ターミナルの入力コードは KANJI に設定されています。
ターミナルのマルチバイト行編集は ENABLE されています。

ファイル名コードセットは sdeckanji に設定されています。
ファイル名解析スタイルは EXTENDED に設定されています。
$
```

各メッセージの詳しい意味については『日本語ユーティリティ利用者の手引き』を参照してください。メッセージ全体が英語で表示される場合には，ユーザの日本語環境は無効になっています。

B.2 日本語環境を有効にする

日本語環境を有効にするには，次のコマンドを入力します。

```
$ JSYCP SET LOCALE ja_JP.sdeckanji
```

このコマンドを実行すると，以下の日本語機能が有効になります。

- 日本語ヘルプの使用
- 日本語ファイル名の使用
- PC の日本語入力機能 (IME など) からの漢字入力を受け付け
- 日本語ユーティリティなどで，自動的に日本語を使用
- 標準の漢字コードが Super DEC 漢字に設定
- プログラム開発に必要なロケールが ja_JP.sdeckanji に設定

ロケールとは

ロケールとは，日本，中国，フランスなど特定の地理的あるいは言語的領域で，文化的な慣習に正確に対応するソフトウェア環境のことのことです。

B.3 DCL コマンド・ラインでかな漢字変換を行う

VT ターミナルのような，日本語入力機能を持たないデバイスから日本語 OpenVMS システムにログインしている場合は，日本語 OpenVMS の持つかな漢字変換を使って日本語を入力することができます。

(PC (パーソナル・コンピュータ) から日本語 OpenVMS システムにログインしている場合は， PC の日本語入力機能 (IME など) を使って日本語を入力することができます。)

DCL コマンド・ラインでかな漢字変換を行うには，次のようにします。

1. DCL プロンプトに対して以下のコマンドを入力します。

```
$ INPUT START /SUBPROCESS /CONVERSION_KEY="@"
```

   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，かな漢字変換ユーティリティが起動されたことを示します。

```
Creating subprocess...
Process YAMADA_1 spawned
Enter LOGOUT command to exit. Type CTRL @ to enter Kanji.
```

2. Ctrl + @ キーまたは Ctrl + Space キーを押します。

   画面の下方に以下のプロンプトが表示されるので，このプロンプト上で，かな漢字変換を行うことができます。

```
変換入力 :
```

   標準ではコントロール・キーを利用して漢字変換やカタカナ変換を行います。

3. 入力したい文字列が確定したら Return キーを押します。

変換結果が DCL コマンド・ライン上の元のプロンプトに入力されます。

4. かな漢字変換ユーティリティを停止します。

   かな漢字変換を行う必要がなくなったら，以下のコマンドを入力すると，かな漢字変換ユーティリティを停止できます。

```
$ LOGOUT
```

注意 1

日本語 EVE エディタなどの日本語を使うことを目的としたアプリケーションは，アプリケーション自身が日本語入力機能を持つため，かな漢字変換ユーティリティを終了してから起動してください。

注意 2

日本語用に設計されていないアプリケーションに日本語を入力した場合，意図しない動作をしたり，エラーが発生する場合があります。そのようなアプリケーションに対しては日本語を使わないでください。

B.4 カナ入力の設定を行う

VT ターミナル・デバイスや CDE の漢字端末エミュレータで，日本語 OpenVMS の持つかな漢字変換を使ってカナ入力モードで文字を入力するには，第 B.2 節に説明されている設定を行った上で， KANJIGEN ユーティリティの次のコマンドを使用して，かな入力を有効にします。

```
$ KANJIGEN == "$ JSY$SYSTEM:KANJIGEN.EXE"
$ KANJIGEN SET /INPUT=KANA /EDIT=DISABLE
```

(コマンドの詳しい説明については『フォント管理ユーティリティ利用者の手引き』をご覧ください。)

これで，かな入力が有効になります。同時に PC からの漢字入力は無効になり， IME やカット&ペーストによる漢字入力はできなくなります。

かな入力を無効に戻すには，次のようにします。

```
$ KANJIGEN SET /INPUT=KANJI /EDIT=ENABLE
```

なおローマ字入力は，この設定にかかわらず常に利用可能です。

C

ネットワークを利用したファイル操作

日本語 OpenVMS は TCP/IP と DECnet の 2 種類のネットワークをサポートしています。ここでは，これらのネットワークを使用して他の日本語 OpenVMS システムにアクセスする方法を説明します。ネットワークを使用してアクセスされる他の日本語 OpenVMS システムのことを**リモート・システム**と呼びます。

C.1 TCP/IP を利用する

日本語 OpenVMS システムで TCP/IP を利用するには，あらかじめ TCP/IP ソフトウェアがシステムにインストールされ，動作している必要があります。 TCP/IP が動作しているかどうかは，次のようにして知ることができます。

```
$ SHOW NETWORK

Product:  TCP/IP        Node:  tko.honsha.com       Address(es):  192.168.0.123
$
```

Product: の項目に TCP/IP と表示されれば， TCP/IP が動作しています。

C.2 TCP/IP を利用してリモート・システムへログインする

現在ログインしている日本語 OpenVMS システムから，リモート・システムにログインするには telnet コマンドや rlogin コマンドを使います。

```
$ TELNET OSAKA.SHITEN.COM

%TELNET-I-TRYING, Trying ... 192.168.0.124
%TELNET-I-SESSION, Session 01, host osaka.siten.com, port 23
-TELNET-I-ESCAPE, Escape character is ^]

 Welcome to OpenVMS (TM) Alpha Operating System, Version V7.3-1

Username:
```

リモート・システムでの作業が完了し，そのシステムをログアウトすると，元の日本語 OpenVMS システムに制御が戻ります。

C.3 TCP/IP を利用してリモート・システム間でファイルを転送する

DIRECTORY や COPY などの DCL コマンドはネットワークに対応しているので，簡単な操作でリモート・システムへアクセスできます。ユーザ名とパスワードは二重引用符 (") で囲み，コマンド・ライン上で指定します。

```
$ コマンド /FTP リモート・システム名"ユーザ名 パスワード"::
```

C.3.1 TCP/IP を利用してファイルの一覧を取得する

ファイルの一覧を取得するには，次のようにします。

```
$ DIR /FTP OSAKA"YAMADA パスワード"::

Directory osaka.shiten.com"yamada password"::

LOGIN.COM;1                1   9-MAY-2001 15:59:46.53
MAIL.MAI;1                27  22-JAN-2002 10:41:56.21

Total of 2 files, 28 blocks.
$
```

C.3.2 TCP/IP を利用してファイルをコピーする

ファイルをコピーするには，次のようにします。

```
$ COPY /FTP OSAKA"YAMADA パスワード"::LOGIN.COM []
$
```

コマンド・ライン上でパスワードを指定すると，第 3 者にパスワードを知られてしまう可能性があります。そのような危険を避けるには， FTP ユーティリティを使います。 FTP ユーティリティはパスワードを画面に表示しないので，セキュリティが重要な場合に使うと便利です。

C.3.3 FTP ユーティリティを使ってファイルをコピーする

FTP ユーティリティを使ってファイルをコピーするには，次のようにします。

```
$ FTP
FTP> connect osaka
220 osaka.shiten.com FTP Server (Version 5.1) Ready.
Connected to osaka.
Name (oska:yamada):
331 Username system requires a Password
Password:
230 User logged in.
FTP> ls login.com
200 PORT command successful.
150 Opening data connection for SYS$SYSROOT:[SYSMGR]login.com;*
                                          (192.168.0.124,49896)
SYS$COMMON:[SYSMGR]LOGIN.COM;1

226 NLST Directory transfer complete
96 bytes received in 00:00:00.00 seconds (46.87 Kbytes/s)
FTP> get login.com
200 TYPE set to IMAGE.
200 PORT command successful.
150 Opening data connection for SYS$COMMON:[SYSMGR]login.com;
                                        (192.168.0.124,49897)
226 Transfer complete.
local: SYS$COMMON:[SYSMGR]LOGIN.COM;2  remote: login.com
992 bytes received in 00:00:00.00 seconds (322.91 Kbytes/s)
FTP> exit
221 Goodbye.
$
```

C.4 TCP/IP を利用したファイル・アクセスの注意点

TCP/IP（おもに FTP）を使用してファイルをコピーする場合，そのファイルがテキスト・ファイルであるかバイナリ・ファイルであるか注意する必要があります。特に日本語 OpenVMS 以外のシステムとの間でファイルのコピーを行う場合には非常に重要になります。間違ったファイル形式でコピーを行うと，ファイルの内容は正しくコピーされません。

日本語のテキスト・ファイルをコピーする場合は，漢字コードにも注意する必要があります。日本語 OpenVMS では EUC に類似した Super DEC 漢字コードセットをサポートしています。 Windows 等の Shift JIS を使うシステムとの間で日本語のファイルを転送する場合には，たとえば次のように漢字コードの変換を行う必要があります。

```
$ ICONV CONVERT LIST.TXT /FROMCODE=SJIS LIST-1.TXT /TOCODE=SDECKANJI
```

日本語 OpenVMS ではファイル名にも日本語を使うことができますが，リモート・システムに置かれた日本語のファイル名を持つファイルにはアクセスできません。このような場合は英語のファイル名を使用してください。

C.5 FTP とレコード・フォーマット

Windows と Linux との間で FTP を用いてファイルを転送する場合には，転送したいファイルがテキストなのかバイナリなのかに注意する必要があります。日本語 OpenVMS の場合も同様に，日本語 OpenVMS と Windows もしくは Linux との間で FTP を用いてファイルを転送する場合には，転送したいファイルがテキストなのかバイナリなのかに注意する必要があります。

日本語 OpenVMS のディスク上にある各ファイルは，それぞれレコード・フォーマットを持っています。 OpenVMS では多数のレコード・フォーマットがサポートされていますが，一般的には以下のフォーマットが使用されます。

表 C–1 PC と互換性のあるレコード・フォーマット

ファイル形式	レコード・フォーマット	レコード属性
テキスト・ファイル	Variable length	CRLF
	Stream_LF	CRLF
バイナリ・ファイル	Fixed length	None

ファイルのレコード・フォーマットは DIR /FULL コマンドで知ることができます。ファイル転送の際にテキストとバイナリを間違えずに指定していれば，日本語 OpenVMS と Windows や Linux システムとの間で，ファイルは正しく転送されます。

テキストやバイナリの指定を間違えて転送してしまうと，実際のファイルの中身とファイル・フォーマットとが違い，ファイルの内容が壊れてしまいます。そのような場合には，再度正しく転送し直すことをおすすめしますが，再送しないで修正するには，以下の方法を試してみてください。

C.5.1 テキスト・ファイルをバイナリ・ファイルとして転送した場合

Windows または Linux から日本語 OpenVMS へファイルを転送して，ファイルの内容が壊れてしまった場合には，日本語 OpenVMS のレコード・フォーマットについての高度な知識があれば，ファイルを修復できる場合があります。特にテキスト・ファイルをバイナリ・ファイルとして日本語 OpenVMS システム上に転送した場合は，理論上はデータの欠落がないので修復可能です。以下のコマンドを実行してください。

```
$ set file text.bin /attribute=(rfm:stm,rat:cr)
```

このコマンドを実行すると，ファイルのレコード・フォーマットは Stream となり，エディタで編集可能となります。以下のコマンドでファイルの文字コードセットを適切に変換すれば，正しいテキスト・ファイルを得ることができます。

```
$ iconv convert text.bin /fromcode=sjis text.txt /tocode=sdeckanji
```

C.5.2 バイナリ・ファイルをテキスト・ファイルとして転送した場合

Windows 上のバイナリ・ファイルをテキスト・ファイルとして日本語 OpenVMS へ転送してしまった場合， Windows 側のファイル転送ユーティリティによって漢字コード変換が行なわれるため，殆どの場合はファイル内容が壊されてしまいます。

一部の UNIX のように，漢字コードに EUC を使っている場合は，日本語 OpenVMS へのファイル転送の際に漢字コード変換が必要ないため，ファイルを修復できる場合があります。実際に修復できるかどうかは，日本語 OpenVMS 上に作成されたファイルの状態に依存します。

テキスト・ファイルとして日本語 OpenVMS へ転送された結果できあがったファイルが， Stream_LF フォーマットになっている場合は，以下のコマンドで修復できる場合があります。

```
$ set file binary.txt /attribute=(rfm:fix,rat:none)
```

もし Variable length フォーマットになっている場合には，データの一部が失われている可能性が大きく，修復は困難です。

C.6 DECnet を利用する

DECnet は旧 DEC 社の独自のネットワークです。日本語 OpenVMS のほとんどの DCL コマンドは DECnet をサポートしています。

日本語 OpenVMS システムで DECnet を利用するためには，あらかじめ DECnet ソフトウェアがシステムにインストールされ，動作している必要があります。 DECnet が動作しているかどうかは，次のようにして知ることができます。

```
$ show network

Product:  DECNET        Node:  TKO                 Address(es):  12.345
$
```

Product: の項目に DECnet と表示されれば， DECnet が動作しています。

C.7 DECnet を利用してリモート・システムへログインする

現在ログインしている日本語 OpenVMS システムから，リモート・システムにログインするためには SET HOST コマンドを使います。

```
$ SET HOST OSAKA

 Welcome to OpenVMS (TM) Alpha Operating System, Version V7.3-1

Username:
```

リモート・システムでの作業が完了し，そのシステムをログアウトすると，元の日本語 OpenVMS システムに制御が戻ります。

C.8 DECnet を利用してリモート・システム間でファイルを転送する

DIR や COPY など，ほとんどの DCL コマンドは DECnet に対応しているので，簡単な操作でリモート・システムへアクセスできます。ユーザ名とパスワードは二重引用符 (" ") で囲み，コマンドライン上で指定します。

```
$ コマンド リモート・システム名"ユーザ名 パスワード"::
```

C.8.1 DECnet を利用してファイルの一覧を取得する

ファイルの一覧を取得するには，次のようにします。

```
$ DIR OSAKA"YAMADA パスワード"::

Directory OSAKA"yamada password"::

LOGIN.COM;1               1   9-MAY-2001 15:59:46.53
MAIL.MAI;1               27  22-JAN-2002 10:41:56.21

Total of 2 files, 28 blocks.
$
```

C.8.2 DECnet を利用してファイルをコピーする

ファイルをコピーするには，次のようにします。

```
$ COPY OSAKA"YAMADA PASSWORD"::LOGIN.COM []
$
```

C.9 DECnet を利用してパスワード入力なしにアクセスする

セキュリティが重要なリモート・システムに対し，コマンド・ラインでパスワードを入力せずに DECnet を使用してファイル・アクセスを行う方法があります。

コマンド・ラインでパスワードを入力しないで， DECnet を使用してリモート・システムにアクセスするには，使用しているこちら側のシステムの名前とアクセスしたいユーザ名とを，あらかじめリモート・システム側に登録しておく必要があります。これをプロキシと呼びます。

プロキシの登録はシステム管理者が行います。プロキシの登録が完了すると，以下のようにユーザ名もパスワードも入力せずに，リモート・システムへのアクセスができるようになります。

```
$ DIR OSAKA::

Directory OSAKA::USER$DISK:[YAMADA]

LOGIN.COM;1               1   9-MAY-2001 15:59:46.53
MAIL.MAI;1               27  22-JAN-2002 10:41:56.21

Total of 2 files, 28 blocks.
$
```

C.10 日本語 OpenVMS がサポートするネットワーク・プロトコルとネットワーク製品

日本語 OpenVMS がサポートするネットワーク・プロトコルとネットワーク製品には次のものがあります。

TCP/IP

- TCP/IP Services for OpenVMS (旧製品名： UCX)
- ANET+
- PathWay (旧製品名： WIN/TCP)
- MultiNet

DECnet

- DECnet-Plus (旧製品名： DECnet/OSI Phase V)
- DECnet Phase IV

D

JMAIL ユーティリティを使用する

JMAIL ユーティリティは，日本語 OpenVMS が提供しているユーザ間で電子メールをやりとりするためのユーティリティです。ここでは， JMAIL の基本操作について説明します。 (JMAIL について詳しくは『日本語ユーティリティ 利用者の手引き』または『OpenVMS ユーザーズ・マニュアル』の「Mail ユーティリティ」をご覧ください。)

JMAIL ユーティリティを使用する前に JMAIL ユーティリティで使用する用語を次に説明します。

◆メール・メッセージ

JMAIL ユーティリティを使用して情報を扱う際の情報の最小の単位を，**メール・メッセージ**といいます。

◆発信人

受け取ったメール・メッセージには，そのメール・メッセージの**発信人**の名前が付けられています。発信人の名前の形式は，次のとおりです。

```
ノード名::ユーザ名       "個人名"
```

ユーザが個人名を設定していると，ユーザ名の後に個人名が付きます。

◆ヘッダ

ヘッダには次の項目があります。

- 宛先

  メール・メッセージを送るときは，相手の宛先を指定します。宛先には，ノード名とユーザ名を指定します。

  ただし，相手が同じノードを使用している場合は，ユーザ名だけを指定します。

- CC

  CC は，カーボン・コピー (Carbon Copy) の略です。「参考までに」送信する相手を指定します。

- 表題

  メール・メッセージを送る場合は，表題を付けて送ることができます。表題を付けて送ると，メール・メッセージの識別が容易になります。

◆フォルダ

メール・メッセージを整理して保存しておく入れ物を，**フォルダ**といいます。以下のフォルダは，あらかじめ用意されています。

- NEWMAIL フォルダ

  まだ読んでいないメール・メッセージが入っているフォルダ

- MAIL フォルダ

  すでに読んだメール・メッセージが入っているフォルダ

- WASTEBASKET フォルダ

  削除したメール・メッセージが入っているフォルダ

これらのフォルダの他に，ユーザが任意にフォルダを作成することもできます。フォルダの中のメール・メッセージがすべてなくなると，フォルダも自動的になくなります。

◆メール・ファイル

メール・メッセージとフォルダを管理するファイルを，**メール・ファイル**といいます。

◆メール・ディレクトリ

メール・メッセージは，通常， MAIL$xxxxxxxxxx.MAI というようなファイル名 (MAIL$で始まるファイル名) とファイル・タイプ (MAI) で取り扱われます。このファイルが置かれているディレクトリを，**メール・ディレクトリ**といいます。メール・ディレクトリを特に指定しない場合，ログイン・ディレクトリがメール・ディレクトリになります。

D.1 JMAIL ユーティリティを起動する — JMAIL

JMAIL ユーティリティを起動するには， JMAIL コマンドを使用します。次のように入力します。

```
$ JMAIL
```

JMAIL ユーティリティを起動すると，次のように表示されます。

```
$ JMAIL
JMAIL>
 1    2
```

1 JMAIL ユーティリティ専用のプロンプト。

2 **1** のプロンプトに対して，サブコマンドを入力できる状態。

D.2 使用環境を調べる — SHOW ALL

JMAIL ユーティリティを使用する際の条件を，**使用環境**といいます。使用環境を調べるには次のようにします。

```
JMAIL>SHOW ALL
```

使用環境のうちメール・ディレクトリ， CC プロンプト，エディタは， JMAIL ユーティリティを初めて使用するときに設定したほうがよいでしょう。

D.3 メール・ディレクトリを設定する — SET MAIL_DIRECTORY

メール・メッセージのファイルやメール・ファイルは，メール・ディレクトリに置かれます。

メール・ディレクトリを設定しておかないと，ログイン・ディレクトリがメール・ディレクトリになってしまうため，他のファイルやディレクトリと混在してしまいます。これを防ぐため，メール・ディレクトリは，専用のディレクトリにしたほうがよいでしょう。

メール・ディレクトリを設定するには， SET MAIL_DIRECTORY サブコマンドを使用します。次のように入力します。

```
JMAIL> SET MAIL_DIRECTORY ディレクトリ
```

ディレクトリは，メール・ディレクトリにするディレクトリの名前です。必ず[.**ディレクトリ**]の形式で指定します。

- 存在するディレクトリを指定した場合，次のようなメッセージが表示されます。このメッセージは確認のためのメッセージであり，エラーではありません。

```
JMAIL> SET MAIL_DIRECTORY [.MAILBOX]
%MAIL-I-EXISTS, USER$:[YAMADA.MAILBOX] はすでに存在します。
```

- 存在しないディレクトリを指定した場合，自動的にディレクトリが作成されます。このとき，次のようなメッセージが表示されます。このメッセージは確認のためのメッセージであり，エラーではありません。

```
JMAIL> SET MAIL_DIRECTORY [.MAILBOX]
%MAIL-I-CREATED, USER$:[YAMADA.MAILBOX] が作られました。
```

SHOW MAIL_DIRECTORY サブコマンドを使用して，メール・ディレクトリを確認することができます。次のように入力します。

```
JMAIL> SHOW MAIL_DIRECTORY
あなたのメール・ファイル・ディレクトリは USER$:[YAMADA.MAILBOX] です。
```

D.4 CC プロンプトを使用する — SET CC_PROMPT

CC は，カーボン・コピー (Carbon Copy) の略です。 CC プロンプトに対してユーザ名を入力すると，そのユーザに対して，「参考までに」メール・メッセージを送ることができます。また，何の設定もしていないと，メール・メッセージは手元に残りません。確認のために，自分自身を CC に入れて，メールを保存してもよいでしょう。

CC プロンプトを使用するには， SET CC_PROMPT サブコマンドを使用します。次のように入力します。

```
JMAIL> SET CC_PROMPT
```

D.5 自分自身へのコピーの設定 — SET COPY_SELF

自分自身へのコピーの設定をしておかないと，送ったメール・メッセージは手元に残りません。自分自身へのコピーを手元に残すように設定するには， SET COPY_SELF サブコマンドを使用します。次のように入力します。

```
JMAIL> SET COPY_SELF サブコマンド[,...]
```

サブコマンドは，コピーを残すサブコマンドを指定します。指定できるサブコマンドは， SEND， REPLY， FORWARD です。

```
JMAIL> SET COPY_SELF SEND,REPLY,FORWARD
```

D.6 エディタを設定する — SET EDITOR

メール・メッセージを送る場合，エディタを使用してメール・メッセージを作成，編集してから送ることができます。エディタを設定するには， SET EDITOR サブコマンドを使用し，次のように入力します。

```
JMAIL> SET EDITOR エディタ名
```

D.7 エディタを確認する — SHOW EDITOR

エディタが設定されたことを確認するには， SHOW EDITOR サブコマンドを使用します。次のように入力します。 (この例では XTPU を指定。)

```
JMAIL> SHOW EDITOR
あなたのエディタは XTPU です。
```

D.8 個人名の設定 — SET PERSONAL_NAME

個人名を設定することもできます。個人名を設定するには， SET PERSONAL_NAME サブコマンドを使用し，次のように入力します。

```
JMAIL> SET PERSONAL_NAME "個人名"
```

個人名は，ユーザ名の後に付ける個人名です。任意の文字列を指定できます。必ず，二重引用符 (") で囲みます。

```
JMAIL> SET PERSONAL_NAME "山田太郎/開発1課/東京"
```

D.9 JMAIL ユーティリティを終了する — EXIT

JMAIL ユーティリティを終了するには，次のいずれかの方法を使用します。

- JMAIL> に対して， EXIT サブコマンドを入力する。
- JMAIL> に対して，Ctrl + Z を入力する。

JMAIL ユーティリティを終了すると， DCL コマンドが入力できる状態に戻り，$プロンプトが表示されます。

```
JMAIL> EXIT
$ _
```

D.10 まだ読んでいないメール・メッセージを読む

まだ読んでいないメール・メッセージを読むには， READ サブコマンドを使用します。次のように入力します。

```
JMAIL> READ
```

ただし， JMAIL ユーティリティの使用中に，新たにメール・メッセージが届いた場合は，次のように入力します。

```
JMAIL> READ/NEW
```

D.11 すでに読んだメール・メッセージを読む

すでに読んだメール・メッセージを再度読む場合は， READ サブコマンドを使用します。ただし，そのときに新たにメール・メッセージが届いているかどうかによって方法が異なります。

- 新たにメール・メッセージが届いていないときは，次のように入力します。

```
JMAIL> READ
```

- 新たにメール・メッセージが届いているときは，次のように入力します。

```
JMAIL> READ MAIL［メール・メッセージ番号］
```

メール・メッセージ番号は，これから読むメール・メッセージの番号です。メール・メッセージ番号は省略できます。省略した場合，1 を指定したものとみなされます。

- 任意のフォルダに入っているメール・メッセージを読む場合は，次のように入力します。

```
JMAIL> READ フォルダ名［メール・メッセージ番号］
```

フォルダ名は，メール・メッセージの入っているフォルダの名前です。

メール・メッセージ番号は，これから読むメール・メッセージの番号です。メール・メッセージ番号は省略できます。省略した場合は，1 を指定したものとみなされます。

フォルダについては，第 D.17 節以降を参照してください。

D.12 メール・メッセージを続けて読む

メール・メッセージを続けて読む場合は，単に Return キー (または Enter キー) を押すだけで，次のメール・メッセージを読むことができます。

```
JMAIL>
```

D.13 メール・メッセージの一覧を表示する

メール・メッセージの一覧を表示するには，次のようにします。

```
JMAIL> DIRECTORY［フォルダ名］
```

フォルダ名は，一覧を表示するフォルダの名前です。

フォルダ名は省略できます。省略した場合，現在選択しているフォルダを指定したものとみなされます。フォルダを選択していない場合，NEWMAIL フォルダを指定したものとみなされます。ただし，まだ読んでいないメール・メッセージがない場合は，MAIL フォルダを指定したものとみなされます。

D.14 メール・メッセージを送る — SEND

メール・メッセージを送るには，SEND サブコマンドを使用します。SEND サブコマンドを使用してメール・メッセージを送る場合，次の方法があります。

方法	操作
その場で簡単に作成して送る	JMAIL> SEND
エディタを使用して編集してから送る	JMAIL> SEND/EDIT
既存のファイルをそのまま送る	JMAIL> SEND **ファイル**

D.15 メール・メッセージの返事を送る — REPRY

メール・メッセージに返事を送るには，REPLY サブコマンドを使用します。REPLY サブコマンドを使用してメール・メッセージに返事を送る場合，次のような方法があります。

方法	操作
その場で簡単に作成して送る	JMAIL> REPLY
エディタを使用して編集してから送る	JMAIL> REPLY/EDIT
受け取ったメール・メッセージを引用して送る	JMAIL> REPLY/EXTRACT
既存のファイルをそのまま送る	JMAIL> REPLY **ファイル**

D.16 受け取ったメール・メッセージを他の人に送る — FORWARD

受け取ったメール・メッセージを他の人に送るには，FORWARD サブコマンドを使用します。FORWARD サブコマンドを使用して，受け取ったメール・メッセージを他の人に送る場合，次のような方法があります。

方法	操作
受け取ったメール・メッセージをそのまま送る	JMAIL> FORWARD
受け取ったメール・メッセージを編集して送る	JMAIL> FORWARD/EDIT

D.17 メール・メッセージをフォルダに入れる — MOVE

メール・メッセージをフォルダに入れるには，MOVE サブコマンドを使用します。次のように入力します。

```
JMAIL> MOVE フォルダ名
```

フォルダ名は，直前に読んだメール・メッセージを入れるフォルダの名前です。フォルダが存在しない場合は，次のような問い合わせメッセージが表示されます。

```
JMAIL> MOVE OSAKA_GIJIROKU
フォルダ OSAKA_GIJIROKU は存在しません。
          このフォルダを作成しますか （Y/N，省略時は N です）？
```

ここで，フォルダを新たに作成するかどうかを答えます。

- 作成する場合は， Y を入力します。その結果，次のように表示され，フォルダが作成されます。

  ```
  %MAIL-I-NEWFOLDER, フォルダ OSAKA_GIJIROKU が作成されました。
  ```

- 作成しない場合は， N を入力します。その結果，新しいフォルダは作成されず，メール・メッセージは MAIL フォルダに入れられます。

D.18 フォルダの一覧を表示する — DIRECTORY/FOLDER

現在存在するフォルダの一覧を表示することができます。フォルダが存在するということは，その中にメール・メッセージが入っていることを意味します。メール・メッセージの入っていない空のフォルダはありません。フォルダの一覧を表示するには，DIRECTORY/FOLDER サブコマンドを使用します。次のように入力します。

```
JMAIL> DIRECTORY/FOLDER
```

D.19 フォルダを選択する — SELECT

メール・メッセージの入っているフォルダを選択するには， SELECT サブコマンドを使用します。次のように入力します。

```
JMAIL> SELECT フォルダ名
```

D.20 メール・メッセージを削除する — DELETE

メール・メッセージを削除するには， DELETE サブコマンドを使用します。次のように入力します。

```
JMAIL> DELETE [メール・メッセージ番号]
```

メール・メッセージ番号は，削除したいメール・メッセージの番号です。複数のメール・メッセージ番号を指定することもできます。複数のメール・メッセージ番号を指定する場合，コンマで区切ります。

現在読んでいるメール・メッセージの番号は省略することもできます。

D.21 誤って削除したメール・メッセージの復元 — SELECT WASTEBASKET

メール・メッセージを誤って削除した場合，JMAIL ユーティリティをまだ終了していなければ，そのメール・メッセージを復元することができます。

削除したメール・メッセージは，JMAIL ユーティリティを終了するまでは，WASTEBASKET フォルダに一時的に保存されます。これらのメール・メッセージは，JMAIL ユーティリティの終了時にまとめて実際に削除されます。メール・メッセージの復元は，WASTEBASKET フォルダからメール・メッセージを取り出すことにより行います。

次の手順で行います。

1. WASTEBASKET フォルダを選択します。

```
JMAIL> SELECT WASTEBASKET
```

2. メール・メッセージの一覧を表示します。

```
JMAIL> DIRECTORY
```

3. 復元したいメール・メッセージを読みます。

```
JMAIL> READ
```

4. 復元したいメール・メッセージを，他のフォルダに入れます。

```
JMAIL> MOVE MEMO
```

D.22 メール・メッセージをファイルに落とす — EXTRACT

メール・メッセージをファイルにするには，EXTRACT サブコマンドを使用します。次のように入力します。

```
JMAIL> EXTRACT ファイル名
```

ファイル名は，メール・メッセージをファイルにするときのファイルの名前です。

バージョン番号は省略できます。バージョン番号を省略したときにファイルが存在しない場合は，バージョンが 1 のファイルを新たに作成します。バージョン番号を省略したときにファイルが存在する場合は，最新のバージョンに 1 を加えたバージョンのファイルを作成します。

D.23 配布リストを利用する

メール・メッセージを送る複数の相手の宛先をまとめて 1 つのファイルにしたものを，**配布リスト**といいます。配布リストは，エディタを使用して作成します。配布リストのファイル名は任意です。ファイル・タイプには，なるべく DIS を指定してください。

配布リストを作成する際は， 1 行に 1 ユーザ名のみを入力してください。 (感嘆符 (!) に続けてコメントを入力できます。)

D.24 配布リストの指定方法

配布リストを指定する場合，アットマーク (@) を付けて指定します。次のように指定します。

```
@[ディレクトリ]ファイル名.ファイル・タイプ;バージョン番号
      1          2          3            4
```

1 現在のディレクトリに配布リストが置かれている場合，ディレクトリは省略できます。

2 ファイル名は任意で付けられます。

3 ファイル・タイプが DIS の場合，ファイル・タイプは省略できます。

4 バージョン番号は省略できます。省略した場合，最新のバージョンを指定したものとみなされます。

配布リストは，ユーザ名と同様に，宛先 (To:， CC:) に指定できます。たとえば，MEMBER.DIS という名前で配布リストを作成したとします。配布リストは，ユーザ名と同様に使用できるので，次のように入力することができます。

```
To:      @MEMBER,OSAKA::TANAKA
CC:
Subj:
```

D.25 JMAIL のヘルプを使用する

JMAIL ユーティリティでは，サブコマンドの使い方と働きを調べるためのオンライン・ヘルプが用意されています。ヘルプの使用方法は DCL コマンドの場合と同様です。 JMAIL のヘルプを使用する場合は，次のように入力します。

```
JMAIL> HELP
```

ヘルプが開始され，次のように表示されます。

```
HELP

    HELP コマンドは，日本語メール・ユーティリティに関する情報が欲しいと
    きに使用します。

    すべての JMAIL コマンドに関する情報が必要なときは，次のコマンドを入
    力してください:

          JMAIL> HELP

    個々のコマンドまたはトピックについて知りたいときは，HELP の後にコマ
    ンド名またはトピック名を続けて入力してください。

    形式

      HELP  [トピック]


  Additional information available:

  ANSWER      ATTACH      BACK        COMPRESS    COPY        CURRENT     DEFINE
  DELETE      DIRECTORY   EDIT        ERASE       EXIT        EXTRACT     FILE
  Files       FIRST       Folders     FORWARD     Getting_Started         HELP
  JMAIL_Commands          Keypad      LAST        MAIL        MARK        MOVE
  NEXT        Overview    PRINT       PURGE       QUIT        READ        REMOVE
  REPLY       SEARCH      SELECT      SEND        SET-SHOW    SHOW        SPAWN
  UNMARK      Usage_Summary

Topic? _
```

1. "Topic?"の後に，調べたいサブコマンドを入力し，Return (またはEnter) キーを入力します。

   サブコマンドの説明と "サブコマンド名 Subtopic?"が表示されます。

2. "サブコマンド名 Subtopic?"の後に，調べたい修飾子などを入力します。

ヘルプの終了方法

ヘルプを終了するには，次のいずれかの方法を使用します。

- "Topic?"に対して，Return (または Enter) キーを押す。
- 任意の箇所で，Ctrl + Z を入力する。

E

レキシカル関数一覧

レキシカル関数は，DCL コマンド・ラインやコマンド・プロシージャ内で使うことのできる関数です。レキシカル関数は，現在のプロセスに関する情報 (たとえば，デフォルトのディレクトリなど)，あるいは別の文字列に関する情報 (たとえば，文字列の長さや位置) を得ることができます。

ここでは，よく使用されるレキシカル関数をまとめます。

各レキシカル関数についての詳細は，『DCL ディクショナリ』を参照してください。

表 E–1 システムとその処理に関する情報入手のためのレキシカル関数

レキシカル関数	使用目的
F$FILE_ATTRIBUTES(ファイル指定, 項目)	特定のファイルに関する情報を得る
F$MESSAGE(コード)	特定のシステムの状態値に対応するメッセージを得る
F$GETDVI(デバイス名，項目)	特定のデバイスの情報を得る
F$GETJPI(プロセス識別コード，項目)	特定のプロセスの会計情報，状態値情報，識別情報を得る
F$GETQUI(ファンクション, [アイテム]，[オブジェクト ID]，[フラグ])	キューとジョブに関する情報を得る
F$GETSYI(項目[, ノード名])	システムに関する状態値情報と識別情報を得る
F$PARSE(ファイル指定[, デフォルト指定][, 関連指定][, フィールド][, タイプ])	ファイル記述の指定した部分を入手したり，ファイル記述の一部を変更する
F$PID(プロセス・シンボル)	システムに存在するプロセスの PID を得る
F$SEARCH(ファイル指定[, 文字列識別子])	完全なファイル指定を得る
F$TIME( )	現在の日付と時刻を dd-mmm-yyyy hh:mm:ss.ccの形式で得る
F$IDENTIFIER(UIC，変換タイプ)	UIC を数値から文字列に，またはその逆の変換を行う

表 E–2 利用者プロセスについての情報入手のためのレキシカル関数

レキシカル関数	使用目的
F$DIRECTORY( )	現在のデフォルト・ディレクトリ名を得る

(次ページに続く)

表 E–2 (続き) 利用者プロセスについての情報入手のためのレキシカル関数

レキシカル関数	使用目的
F$MODE( )	現在のプロセスのモード（会話，バッチ，ネットワーク，その他のいずれか）を得る
F$PRIVILEGE(特権状態)	プロセスが指定した特権を持っているかどうかを判断する
F$PROCESS( )	現在のプロセスの名前を得る
F$SETPRV(特権状態)	プロセスの特権を設定し，もとの設定を得る
F$USER( )	現在のプロセスの UIC を得る
F$VERIFY([プロシージャ値][, イメージ値])	VERIFY 機能をオンかオフに設定したり，判断したりする
F$ENVIRONMENT(項目)	DCL コマンド実行時の環境を得る
F$TRNLNM(論理名[, テーブル] [, インデックス][, モード][, ケース] [, 項目])	論理名の翻訳を 1 レベルだけ行う

表 E–3 文字列や整数の操作のためのレキシカル関数

レキシカル関数	使用目的
F$CVSI(ビット位置，幅，文字列)	文字列の一部を取り出し，それを符号の付いた 10 進整数に変換する
F$CVUI(ビット位置，幅，文字列)	文字列の一部を取り出し，それを符号のない整数に変換する
F$CVTIME([入力時刻][, 出力時刻] [, フィールド])	dd-mmm-yyyy hh:mm:ss.cc という形式の ASCII 文字列を指定した形式の文字列に変換する
F$EXTRACT(相対位置，けた数，文字列)	指定した文字列の一部を取り出す
F$FAO(制御文字列[, アーギュメント 1, アーギュメント 2，…，アーギュメント 15])	文字列を指定形式の出力文字列に合成する
F$INTEGER(式)	数値文字列をそれに対応する整数に変換する
F$LENGTH(文字列)	文字列の長さを得る
F$LOCATE(部分文字列，文字列)	与えられている文字列の中で指定されている部分文字列の位置を見つける
F$STRING(式)	整数演算式を数値列に変換する
F$EDIT(文字列，リスト)	文字列をリストで指定した形式で編集する
F$ELEMENT(番号，区切り文字，文字列)	指定された区切り文字で区切られた要素から文字列を取り出す
F$TYPE(シンボル名)	シンボルのデータ・タイプを決定する

F

Linux または UNIX，MS–DOS，日本語 OpenVMS コマンド対応表

ここでは， Linux や UNIX，MS–DOS，日本語 OpenVMS でよく使用されるコマンドの対応表を示します。

表中 ( ) で囲まれたコマンドは，類似機能を持ったコマンドです。また空欄は対応するコマンドがないことを示します。

表 F–1 コマンド対応表 (一般コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
!		RECALL	過去に入力したコマンドを呼び出す
|	|	(PIPE)	コマンドの標準出力を次のコマンドに引き渡す
>	>	(PIPE)	出力をリダイレクトする
>>	>>	(PIPE)	出力をファイルへ追加する
<	<	(PIPE)	入力をリダイレクトする
<<	<<	(PIPE)	入力終端文字列を指定する
&		(SPAWN/NOWAIT)	コマンドをバック・グラウンドで実行する
alias	DOSKEY	= =, := =	コマンドの別名を定義したり，マクロを作成する
bg		(ATTACH)	ジョブをバック・グラウンドで実行するよう切り替える
source	CALL	@	親プロセスにスクリプトを起動させる バッチファイル中から，別のバッチファイルを呼び出す
cd	CD， CHDIR	SET DEFAULT	デフォルト・ディレクトリの変更（移動）
date	DATE, TIME	SET TIME, SHOW TIME	日時を設定，表示する
exit	EXIT	LOGOUT	ログアウトする
fg		(ATTACH)	ジョブをフォア・グラウンドで実行するよう切り替える
	GOTO	GOTO	バッチファイル（コマンド・プロシージャ）内でラベルが付けられた行へコマンドインタプリタの実行を移す
	IF	IF	バッチファイル内で，条件処理を実行する
history	DOSKEY	RECALL	過去に入力したコマンドを再呼び出しする

(次ページに続く)

表 F–1 (続き) コマンド対応表 (一般コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
jobs		SHOW SYSTEM	現在のプロセスの情報を表示する
	PAUSE	(INQUIRE)	バッチファイルの処理を一時停止し，メッセージを表示する
man	HELP	HELP	オンライン・ヘルプを参照する
passwd		SET PASSWORD	パスワードを変更する
popd [csh 内]	POPD		スタックに保存したディレクトリに戻る
	PROMPT	SET PROMPT	プロンプトを変更する
pushd [csh 内]	PUSHD		カレント・ディレクトリをスタックに保存して移動する
	REM	!	バッチファイル（コマンド・プロシージャ）の中で，コメント（注釈）を記入する
setenv	SET	= =, := =, DEFINE, SHOW LOGICAL, SHOW SYMBOL	環境変数を表示，設定，または削除する
stty, tset	MODE	SET TERMINAL	ターミナル属性を設定する
unalias		DELETE/SYMBOL	alias 名（シンボル定義）を削除する
	VER	(SHOW SYSTEM /NOPROC)	システムのバージョンを表示する
vmstat		MONITOR	メモリや CPU の負荷率や使用状況を表示する
w		SHOW PROCESS	ログインしているユーザと使用状況を表示する
wait		SYNCHRONIZE	プロセスおよびジョブの終了を待つ
who		SHOW USER	現在ログインしているユーザ名を表示する
write		(REPLY/USER)	ログイン中のユーザにメッセージを送る

表 F–2 コマンド対応表 (ファイル管理コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
basename			ディレクトリや末尾の文字列を削除したファイル名だけを出力する
ls -l	CACLS	DIR/SECURITY, SET FILE/ACL	ファイルのアクセス制御リスト（ACL）等を表示または変更する
chgrp		SET FILE/OWNER	ファイルやディレクトリのグループを変更する
chmod	ATTRIB	SET FILE/PROT	ファイルやディレクトリのアクセス権を変更する
chown		SET FILE/OWNER	ファイルやディレクトリの所有者を変更する
compress			ファイルのサイズを縮小する

(次ページに続く)

表 F–2 (続き) コマンド対応表 (ファイル管理コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
cp	COPY	COPY	ファイルやディレクトリをコピーする
dd		CONVERT	ファイルの変換とコピーを行う
df		SHOW DEVICE	ディスク・ドライブの使用量を表示する
	DISKCOPY	BACKUP/IMAGE	ディスクの内容を別のディスクにコピーする
diff	COMP, FC	DIFFERENCE	ファイルの内容を比較する
du	DIR/S	DIR/SIZE/TOTAL	ディレクトリ内のファイル容量を表示する
find	DIR/S	DIR	ファイルやディレクトリを検索する
	LABEL	SET VOLUME	ディスクのボリュームラベルを作成，変更，または削除する
ln		(SET FILE/ENTER)	ファイルやディレクトリにリンクを張る
ls	DIR	DIR	ファイルやディレクトリの情報を表示する
mkdir	MD, MKDIR	CREATE/DIRECTORY	ディレクトリを作成する
mv	MOVE, REN, RENAME	RENAME	ファイルやディレクトリの移動，名前の変更を行う
od		DUMP	バイナリ・ファイルの内容を表示する
pwd	CD	SHOW DEFAULT	現在のディレクトリの場所を確認する
rm	DEL, ERASE	DELETE	ファイルやディレクトリを削除する
rmdir	DEL, RD, RMDIR	DELETE	ディレクトリを削除する
split			ファイルを分割する
	SUBST	DEFINE	パスをドライブ名で置き換える（論理名に等価名を対応させる）
tar		BACKUP	複数のディレクトリやファイルを 1 つのファイルに格納または復元する
touch			ファイルのタイムスタンプを変更する
	VOL	(SHOW DEVICE)	ディスクのボリュームラベルとシリアル番号を表示する
cp-r	XCOPY	COPY, BACKUP	ファイルやディレクトリ構造をコピーする

表 F–3 コマンド対応表 (テキスト・ファイル操作コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
cat	TYPE	TYPE	テキスト・ファイルの内容を表示する
cut			テキスト・ファイルの各行から文節を取り出す

(次ページに続く)

表 F–3 (続き) コマンド対応表 (テキスト・ファイル操作コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
grep	FIND, FINDSTR	SEARCH	文字列を検索する
iconv		KCODE, ICONV	文字コードを変換する
head		(TYPE/PAGE)	テキスト・ファイルの先頭部分を表示する
less		TYPE/PAGE=SAVE	テキスト・ファイルの内容をページ単位で表示する
more	(MORE)	TYPE/PAGE	テキスト・ファイルの内容をページ単位で表示する
nkf		KCODE, ICONV	文字コードを変換する
sort	SORT	(SORT)	行を並べ替える
tail		TYPE/TAIL	テキスト・ファイルの末尾部分を表示する
uniq		(SORT /NODUPLICATE)	重複した行を削除する
vi		EDIT	テキスト・ファイルを編集する
wc			テキスト・ファイルの行数，単語数，バイト数を表示する

表 F–4 コマンド対応表 (ジョブ・プロセス管理コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
at	AT	SUBMIT	指定時刻にジョブを実行する
atq	AT	SHOW ENTRY	実行待ちジョブを表示する
atrm	AT/DELETE	DELETE/ENTRY	実行待ちジョブを削除する
batch		SUBMIT	自動的にジョブを実行する
crontab			プログラムを定期的に実行する crond 用の設定ファイルを編集する
kill		STOP	プロセスおよびジョブを強制終了する
nice			優先順位を決めてコマンドを実行する
nohup			ログアウトした後もコマンドを実行し続ける
ps		SHOW PROCESS, SHOW SYSTEM	実行中のプロセスを表示する
sleep		WAIT	指定された時間が過ぎるまで現在のプロセスを待ち状態にする
stop [csh 内]		(STOP)	バック・グラウンドのジョブを停止する
top		MONITOR	現在のシステムの状況の表示やプロセスの管理をする

表 F–5　コマンド対応表 (ネットワーク関連コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
ftp	FTP	FTP	FTP サーバに接続し，ファイル転送または表示を行う
hostname		(TCPIP)	ホスト名を登録する
ping	PING	(TCPIP PING)	パケットを送りネットワークの状況を調べる
rcp		RCP	リモート・マシン間でファイルをコピーする
rlogin		RLOGIN	リモート・マシンにログインする
rsh		RSH	リモート・マシンのコマンドを実行する
telnet	TELNET	TELNET	他のホストと通信をする

表 F–6　コマンド対応表 (印刷関連コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
lpc		SET ENTRY	印刷を制御する
lpq		SHOW QUEUE	印刷キューを確認する
lpr	PRINT	PRINT	プリンタで印刷をする
lprm		DELETE/QUEUE	印刷キューを削除する

表 F–7　コマンド対応表 (システム管理コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
disklabel	FDISK		ハードディスクのパーティションを設定する
fastboot			システムを高速で再起動する
fasthalt			システムを高速でシャットダウンする
fdisk	FDISK		ハードディスクのパーティションを設定する
finger			ユーザ情報を表示する
free		SHOW MEMORY	メモリの使用状況を表示する
fsck	CHKDSK, SCANDISK	ANALYZE/DISK	ディスク検査と修復を行う
groupadd		AUTHORIZE	グループを追加する
groupdel		AUTHORIZE	グループを削除する
groupmod		AUTHORIZE	グループ情報を変更する
halt		(SHUTDOWN)	プロセッサを停止する
id		(AUTHORIZE)	ユーザやグループ ID を出力する
last			最近ログインしたユーザの情報を表示する
lastlog		(AUTHORIZE)	各ユーザの最後にログインした日付を表示する
login		(SET HOST)	ログインする

(次ページに続く)

表 F–7 (続き) コマンド対応表 (システム管理コマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
mkfs	FORMAT	INITIALIZE	ファイル・システムを作成する
mount		MOUNT	ファイル・システムをマウントする
newfs	FORMAT	INITIALIZE	ファイル・システムを作成する
reboot		(REBOOT)	システムを再起動する
shutdown		(SHUTDOWN)	システムをシャットダウンあるいは再起動する
su			ユーザ ID 一時的に切り替える
uname			システム情報を表示する
umount		DISMOUNT	ファイル・システムをアンマウントする
useradd		AUTHORIZE	ユーザを追加する
userdel		AUTHORIZE	ユーザを削除する
usermod		AUTHORIZE	ユーザのアカウント情報を変更する
vipw			passwd ファイルを編集する

表 F–8 コマンド対応表 (その他のコマンド)

Linux または UNIX	MS–DOS	日本語 OpenVMS	機能
banner			指定された文字を拡大して表示する
	BREAK	(SET CONTROL)	拡張 CTRL/C チェックを設定または解除する
cal			カレンダを表示する
	CHCP	(LOCALE)	有効なコードページ番号を表示または設定する
echo	ECHO		引数に与えられた文字列を表示する
tee			標準入力を標準出力とファイルに出力する
which			コマンドを探す

G

関連アプリケーション

ここでは，日本語 OpenVMS にさらに便利な機能を追加することのできる，関連アプリケーションを紹介します。

ポストスクリプト (PostScript) プリンタを使う

日本語 DECprint Supervisor for OpenVMS を導入します。 PostScript ファイルおよび漢字テキスト・ファイルが日本語 PostScript プリンタで印刷できるようになります。また，トレイの選択，両面印刷や印刷フォーマットの指定などの特別な機能もサポートされます。

ファイル・サーバとして Windows PC から利用する

OpenVMS Alpha では日本語 Advanced Server for OpenVMS を導入します。 WindowsNT Server 4.0 と同等の機能を実現します。日本語 OpenVMS のディレクトリが共有フォルダとして PC から利用できるようになります。

OpenVMS Integrity では CIFS for OpenVMS を導入します。

Web サーバとして利用する

Secure Web Server for OpenVMS を導入します。この製品は Apache web server を OpenVMS 向けに強化したものです。 Perl， Java， PHP などもサポートします。

アプリケーションをネットワーク対応にする

BridgeWorks を導入します。現在使用しているアプリケーションが簡単にネットワーク越しに利用できるようになります。

データベースをネットワーク対応にする

Attunity Connect を導入します。現在使用しているデータベースが簡単にネットワーク越しに利用できるようになります。

マウスを使う

DECwindows Motif for OpenVMS を導入します。 CDE， Motif のウィンドウ・システムが利用できます。

ブラウザを使う

Netscape Navigator を導入します。 (バージョンが古いので最近の Web はうまく見えないことがあります。) 現在 Mozilla が開発中であり， Technology Demonstration 版が公開されています。端末であれば LYNX が使用できます。

データベースを使う

Oracle Rdb または Oracle8i を導入します。

XML を使う

Apache XML Project の XML パーサ (Xerces) および XSLT スタイル・シート・プロセッサ (Xalan) が Java， C++ で使用できます。

ディレクトリ・サービスを使う

OpenVMS Enterprise Directory for e-Business を導入します。業界標準の LDAPv3 と X.500 の両方が使え，強力で拡張性のあるディレクトリ・サービスが構築できます。

Freeware を使う

OS 添付の Freeware CD sets が利用できます。また OpenVMS の Web サイトからも利用できます。

プログラムを開発する

次のようなコンパイラや DECset(プログラム支援ツール) などがあります。

◆言語

C， C++， Cobol， Fortran， Java など

◆ DECset

CMS， LSE， MMS， SCA など

ツールについての説明は，日本語 OpenVMS のオンライン・ヘルプ等をご覧ください。

H

PC 関連製品

ここでは，日本語 OpenVMS とともに使用できるパーソナル・コンピュータ (PC) 関連製品を紹介します。

H.1 PC 用の VT ターミナル・エミュレータ製品

Tera Term Pro

フリーウェア
以下のサイトからダウンロードできます。
http://hp.vector.co.jp/

Reflection for UNIX and OpenVMS

WRQ 社製品
販売代理店：サイバネットシステム株式会社 Web サイト
http://www.cybernet.co.jp/products/network/index.html
（評価版のダウンロード可能）

H.2 PC 用の X サーバ製品

Reflection X

WRQ 社製品
販売代理店：サイバネットシステム株式会社 Web サイト
http://www.cybernet.co.jp/products/network/index.html
（評価版のダウンロード可能）

eXcursion (PATHWORKS 32 のコンポーネントの一部)

弊社製品

Exceed

Hummingbird Communication Ltd 社製品
販売代理店：株式会社マクニカ Web サイト
http://www.net.macnica.co.jp/

日本語 OpenVMS クイック・リファレンス

表 I–1　DCL コマンドの形式

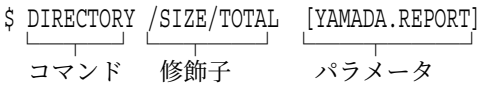
項目	内容
コマンド	DCL コマンドの名前
修飾子	/ で始まる単語。処理の内容を細かく指定する。 省略または複数指定できる。複数指定する場合は，連続して指定する。
パラメータ	処理する対象を省略または複数指定できる。
例	

表 I–2　一般的な操作の DCL コマンド

コマンド	説明
APPEND	複数のファイルの内容を 1 つのファイルに追加し，まとめる
COPY	指定されたファイルをコピーする
CREATE	ファイルまたはディレクトリを作成
CREATE /DIRECTORY	ディレクトリを作成
DEFINE/KEY	コマンド列をキーに割り当てる
DELETE	指定されたファイルをディレクトリから削除
DELETE/ENTRY	ジョブ・エントリを削除
DIFFERENCES	ファイルの内容を比較する
DIRECTORY	ディレクトリの内容 (ファイルの一覧) を表示する
EDIT	テキスト・ファイルを編集する
HELP	オンライン・ヘルプを見る
JMAIL	JMAIL (日本語メール) ユーティリティを起動する
LOGOUT	セッションを終了する
PRINT	指定されたファイルをプリンタに送ってプリントする
PURGE	最新バージョン以外のファイルを削除
RECALL	コマンドを呼び出す
RENAME	指定されたファイルの名前や置かれる場所 (ディレクトリ) を変更する

(次ページに続く)

表 I–2 (続き) 一般的な操作の DCL コマンド

コマンド	説明
SEARCH	ファイルの内容を検索する
SET DEFAULT	現在のデフォルト・ディレクトリを変える
SET KEY	コマンド列をキーに割り当てる
SET PASSWORD	パスワードを設定する
SET SECURITY	ファイルの保護を行う
SET TERMINAL	ターミナルの設定を変更する
SHOW DEFAULT	現在のデフォルト・ディレクトリを調べる
SHOW ENTRY	ジョブ・エントリを調べる
SHOW LOGICAL	指定した論理名の等価文字列などを表示する
SHOW QUOTA	現在のディスク・クォータを調べる
SHOW SYMBOL	シンボル名を調べる
SHOW SYSTEM	現在のプロセス情報を表示する
SHOW TERMINAL	ターミナルの現在の属性を表示する
SHOW USER	ユーザ名，ノード名 (OpenVMS Cluster 環境の場合) を表示する
SORT	ファイルの内容を並べかえる
TYPE	指定されたファイルの内容を画面に表示する

表 I–3 コマンドの再呼び出し方法

方法 1： Ctrl + B を押す

方法 2： 上下の矢印キーを使用する

方法 3： RECALL コマンドを入力する

- RECALL/ALL
- RECALL 数字
- RECALL 英文字
- RECALL/ERASE (再呼び出しバッファをクリア)

表 I–4 完全なファイル指定

ノード:: デバイス:[ディレクトリ]ファイル名. ファイル・タイプ; バージョン

表 I–5 ファイル指定の各要素

要素	説明
ノード	ネットワーク・ノード名

(次ページに続く)

表 I–5 (続き) ファイル指定の各要素

要素	説明
デバイス	ファイルが格納または書き込まれる物理デバイスの名前
ディレクトリ	ファイルを登録するディレクトリの名前。ディレクトリ名は，[ ] で区切る
ファイル名	ファイルの名前
ファイル・タイプ	ファイルの中の構造またはデータのタイプ
バージョン	ファイルのバージョン番号

表 I–6 DCL コマンドが使用するデフォルトのファイル・タイプ

ファイル・タイプ	説明
.CLD	コマンド定義ファイル
.COM	コマンド・プロシージャ・ファイル
.DAT	データ・ファイル
.DIF	DIFFERENCES コマンドによって作成される出力ファイル
.DIR	ディレクトリ・ファイル
.DIS	JMAIL ユーティリティで使用する配布リスト・ファイル
.EVE	日本語 EVE エディタのスタートアップ・コマンド・ファイル
.EXE	リンカ・ユーティリティによって作成される実行可能プログラム・イメージ・ファイル
.HLB	ヘルプ・テキスト・ライブラリ・ファイル
.HLP	ヘルプ・ライブラリのソース・ファイル
.INI	初期化ファイル
.JOU	エディタによって作成されるジャーナル・ファイル
.LIS	コンパイラまたはアセンブラによって作成されるリスト・ファイル，または PRINT コマンドと TYPE コマンドの省略時の入力ファイル
.LOG	バッチ・ジョブ出力ファイル
.MAI	MAIL メッセージ・ファイル
.MAP	リンカ・ユーティリティによって作成されるメモリ割り当てマップ
.MEM	DIGITAL Standard Runoff (DSR) によって作成される出力ファイル
.OBJ	コンパイラやアセンブラによって作成されるオブジェクト・ファイル
.PS	PostScript 形式のファイル
.REGIS	Regis 形式のファイル
.RNO	DIGITAL Standard Runoff (DSR) の入力ソース・ファイル
.SIX	シクセル・グラフィック・ファイル
.SYS	システム・イメージ
.TJL	DECTPU と ACL エディタによって作成されるジャーナル・ファイル
.TLB	テキスト・ライブラリ・ファイル
.TMP	一時的ファイル

(次ページに続く)

表 I–6 (続き) DCL コマンドが使用するデフォルトのファイル・タイプ

ファイル・タイプ	説明
.TPU	EVE エディタのコマンド・ファイル
.TPU$JOURNAL	EVE エディタによって作成されるジャーナル・ファイル
.TXT	テキスト・ファイル

表 I–7 よく使用されるキーの組み合わせによる操作 (DCL コマンド・ラインで使用)

組み合わせキー	説明
Ctrl + A	上書モードと挿入モードを切り替える
Ctrl + B	以前に入力したコマンドを再呼び出しする
Ctrl + C	現在実行中の処理を取り消す
Ctrl + D	カーソルを 1 文字だけ左に移動
Ctrl + E	カーソルを行末に移動
Ctrl + F	カーソルを 1 文字だけ右に移動
Ctrl + H	カーソルを行頭に移動
Ctrl + I	カーソルを次のタブ・ストップに移動
Ctrl + J	カーソルの左にある単語 1 語分を削除
Ctrl + M	Return (Enter)
Ctrl + O	画面の表示を一時停止または再開する
Ctrl + Q	画面出力を再開する
Ctrl + S	画面出力を一時停止する
Ctrl + T	プロセスについての統計情報を表示する
Ctrl + U	カーソルのある位置から左のテキストを行頭まで削除
Ctrl + V	制御文字を入力する。 (日本語 EVE エディタで)
Ctrl + Y	現在実行中の処理を中断する。 (Ctrl + Cよりも強力)
Ctrl + Z	処理を正常終了させる。

表 I–8 ファイルの保護

区分	説明
所有者区分	SYSTEM (S), OWNER (O), GROUP (G), WORLD(W)。 各区分は，最初の 1 文字に短縮できる。
アクセス区分	読込み (R), 書込み (W), 実行 (E), 削除 (D)。 アクセス区分は，それぞれの所有者区分に対して指定する。何も指定しない場合は，どのアクセスもできない。

J

日本語 OpenVMS マニュアル・クイック・リファレンス

日本語 OpenVMS で，ある操作をしたい時にどのマニュアルを参照すればよいか，代表的な操作や事柄と参照マニュアルの一覧を表 J–1 にまとめます。

マニュアルは随時アップデートされたり，新規のものが提供されたりしています。最新のものは製品版 CD-ROM または，次の Web サイトのオンライン・ドキュメントのページでご覧ください。

http://www.hp.com/jp/openvms/

表 J–1 日本語 OpenVMS マニュアル・クイック・リファレンス

代表的な操作/事項	参照するマニュアル
製品の概要を知りたい	『OpenVMS をご使用のお客様へ』 『ソフトウェア仕様書 (SPD)』 『概説書』 『ユーザーズ・マニュアル』 『新機能説明書』
マニュアルの概要を知りたい	『新機能説明書』
システムをインストールしたい	『OpenVMS をご使用のお客様へ』 『インストレーション・ガイド』 『リリース・ノート』 『リリース・ノート[翻訳版]』 『ソフトウェア仕様書 (SPD)』
システム管理について知りたい	『システム管理者マニュアル』 『システム管理ユーティリティ・リファレンス・マニュアル』
ターミナルの設定をしたい	『概説書』
文書を作成したい	『日本語 EVE ユーザーズ・ガイド』 『日本語 EVE リファレンス・マニュアル』 『日本語 EVE かな漢字変換入門』 『DEC XTPU リファレンス・マニュアル』 『ユーザーズ・マニュアル』

(次ページに続く)

表 J–1 (続き) 日本語 OpenVMS マニュアル・クイック・リファレンス

代表的な操作/事項	参照するマニュアル
個人辞書の編集をしたい	『日本語ユーティリティ　利用者の手引き』
メールを送りたい	『日本語ユーティリティ　利用者の手引き』 『ユーザーズ・マニュアル』
ファイルを印刷したい	『日本語ユーティリティ　利用者の手引き』
DCL コマンドについて詳しく知りたい	『DCL ディクショナリ』
キーの定義を変えたい	『ユーザ・キー定義　利用者の手引き』
日本語ファイル名サポートついて詳しく知りたい	『概説書』 『日本語ライブラリ 利用者の手引き』
日本語入力プロセスについて詳しく知りたい	『日本語入力プロセス 利用者の手引き』
日本語ユーティリティのエラーメッセージの内容を知りたい	『日本語メッセージ　利用者の手引き』
日本語処理プログラムを書きたい	『日本語ライブラリ　利用者の手引き』 『IMLIB/OpenVMS ライブラリ リファレンス・マニュアル』
日本語ビデオ・ターミナルでメニューの作成をしたい	『日本語画面管理ライブラリ利用者の手引』
フォントを作成/変更/管理したい	『フォント管理ユーティリティ 利用者の手引き』
漢字コードについて知りたい	『漢字コード表』
デバッグをしたい	『日本語ユーティリティ　利用者の手引き』 『デバッガ説明書』 『デバッガ・コマンド・ディクショナリ』

(次ページに続く)

表 J–1 (続き) 日本語 OpenVMS マニュアル・クイック・リファレンス

代表的な操作/事項	参照するマニュアル
XPG4 に準拠したユーティリティで, アプリケーションのローカライズをしたい	『C 国際化ユーティリティ・リファレンス・マニュアル』
VAX システムから Alpha システムへシステムの移行をしたい	『OpenVMS VAX から OpenVSM Alpha へのアプリケーションの移行』
VAX で使用したアプリケーションを Alpha システムで使用したい	『OpenVMS VAX から OpenVSM Alpha へのアプリケーションの移行』
64 ビット・アドレッシングについて詳しく知りたい	『OpenVMS Alpha 64 ビット・アドレッシングおよび VLM 機能説明書』
VLM 機能について詳しく知りたい	『OpenVMS Alpha 64 ビット・アドレッシングおよび VLM 機能説明書』
Extended File Specifications 機能について詳しく知りたい	『OpenVMS Extended File Specifications の手引き』
OpenVMS Galaxy 機能について詳しく知りたい	『OpenVMS Alpha パーティショニングおよび Galaxy ガイド』
OpenVMS Cluster 構成の設計について詳しく知りたい	『OpenVMS Cluster 構成ガイド』
OpenVMS Cluster シュステム管理について詳しく知りたい	『OpenVMS Cluster システム』
Volume Shadowing について詳しく知りたい	『Volume Shadowing for OpenVMS 概説書』

索引

T

U

V

W

X

ア

イ

エ

オ

カ

キ

ク

ケ

コ

サ

シ

ソ

タ

テ

ト

ニ

ネ

ハ

ヒ

フ

ヘ

ホ

マ

メ

モ

ユ

ヨ

リ

ル

レ

ロ

ワ

はじめよう！日本語 OpenVMS

2009 年 11 月　発行

日本ヒューレット・パッカード株式会社

〒 102-0076 東京都千代田区五番町 7 番地

電話　(03)3512-5700 (大代表)

AA-RSTYA-TE.2