Solaris ユーザーズガイド (上級編)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–3946–11
2003 年 8 月

原典: *Solaris Advanced User's Guide*

Part No: 806-7612-11

Revision A

Adobe PostScript

030608@5943

目次

はじめに

このマニュアルは、Solaris™ オペレーティング環境のコマンド行インタフェース (CLI) を使用してさまざまなシステム作業を行う方法について説明しています。

注 – Solaris オペレーティング環境は、2 種類のプラットフォーム、SPARC™ と x86 で動作します。このマニュアルで説明する情報は、章、節、注、箇条書き、図、表、例、またはコード例において特に明記しない限り、両方のプラットフォームに該当します。

対象読者

このマニュアルは Solaris 9 オペレーティング環境のユーザを対象としており、コマンド行インタフェースの使用方法を調べるために利用できます。

お読みになる前に

システムがインストール済みで、使用可能な状態になっている必要があります。システムに Solaris オペレーティング環境をインストールしていない場合は、先にシステムに付属のインストールマニュアルを参照してください。

内容の紹介

このマニュアルの内容は以下のとおりです。

- 第 1 章 では、Solaris オペレーティング環境でサポートされるコマンド行インタフェースと 2 つのグラフィカルユーザインタフェース (共通デスクトップ環境 (CDE) および GNOME) について概要を述べています。
- 第 2 章 では、システムのログインとログアウトの方法、および基本的なコマンドによるシステム作業の方法について説明しています。

  Solaris オペレーティング環境がデフォルト以外の場所にインストールされている場合は、インストールされているファイルにアクセスできるように既定のパスを変更してください。
- 第 3 章 では、ファイルとディレクトリの作成、コピー、移動、削除について説明しています。この章では、ファイルの検索、ファイルの相違比較、ファイルおよびディレクトリのアクセス権の設定などについても説明しています。
- 第 4 章 では、`grep` コマンドを使用してファイル内で特定の文字列を検索する方法について説明しています。
- 第 5 章 では、システムのプロセスとディスク使用を監視 / 管理する方法について説明しています。
- 第 6 章 では、`vi` テキストエディタを使用してテキストファイルの作成、編集、保存、印刷を行う方法について説明しています。
- 第 7 章 では、`mailx` ツールを使用して電子メールの表示、作成、送信、印刷を行う方法について説明しています。
- 第 8 章 では、ファイルの印刷とプリンタステータスの管理について説明しています。
- 第 9 章 では、リモートマシンからシステム作業を行う方法について説明しています。
- 第 10 章 では、システム初期化ファイルおよび環境変数を変更して作業環境をカスタマイズする方法について説明しています。
- 付録 A では、キーボードのキーマッピングをカスタマイズする方法について説明します。

関連マニュアル

Solaris オペレーティング環境には、オンラインで利用できる Solaris ソフトウェアに関するマニュアルが多数用意されています。これらのマニュアルの分類は次のとおりです。

- 「*Solaris 9 System Administrator Collection - Japanese*」

  さまざまなシステム構成 (Sun のワークステーションを使用した大規模ネットワークなど) のインストールとシステム管理に関する詳しい情報を提供します。

- 「*Solaris 9 Software Developer Collection - Japanese* 」

  ソフトウェア開発者がシステム上のプログラムの作成、デバッグ、維持管理などを行うために必要な情報を提供します。

- 「*Solaris 9 Reference Manual Collection - Japanese* 」

  SunOS の全コマンドについて解説します。一般にマニュアルページと呼ばれるこれらの解説は、オンラインマニュアルとしてインストールすることもできます。

- 「*Solaris 9 User Collection - Japanese* 」

  Solaris オペレーティング環境についての各種の情報を掲載しています。主なものを以下に示します。

  - SunOS™ コマンドの使用方法
  - ウィンドウ環境での作業方法
  - 作業環境のカスタマイズ方法
  - シェルスクリプトの作成方法
  - 電子メールの使用方法
  - ネットワーク上での作業方法

Sun のオンラインマニュアル

`docs.sun.com` では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、`http://docs.sun.com` です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェル

  machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

  machine_name# **command y|n** [*filename*]

- Bourne シェルおよび Korn シェル

  $ **command y|n** [*filename*]

- Bourne シェルおよび Korn シェルのスーパーユーザー

  # **command y|n** [*filename*]

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、「x86」という用語は、Intel 32 ビット系列のマイクロプロセッサチップ、および AMD が提供する互換マイクロプロセッサチップを意味します。

第 1 章

コマンド行インタフェースとグラフィカルユーザインタフェースの違い

これらのインタフェースを利用するためには、自分のハードディスクまたは利用できるネットワークサーバにデスクトップソフトウェアがあらかじめインストールされている必要があります。デスクトップソフトウェアにアクセスできるか不明な場合は、システム管理者に尋ねるか、あるいは使用しているプラットフォームのインストールマニュアルを参照してください。

この章では、コマンド行インタフェースとデスクトップ環境の違いについて簡単に説明しています。

コマンド行インタフェース

コマンド行インタフェース (CLI) を利用すると、端末またはコンソールのウィンドウにコマンドを入力してオペレーティングシステムと対話方式で処理を行えます。コマンド行インタフェースでは、表示されるプロンプトに対して指定行にコマンドを入力するという方法で応答し、システムの返答を受け取ります。ユーザは、実施したい作業ごとに 1 つ以上のコマンドを入力する必要があります。

このマニュアルでは、コマンド行インタフェースを使用して各種のシステム作業を行う方法について説明しています。

グラフィカルユーザインタフェース

グラフィカルユーザインタフェース (GUI) は、プログラムにおいてキーボードやマウスに加えてグラフィックスを使用して操作を簡単に行えるようにしたインタフェースです。GUI は、オペレーティングシステムまたはアプリケーションとの情報伝達が可能なように、ウィンドウ、プルダウンメニュー、ボタン、スクロールバー、アイコンイメージ、ウィザード、各種アイコン、マウスなどを提供します。

Solaris 9 オペレーティング環境は、2 つの GUI、共通デスクトップ環境 (common Desktop Environment、CDE) と GNOME デスクトップをサポートします。

共通デスクトップ環境

共通デスクトップ環境 (CDE) には、作業を体系的に管理するのに便利なウィンドウ、ワークスペース、コントロール、メニュー、フロントパネルなどがあります。CDE GUI は、ファイルとディレクトリの整理、電子メールの読み書きや送信、ファイルのアクセス、システムの管理などに使用できます。

詳細は、『*Solaris* 共通デスクトップ環境 ユーザーズ・ガイド』を参照してください。

GNOME デスクトップ

GNOME (GNU Network Object Model Environment) は GUI であり、コンピュータデスクトップアプリケーションの集合体です。GNOME のデスクトップ、パネル、アプリケーション、およびツールセットは、作業環境のカスタマイズやシステム作業の管理などに利用できます。GNOME には、ワードプロセッサ、スプレッドシートプログラム、データベースマネージャ、プレゼンテーションツール、Web ブラウザ、電子メールプログラムといった各種のアプリケーションもあります。

第 2 章

ログインと基本的な SunOS コマンドの使用

この章の内容は次のとおりです。

コマンドの入力には、端末かウィンドウを使用してください。端末またはウィンドウの起動方法を知りたい場合は、使用しているデスクトップ環境のマニュアルを参照してください。

ログインの方法

一般に作業セッションとは、ユーザがシステムにログインしてからログアウトするまでを指します。SunOS のマルチユーザ環境では、システムを利用するたびに識別名を入力することが必要です。システムがユーザを識別し、またそのユーザをほかのユーザと区別するための識別名として、ログイン名 (ユーザ名またはアカウントとも呼ばれる) が使われます。パスワードは、それを知っているユーザだけがアカウントを利用できるように制限するものです。ログイン名とパスワードをまだ持っていない場合は、システム管理者に問い合わせてアカウントを設定してもらってください。ログイン名とパスワードが設定されれば、ログインできる状態になります。

システムにログインするときは、画面に次のような文字が表示されていなければなりません。

hostname `console login:`

システム管理者によって割り当てられたログイン名を入力して Return キーを押します。たとえば、ログイン名が spanky の場合は次のように入力します。

hostname console login: **spanky**

それから、Return キーを押します。次に、パスワードを入力するよう要求されます。

hostname console login: **spanky**

Password:

プロンプトに対してパスワードを入力して Return キーを押します。(アカウントにまだパスワードが割り当てられていない場合は、パスワードを入力しなくてもログインできます。)入力したパスワードが画面に表示 (エコー) されることはありません。これは、ほかのユーザにパスワードを知られないようにするためです。

ログインシェル

2 章以降では、SunOS のコマンドを使用します。システムに対してコマンドを入力すると、実際にはシェルと呼ばれるコマンド・インタプリタプログラムに情報を提供することになります。シェルプログラムは提供された情報を読み取り、適切なアクションをシステム内部で実行させます。

SunOS のデフォルトシェルは Bourne シェルですが、Solaris オペレーティング環境は次のシェルもサポートしています。

- GNU Bourne Again シェル (bash)
- C シェル (csh)
- Korn シェル (ksh)
- TC シェル (tcsh)
- Z シェル (zsh)

これらのシェルには、それぞれ独自の機能があります。

注 – man コマンドで各シェルプログラムを含む SunOS の各コマンドのマニュアルページを表示できます。マニュアルページについての詳細は、29 ページの「man コマンドによるマニュアルページの表示」を参照してください。

初めてシステムにログインしたとき (あるいは端末またはウィンドウを新たに開いたとき) にコマンドプロンプトが表示された場合は、シェルプログラムが自動的に起動されたことを意味します。このシェルは、ログインシェルと呼ばれます。システム管理者が別のシェルを指定している場合は、ログインシェルは SunOS のデフォルト (Bourne シェル) になりません。

あるシェルで利用できるコマンドや手続きがほかのシェルでは利用できない場合もあります。特に明記されていないかぎり、このマニュアルで説明されているコマンドと手続きはすべて Bourne シェルで利用できます。

ログアウトの方法

作業セッションを完了してオペレーティングシステムを終了する準備ができたら、次のように入力してログアウトします。

```
$ exit
```

しばらくすると、再びログインプロンプトが表示されます。

```
$ exit

hostname console login:
```

ログインプロンプトが表示された場合は、正常にログアウトできたことを意味します。これで任意のユーザが続けてログインできる状態になります。

注 – SunOS オペレーティングシステムでは、ワークステーションや端末の電源を切ってもログアウトできるとは限りません。明示的にログアウトしない場合、システムにログインしたままになることがあります。

ショートカットキー

操作の中には、特定のキーの組み合わせ (キーボードアクセラレータと呼ぶ) を使用して操作をスピードアップできるものが多数あります。この機能を利用すると、マウスやメニューと同等の操作を行なったり、あらかじめ定義されているキーボードのキー操作を行なったりできます。

注 – メタキーは SPARC キーボード上では <> キーを意味し、x86 キーボード上では Ctrl-Alt キーを押すことを意味します。

キーボードアクセラレータを使用するには、最初のキー (メタキー、または Ctrl-Alt キー) を押しながら 2 番目のキーを押します。たとえば、選択したテキストをカットするには、SPARC システムではメタキーを押しながら *x* キーを押します。x86 システムでは Ctrl キーと Alt キーを一緒に押しながら *x* キーを押します。

コマンドプロンプト

ログイン直後には、画面またはウィンドウに初期プロンプトが表示されます。このプロンプトの形状は、使っているシェルの種類やシステム管理者による初期設定によって異なります。SunOS で使われるデフォルトのコマンドプロンプトはドル記号 ($) なので、このマニュアルで紹介する大部分の例ではこの `$` プロンプトを使います。

コマンドプロンプトを変更する場合は、142 ページの「コマンドプロンプトの変更」を参照してください。

コマンドの入力

コマンドプロンプトが表示されているときは、システムはコマンドの入力待ちの状態にあります。次の例に示すように、コマンドプロンプトに対して `date` というコマンドを入力してみてください (`date` と入力して Return キーを押します)。

```
$ date

Mon Sep 17 10:12:51 PST 2001

$
```

このコマンドは、現在の日時を表示します。同じコマンドの先頭の文字を大文字で入力すると、次のメッセージが表示されます。

```
$ Date

Date: Command not found.

$
```

Solaris オペレーティング環境は大文字の `D` を小文字の `d` とは異なるものとして解釈するため、`Date` コマンドは失敗します。Solaris オペレーティング環境のほとんどのコマンドは小文字です。

誤った入力の訂正

ユーザが入力したコマンドは、Return キーを押すまではシステムに送信されません。コマンドを間違って入力しても、Return キーをまだ押していなければ次の方法で入力を訂正できます。

- Delete キーか Back Space キーを押して間違った箇所まで戻ります。
- Ctrl-U を使用して行全体を消去し、最初から入力し直します (Control キーを押したまま u キーを押す)。

これら 2 つの方法を試して動作を確認してください (Delete キーと Backspace キーの動作はシステムによって異なる場合があります。Ctrl-U はほとんどのシステムで同様に動作します)。

複数のコマンドと長いコマンドの入力

1 行に複数のコマンドを入力することもできます。次の date コマンドと logname コマンドの例に示すように、コマンドの間にセミコロン (;) を挿入してください。

```
$ date; logname

Tue Oct 31 15:16:00 MST 2000

spooky
```

このコマンド入力では、現在の日付と時間 (date コマンドの出力) およびシステムに現在ログインしているユーザのログイン名 (logname コマンドの出力) が続けて表示されます。

長いコマンドを入力する場合は、バックスラッシュ文字 (\) を使うことによって次の行に続けて入力できます。次に例を示します。

```
$ date; \

logname

Tue Oct 31 15:17:30 MST 2000

spooky
```

date と logname は長いコマンドではありませんが、この例では複数のコマンドを次の行に続けるという概念を示すためにこれらのコマンドを使用しています。入力するコマンドが画面の幅よりも実際に長いときは、バックスラッシュ文字を使うと便利です。

注 – デスクトップウィンドウを使用する場合は、次の行にコマンドを続けて入力するためにバックスラッシュ文字を使う必要はありません。入力中のコマンドが行の終わりにくると、次の行に自動的に改行され、Return キーを押したときにそれらの全コマンドが実行されます。

前回のコマンドを繰り返す

Korn、Bourne Again、C、TC、および Z シェルには、入力されたコマンドの履歴を保持し、以前に入力されたコマンドを繰り返すことができるようにする機能があります。

注 – Bourne シェル (sh) では、history コマンドは使用できません。

Bourne Again、C、TC、Z シェルでコマンドを繰り返す

Bourne Again、C、TC、または Z シェルを使用している場合は、!! と入力して Return キーを押すことにより最後に入力したコマンドを繰り返し実行することができます。

```
example%!!

date

Tue Oct 31 15:18:38 MST 2000

example%
```

!*x* と入力すると、以前に入力した任意のコマンドを繰り返し実行できます。*x* は、繰り返すコマンドに対応する履歴リスト上のコマンド番号です。履歴リストを参照するには、history と入力して Return キーを押します。次の例に示すようなリストが表示されます。

```
example% history

1  pwd

2  clear

3  ls -l

4  cd $HOME

5  logname

6  date

7  history
```

注 – Z シェルは、履歴リストに history コマンドを表示しません。

また、! のあとに負の番号を入力しても履歴リスト上のコマンドを繰り返すことができます。たとえば、履歴リスト上の最後のコマンドから数えて 2 番目のコマンドを実行するには、次のコマンドを入力します。

```
example% !-2

date

Tue Oct 31 15:20:41 MST 2000

example%
```

注 – Z シェル内で history コマンドの直後にこのコマンドを繰り返し実行するには、! のあとの負の数を 1 つ増やしてください (!-3)。

上記の履歴リストの例では、date コマンドが繰り返し実行されます。

! の後に以前に入力したコマンドの先頭の数文字を入力してもコマンドを再実行できます。たとえば、以前に clear コマンドを入力して画面をクリアした場合は、!cl と入力すれば画面を再度クリアできます。ただし、この方法によるコマンドの繰り返しでは、繰り返したいコマンドを履歴リスト上で一意に識別できる文字数を指定しなければなりません。! のあとに 1 文字しか指定しなかった場合は、その文字で始まるコマンドのうちで最後に入力したものが繰り返されます。

Korn シェルでのコマンドの繰り返し

Korn シェルを使用する場合、次のコマンドを使用して以前のコマンドを繰り返し実行します。

```
$ fc -s -

date

Tue Oct 31 15:18:38 MST 2000

$
```

fc -s *x* と入力すると、以前に入力した任意のコマンドを繰り返し実行できます。*x* は、繰り返すコマンドに対応する履歴リスト上のコマンド番号です。履歴リストを参照するには、fc -l と入力して Return キーを押します。履歴リストの使用例を次に示します。

```
$ fc -l

344  pwd

345  clear

346  ls -l
```

```
347  cd $HOME

348  logname

349  date

350  history

$
```

履歴リストからコマンドを繰り返し実行するには、fc -s コマンドにマイナスの番号を付けて実行します。たとえば、履歴リスト上の最後のコマンドから数えて 2 番目のコマンドを実行するには、次のコマンドを入力します。

```
$ fc -s -2

date

Tue Oct 31 15:20:41 MST 2000

$
```

上記の履歴例では、date コマンドが繰り返し実行されます。

fc -s コマンドに以前のコマンドの最初の 2、3 の文字を付けることによっても実行できます。たとえば、以前に date コマンドを使用して現在の日時を表示した場合、fc -s da と入力すると日時を再度表示できます。ただし、履歴リスト内でコマンドを特定するのに十分な文字数を入力する必要があります。fc -s のあとに 1 文字しか指定しなかった場合は、その文字で始まるコマンドのうちで最後に入力したものが繰り返されます。

コマンドオプションの指定

多くのコマンドにはオプションがあり、そのコマンドに関連する特別機能を実行できます。たとえば、date コマンドには、ローカル時間ではなくグリニッジ標準時間で日付を表す -u というオプションがあります。

```
$ date -u

Tue Oct 31 22:33:16 GMT 2000

$
```

大部分のオプションは、1 文字の前にダッシュ (-) を付けたものです。オプションを持たないコマンドもあれば、複数のオプションを持つコマンドもあります。コマンドに対して引数を持たないオプションを複数指定する場合は、それらのオプションを個別に入力しても (-a -b) まとめて入力しても (-ab) かまいません。

コマンド出力のリダイレクトとパイプ

特に指定しない限り、コマンドの出力結果は画面上に表示されます。特別な記号を使って、コマンドの出力をリダイレクト (出力先を変更) できます。たとえば、コマンド出力を画面に表示せずにファイルに保存できます。次の例は、リダイレクト記号 (>) の使い方を示しています。

```
$ date> sample.file

$
```

この例では、date コマンドからの出力が sample.file という新規ファイルにリダイレクトされています。more を入力すると、 sample.file の内容を表示できます。

```
$ more sample.file

Tue Oct 31 15:34:45 MST 2000

$
```

この例に示すように、sample.file には date コマンドからの出力結果が入っています。more コマンドについての詳細は、第 3 章を参照してください。

あるコマンドの出力を別のコマンドの入力としてリダイレクトすることもできます。このような方法で複数のコマンドを連結したものをパイプラインと呼びます。パイプラインは、パイプと呼ばれる垂直バー (|) を使って構成します。

たとえば、コマンドの出力結果をファイルに保存する代わりに、プリントコマンド (lp) の入力としてリダイレクトする場合、パイプ記号 (|) を使えます。date コマンドの出力を直接プリンタに送るには、次のように入力します。

```
$ date | lp

request id is jetprint-46 (1 file)

$
```

このパイプラインにより、date コマンドの結果が印刷されます。lp コマンドを使ったファイルの印刷方法については、第 8 章の 111 ページの「デフォルトプリンタへの印刷要求の実行」を参照してください。

ここで紹介したコマンドのリダイレクト例は非常に簡単なものですが、さらに高度なコマンドを学習するに従って、リダイレクトとパイプのさまざまな用途が理解できます。

バックグラウンドでコマンドを実行する

コマンドを入力して Return キーを押すと、システムはそのコマンドを実行し、コマンドがタスクを完了するのを待ち、その後別のコマンドを入力するようプロンプトを表示します。しかし、中には完了するまでにかなりの時間がかかるコマンドもあるため、その間に別のコマンドを実行する方が効率的です。前のコマンドを実行している間、別のコマンドを実行するには、バックグラウンドで実行できます。

バックグラウンドでコマンドを実行するには、次の例のようにアンパサンド (&) をコマンドの後に入力します。

```
$ bigjob &

[1] 7493

$
```

その下に表示されている番号は、プロセス ID です。この例では、コマンド `bigjob` はバックグラウンドで実行され、他のコマンドの入力を続行できます。バックグラウンドのジョブが終了すると、次にコマンド (この例では `date`) を入力したときに下記のようなメッセージが表示されます。

```
$ date

Tue Oct 31 15:44:59 MST 2000

[1]    Done                  bigjob

$
```

バックグラウンドのジョブが完了する前にログアウトする可能性がある場合は、下記の例に示すように、`nohup` コマンド (no hangup) を使ってジョブを起動すると、ログアウトしてもそのジョブを完結させることができます。`nohup` コマンドを使用しないと、ログアウトする時点でバックグラウンドのジョブは終了してしまいます。

```
$ nohup bigjob &

[3] 7495

$
```

パスワードの使い方

システムのセキュリティ保護のため、Solaris オペレーティング環境ではパスワードを使用してシステムにアクセスする必要があります。1 年に何度かパスワードを変更するとより安全です。

注 – 自分のアカウントを許可なく使ったユーザがいるようであれば、即座にパスワードを変更してください。

パスワードは次の点を考慮して選択します。

- 書き留めなくても覚えられるパスワードを選択する。覚えられないようなパスワードは、簡単に推測できるパスワードよりも不適切です。
- 6 文字以上で、そのうち 1 文字以上が数字からなるパスワードを選択する。
- 自分の名前や頭文字、または配偶者の名前や頭文字は使わない。
- ペットの名前や一般的な趣味に関係する名前は使わない。
- 大文字だけからなるパスワードは使わない。
- 複数のアカウントを持っている場合は、各アカウントごとにパスワードを変える。
- パスワード内に Ctrl-C、Ctrl-Z、Ctrl-U、Ctrl-S、Esc、Tab、#、および @ は使用しない。端末によっては、これらの文字が通常のテキスト文字ではなくシグナルとして解釈されるため、パスワードを正しく入力できなくなる場合があります。

パスワードの変更方法

各ユーザのパスワードを変更するには、passwd コマンドを使います。

```
$ passwd

passwd:  Changing password for user2

Enter login password:

New password:

Re-enter new password:

passwd (SYSTEM): passwd successfully changed for user2

$
```

1. **Enter login password:** というプロンプトが表示されたら、現在のパスワードを入力します。

   現在アカウントにパスワードが割り当てられていない場合 は、Old Password: プロンプトは表示されません。

   パスワードは画面にエコー (表示) されないため、ほかの人にパスワードを知られなくてすみます。

2. **New Password:** というプロンプトが表示されたら、新しいパスワードを入力します。

   このときも、入力したパスワードは画面上に表示されません。

3. 最後に **Retype new password:** というプロンプトに対して、新しいパスワードをもう一度入力します。

   これは、意図したパスワードを入力したかを確認するためです。

   前のプロンプトで入力したパスワードのとおり入力しないと、パスワード変更は拒否され、次のメッセージが表示されます。

   ```
   passwd: They don't match; try again.
   ```

   このメッセージが繰り返し表示される場合は、システム管理者に連絡して新しいパスワードを設定してください。

注 – 6 文字未満のパスワードは使えません。また、新しいパスワードは古いパスワードと比較して 3 文字以上は異なっていなければなりません。

パスワードの有効期限を設定する

パスワードの有効期限を設定 (passwd コマンドのオプションを使うと設定できる) している場合、パスワードには最長有効期限のみ、または最長有効期限と最短有効期限の両方を持つものがあります。パスワードの有効期限は、システム管理者が設定します。

パスワードが有効期限に達すると、ログイン時にパスワードの変更を求める次のメッセージが表示されます。

```
Your password has expired. Choose a new one.
```

次に、自動的に passwd プログラムが実行され、新しいパスワードを入力するよう要求されます。

また、たとえば、パスワードの最短有効期限が 2 週間と設定されている場合、その最短期間が経過する前にパスワードを変更しようとすると、次のメッセージが表示されます。

```
Sorry, less than 2 weeks since the last change.
```

passwd(1) と、パスワードの有効期限の設定についての詳細は、『*man pages section 1: User Commands*』を参照してください。

OS コマンドによるヘルプ情報の表示

この節では、さまざまなオンラインヘルプ機能について説明します。オンラインヘルプ機能を使うことによって、ワークステーションや端末からリファレンス情報を読むことができます。

man コマンドによるマニュアルページの表示

コマンドの名前は知っていてもその機能について不明な場合は、man コマンドを使うと便利です。man コマンドの詳細機能を表示するには、次のように入力します。

```
$ man man
```

このコマンドは、SunOS マニュアルページ man(1) の最初の画面をウィンドウの表示領域に出力します。次の画面を表示するにはスペースバーを押します。終了してコマンドプロンプトに戻るには q キーを押します。マニュアルページは、利用できる全オプションと適切なコマンド構文を記述しています。マニュアルページによっては、コマンドのさまざまな用途を説明した例もあります。

whatis コマンドによる 1 行要約の表示

コマンド機能の 1 行要約だけを参照する場合は、次に示すような whatis コマンドを使います。

```
$ whatis date

date (1)           -display or set the date

$
```

上記の例で、コマンド名のあとの括弧内に数字がありますが、この数字はこのコマンドが属するリファレンスマニュアルセクションを示します。コマンドは、その機能に応じてさまざまなセクションに分類されています。大部分のユーザコマンドは、セクション 1 にあります。共通の表記法により、セクション番号はコマンド名のあとの括弧内に表示されます。紙のマニュアルページでは、同じセクション内のコマンドはアルファベット順に並んでいます。

注 – whatis コマンドを利用できるのは、システム管理者がコマンド説明を集めた特殊なデータベースを設定している場合だけです。

---

apropos によるキーワードの検索

実行したい機能はわかっていても、どのコマンドを使えばよいかわからない場合は、apropos コマンドでキーワード検索によってコマンドを見つけることができます。apropos を実行すると、入力したキーワードを要約行に含むコマンドが、すべて一覧表示されます。キーワードが多数の場所に現れる場合は、apropos の出力が非常に長くなることもあります。

注 – apropos コマンドを利用できるのは、システム管理者がコマンド説明を集めた特殊なデータベースを設定している場合だけです。

---

apropos の出力例を確認するには、次のコマンドを入力してください。

- `apropos who`
- `apropos execute`
- `apropos apropos`

入力するキーワードによって表示が非常に長くなる場合は、Ctrl-C を押せば (Ctrl キーを押したままで c キーを押す)、表示が中止されコマンドプロンプトに戻ります。

第 3 章

ファイルとディレクトリの操作

ファイルとディレクトリの作業、編成、管理は、SunOS コマンドを使って行えます。特定の操作を行うには、ファイルやディレクトリを指定して SunOS コマンドを実行します。コマンド行の操作は、デスクトップのファイルマネージャとは異なります。ファイルマネージャでは、ファイルはマウスクリックや移動が可能なアイコンとして表示され、コマンドはメニューから選択できます。

この章の内容は次のとおりです。

ファイルの概念

ファイルは、SunOS オペレーティングシステムの基本的な単位です。次に示す要素を始めとして、ほとんどあらゆるものがファイルとして扱われます。

- ドキュメント – これには、手紙やレポートのようなテキストファイル、コンピュータのソースコードなど、ユーザが書き込みや保存を行うあらゆるドキュメントが含まれます。
- コマンド — ほとんどのコマンドは実行可能ファイル (特定のプログラムを起動するために実行されるファイル) です。たとえば、第 2 章で説明した `date` コマンドは、現在の日付を出力するプログラムを起動するための実行可能ファイルです。
- デバイス — 端末、プリンタ、ディスクドライブなどは、すべてファイルとして取り扱われます。
- ディレクトリ – ディレクトリは、ほかのファイルを格納するためのファイルです。

次の節では、ファイルの作成、一覧表示、コピー、移動、削除に利用できるコマンドについて説明します。また、ファイル内容を表示する方法やファイルの特性を確認する方法などについても述べています。

ファイル操作コマンドの使い方

この節の各コマンド説明には、その使用例が示されています。本文を読みながら、これらの例を試してください。

始める前に

ファイル操作を行う前に、ホームディレクトリにいることを確認してください。ホームディレクトリは、ユーザアカウントの作成時にシステム管理者によって作成されます。ほかのユーザに影響を与える変更を避けるには、自分のホームディレクトリ内で以下の作業を実行してください。

ホームディレクトリにいることを確認するため、cd (change directory) コマンドを入力します。このコマンドを実行すると、自分のホーム (デフォルト) ディレクトリに移動します。次に pwd (print working directory) コマンドを入力して、ファイルシステム内での現在位置を表示します。このとき表示されるディレクトリが各ユーザのホームディレクトリです。

```
$ cd

$ pwd

/export/home/username
```

上記の例では、ユーザのホームディレクトリは /export/home/*username* です。*username* は、このホームディレクトリを所有しているユーザの名前です。

テスト用ファイルの作成

空のファイルを作成するには、touch コマンドを使います。

```
$ touch tempfile

$
```

名前を指定したファイルが存在しない場合、touch コマンドが空のファイルを作成します。

注 – ファイルが存在する場合は、最終アクセス時間が更新されます。

ファイルの一覧表示 (ls)

ファイルが作成されたことを確認するために、ls コマンドを使ってファイル名を表示します。

```
$ ls tempfile

tempfile
```

ls コマンドを引数なしで入力すると、現在のディレクトリ内のファイルの一覧が表示されます。特定のファイル名を指定して ls コマンドを入力すると、そのファイルが存在する場合は指定されたファイルだけが表示されます。

ファイル表示の詳細は、ls(1) のマニュアルページを参照してください。

ファイルのコピー (cp)

tempfile を copyfile というファイルにコピーするには、cp コマンドを使います。

```
$ cp tempfile copyfile

$
```

ここで両方のファイルの一覧を表示してみましょう。どちらのファイル名も「file」という文字列で終わることに注目してください。任意の 1 文字または文字列を表すには、ワイルドカード文字のアスタリスク (*) が使えます。したがって、ls *file というコマンドを入力すると、tempfile と copyfile の両方が表示されます。この場合、現在のディレクトリ内にほかに file で終わる名前のファイルがあれば、そのファイルも表示されます。

```
$ ls *file

copyfile    tempfile
```

copyfile が最初に表示されることに注意してください。ファイルはアルファベット順に表示されます。ただし、大文字と数字は小文字より先に表示されます。

ファイルのコピーについての詳細は、cp(1) のマニュアルページを参照してください。

ファイルの移動とファイル名の変更 (mv)

mv (move) コマンドを使って、ファイルの移動とファイル名の変更が行えます。次の例では、mv コマンドを使ってファイル名を tempfile から emptyfile に変更しています。

```
$ mv tempfile emptyfile

$
```

上記の変更を確認するために、両方のファイルのリストを再表示します。

```
$ ls *file

copyfile    emptyfile
```

tempfile が emptyfile で置き換えられます。

ファイルの移動と名前変更の詳細は、mv(1) のマニュアルページを参照してください。

ファイルの削除 (rm)

rm (remove) コマンドを使って copyfile を削除し、ls コマンドでその結果を確認します。

```
$ rm copyfile

$ ls *file

emptyfile
```

注意 – rm コマンドの使用には注意を払ってください。特に、ワイルドカード文字 (*) を指定してこのコマンドを使用する場合は細心の注意を払う必要があります。rm を使用して削除したファイルの復元は行えません。

rm(1) コマンドについての詳細は、『*man pages section 1: User Commands*』を参照してください。

ファイル内容の表示 (more、cat)

ファイルの内容を表示するには、more コマンドを使います。more のあとに、表示するファイル名を指定します。ファイルの内容は、画面に表示されます。ファイルが 1 画面より長い場合は、次のメッセージが表示されます。

```
--More--(nn%)
```

nn はすでに表示されたファイルの割合です。

ファイル内容の表示には cat (concatenate) コマンドも使えますが、このコマンドでは連続してすばやくファイル内容が表示されます。cat コマンドは、ファイルの表示よりも 2 つ以上のファイルを 1 つの大きいファイルに連結する場合によく使われます。ファイルの連結例を以下に示します。

```
$ cat file1 file2 file3> bigfile

$ ls *file

bigfile

file1

file2

file3

$
```

more(1) コマンドと cat(1) コマンドに ついての詳細は、『*man pages section 1:User Commands*』を参照してください。

ファイル形式の表示 (file)

ファイルによっては、バイナリファイルや実行可能ファイルのように、プリンタに印刷したり画面上に表示したりできないものがあります。file コマンドは、ファイル形式の表示に使います。

```
$ file *

myscript:      executable shell script

print.ps:      PostScript document

save.txt:      ascii text
```

ディレクトリとディレクトリ階層

この節では、ファイルの管理や整理のために Solaris オペレーティング環境で使用するディレクトリ階層について説明します。

ディレクトリ階層

ファイルはディレクトリにまとめられますが、ディレクトリも階層構造になっています。この階層の最上位は「ルート」ディレクトリと呼ばれ、「/」で表されます。

次の図 3–1 の例では、ファイルシステム内のディレクトリごとに多数のサブディレクトリがあります。文字「/」は、ディレクトリのレベルを識別するために使用されます。/ (ルート) ディレクトリには、/usr、/bin、/export/home、/lib というサブディレクトリなどがあります。さらに /export/home サブディレクトリには、user1、user2、user3 などのサブディレクトリがあります。

コマンドを実行するには、その対象となるファイルまたはディレクトリを含むディレクトリを指定する必要があります。指定されるディレクトリの下に存在するディレクトリとファイルの名前は、区切り文字であるスラッシュで結合されてパス名を形成します。たとえば、次の図の user3 ディレクトリのパス名は /export/home/user3 です。

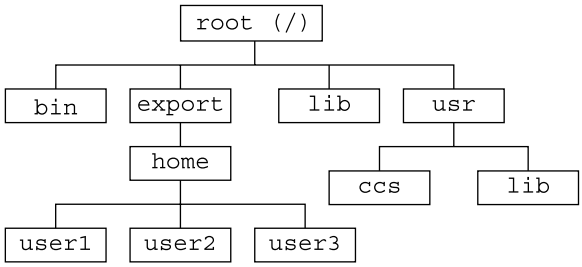

図 **3–1** ファイルシステム階層

同一ディレクトリ内のすべてのサブディレクトリとファイルの名前は一意でなければなりませんが、ディレクトリが異なればこれらの名前が同じであってもかまいません。たとえば、/usr ディレクトリには /usr/lib というサブディレクトリがあります。/usr/lib と /lib ではパス名が異なるため、これらの名前に重複はありません。

ファイルのパス名の機能は、ディレクトリのパス名とまったく同じです。ファイルのパス名は、ファイルシステム階層におけるそのファイルの位置を表します。たとえば、/export/home/user2 ディレクトリに report5 というファイルが存在する場合、このファイルのパス名は /export/home/user2/report5 となります。このパス名は、ファイル report5 はディレクトリ user2 の中にあり、user2 はディレクトリ home の中にあり、home はルート (/) ディレクトリの中にあることを示しています。

作業用ディレクトリの表示 (pwd)

pwd (print working directory) コマンドは、ファイルシステム階層内でのユーザの現在位置を表示します。

```
$ pwd

/home/user1
```

実際の出力は、ディレクトリ構造の違いのためにこの例とは異なることがあります。作業用ディレクトリが、ファイルシステム階層内でのユーザの現在位置となります。

ユーザのホームディレクトリ

各ユーザはホームディレクトリを持っています。端末またはウィンドウを始めて開いたときには、最初の位置がホームディレクトリになります。ホームディレクトリは、ユーザアカウントの作成時にシステム管理者によって作成されます。

作業用ディレクトリの変更 (cd)

cd (change directory) コマンドを使うと、ファイルシステム階層内を自由に移動できます。

```
$ cd /usr/lib

$ pwd

/usr/lib
```

cd コマンドを引数なしで入力すると、自分のホームディレクトリに戻ります。たとえば、ホームディレクトリが /export/home/user1 の場合は、次のようになります。

```
$ cd

$ pwd

/home/user1
```

ホームディレクトリは /export/home/user1 ですが、cd コマンドはユーザを /home/user1 に返します。これは、ホームディレクトリがオートマウンタによって /home ディレクトリにマウントされているためです。

Bourne Again、C、Korn、TC、および Z シェルでは、ホームディレクトリを示す短縮名としてチルド文字 (~) が使用されます。たとえば、次のように入力するとホームディレクトリ内の music サブディレクトリに移動できます。

```
example% cd ~/music
```

この短縮名を使ってほかのユーザのホームディレクトリを指定することもできます。次に例を示します。

```
example% cd ~username
```

username はほかのユーザのログイン名です。このコマンドによって、そのユーザのホームディレクトリに移動します。

注 – Bourne シェルを使用している場合は、この短縮名「~」は機能しません。

Bourne シェルを使用している場合は、$home と入力してホームディレクトリを指定できるようにシステム管理者がすでに設定している場合があります。その場合は、次のコマンドを入力すると作業ディレクトリをホームディレクトリ内の music サブディレクトリに移動できます。

```
$ cd $home/music
```

サブディレクトリのすぐ「上」のディレクトリは、親ディレクトリと呼ばれます。前記の例では、/home が /home/user1 の親ディレクトリです。親ディレクトリは、シンボル .. (ピリオドを 2 つ) で表します。したがって、cd .. というコマンドを入力すると、次の例で示すように、作業用ディレクトリが親ディレクトリに変わります。

```
$ pwd

/home/user1

$ cd ..

$ pwd

/home
```

現在の作業ディレクトリが /home/user1 のときに、/home/user2 にあるファイルの処理を行いたいという場合は、次のコマンドを入力します。

```
$ pwd

/home/user1

$ cd ../user2

$ pwd

/home/user2
```

../user2 は、user2 の親ディレクトリを経由するようシステムに指示します。この例に示すように、絶対パス名 /home/user2 を入力するより簡単です。

ディレクトリの作成 (mkdir)

新しいディレクトリを作成するには、mkdir コマンドのあとに新しいディレクトリの名前を指定します。

```
$ mkdir veggies

$ cd veggies
```

```
$ mkdir broccoli

$ cd broccoli

$ pwd

/home/user2/veggies/broccoli
```

相対パス名

ファイルやディレクトリの絶対パス名はスラッシュ (/) で始まり、そのファイルまたはディレクトリとルートディレクトリとの間のディレクトリ構造全体から見た関係を表します。ただし、現在の作業用ディレクトリに対するファイルやディレクトリの相対関係を表す省略名 (相対パス名) を使うこともできます。

親ディレクトリにいる場合は、絶対パス名ではなくディレクトリ名だけを指定すればサブディレクトリに移動できます。前記の例では、cd veggies というコマンドで veggies ディレクトリの相対パス名が使われています。現在の作業用ディレクトリが /home/user2 の場合、このディレクトリの絶対パス名は /home/user2/veggies となります。

別個のサブディレクトリをいくつか作成し、そのディレクトリ構造の中を移動してみてください。移動の際には絶対パス名と相対パス名の両方を使ってみてください。現在位置は pwd コマンドで確認します。

ディレクトリの移動とディレクトリ名の変更

ディレクトリを別の名前に置き換えることによって、ディレクトリ名を変更できます。ディレクトリ名の変更には mv コマンドを使います。

```
$ pwd

/home/user2/veggies

$ ls

broccoli

$ mv broccoli carrots

$ ls

carrots
```

mv コマンドを使って、ディレクトリをすでにある別のディレクトリ内に移動することもできます。

```
$ pwd
```

```
/home/user2/veggies

$ ls

carrots

$ mv carrots ../veggies2

$ ls ../veggies2

carrots
```

この例では、mv コマンドを使って、carrots ディレクトリを veggies から veggies2 に移動しています。

ディレクトリのコピー

ディレクトリとその下にあるファイルをコピーする場合は、cp -r コマンドを使います。

```
$ cp -r veggies veggies3

$
```

上記のコマンドは、veggies ディレクトリに含まれるすべてのファイルとサブディレクトリを新しいディレクトリ veggies3 にコピーします。これは「再帰的コピー」と呼ばれ、-r (recursive) オプションで指定します。このオプションを指定しないでディレクトリをコピーしようとすると、エラーメッセージが出されます。

ディレクトリの削除 (rmdir)

空のディレクトリを削除する場合は、rmdir コマンドを使います。

```
$ rmdir veggies3

$
```

ディレクトリにファイルやサブディレクトリが含まれている場合、rmdir コマンドではディレクトリは削除できません。

ディレクトリとそのすべての内容 (すべてのサブディレクトリおよびファイル) を削除する場合は、再帰的削除を指示する -r オプションを指定して rm コマンドを実行します。

```
$ rm -r veggies3

$
```

注意 – rmdir コマンドで削除したディレクトリは復元できません。rm -r コマンドで削除したディレクトリとその内容についても同様に復元できません。

ファイル間の相違点を検出する (diff)

似ているファイル間の違いを確認するには diff コマンドを使用します。次のコマンドは leftfile と rightfile の各行を走査して相違点を表示します。

```
$ diff leftfile rightfile
```

異なる行が見つかると、diff はその相違の原因 (行の追加、削除、または変更) と影響を受けた行を調べ、各ファイル内の個々の相違箇所の行番号と該当テキストを表示します。

相違の原因が行の追加による場合、diff は次の形式で行番号を表示します。

l[,*l*] a *r*[,*r*]

l は leftfile の行番号、*r* は rightfile の行番号です。

相違の原因が行の削除による場合は、上記の a の代わりに d が表示され、行の変更による場合は c が表示されます。

各ファイル内の該当テキストは、行番号情報の次の行に表示されます。leftfile 内のテキストの前には左角括弧 (<) が付き、rightfile 内のテキストの前には右角括弧 (>) が付きます。rightfile 内のテキストの前には右角括弧 (>) が付きます。

次の例では、最初に 2 つのファイルの内容を表示し、その後で diff コマンドを使ってそれらの相違点を表示しています。

```
$ cat sched.7.15

Week of 7/15


Day:  Time:       Action Item:          Details:


T     10:00       Hardware mtg.         every other week

W     1:30        Software mtg.

T     3:00        Docs. mtg.
```

```
F     1:00         Interview
$ cat sched.7.22
Week of 7/22


Day:  Time:        Action Item:          Details:


M     8:30         Staff mtg.            all day
T     10:00        Hardware mtg.         every other week
W     1:30         Software mtg.
T     3:00         Docs. mtg.
$ diff sched.7.15 sched.7.22
1c1
< Week of 7/15
---
> Week of 7/22
4a5
> M     8:30         Staff mtg.            all day
8d8
< F     1:00         Interview
```

比較される 2 つのファイルの内容が同じであれば、`diff` コマンドを実行しても何も出力されません。

`diff1` コマンドについての詳細は、『*man pages section 1: User Commands*』を参照してください。

3 つのファイルを比較する (`diff3`)

あるファイルの 3 つのバージョンを比較するには、`diff3` コマンドを使用します。

`$ diff3` *file1 file2 file3*

`diff3` コマンドはファイルの 3 つのバージョンを比較し、内容が異なる範囲に以下のコードを付加して表示します。

==== 3 つのファイルがすべて異なる

====1 *file1* がほかの 2 つと異なる

====2 *file2* がほかの 2 つと異なる

====3 *file3* がほかの 2 つと異なる

大きいファイルを bdiff を使って比較する

大きなファイルを比較する場合は、diff の代わりに bdiff コマンドを使います。bdiff の使用方法は、diff コマンドと同じです。

```
$ bdiff leftfile rightfile
```

3500 行を超えるファイルでは、diff ではなく bdiff を使ってください。

ファイルの検索 (find)

find コマンドを使うと、指定した条件に合致するファイル名の検索を、特定のディレクトリから開始できます。たとえば、特定のパターンに一致するファイル名や一定の期間内に変更されたファイル名などを検索できます。

ほかの大部分のコマンドとは違って、find のオプションはいくつかの文字からなり、コマンド行ではオプションより前に、検索を開始するディレクトリ名を指定する必要があります。

```
$ find directory options
```

directory は検索を開始するディレクトリの名前、*options* は find コマンドのオプションです。

各オプションは、ファイルの選択基準を指定します。選択されるファイルは、すべての基準を満たしている必要があります。指定するオプションの数が多くなるほど、選択されるファイルの範囲は狭くなります-print オプションを指定すると、検索結果が表示されます。

-name *filename* オプションを指定すると、*filename* に一致するファイルだけが選択されます。この *filename* は、絶対パス名のファイルの一番右側の要素と見なされます。たとえば、/usr/bin/calendar の一番右側の要素は calendar です。通常、絶対パス名中のこの要素は、ファイルのベース名と呼ばれます。

たとえば、現在のディレクトリとそのサブディレクトリ内で、文字 s で終わるファイルを検索するには、次のように入力します。

```
$ find . -name '*s' -print
./programs
./programs/graphics
./programs/graphics/gks
./src/gks
$
```

次の表は、find コマンドのその他のオプションを示しています。

表 **3–1** find オプション

オプション	説明
-name *filename*	絶対パス名の一番右側の要素 (ベース名) が *filename* に一致するファイルを選択します。*filename* にファイル名の置換パターン (正規表現) が含まれる場合は、*filename* を単一引用符 (') で囲みます。
-user *userid*	*userid* が所有しているファイルを選択します。*userid* はログイン名、ユーザ ID 番号のどちらでもかまいません。
-group *group*	*group* に属するファイルを選択します。
-mtime *n*	*n* 日前に変更されたファイルを選択します。
-newer *checkfile*	*checkfile* よりあとに変更されたファイルを選択します。

\(*options*\) のように、エスケープされた括弧でオプションを囲むことによって、オプションの優先度を指定できます。エスケープされた括弧内では、各オプションの間に -o フラグを挿入して、両方のカテゴリではなく、どちらか一方のカテゴリに分類されたファイルを選択するよう find に指定できます。

```
$ find . \( -name AAA -o -name BBB \) -print
./AAA
./BBB
```

上記の例では、find コマンドは . ディレクトリ内で AAA というすべてのファイル探し、続いて BBB というすべてのファイルを探し、そのあとで両方の検索結果を表示します。

オプションの前にエスケープされた感嘆符を付けることによって、オプションの意味を逆にすることができます。この場合、find はオプションで指定された条件に合致しないファイルを選択します。

```
$ find . \!-name BBB -print

./AAA
```

また、次のオプションを使用すると、find で検索されたファイルに対してコマンドを実行することもできます。

-exec *command* '{}' \;

このオプションは、エスケープされたセミコロン (\;) で終了します。引用符付きの大括弧 ('{}') は、find によって選択されたファイル名で置き換えられます。

find コマンドを利用すれば一時的な作業用ファイルを自動的に削除できます。一定の規則に従って一時ファイル名を指定しておくことにより、その存在場所にかかわらず、find を使ってそれらの一時ファイルを検索した上で破棄できます。たとえば、一時ファイル名を junk または dummy と指定した場合は、次のコマンドによってそれらの検索と削除を実行できます。

```
$ find . \( -name junk -o -name dummy \) -exec rm '{}' \;
```

ファイル検索の詳細は、find(1) のマニュアルページを参照してください。

ファイルとディレクトリのセキュリティ

ファイルのアクセス権は、ファイルとディレクトリを許可なく読み書きされないように保護します。ほかのユーザに対してファイルの読み取りは許可しても変更は許可したくない場合があります。また、実行可能ファイル (プログラム) を共有したい場合もあります。このような場合、ファイルのアクセス権を利用して、ファイルに対するアクセスを管理できます。

ファイルとディレクトリに関する基本的なアクセス権の種類は次のとおりです。

- r – 読み取り権。ファイルの内容を参照したりコピーしたりするには、その ファイルが読み取り可能でなければなりません。また、ディレクトリの内容を表示するには、そのディレクトリが読み取り可能でなければなりません。
- w – 書き込み権。ファイルを変更、削除したり、ファイル名を変更するには、そのファイルが書き込み可能でなければなりません。また、ディレクトリ内のファイルを追加したり削除したりするには、そのディレクトリが書き込み可能でなければなりません。
- x – 実行権。実行権付きのファイル (プログラムなど) は実行することができます。ユーザがディレクトリの任意のサブディレクトリにアクセスするには、そのディレクトリが実行可能でなければなりません。

アクセス権の設定対象となるユーザは、次の 3 つに分類できます。

- ユーザ – ファイルの所有者
- グループ – ユーザと同じグループに属するその他のユーザ (例: 特定部署のスタッフ全員など)。グループの作成と管理は、システム管理者が行います。
- その他 – すべてのユーザ

アクセス権と状態の表示 (`ls -l`)

ファイルとディレクトリの詳細一覧をアルファベット順に出力するには、`-l` を指定して `ls` コマンドを実行します。

```
$ pwd
/home/hostname/user2
$ ls -l
total 8
drwxr-xr-x   2   user2   users     1024  Feb  9  14:22  directory1
-rw-r--r--   1   user2   users        0  Feb 10  10:20  emptyfile
-rw-r--r--   1   user2   users   104357  Feb  5  08:20  large-file
drwxr-xr-x   3   user2   users     1024  Feb 10  11:13  veggies2
      |          |     |       |       |       |       |          |
  アクセス権   リンク 所有者   グループ   サイズ     日付     時刻   ファイル名または
                                                          ディレクトリ名
```

図 **3–2** アクセス権とステータスの表示

各行の最初の文字は、ファイル形式を示します。ダッシュ (-) は通常ファイル、`d` はディレクトリ、その他の文字はほかの特殊なファイル形式を表します。

次の 9 文字は、ファイルやディレクトリのアクセス権を示します。これら 9 文字は、3 文字ずつの 3 つのグループから構成され、それぞれ所有者、所有者のグループ、その他の全ユーザのアクセス権を表します。たとえば、`emptyfile` のアクセス権は、`rw-r--r--` となっています。このファイルの場合、所有者は読み取りと書き込みが行え、その他のすべてのユーザは読み取りだけは行えますが、どのユーザも実行はできません。また、ディレクトリ `veggies2` のアクセス権は、`rwxr-xr-x` となっています。このディレクトリの場合、どのユーザでも読み取りと実行が行えますが、書き込みが行えるのは所有者に限られます。

ファイルのアクセス権の他に、上記のリストには次の情報が含まれています。

- ファイルやディレクトリに対するリンクの数
- 所有者の名前 (この例では `user2`)
- グループ所有者の名前 (この例では `users`)
- ファイル内のバイト数 (文字数)
- ファイルやディレクトリの最新更新日付と時間

- ファイルやディレクトリの名前

  cd コマンドを使って自分のホームディレクトリに移動し、ls -l コマンドを入力してみてください。

  次に、以下のように入力してみてください。*dirname* には、使用しているファイルシステムに実際に存在するディレクトリの名前を指定します。

  ```
  $ ls -l dirname
  ```

ディレクトリ名を指定すると、ls -l コマンドは、そのディレクトリ内にあるディレクトリとファイルに関する情報を出力します。

隠しファイルの表示 (ls -a)

ls コマンドで表示されないファイルもあります。これらのファイルは、.cshrc、.login、.profile のように . (ドット) で始まる名前を持っています。 このようなドットファイルを表示するには、ls -a コマンドを使います。

```
$ ls -a

.

..

.cshrc

.login

.profile

emptyfile
```

. で始まるファイルは、他のファイルよりも前に表示されることに注意してください。. ファイルは現在のディレクトリを示し、.. ファイルは親ディレクトリを示します。

一般に、. で始まるファイルはシステムユーティリティによって使われ、通常はユーザが変更するものではありません。この規則には例外もいくつかあります。

アクセス権の変更 (chmod)

ファイルやディレクトリのアクセス権を変更する場合は、chmod コマンドを使います。アクセス権を変更するには、そのファイルやディレクトリの所有者であるか、root にアクセスできる必要があります。chmod コマンドの一般的な形式は次のとおりです。

chmod *permissions name*

permissions は変更用のアクセス権、*name* は影響を受けるファイルやディレクトリの名前を示します。

アクセス権を指定するには、いくつかの方法があります。ここでは簡単な方法を紹介します。

1. 次の文字のどれかを使ってユーザのタイプを指定します。
   - u (所有者)
   - g (グループ)
   - o (ほかのユーザ)
   - a (上記 3 つのカテゴリの全ユーザ)
2. アクセス権を付加するか (+) 除去するか (-) を指定します。
3. 次のいずれかの文字を使ってアクセス権を指定します。
   - r (読み取り)
   - w (書き込み)
   - x (実行)

次の例では、carrots ディレクトリに対する書き込み権を、同じグループに属するユーザに付与しています (つまり *permissions* は g+w、*name* は carrots)。

```
$ cd veggies2

$ ls -l

drwxr-xr-x   2 user2    users        512 Nov  1 09:11 carrots

$ chmod g+w carrots

$ ls -l

drwxrwxr-x   2 user2    users        512 Nov  1 09:11 carrots

$
```

上記の例の **chmod g+w carrots** コマンドは、ファイル carrots に対する書き込み権をこのグループに与えます。グループのアクセス権のハイフン (-) が w に変わっています。

この同じディレクトリに対する読み取り権と実行権を、自分のグループに属さないほかのユーザから除去するには、次のように入力します。

```
$ ls -l

drwxrwxr-x   2 user2    users        512 Nov  1 09:11 carrots

$ chmod o-rx carrots

$ ls -l

drwxrwx---   2 user2    users        512 Nov  1 09:11 carrots

$
```

他のユーザのアクセス権の r (読み取り権) と x (実行権) が、両方ともハイフン (-) に変更されたことが分かります。

新しいファイルを作成すると、次のアクセス権がシステムによって自動的に割り当てられます。

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

新しいディレクトリを作成すると、次のアクセス権がシステムによって自動的に割り当てられます。

```
drwxr-xr-x
```

たとえば、新しいファイル turnip をその所有者 (user2) が実行できるようにするには、次のように入力します。

```
$ ls -l turnip

-rw-r--r--   1 user2    users        124 Nov  1 09:14 turnip

$ chmod u+x turnip

$ ls -l turnip

-rwxr--r--   1 user2    users        124 Nov  1 09:14 turnip

$
```

3 つのカテゴリのすべてのユーザについてアクセス権を変更するには、chmod コマンドに -a オプションを指定します。garlic という新規ファイルの実行権を全ユーザに与えるには、次のように入力します。

```
$ ls -l garlic

-rw-r--r--   1 user2    users        704 Nov  1 09:16 garlic

$ chmod a+x garlic

$ ls -l garlic

-rwxr-xr-x   1 user2    users        704 Nov  1 09:16 garlic

$
```

ls -l コマンドの出力に表示されている x は、全ユーザが garlic を実行できることを示します。

ワイルドカード文字の * を使って、複数のファイルやディレクトリのアクセス権を変更することもできます。たとえば、現在のディレクトリ veggies 内の全ファイルのアクセス権を変更して所有者だけがファイルの書き込みをできるようにする場合は、次のように入力します。

```
$ pwd
```

```
/home/user2/veggies

$ ls -l

-rwxrwxrwx   1 user2    users       5618 Nov  1 09:18 beets

-rwxrwxrwx   1 user2    users       1777 Nov  1 09:18 corn

-rwxrwxrwx   1 user2    users       3424 Nov  1 09:18 garlic

-rwxrwxrwx   1 user2    users      65536 Nov  1 09:18 onions

$ chmod go-w *

$ ls -l

total 152

-rwxr-xr-x   1 user2    users       5618 Nov  1 09:18 beets

-rwxr-xr-x   1 user2    users       1777 Nov  1 09:18 corn

-rwxr-xr-x   1 user2    users       3424 Nov  1 09:18 garlic

-rwxr-xr-x   1 user2    users      65536 Nov  1 09:18 onions

$
```

注 – chmod は、現在のディレクトリでだけ実行してください。

絶対アクセス権の設定

前節では、chmod コマンドを使用してファイルアクセス権を現在の設定に対して相対的に変更する方法を説明しました。chmod コマンドに数字コードを指定することにより、ファイルまたはディレクトリのアクセス権を設定することもできます。

この形式の chmod コマンドの構文は次のとおりです。

chmod *numcode name*

この例では、*numcode* は数値コード、*name* はアクセス権が変更されるファイルやディレクトリの名前です。

数値コードは、3 つの数字からなります。1 つの数字が 3 つのユーザカテゴリ (所有者、グループ、他のユーザ) の各々に適用されます。たとえば、次のコマンドは、所有者とグループに対して読み取り権、書き込み権、実行権の絶対アクセス権を設定し、ほかのユーザに対して実行権だけを設定しています。

```
$ chmod 771 garlic
```

表 3–2 は、garlic のアクセス権が 771 というコードによってどのように表現されるかを示しています。

表 **3–2** garlic のアクセス権

アクセス権	ユーザー	グループ	その他のユーザー
読み取り	4	4	0
書き込み	2	2	0
実行	1	1	1
合計	7	7	1

表 3–2 の各列は、所有者、グループ、他のユーザというユーザカテゴリにそれぞれ対応しています。読み取り権を設定するには、適切な列に 4 を挿入します。同様に、書き込み権を設定するには 2 を挿入し、実行権を設定するには 1 を挿入します。表の最終行に表示された各列のそれぞれの合計値が最終的な数値コードになります。

数値コードを使用して絶対アクセス権を設定するもう 1 つの例を次に示します。ここでは chmod の結果を確認するために、ls -l コマンドを実行しています。

```
$ ls -l onions

-rwxr-xr-x   1 user2    users      65536 Nov  1 09:18 onions

$ chmod 755 onions

$ ls -l onions

-rwxr-xr-x   1 user2    users      65536 Nov  1 09:18 onions

$
```

chmod 755 onions と入力すると、ファイル onions のアクセス権を、そのユーザには読み取り、書き込み、実行可能、グループメンバーには読み取りと実行可能、その他のユーザには読み取りと実行可能に設定できます。表 3–3 に、onions に対するアクセス権を設定するのに使用する数値コードを示します。

表 **3–3** onion のアクセス権

アクセス権	ユーザー	グループ	その他のユーザー
読み取り	4	4	4
書き込み	2	0	0
実行	1	1	1
合計	7	5	5

所有者、グループ、およびその他のユーザ全員にファイル cabbage に対する読み取り、書き込み、および実行の権限を与えるには、次のように指定します。

```
$ ls -l cabbage
-rw-r--r--   1 user2    users         75 Nov  1 09:28 cabbage
$ chmod 777 cabbage
$ ls -l cabbage
-rwxrwxrwx   1 user2    users         75 Nov  1 09:28 cabbage
$
```

表 3–4 は、上記の例でアクセス権設定に使用されている数値コードを示しています。

表 **3–4** cabbage のアクセス権

アクセス権	ユーザー	グループ	その他のユーザー
読み取り	4	4	4
書き込み	2	2	2
実行	1	1	1
合計	7	7	7

777 という数値コードが、指定できる最大のアクセス権の値です。

相対アクセス権の変更の場合と同様に、ワイルドカード文字の * を使って現在のディレクトリ内の全ファイルの絶対アクセス権を設定することもできます。たとえば、次に示すように、現在のディレクトリ内のすべてのファイルについて絶対的なアクセス権を設定するとします。

- 所有者 – 読み取り権、書き込み権、および実行権
- グループ – 読み取り権と書き込み権
- その他のユーザ – 実行権

これらのアクセス権を設定するには、次のように入力します。

```
$ pwd
/home/user2/veggies
$ ls -l
-rwxrwxrwx   1 user2    users       5618 Nov  1 09:18 beets
-rwxrwxrwx   1 user2    users       1777 Nov  1 09:18 corn
-rwxrwxrwx   1 user2    users       3424 Nov  1 09:18 garlic
-rwxrwxrwx   1 user2    users      65536 Nov  1 09:18 onions
$ chmod 751 *
```

```
$ ls -l

-rwxr-x--x   1 user2    users       5618 Nov  1 09:18 beets

-rwxr-x--x   1 user2    users       1777 Nov  1 09:18 corn

-rwxr-x--x   1 user2    users       3424 Nov  1 09:18 garlic

-rwxr-x--x   1 user2    users      65536 Nov  1 09:18 onions

$
```

上記の例では、この chmod 操作をこれからどのディレクトリで実行するのか確認する必要があることを示すために、pwd コマンドを最初に実行しています。最初の `ls -l` コマンドは、アクセス権の変更結果を明らかにするために実行しています。相対アクセス権とは違って、絶対アクセス権を設定するときは、アクセス権の現在の設定状態を知る必要はありません。

`chmod(1)` コマンドについての詳細は、『*man pages section 1: User Commands*』を参照してください。

第 4 章

ファイルの内容の検索

この章では、grep コマンドを使って、ディレクトリとファイルの内容を検索し、キーワードや文字列を見つける方法について説明します。

grep によるパターンの検索

ファイル内の特定の文字列を検索するには、grep コマンドを使います。grep コマンドの基本構文は次のとおりです。

```
$ grep string file
```

string は検索するワードや句、*file* は検索されるファイルです。

注 – 文字列とは 1 文字以上の文字の配列です。ワードや文と同様に、文字が 1 つでも文字列とします。文字列には、空白文字、句読文字、制御文字 (表示できない文字) も入ります。

たとえば、Edgar Allan Poe の内線番号を検索する場合は、grep のあとに名前の一部または全体 (以下の例では Poe)、そのあとに内線番号の情報を含んでいるファイル名 (以下の例では extensions) を指定します。

```
$ grep Poe extensions

Edgar Allan Poe     x72836

$
```

ただし、指定されたパターンに一致するのは 1 行だけとは限らないので、注意してください。

```
$ grep Allan extensions

David Allan         x76438

Edgar Allan Poe     x72836

$ grep Al extensions

Louisa May Alcott   x74236

David Allan         x76438

Edgar Allan Poe     x72836

$
```

grep では大文字と小文字が区別されます。したがって、大文字と小文字の区別に注意してパターンを指定しなければなりません。

```
$ grep allan extensions

$ grep Allan extensions

David Allan         x76438

Edgar Allan Poe     x72836

$
```

上記の最初の例では、小文字の a で始まるエントリがないために検索は失敗に終わっています。

フィルタとして grep を使う

ほかのコマンドで grep コマンドをフィルタとして使用すれば、コマンドの出力結果から不要な情報を除去できます。grep をフィルタとして使うには、ほかのコマンドの出力結果を grep を通してパイプする必要があります。パイプの記号は | です。

次の例では、ファイル名が .ps で終わるファイルのうち、9 月 (September) に作成されたファイル名とその詳細情報を表示させます。

```
$ ls -l *.ps | grep Sep
```

上記のコマンドの最初の部分は、.ps で終わるファイルの一覧を出力します。

```
$ ls -l *.ps

-rw-r--r--   1 user2    users     833233 Jun 29 16:22 buttons.ps

-rw-r--r--   1 user2    users      39245 Sep 27 09:38 changes.ps

-rw-r--r--   1 user2    users     608368 Mar  2  2000 clock.ps
```

```
-rw-r--r--   1 user2    users     827114 Sep 13 16:49 commands.ps

$
```

コマンド行の次の部分は、grep を使用してパイプ処理を行い、Sep というパターンを検索します。

```
| grep Sep
```

この検索の結果、次のように表示されます。

```
$ ls -l *.ps | grep Sep

-rw-r--r--   1 user2    users      39245 Sep 27 09:38 changes.ps

-rw-r--r--   1 user2    users     827114 Sep 13 16:49 commands.ps

$
```

複数ワード文字列の検索

複数のワードからなるパターンを grep で検索する場合は、単一引用符または二重引用符でそのパターンを囲みます。

```
$ grep “Louisa May” extensions

Louisa May Alcott     x74236

$
```

grep コマンドは、複数のファイル内で文字列を検索できます。複数のファイル内でパターンに一致する文字列が見つかった場合は、ファイル名とパターンに一致する行をコロンで区切って出力します。

```
$ grep ar *

actors:Humphrey Bogart

alaska:Alaska is the largest state in the United States.

wilde:book.  Books are well written or badly written.

$
```

特定の文字列を含まない行の検索

特定の文字列を含まない行をファイル内で検索するには、grep の -v オプションを使います。次の例は、あるディレクトリのすべてのファイル内で、文字 e を含まない行すべてを検索する方法を示しています。

```
$ ls

actors    alaska    hinterland    tutors    wilde

$ grep -v e *

actors:Mon Mar 14 10:00 PST 1936

wilde:That is all.

$
```

grep における正規表現の使用

grep コマンドは、正規表現を使ったパターンの検索にも利用できます。正規表現は、grep に対して特別な意味を持つ特殊文字に、数字や英字を組み合わせて作成します。メタキャラクタと呼ばれるこれらの特殊な文字は、システムにとっても特別な意味を持ちます。grep コマンドで正規表現を使用する場合は、これらのメタキャラクタをエスケープしてこれらの文字の特殊な意味を無視するようにシステムに伝える必要があります。コマンドプロンプトで grep の正規表現を使用する場合は、引用符で囲んでください。メタキャラクタ (& ! . * $? \ など) は、バックスラッシュ (\) でエスケープしてください。メタキャラクタのエスケープについての詳細は、59 ページの「メタキャラクタの検索」 を参照してください。

- キャレット (^) メタキャラクタは、行の先頭を表します。たとえば、次のコマンドはファイル list 内の b という文字で始まる行を見つけ出します。

  ```
  $ grep '^b' list
  ```

- ドル記号 ($) は、行の終わりを示します。たとえば、次のコマンドは b で終わる行を表示します。

  ```
  $ grep 'b$' list
  ```

  また、次のコマンドは、ファイル list 内の行のうち b という文字しか含まれない行を表示します。

  ```
  $ grep '^b$' list
  ```

- 正規表現内のドット (.) は任意の 1 文字を表します。したがって、次のコマンドは最初の 2 文字が an である任意の 3 文字を含む文字列を検索します。検索される文字列には、たとえば 「any」、「and」、「management」、「plan」 (空白も 1 文字と数えられるため) などがあります。

  ```
  $ grep 'an.' list
  ```

- ある文字のすぐ後にアスタリスク (*) が続いている場合、grep は * を「その直前の文字のゼロ個以上の繰り返し」と解釈します。正規表現の後にアスタリスクが続いている場合、grep は * を「正規表現のパターンに一致する文字のゼロ個以上の繰り返し」と解釈します。

「ゼロ個以上の繰り返し」という表現のため、その使い方が直観的には理解しにくいかも知れません。たとえば、qu という文字列を含むワードをすべて検索するには次のように入力します。

```
$ grep 'qu*' list
```

しかし、n という文字を含むワードをすべて検索するには、次のように入力します。

```
$ grep 'nn*' list
```

また、nn というパターンを含むワードをすべて検索するには、次のように入力します。

```
$ grep 'nnn*' list
```

- ファイル `list` 内で任意の文字のゼロ個以上の繰り返しを検索するには、次のように入力します。

```
$ grep .* list
```

メタキャラクタの検索

`grep` コマンドを使用してメタキャラクタ (`& ! . * ? \` など) を検索するには、メタキャラクタの前にバックスラッシュ (\) を付けます。バックスラッシュを付けることで、`grep` はそのメタキャラクタをエスケープし、通常の文字として扱います。

たとえば、次の正規表現は、ピリオド (.) で始まる行を検索するため、`nroff` または `troff` の書式要求 (ピリオドで始まるもの) を検索するときに非常に便利です。

```
$ grep '^\.'
```

表 4–1 に、`grep` で利用できる検索パターン文字のうち頻繁に使われるものを示します。

表 4–1 `grep` の検索パターン文字

文字	説明
^	テキスト行の先頭
$	テキスト行の終わり
.	任意の 1 文字
[...]	角括弧内のリストまたは範囲に含まれる任意の 1 文字
[^...]	角括弧内のリストまたは範囲に含まれない任意の 1 文字
*	その直前の文字または正規表現のゼロ個以上の繰り返し
.*	任意の 1 文字のゼロ個以上の繰り返し

表 **4–1** grep の検索パターン文字　*(続き)*

文字	説明
\	そのあとの文字が持つ特殊な意味を無効にする

これらの検索パターン文字は、vi テキストエディタ内での検索にも使えます。

コマンド行で単一引用符と二重引用符を使う

すでに述べたように、grep に 1 つの単位として解釈させるテキストは引用符 (") で囲みます。たとえば、grep を使って「dang it, boys」という句を含むすべてのファイルを検索するには、次のように入力します。

```
$ grep "dang it, boys" *
```

複数のワードからなる句を 1 つの単位にまとめる場合は、単一引用符 (') も使えます。単一引用符を使うと、$ などのメタキャラクタを単なる文字として解釈させることもできます。

注 – history コマンドのメタキャラクタ ! は、バックスラッシュでエスケープしないと、引用符の中にあっても常に特殊文字として解釈されます。

& ! $? . ; \ などの特殊文字を通常の印字文字として解釈させるときは、これらの文字をエスケープしてください。

たとえば、次のように入力すると、ファイル list 内のすべての行が表示されます。

```
$ grep '$' list
```

しかし、次のように入力すると、$ という文字を含む行だけが表示されます。

```
$ grep '\$' list
```

grep(1) コマンドについての詳細は、『*man pages section 1: User Commands*』を参照してください。

第 5 章

プロセスとディスク利用の管理

この章の内容は次のとおりです。

プロセスと PID

各コマンドを解釈したあと、システムは一意のプロセス識別番号 (PID) を持つ独立したプロセスを生成してコマンドを実行します。システムは PID を使って各プロセスの現在の状態を追跡します。

現在実行されているコマンドの表示 (ps)

現在実行されているプロセスを調べるには、ps コマンドを使います。ps コマンドでは、コマンドの入力後生成される、ユーザが所有する各プロセスのプロセス識別番号 (PID の下に表示される) が表示されます。また、プロセスが起動された 端末 (TTY)、現在までにプロセスが使った *CPU* 時間 (TIME)、プロセスが実行しているコマンド (COMMAND) などが表示されます。

ps コマンドに -l オプションを指定すると、実行中の各プロセスの情報 (S の下に示される各プロセスの状態など) が表示されます。プロセスの状態を示すコードは次のとおりです。

- O – プロセッサ上で実行中
- S – スリープ状態。プロセスはイベントが終了するのを待っている。
- R - 実行可能状態。プロセスが実行待ち行列内にある。

- I - アイドル状態。プロセスは現在作成中。
- Z – ゾンビ状態。プロセスはすでに終了し、親プロセスは待機していない。
- T – トレース中。親プロセスがそのプロセスをトレース中のため、プロセスはシグナルによって停止されている。
- X – SXBRK 状態。プロセスは主記憶が空くのを待っている。

ps の実行中でも、個々のプロセスの状態は変化します。ps コマンドで取得できる情報はこのコマンドの実行時におけるスナップショットのため、その出力結果は、ps を入力した直後のほんのわずかの時間にしか適用できません。

ps(1) コマンドにはこの節で説明している以外に多数のオプションがあります。詳細については、『*man pages section 1: User Commands*』を参照してください。

プロセスの終了方法 (pkill)

ほとんどのウィンドウ環境には、プロセスを管理するツールが付属しています。ツールの使用方法については、それらのオンラインツールを参照してください。

不要になったコマンドプロセスの特定と終了には、pgrep コマンドと pkill コマンドを使用できます。これらのコマンドは、実行に時間のかかるプロセスを誤って開始した場合に便利です。

プロセスを終了するには、次の手順を実行します。

1. **pgrep** コマンドを入力して終了したいプロセスの **PID** を検索します。
2. **PID** を指定して **pkill** コマンドを入力します。

次に、特定の名前 (xterm) のプロセスをすべて見つけ、最後に開始された xterm プロセスを停止する方法を示します。

```
$ pgrep xterm

17818

17828

17758

18210

$ pkill -n 18210

$
```

あるプロセスを強制的に終了する必要がある場合は、pkill コマンドに -9 オプションを指定してください。

```
$ pkill -9 -n xterm
```

ディスク記憶容量の管理

ディスク領域は限られたリソースであるため、現在使用されている容量を認識しておくことが必要です。

ディスクの使用状況を表示する (df -k)

df -k コマンドを入力すると、システムにマウントされている (直接アクセスできる) 各ディスクの現在の使用容量が表示されます。システムにマウントされている各ディスクの全容量、未使用領域の容量、および使用中の領域の割合を調べる場合は、df -k コマンドを使用してください。

```
$ df -k
```

df -k コマンドの出力でファイルシステムの全容量の 90% 以上が使用中であることが判明した場合は、不要なファイルを削除してください。このためには、不要なファイルをディスクやテープに移動したり (cp でコピーしてから rm で削除する)、あるいはそれらのファイルを単に rm コマンドで削除したりします。この種のハウスキーピング処理は、自分が所有するファイルに対してだけ実行してください。

ディレクトリの使用状況を表示する (du)

du コマンドを使用すると、ディレクトリとそのサブディレクトリの使用状況を 512 バイト単位のブロック数で表示できます。

du コマンドは、各サブディレクトリ内でのディスク使用状況を表示します。ファイルシステム内のサブディレクトリの一覧を表示するには、そのファイルシステムに対応するパスに移動し、次のパイプラインを実行します。

```
$ du | sort -r -n
```

sort コマンドの *reverse* (-r) オプションと *numeric* (-n) オプションを使用したこのパイプラインにより、サイズの大きなディレクトリを見つけることができます。各ディレクトリにあるファイルの大きさ (バイト単位) と更新日付を調べるには、ls -l を実行します。古いファイルや 100 キロバイト以上のテキストファイルは、ディスク容量不足の原因となることがよくあります。

第 6 章

vi エディタの使い方

vi (発音は「ブイアイ」、visual display editor の略) は、SunOS の標準的なテキストエディタです。vi はウィンドウベースではないため、各種の端末でさまざまなファイルの編集に使えます。

vi を使ってテキストの入力と編集ができますが、vi は文書処理プログラムではありません。vi は、一般の文書処理プログラムのような手軽な方法で書式付きテキストを処理できるようには設計されていません。書式付きテキストを出力する場合は、nroff、troff、ditroff などの植字エミュレーションプログラムが必要になります。テキスト中にエミュレータによって解釈されるコードを挿入することで、これらの植字プログラムを使って書式化することができます。

vi には多様なコマンドが使用されますが、そのうち機能の重複するものが相当あります。この章では、もっとも重要な vi コマンドの概要を述べます。vi は非常に強力なテキストエディタで、熟練するには多少時間がかかります。

注 – vi には、view ユーティリティという読み取り専用のバージョンがあります。view を使ってファイルを開くと、vi コマンドは使えますが、変更を保存してファイルを誤って書き換えることがありません。

---

vi の起動

以下の各節では、vi の起動方法、ファイルへのテキストの入力方法、ファイルの保存 (書き込み) 方法、vi の終了方法について説明します。

ファイルの作成

この例では、vi を起動して paint というファイルを編集します。

```
$ vi paint
```

ファイル paint がすでに存在する場合、vi は既存のファイルを開きます。存在しない場合は、paint ファイルが作成されます。この例では、paint を新規のファイルとします。

次のような vi の編集画面が表示されます。

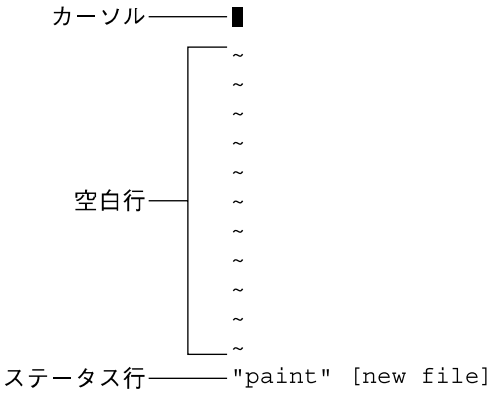

図 **6–1** vi の編集画面

カーソルは、画面の左上隅に表示されます。未入力の空白行は、チルド (~) によって示されます。

vi は、ファイル名を指定せず vi と入力するだけでも起動できます。その場合は、vi を終了するときにファイル名を指定できます。

状態表示行

画面の最終行は状態表示行と呼ばれ、ここにはファイル名、ファイルの行数と文字数が表示されます。この例のように新しいファイルを作成した場合は、状態表示行に新規ファイルである旨 ([new file]) が表示されます。

vi の 2 種類のモード

vi には、入力モードとコマンドモードという 2 種類の操作モードがあります。入力モードは、ファイルにテキストを入力するために使います。コマンドモードは、vi 特有の機能を実行するコマンドを入力するために使います。vi のデフォルトモードは、コマンドモードです。

vi ではユーザが現在どちらのモードにいるのか明示されないため、vi の初心者にとっては、コマンドモードと入力モードの区別がおそらく最大の混乱要因となります。ただし、早いうちから最低限の基本概念さえ覚えておけば、そのような混乱をしないですみます。

vi を使って最初にファイルを開くと、常にコマンドモードになります。ファイルにテキストを入力するには、vi の入力コマンドを 1 つ入力します。入力コマンドには、現在のカーソル位置にテキストを挿入する i (insert) 、現在のカーソル位置の後にテキストを追加する a (append) などがあります。これらの vi 入力コマンドについては、この章の後半で詳細に説明します。

vi のコマンドモードに戻る場合は、Esc キーを押します。現在どちらのモードにいるのか明らかでない場合は、Esc キーを押してコマンドモードにした上で作業を続けます。vi がすでにコマンドモードのときに Esc キーを押すと、警告音が鳴って画面がフラッシュしますが実害はありません。

入力モード

サンプルファイルの paint にテキストを入力するには、vi の「挿入 (insert)」コマンドである i を入力します。このコマンドにより、vi はコマンドモードから入力モードに切り替わります。

ここで数行の短いテキストを入力し、各行の終わりで Return キーを押してみてください。入力した文字はカーソルの左側に表示され、既存の文字は右側に押し出されます。入力途中ならば、Return キーを押す前に Back Space キーで後退して再入力することによって誤りを修正できます。vi におけるテキストの編集については、74 ページの「テキストの変更」を参照してください。

paint にテキストを入力し終えたら、Esc キーを押してコマンドモードに戻ります。このとき、カーソルは入力した最後の文字に移動します。これで次の vi のコマンドを入力できる状態になります。

vi が予期しない動作をする場合は、「Caps Lock」モードになっていないか確認してください。このモードになっていると、入力した文字がすべて大文字になります。システムによっては、F1 キー (通常 Esc キーの隣り) が Caps Lock として機能するものがあります。そのため、Esc キーの代わりに F1 キーを押してしまうのはよくある誤りです。

注 – 余分なシステムメッセージを削除するためなどで、画面をクリア (再描画) するよう vi に指示しなければならない場合があります。画面をクリアするには、コマンドモードで Ctrl-L を押します。

コマンドモード

vi を使ってファイルを開くと、コマンドモードになります。このモードでは、広範囲にわたって機能を実行するさまざまなコマンドを入力できます。vi の大部分のコマンドは、1 文字か 2 文字と、任意に指定できる数値からなっています。通常、大文字のコマンドと小文字コマンドは関連した処理を行いますが、異なる機能を実行します。たとえば、小文字の a を押すとカーソルの右側にテキストを追加できますが、大文字の A を押すとその行の最後にテキストを追加できます。

vi のコマンドの大部分は、Return キーを押さなくても実行されます。ただし、コロン (:) で始まるコマンドは、最後に Return キーを押す必要があります。vi に関する説明では、コロンで始まるコマンドを 3 番目の独立したモード (最終行モード) として取り扱う場合もあります。これは、コマンドモードでコロンを入力すると、コロンとそのあとに入力した文字が状態表示行に表示されるためです。ただし、この章の説明では、vi の全コマンドはコマンドモードから起動されるものとします。

コロンで始まるコマンドは、ex のコマンドです。vi と ex は、異なるインタフェースを持つテキスト編集プログラムです。vi が画面指向のインタフェースであるのに対し、ex は行指向のインタフェースです。ex のコマンドは vi 内部から利用できます。コロンを入力した場合は、実際には行指向の ex インタフェースに切り換えたことになります。この切り換えによって、vi を終了しなくても多数のファイルを操作するコマンドを実行できます。詳細については、この章で後述されている 78 ページの「ex コマンドの使い方」を参照してください。

セッションの終了

vi でファイルを編集する場合は、ファイルそのものが直接変更されるのではなく、vi が一時メモリ内に作成するファイルのコピー (バッファと呼ばれる) が変更されます。ディスク上にある恒久的なファイルのコピーは、バッファの内容を書き込んだ (保存した) ときだけ変更されます。

これには長所と短所があります。長所は、編集セッション中の変更内容をすべて破棄し、ディスク上のファイルに影響を与えずにファイル編集を中止することができるという点です。短所は、システムがクラッシュした場合は、保存していない作業バッファの内容が失われてしまうことです。特に、電話回線で接続されたリモート端末上の vi ユーザは、予期しない障害にさらされる危険があります。

特に重要な変更を行う場合などは、ファイルを頻繁に保存することをお勧めします。

注意 – 同一ファイルに対して複数の vi セッションを同時に実行することはできますが、あまりお勧めできる方法ではありません。ファイルに書き込まれた変更とほかの同時セッションによって上書きされた変更とを見分けることは、非常に困難です。

ファイルの保存と vi の終了

vi には、バッファ内容のファイル保存と vi の終了を制御する類似の意味を持つコマンドが多数存在します。これらのコマンドを使うと、「ファイルの保存」、「保存と終了」、「保存しないで終了」などを選択できます。

保存

バッファの内容を保存する (バッファの内容をディスク上のファイルに書き込む) には、次のように入力します。

:w

Return キーを押します。

保存と終了

次のように入力して保存し、終了します。

:wq

Return キーを押します。上記の代わりに、ZZ を続けて押すこともできます。

ZZ コマンドについては、前にコロンを入力する必要も、Return キーを押す必要もありません。

保存しないで終了

ファイルを変更していない場合は、次のように入力して終了します。

:q

Return キーを押します。ファイルを変更した場合は、:q コマンドでは vi を終了できません。このコマンドを入力すると、次のメッセージが表示されます。

```
No write since last change (:quit! overrides)
```

変更内容を保存しなくてもよい場合は、次のように入力します。

```
:q!
```

Return キーを押します。

ファイルの印刷

ファイルを編集して vi を終了したあとは、次のコマンドを使ってファイル内容を印刷できます。

```
$ lp filename
```

filename は、印刷する vi ファイルの名前です。このコマンドによってファイルはデフォルトプリンタに出力されます。ファイルは書式化されず画面上に表示されているとおり各行ごとに印刷されます。プリンタのコマンドについての詳細は、第 8 章を参照してください。

vi の基本的なコマンド

以降の節では、次に示す、vi コマンドのさまざまな機能について説明します。

- ファイル内の移動
- テキストの挿入
- テキストの変更と置換
- テキストの変更の取り消し
- テキストの削除
- スペルのチェック
- ファイル出力の書式の指定
- コマンドの繰り返し

ファイル内の移動

前節では、ファイルの作成、保存、印刷の方法と vi の終了方法を学習しました。ファイルが作成されたので、次にファイル内を移動するために必要な概念を理解しておく必要があります。練習用ファイルを開いて、この節で説明する各コマンドを実際に試してみてください。

カーソルの移動

vi の起動時には、カーソルは vi 画面の左上隅に表示されます。コマンドモードでは、複数のキーボードコマンドを使ってカーソルを移動できます。カーソルの移動には、特定の文字キー、矢印キー、Return キー、Back Space キー (または Delete キー)、Space Bar キーが使えます。

注 – vi の大部分のコマンドでは大文字と小文字が区別されます。そのため、同じキーによるコマンドであっても、大文字と小文字とでは異なる結果となる場合があるので注意してください。

矢印キーによる移動

キーボードに矢印キーが付いている場合は、これらを使ってみてください。上下左右の矢印キーを使うと、画面内で自由自在にカーソルを移動できます。カーソルの移動ができるのは、すでに表示されているテキストまたは入力領域内に限られます。

リモート端末から vi を使う場合は、矢印キーが正しく動作しないことがあります。矢印キーの動作は、端末エミュレータによって異なります。矢印キーが正しく動作しない場合は、次のような方法でカーソルを移動します。

- 左に移動するには、h を押します。
- 右に移動するには、l を押します。
- 下に移動するには、j を押します。
- 上に移動するには、k を押します。

ワード単位の移動

ワード単位で右にカーソルを移動する場合は、w (「word」を表す) を押します。(Solaris 2.5 and ... という文の場合、w を押すごとに S 2 . 5 a の順に移動します。)

ワード単位で左にカーソルを移動するには、b (「back」を表す) を押します。

いちばん近くにある区切り文字を通り越して右または左のワード (スペースで区切られているワード) の最初の文字にカーソルを移動するには、W または B を押します。(Solaris 2.5 and ... という文の場合、W を押すごとに S 2 a の順に移動します。

現在カーソルがあるワードの最後の文字にカーソルを移動するには、e (「end」を表す) を押します。

行頭または行末への移動

現在カーソルがある行の先頭に、カーソルを移動するには、^ を押します。

現在カーソルがある行の末尾に、カーソルを移動するには、$ を押します。

1 行下への移動

1 行下の、最初の (空白ではない) 文字にカーソルを移動するには、Return キーを押します。

左への移動

1 文字分左にカーソルを移動するには、Back Space キーを押します。

右への移動

1 文字分右にカーソルを移動するには、Space Bar キーを押します。

先頭行への移動

画面の先頭行にカーソルを移動するには、H (「high」を表す) を押します。

中央行への移動

画面の中央行にカーソルを移動するには、M (「middle」を表す) を押します。

最終行への移動

画面の先頭行にカーソルを移動するには、L (「low」を表す) を押します。

ページングとスクロール

画面の最終行にカーソルがあるときに下に移動する場合、または画面の先頭行にカーソルがあるときに上に移動する場合は、テキストがスクロールアップまたはスクロールダウンされます。短いファイルの場合にはこのスクロールが効果的ですが、長いファイル内を移動する場合には少々手間がかかります。

ファイル内で 1 画面分または半画面分まとめて上方向や下方向にカーソルをページング、またはスクロールすることができます。これらのコマンドを `paint` というファイルで試すには、ファイルが十分長くなるようにテキストを追加します。

ページングとスクロールは基本的に異なります。テキストのスクロールでは、カーソルを一度に行単位で上下に移動しますが、ページングではカーソルを一度に画面単位で上下に移動します。高速のシステム上では、この違いに気付かないこともありますが、リモート端末から `vi` を使っている場合やシステムの実行速度が通常より遅い場合は、この違いがきわめて明白になります。

次の *1* 画面へのページング

1 画面先 (下方向) にカーソルを移動するには、Ctrl-F を押します (Control キーを押しながら F キーを押します)。カーソルは、新しい画面の左上隅に移動します。

次の半画面へのスクロール

半画面先 (下方向) にスクロールするには、Ctrl-D を押します。

前の *1* 画面へのページング

1 画面前 (上方向) にカーソルを移動するには、Ctrl-B を押します。

前の半画面へのスクロール

半画面先 (下方向) にスクロールするには、Ctrl-U を押します。

テキストの挿入

`vi` にはテキストを挿入するためのコマンドが多数あります。この節では、これらのコマンドのうち最も便利なものを紹介します。これらのコマンドが実行するモードは、入力モードです。これらのコマンドを使う場合、最初はコマンドモードでなければなりません。Esc キーを押してコマンドモードであることを確認してください。

追加

カーソルの右側にテキストを追加するには、`a` (append) コマンドを使います。追加したい位置の左にカーソルを移動して `a` を押し、追加するテキストを続けて入力してみてください。入力が終わったら Esc キーを押します。

行の末尾にテキストを追加するには、`A` コマンドを使います。このコマンドの動作を確認するために、追加したい行にカーソルを移動して (その行内ならカーソルはどこにあってもよい)、`A` を押します。カーソルは行の末尾に移動し、その位置からテキストを続けて入力できます。入力が終わったら Esc キーを押します。

挿入

カーソルの左側にテキストを挿入するには、コマンドモードで `i` (insert) コマンドを使います。

行の先頭にテキストを挿入するには、`I` を押します。このコマンドを使うと、その行の任意の位置からカーソルが移動します。テキストを入力したあとにコマンドモードに戻るには、Esc キーを押します。

行の挿入

カーソルがある行の上または下に新規の行を挿入するには、次のコマンドを使います。

カーソルがある行の下に新規の行を挿入するには、o コマンドを使います。o を入力したあとにテキストを入力します。複数行にわたるようなテキストも入力することができます。入力が終わったら Esc キーを押します。

カーソルがある行の上に新規の行を挿入するには、O コマンド (大文字) 使います。

テキストの変更

テキストの変更とは、テキストの一部分を別のテキストで置換することです。vi には状況に応じてテキストを変更するための方法が複数あります。(テキスト変更のコマンドを入力後、置換される部分を明らかにするために、1 文字の場合はその文字が、テキストなどの場合はその最後の文字が、$ で表示されます。

ワードの変更

ワード全体を置換するには、置換するワードの先頭にカーソルを移動し、cw と続けて入力してから新しいワードを入力します。入力が終わったら Esc キーを押します。

ワードの後半部分だけを変更するには、保存する部分の右側にカーソルを移動し、cw と続けて入力してから新しいテキストを入力します。入力が終わったら Esc キーを押します。ワードの前半部分だけを変更するには、保存する部分の先頭にカーソルを移動し、cb と続けて入力してから新しいテキストを入力します。入力が終わったら Esc キーを押します。

行の変更

行全体を置換するには、変更したい行に移動して (その行内ならカーソルはどこにあってもよい)、cc と続けて入力します。すると、その行内の文字はすべて消えます (置換される部分を示す $ は表示されません)。新しいテキスト (長さは任意) の入力が終わったら Esc キーを押します。

1 文字以上の置換

カーソルがある文字を、1 つ以上の文字からなるテキストに置換するには、s を押してから新しいテキストを入力します。コマンドモードに戻るには、Esc キーを押します。

1 文字の置換

カーソルがある文字を、別の 1 文字に置換するには、r コマンドを使います。置換する文字の上にカーソルを移動し、r を押してから 1 文字だけ入力します。この置換が終了すると、vi は自動的にコマンドモードに戻ります (Esc キーを押す必要はありません)。

変更の取り消し

vi でテキストを編集してファイルを変更するときは、誤った入力を取り消したいと思う場合が必ずあります。vi の u (「undo」を表す) コマンドを使うと、直前の操作を取り消した状態から編集作業を再開できます。

前回のコマンドの取り消し

vi で入力を誤ったり、特定の操作を実行したあとで気が変わった場合は、コマンドを実行した直後に u を押すことによってそのコマンドを取り消しできます (u を押したあとで Esc キーを押す必要はありません)。もう一度 u を押すと、u コマンド自身が取り消されます。

行に対する変更の取り消し

特定の行に対する変更をすべて取り消すには、U コマンドを使います。このコマンドは、その行からカーソルを移動していない場合にだけ有効です (U を押したあとで Esc キーを押す必要はありません)。

テキストの削除

vi の削除コマンドを使って、指定した文字、ワード、行の削除ができます。削除後は、vi はコマンドモードのままなので、入力モードにするためのコマンドを入力してからテキストを挿入してください。

1 文字の削除

1 文字を削除するには、削除する文字の上にカーソルを移動して x を押します。

x コマンドは、その文字自身と、その文字が占めていたスペースも削除します。つまり、ワードの真ん中から 1 文字を削除すると、その 1 文字分のスペースも削除されて残りの文字が詰められます。x コマンドを使って行内の空白 (つまりその空白自身と、その空白が占めていたスペース) を削除することもできます。

カーソルの前方 (左側) の文字を 1 文字削除する場合は、大文字の x を押します。

ワードまたはワードの一部分の削除

ワードを 1 つ削除するには、削除するワードの先頭にカーソルを移動して dw と続けて入力します。ワードとそれが占めていたスペースが削除されます。

ワードの一部分だけを削除するには、保存する (削除せずにそのままにしておく) 部分の右にカーソルを移動して dw と続けて入力します。ワードの残りの部分とそれが占めていたスペースが削除されます。

行の削除

行を削除するには、削除したい行に移動して (その行内ならカーソルはどこにあってもよい)、dd と続けて押します。その行と、その行が占めていたスペースが削除されます。

テキストのコピーと移動 — yank、delete、put

大部分の文書処理プログラムと同様に、vi エディタでも、テキスト行の「コピー&ペースト」と「カット&ペースト」ができます。vi で「コピー&ペースト」を行う場合は、yank と put コマンドを使います。また、「カット&ペースト」の場合は、delete と put を使います。

vi で比較的短いテキストブロックのコピーや移動を行う場合は、 yank、delete、put の各コマンドを組み合わせて使います。

行のコピー

行をコピーするには、「yank」を表す yy (または Y) と、「put below」を表す p (または「put above」を表す P) という 2 つのコマンドが必要です。Y と yy は同じ働きをします。

1 行をバッファに取り込む (yank) には、次の手順を実行します。

1. 取り込みたい行の任意の位置にカーソルを移動させます。
2. yy と入力します。
3. 取り込んだ行を挿入 (コピー) する位置の上の行に移動します。
4. p と入力します。

   その取り込んだ行が p と押した行の下に表示されます。

   取り込んだ行をカーソルがある行の上にコピーするには、P (大文字) を押します。

yy コマンドでは行数を指定できます。たとえば、バッファに 11 行を取り込むには、11yy と続けて入力します。すると、カーソルがある行から下方向に数えて 11 行目までがバッファに取り込まれ、画面の最終行にその旨を示すメッセージ 「11 行をバッファにコピーしました」が表示されます。

先に説明した任意の削除コマンドの直後に p または P コマンドを使うと、削除したテキストをカーソルがある行の下の行または上の行に挿入できます。

注意 – コピーや移動 (yank、delete、put) の操作中は、カーソルの移動コマンドだけを使ってください。テキストを新しい位置に挿入する前にほかのテキストをコピーしたり移動したりすると、先にコピーや移動のためにバッファに取り込んだ内容が失われてしまいます。

行の移動

行の移動についても、dd (「delete」を表す) と、p (または P) という 2 つのコマンドが必要です。

行を移動するには、まずその移動したい行に移動して (その行内ならカーソルはどこにあってもよい)、`dd` と続けて入力します。たとえば、その行とその行から数えて 5 行目までの 5 行分を移動の対象行としたい場合は、その行で `5dd` と続けて入力します。このコマンドによって、移動の対象行が削除されます。

次に、その削除した行を挿入する位置の上の行に、カーソルを移動して `p` を押します。その行がカーソルのあった行の下に挿入されます。

また、その削除した行をカーソルのある行の上に挿入するには、`P` を押します。

カウントを使ってコマンドを繰り返す

ほとんどの `vi` のコマンドの前には、カウントと呼ばれるリピートファクタ (コマンドの前に付ける数値で、その操作を何回繰り返すかを指示する) を指定できます。

これまでの節で説明した大部分のコマンドに、カウントを指定できます。たとえば、`3dd` は行を削除する操作を 3 回繰り返します (したがって、カーソルがあった行も含めて 3 行が削除されます)。`2dw` はカーソルがあった位置のワードも含めて 2 つのワードを削除し、`4x` はカーソルがあった位置も含めて 4 つの文字または空白を削除します。またカウントを移動コマンドの前に付ければ、`3w` や `2Ctrl-F` (`2` を押してから、Ctrl-F を押す) のように入力してカーソルを移動できます。

ピリオド (`.`) を押すと、前回実行したテキスト編集コマンドが繰り返されます。たとえば、`dd` を使ってある行を削除した場合は、カーソルを別の行に移動してピリオドを押せば、その別の行を削除できます。

ex コマンドの使い方

比較的長いテキストブロックを取り扱う場合は、yank、delete、put よりも ex コマンドを利用した方が正確で便利です。ex コマンドを使うと、画面上で行を数えたり挿入位置を目で探したりするのではなく、行範囲を指示してから挿入位置の上にくる行番号を指定することで移動またはコピーができます。削除コマンドの場合は、挿入位置の指定は必要ありません。

行番号の表示と非表示

行番号を表示させない場合には、:set nu と入力して Return キーを押します。

行番号は左端に表示されます。なお、ファイルを印刷しても、これらの行番号は印刷されません。行番号が見えるのは画面上だけです。

```
1 Oh, when I die, take my saddle from the wall,

2 Put it on my pony, lead him out of the stall.

3 Tie my bones to his back, point our faces to the west,

4 And we'll ride the prairies that we love the best.

5

6 Ride around, little doggies,

7 Ride around real slow.

8 Firey and Snuffy are rarin' to go.

~

~

~

~

~

~

:set nu
```

行番号を表示させない場合は、:set non u と入力して Return キーを押します。

行のコピー

ex コピーコマンドの基本形式は次のとおりです。

:*line#*,*line#* **co** *line#*

コンマで区切られた最初の 2 つの数値は、コピーされる行の範囲を指定します。3 番目の数値は、それを挿入する位置の上の行番号です。

たとえば、1 行目から 5 行目までをコピーしてそのブロックを 12 行目の下に挿入する場合は、次のように入力します。

```
:1,5 co 12
```

続いて Return キーを押します。

行の範囲を指定するときに、次の省略形を使用できます。

- ピリオド (.) は、「現在カーソルがある行」を意味します。
- ドル記号 ($) は、「ファイルの最終行」を意味します。

たとえば、「現在カーソルがある行から 5 行目まで」の範囲をコピーしてそのブロックを 12 行目の下に挿入する場合は、次のように入力します。

```
:.,5 co 12
```

また、「6 行目からファイルの最終行まで」の範囲をコピーしてそのブロックを 2 行目の下に挿入する場合は、次のように入力します。

```
:6,$ co 2
```

行の移動

ex 移動コマンドの基本形式は、上記のコピーコマンドの形式と似ています。

:*line#*,*line#* **m** *line#*

行範囲と挿入位置 (行) の指定方法はコピーコマンドの場合と同様です。省略形のピリオド (.) とドル記号 ($) も使用できます。コピーと移動の機能の違いは、移動の場合はブロックがある位置から削除されてほかの位置に挿入されることです。

たとえば、1 行目から 5 行目までを 12 行目の下に移動する場合は、次のように入力します。

```
:1,5 m 12
```

続いて Return キーを押します。

行の削除

ある範囲の行を削除するには、次の形式のコマンドを使います。

:*line#*,*line#* **d**

たとえば、1 行目から 5 行目までを削除する場合は、次のように入力します。

```
:1,5 d
```

vi による検索と置換

vi には指定された文字列をファイル内で検索するための方法がいくつかあります。また、広範囲にわたった置換機能も利用できます。

文字列の検索

文字列とは、1 つ以上の文字の配列です。文字列には、英字、数字、句読点、特殊文字、空白、タブ、改行を入れることができます。文字列は、文法的に正しいワード、ワードの一部だけのどちらでもかまいません。

文字列を検索するには、/ の後に検索する文字列を入力して Return キーを押します。カーソルがある位置から検索を始め、その文字列の先頭位置にカーソルが移動します。たとえば meta という文字列を検索する場合は、/meta と入力して Return キーを押します。

それと同じ文字列をさらに検索するには、n を押します。カーソル位置よりも前 (逆) 方向に検索するには、N を押します。

ファイル内で逆方向 (画面の上の方向) に文字列を検索する場合は、/ の代わりに ? を使います。その場合、n と N では / の時と方向が逆になります。

通常の検索では大文字と小文字が区別されます。したがって、china の検索では China は見つかりません。検索時に大文字と小文字の区別を無視させる場合は、:set ic と入力します。デフォルトの検索モードに戻すには、:set noic と入力します。

検索する文字列が見つかると、最初の出現位置にカーソルが移動します。文字列が見つからない場合は、状態表示行に「パターンが見つかりません」というメッセージが表示されます。

特殊文字 (/ & ! . ^ * $ \ ?) は検索機能にとって特別な意味を持っているため、検索対象の文字列中で使う場合は「エスケープ」する必要があります。特殊文字をエスケープするには、その文字の前にバックスラッシュ (\) を付けます。たとえば anything? という文字列を検索する場合は、 /anything\? と入力して Return キーを押します。

これらの特殊文字は、検索機能に対するコマンドとして使えます。したがって、これらの文字を 1 つ以上含む文字列を検索する場合は、その文字の前にバックスラッシュを付ける必要があります。バックスラッシュ自身をエスケープするには、\\ と入力します。

高度な検索

次の特性を表すタグを文字列に付けることによって、さらに高度な検索ができます。

- 行の先頭
- 行の終わり
- ワードの先頭
- ワードの終わり
- ワイルドカード文字

行の先頭を表すには、検索文字列の最初にキャレット (^) を付けます。たとえば、Search で始まる行を検索する場合は、次のように入力します。

```
/^Search
```

行の終わりを表すには、検索文字列の最後にドル記号 ($) を付けます。たとえば、search で終わる次の行を検索するには、次のように入力します。

```
/search\.$
```

ピリオドは、バックスラッシュでエスケープしていることに注意してください。

ワードの先頭を表すには文字列の最初に \< を付け、ワードの終わりを表すには文字列の最後に \> を付けます。したがって、文字列ではなくワードを検索する場合は、ワードの先頭を表すタグとワードの終わりを表すタグを検索パターン中で組み合わせて使います。たとえば、search というワード (文字列中にあるものは除く) の出現位置を検索するには、次のように入力します。

```
/\<search\>
```

任意の 1 文字を表すには、文字列中のその文字の位置にピリオド (.) を入力します。たとえば、disinformation または misinformation の出現位置を検索するには、次のように入力します。

```
/.isinformation
```

この検索はワードではなく文字列を対象としているため、上記の検索パターンは misinformationalist や disinformationism という構文とも一致します。

指定した文字のどれかと一致する文字を検索するには、それらの文字を括弧 [] で囲みます。たとえば、/[md]*string* という検索パターンは、m または d で始まる文字列を検索します。また、/[d-m]*string* という検索パターンは、d から m のうちの 1 文字で始まる文字列を検索します。

文字列中にアスタリスク (*) を使うと、* の直前の文字のゼロ個以上の繰り返しを表すことができます。括弧 [] と * を効果的に組み合わせると、検索する文字列を細かく指定できます。たとえば、次の例のように入力すると、文字列 isinformation が出現する位置、および a から z のうちの 1 文字で始まり文字列 isinformation で終わる文字列が出現する位置をすべて検索します。

```
/[a-z]*isinformation
```

文字列の置換

文字列を置換する手順は、上記で説明した検索の手順に基づいています。検索パターンを表す特殊文字はすべて、検索および置換の手順でも使えます。

置換コマンドの基本形式は次のとおりです。

:g/*search-string***/s//***replace-string***/g**

続いて Return キーを押します。

たとえば、disinformation という文字列をすべて newspeak で置換するには、次のように入力します。

```
:g/disinformation/s//newspeak/g
```

続いて Return キーを押します。

このコマンドで、文字列が見つかるたびに検索を中断して置換を行うかどうかを確認するメッセージを表示させることもできます。次のコマンド例では、gc (「consult」を表す c を g に追加したもの) を使って、disinformation が見つかるたびに検索を中断して置換を行うかどうかを確認するメッセージを出すよう設定しています。置換する場合は y、置換しない場合は n を入力してから Return キーを押します。

```
:g/disinformation/s//newspeak/gc
```

注 – この応答形式の検索と置換を中断するには、Ctrl-C を押します。Ctrl-C の入力以前に y と入力されたものに関しては、置換されたままとなります。

指定行へジャンプする

編集ファイルの最終行にジャンプするには、G を押します。ファイルの先頭行に戻るには、1G と続けて押します。

行番号の数字を押した後で G と押せば、指定した行にジャンプできます。

たとえば、あるファイルの 51 行目を編集してから vi を終了したとします。その後、51 行目からまた編集したい場合は、vi でそのファイルを開いてから 51G と続けて押してください。その 51 行目にカーソルが移動し、その行から編集を始めることができます。

編集中のファイルに別ファイルを読み込む

vi では、編集中のファイルに別のファイルを簡単に「読み込む」(挿入する) ことができます。コマンドの一般的な形式は次のとおりです。

: *line#* **r** *filename*

行番号 *line#* を指定しない場合 (**:r** *filename*) は、カーソルがある行の下にファイル *filename* が読み込まれます。

たとえば、orwell というファイルをあるファイルの 84 行目の下に挿入したい場合は、次のように入力します。

```
:84 r orwell
```

または、84 行目にカーソルを移動 (84G と続けて入力) してから、次のように入力することもできます。

```
:r orwell
```

複数のファイルの編集

vi では複数のファイルを編集できます。たとえば、paint と orwell というファイルを交互に編集する場合は、次のようにします。

1. 最初に **paint** の変更内容を保存するために、**:w** と入力して **Return** キーを押します。
2. **orwell** を編集するために、**:n orwell** と入力して **Return** キーを押します。
3. **orwell** を編集して変更内容を保存します。
4. **orwell** の編集を完了して変更内容を保存したあとは、次の **3** つのどれかを選択できます。
   - vi を終了する – :q と入力して Return キーを押します。
   - paint に戻る – :n # と入力して Return キーを押します。
   - 2 つのファイルを交互に移動する – :n # を繰り返し実行します。

注意 – :w *filename* (別の名前のファイルに保存) または :r *filename* (ほかのファイルを読み込む) などを編集中に使うと、:n # としても paint や orwell ではなくそのファイル *filename* になってしまうので注意してください。

vi の基本コマンドの一覧

次の表は、vi コマンドの基本コマンドの一覧です。

表 **6–1** vi の起動

コマンド名	説明
vi *filename*	ファイルを開くまたは新規作成する
vi Return キー	新規ファイルを開く (ファイル名は後で指定)
vi -r *filename*	システムクラッシュ時のファイルを復元して開く
view *filename*	読み取り専用でファイルを開く

表 **6–2** カーソル移動コマンド

コマンド名	説明
h	左に移動
j	下の行の先頭文字 (空白ではない) に移動
k	上に移動
l	右に移動
w	ワード単位で右に移動
W	(スペースで区切られた) ワード単位で右に移動
b	ワード単位で左に移動
B	(スペースで区切られた) ワード単位で左に移動
e	現ワードの最後の文字に移動
Return キー	下の行の先頭文字 (空白ではない) に移動
Back Space キー	左に移動

表 **6–2** カーソル移動コマンド　(続き)

コマンド名	説明
Space Bar キー	右に移動
`H`	画面の先頭行に移動
`M`	画面の中央行に移動
`L`	画面の最後行に移動
Ctrl-F	1 画面先のページを表示
Ctrl-D	半画面先にスクロール
Ctrl-B	1 画面前のページを表示
Ctrl-U	半画面前にスクロール

表 **6–3** 文字と行の挿入

コマンド名	説明
`a`	カーソルの右にテキストを追加
`A`	行の末尾にテキストを追加
`i`	カーソルの左にテキストを追加
`I`	行の先頭にテキストを挿入
`o`	カーソルがある下の行にテキストを挿入
`O`	カーソルがある上の行にテキストを挿入

表 **6–4** テキストの変更

コマンド名	説明
`cw`	カーソルのワード (またはワードの右側の部分) を変更
`cc`	行全体を変更
`C`	カーソル位置から行の末尾までを変更
`s`	カーソルの 1 文字をテキストに変換
`r`	カーソルの文字を別の 1 文字に置換
`J`	カーソルがある行とその下の行を連結
`xp`	カーソルの文字とその右の文字を入れ替える
`~`	大文字または小文字に変える

表 **6–4** テキストの変更　*(続き)*

コマンド名	説明
`u`	前回の変更などのコマンドを取り消す
`U`	カーソルがある行に対する全変更を取り消す
	前回の変更などのコマンドを取り消す

表 **6–5** テキストの削除

コマンド名	説明
`x`	カーソルの文字を削除
`X`	カーソルの左の文字を削除
`dw`	ワード (またはワードの右側の部分) を削除
`dd`	行を削除
`D`	ある行のうちカーソルから右側の部分を削除
`dG`	カーソルがある行からファイルの最終行までを削除
`d1G`	ファイルの先頭行からカーソルがある行までを削除
`:5,10 d`	5 行目から 10 行目までを削除

表 **6–6** テキストのコピーと移動

コマンド名	説明
`yy`	行をコピー
`Y`	行をコピー
`p` (小文字)	コピーまたは移動の対象行をカーソルがある行の下に挿入
`P` (大文字)	コピーまたは移動の対象行をカーソルがある行の上に挿入
`:1,2 co 3` Return キー	1 行目から 2 行目までを 3 行目の下にコピー
`:4,5 m 6` Return キー	4 行目から 5 行目までを 6 行目の下に移動

表 **6–7** 行番号の設定

コマンド名	説明
`:set nu` Return キー	行番号を表示する
`:set nonu` Return キー	行番号を表示しない

表 **6–8** 大文字と小文字の区別

コマンド名	説明
`:set ic` Return キー	検索時に大文字と小文字の区別をしない
`:set noic` Return キー	検索時に大文字と小文字を区別する

表 **6–9** カーソルのジャンプ

コマンド名	説明
`G`	ファイルの最終行にジャンプ
`1G`	ファイルの先頭行にジャンプ
`21G`	21 行目にジャンプ

表 **6–10** 検索と置換

コマンド名	説明
/*string*	文字列を検索
?*string*	文字列を逆方向に検索
`n`	検索方向の前方にある文字列を検索
`N`	検索方向の後方にある文字列を検索
:g/*search*/s//*replace*/g Return キー	文字列の検索と置換

表 **6–11** 画面のクリア

コマンド名	説明
Ctrl-L	乱れた画面をクリア (再表示)

表 **6–12** 編集中のファイルに対するファイルの読み込み

コマンド名	説明
`:r` *filename* Return キー	カーソルがある行の下にファイルを挿入 (読み込み)
`:34 r` *filename* Return キー	34 行目の下にファイルを挿入 (読み込み)

表 **6–13** 保存と終了

コマンド名	説明
`:w` Return キー	変更を保存 (バッファをファイルに書き込む)
`:w` *filename* Return キー	指定されたファイルにバッファーを書き込む
`:wq` Return キー	変更を保存して `vi` を終了
`ZZ`	変更を保存して `vi` を終了
`:q!` Return キー	変更を保存しないで `vi` を終了

第 7 章

メールの使い方

SunOS には、電子メール (email) の送受信に使われる `mailx` というプログラムがあります。`mailx` を使って、メッセージの読み取りと書き込み、送信と受信、および保存と削除ができます。`mailx` プログラムはウィンドウベースではないため、任意の端末上で実行できます。ウィンドウベースのメールのほうが好まれるかもしれませんが、簡単なメッセージを急いで送信する場合は、`mailx` プログラムのほうが便利です。ユーザ専用のメールエイリアスを設定する場合も、この章を参照してください。

注 – ウィンドウ環境でメール・プログラムのアイコンが画面に表示されている場合は、メール・プログラムを終了してからこの章の例を実行してください。終了しないと、2 つのメールプロセスが有効になるため、エラーメッセージや警告が出されることがあります。メール・プログラムのウィンドウ内では正常にメッセージの送受信ができます。しかし、メールを読み取ったあとでメッセージを保存したり削除すると、受信箱が影響を受けるため、メール・プログラムの混乱の原因になります。

`mailx` の基本

この節では、`mailx` の基本作業を行う方法について説明します。この章の後半では、このプログラムを効率良く使う上で役立つ機能と特徴について説明します。

メールの受信者のログイン名とマシン名は、`mailx` プログラム用の一意のアドレスとして使われます。メールの受信者が送信者と同じマシン上にいる場合は、ログイン名だけ分かれば十分です。各ユーザは、メールを受信するためのメールボックスを持っています。通常、このメールボックスは `/var/mail/`*username* ディレクトリ (*username* は各ユーザのログイン名) にあります。

メールが受信されると `mailx` プログラムはその旨をユーザに知らせ、メールボックスにメールを収納します。

mailx の起動

mailx を起動するには、プロンプトに対して次のコマンドを入力し、Return キーを押します。

```
$ mailx
```

メールボックスにメールがない場合は、次のようなメッセージが画面に表示されます。

username 宛のメールはありません。

username は、各ユーザのログイン名です。

自分宛てにメッセージを送信する

mailx の機能を試すには、まず最初に自分自身にメッセージを送信してみましょう。プロンプトに対して mailx コマンドをもう一度入力しますが、今回は自分のアドレス (ログイン名とマシン名) も指定します。たとえば、ログイン名が rose でマシン名が texas の場合、アドレスは rose@texas となります。(@ シンボルは「at」の意味)。ローカルネットワーク上でのログイン名を使うこともできます (疑問点についてはシステム管理者に問い合わせてください)。

```
$ mailx rose@texas
```

mailx プログラムは、Subject: という行を表示します。

```
$ mailx rose@texas
Subject:
```

ここで、メッセージの内容を説明する表題を入力して、Return キーを押します。次に、メッセージの本文を入力します。行の長さは短くおさえて、各行の終わりで Return キー を押します (Return キーを押す前ならば、Back Space キーで後退して再入力することによって文を修正できます)。

メッセージのサンプルを下記に示します (空白行は、Return キーで作成したものです)。

```
$ mailx rose@texas
Subject: to someone who really cares

Dear Rosey,

From the ends of your fingers
To the tip of your nose
You're a cool breeze in August
My sweet Texas Rose.


See you soon,
```

Rose

メッセージを送信するには、メッセージの最終行で Return キーを押してから Ctrl-D を押します。メッセージが送信されると、コマンドプロンプトの状態に戻ります。

自分宛てのメッセージの読み方

自分宛てのメッセージを読むには、`mailx` コマンドをもう一度入力します。画面に以下のようなメッセージが表示されます。

```
$ mailx
Mail version 4.0 Thu Jan 16 12:59:09 PST 1992  Type ? for help.
"/var/mail/rose": 2 messages 1 new
 U  2 hal@uncertain   Fri Feb 14 12:01   14/318 financial status
>N  1 rose@texas      Mon Feb 17 08:12   21/453 to someone who
?
```

最初の行には実行中の `mail` のバージョンが表示され、2 行目には到着するメールが収納されるメールボックス (通常は /var/mail/username にある) の場所が表示されます。この例では 3 行目に、自分に送ったメッセージに関する情報が表示されています。行の先頭の「N」(new を表す) は、これが「新しい」メッセージであることを示します。「U」(unread を表す) は、新しいメッセージが到着したのに、前回 `mailx` プログラムを実行したときには読まれなかったことを意味します (この画面の内容については、92 ページの「メッセージの読み方」で詳しく説明します)。

各メッセージには、受信時に番号が割り当てられます。ユーザ rose の自分宛てのメッセージには、`1` というメール番号が付いています。

メッセージを読むには、次に示すように、`mailx` プロンプトでメッセージ番号を入力します。

```
$ mailx
Mail version 4.0 Thu Jan 16 12:59:09 PST 1992  Type ? for help.
"/var/mail/rose": 1 message 1 new
>N  1 rose@texas   Fri Jul 14 12:01 21/453 to someone who
? 1

To: rose@texas
From: rose@texas
Subject: to someone who really cares

Dear Rose,

From the ends of your fingers
To the tip of your nose
You're a cool breeze in August
My sweet Texas Rose.


See you soon,
```

```
Rose

?
```

mailx の終了

mailx を使い終えたら、 q (quit) または x (exit) コマンドのどちらかを入力してプログラムを終了できます。

mailx プロンプトで q と入力して Return キーを押すと、次のようなメッセージが表示されます。

```
< ...> 個のメッセージを var/mail/username に継続して保存しました。
```

home_directory は、ホームディレクトリのパス名です。

mailx プロンプトに対して q と入力して mailx を終了した場合、mailx は読み終わったメッセージをメールボックスから削除してホームディレクトリ内の mbox ファイルに保存します。メールを変更したり削除した場合は、その内容も保存されます。

mailx プロンプトに対して x と入力して Return キーを押した場合、読み終わったメッセージは mbox ファイルに移動されず、変更や削除の内容も保存されません。

メッセージの読み方

メッセージが存在する場合、ユーザがログインするたびに mailx は次のどちらかのメッセージを表示してその旨を知らせます。

```
You have mail
```

または、

```
You have new mail
```

メッセージを読むには、コマンドプロンプトに対して mailx と入力し、Return キーを押します。メールがない場合は、次のメッセージが表示されます。

username 宛のメールはありません。

メールがある場合は、次のようなリストが表示されます。

```
$ mailx
Mail version 4.0 Thu Jan 16 12:59:09 PST 1992  Type ? for help.
"/var/mail/rose": 4 messages 1 new 2 unread
```

```
    1 rose@texas        Fri Feb 14 12:01 21/453 to someone who
 U  2 hank@fretful      Fri Feb 14 18:31 19/353 so lonely I
 U  3 farmer@freeway    Sat Feb 15 10:22 24/557 looks like my
>N  4 hoover@woofer     Sun Feb 16 23:59 14/280 big old furry

?
```

最初の行には、mailx プログラム自身に関する情報 (バージョン番号と日付) とヘルプの参照方法 (Type ? for help) が表示されます。

次の行には、メールボックスの場所、受信されたメッセージの数とその状態が表示されます。

その次の行には、メールボックス内のメッセージが番号付きで表示されます。各行のカラムの内容は、左から右に以下のとおりです。

- 状態: メッセージの状態を新規 (N)、読まれていない (U)、読まれた (記号なし) で示します。行の先頭にある「>」は、最新のメッセージを示します。削除されたメッセージは、アスタリスク (*) が付いて表示されます。
- 番号: メッセージが受信された順序を示します。
- 送信者: メッセージを送信したユーザ名 (および通常はマシン名も) を示します。
- 時間: メッセージが送信された日付と時刻を示します。
- サイズ: メッセージの行数およびバイト数を示します。
- 表題: 送信者が subject:行で入力したメッセージの表題が表示されます。

メールボックス内のメッセージが複数存在する場合は、このリストにすべてのメールが表示されないこともあります。その場合は、次のように入力します。

- z – メールの見出しリストの次の 1 画面を表示
- h- – メールの見出しリストの前の 1 画面を表示
- h – 任意の時点でメールの見出しリストを再表示

メールボックス内の最新のメッセージ (> 付きで表示されたもの) を参照するには、Return キーを押します。Return キーをもう一度押すと、次のメッセージが表示されます。リスト内の任意のメッセージを参照するには、その番号を入力して Return キーを押します。

メッセージの削除と削除の取り消し

メッセージを読み終えたら、mbox ファイルに保存しないで削除することもできます。デフォルトでは、mailx プログラムの終了時に mbox ファイルに保存されます。

読み終わった最後のメッセージを削除するには、mailx プロンプトに対して d と入力します。メールボックスから特定のメッセージを削除するには、次のコマンドを実行します。

d *number*

たとえば、2 番目のメッセージを削除するには、mailx プログラム内から次のコマンドを実行します。

```
? d 2
```

一度に複数のメッセージを削除することもできます。1 番目と3 番目のメッセージを削除するには、次のコマンドを実行します。

```
? d 1 3
```

一定範囲のメッセージを削除することもできます。たとえば 1 ～ 3 番目のメッセージを削除するには、次のコマンドを実行します。

```
? d 1-3
```

mailx を終了する前なら、メールボックスから削除したメッセージを復元 (削除の取り消し) できます。次の形式のコマンドを入力します。

u *number*

続いて、Return キーを押します。たとえば、2 番目のメッセージを復元するには、次のコマンドを実行します。

```
? u 2
```

前回の削除コマンドを取り消すには、削除した直後に mailx プロンプトに対して u と入力します。たとえば、前回の削除コマンドが d 2-5 の場合は、u と入力すると 2 ～ 5 番目のメールの削除が取り消されます。

q コマンドで mailx を終了すると、すべての削除操作が確定されます。つまり、削除したメッセージを復元することはできなくなります。ただし、x コマンドで mailx を終了すればメールボックスを以前の状態に保つことができます。前述のように、x コマンドで mailx を終了すると、読み終わったメッセージでも U が付けられて表示され、削除したメッセージも削除されなかったことになります。

メッセージの印刷

メッセージをプリンタコマンドにパイプすることによって、メッセージを印刷できます。メッセージの印刷には、mailx プロンプトに対して次の形式のコマンドを実行します。

| *number* lp

| 記号はパイプと呼ばれます。 たとえば、2 番目のメッセージを印刷するには、次のように入力します。

```
? |2 lp
```

続いて、Return キーを押します。メッセージの番号が指定されない場合は、現在のメッセージがプリンタにパイプされます。パイプについての詳細は、第 2 章の 25 ページの「コマンド出力のリダイレクトとパイプ」を参照してください。

メッセージの送信

mailx プログラムを使ってメールを送信する場合は、メッセージの受信者のログイン名が必要です。受信者が別のマシン上にいる場合は、そのユーザのマシン名も必要になります。これらの情報を調べるために、who、finger、rusers などのコマンドを使うことができます。

ファイルサーバに現在ログインしている全ユーザのリストを表示するには、who コマンドを入力します。このリストには、ユーザのログイン名、その端末の型、ログインした日付と時間が表示されます。次に例を示します。

```
$ who
    elmer      tty15        Feb 20 10:22
    susan      tty04        Feb 20 10:37
    stormy     tty07        Feb 20 11:49
    hankw      tty06        Feb 20 12:02
```

who コマンドと同様の情報を詳しく表示するには、finger コマンドを入力します。システム管理者によるこのコマンドの設定状態によって、表示される情報の内容は異なります。たとえば、次のような情報が表示されます。

```
$ finger
    Login     Name           TTY      Idle     When
    elmer     Elmer Brown    tty15     43      Thu 10:22
    susan     Susan Lake     tty04             Thu 10:37
    stormy    Stormy Ball    tty07     12      Thu 11:49
    hankw     Hank Wilson    tty06     22      Thu 12:02
```

rusers コマンドを入力すると、ローカルネットワークに現在ログインしているユーザに関する情報が表示されます。rusers コマンドの使い方についての詳細は、第 9 章を参照してください。

必要なユーザ情報を確認できたら、次の手順に従ってメッセージを送信します。

1. **mailx** コマンドのあとに続けてユーザのアドレスを入力します。

   $ **mailx** *user@machine*

   user は受信者のログイン名、*machine* は受信者のマシン名です。

   - mailx プログラムをすでに起動している場合は、mailx プロンプトに対して m のあとに受信者のログイン名とマシン名を @ で区切って入力します。

     ? **m** *user@machine*

- 同じメッセージを複数の受信者に送信する場合は、次の例のように各アドレスをスペースかコンマで区切ります。

  ```
  $ mailx hank@fretful sally@dakota tex@twister
  ```

  または、

  ```
  $ mailx hank@fretful,sally@dakota,tex@twister
  ```

2. **Return** キーを押すと、**mailx** プログラムはメッセージの表題を入力するよう **subject:** と表示してくるので、表題を入力してもう一度 **Return** キーを押します。

3. メッセージの本文を入力します。改行する場合は、**Return** キーを押します。
   画面上で文が複数行に渡って表示されていても、Return キーを押していなければ改行したとは見なされません。

   注 – メッセージの各行のテキストの最大長は 256 文字です。この制限を超えると、画面が動かなくなります。その場合は、Ctrl-C を押してメッセージの入力を中止してください。

   ---

4. メッセージの入力が終わったら、**Return** キーを押して改行します。次に、**Ctrl-D** を押してメッセージを送信します。

送信できないメッセージ

メッセージを送信するときに正しくないユーザアドレスを指定すると、システムの応答として次のようなメッセージが表示されます。

user@machine`...User unknown`

この場合、メッセージはメールボックスに戻されます。次回、`mailx` コマンドを入力したときに、次のように表示されてメールが届かなかった旨が知らされます。

```
N 1 Mailer-Daemon Fri Jan 3 11:13 8/49 Returned mail: User unknown
```

また、メッセージが届かない場合は、ホームディレクトリ内の `dead.letter` というファイルにそのメッセージがコピーされます。

送信しないメッセージの取り消し

送信前ならば Ctrl-C を 2 回押すことによって、メッセージの入力を取り消すことができます。

カーボンコピーとブラインドカーボンコピーの追加

メッセージを送信する前に、「カーボンコピー」を To: で指定した受信者以外に送信するよう指定できます。また、「ブラインドカーボンコピー」も送信できます。この指定では、メッセージの受信者はカーボンコピーの宛先アドレスを読み取ることができますが、ブラインドカーボンコピーのアドレスは読み取れません。

自分自身にカーボンコピーやブラインドカーボンコピーを送信すれば、独自の記録として送信したメールのコピーを保管できます。

メッセージと一緒にカーボンコピーを送信するには、次の 3 つの方法があります。

- あらかじめテキストエディタを使ってホームディレクトリ内の `.mailrc` ファイルを編集し、次の行を挿入します。

  `set askcc`

  `mailx` プログラムは、表題の入力 (Subject:) プロンプトの下にカーボンコピープロンプト (`Cc:`) を表示します。カーボンコピーの宛先となるユーザのアドレスを入力してください。宛先が複数の場合は、アドレスをスペースまたはコンマで区切ります。

- メッセージの本文を入力したあと Ctrl-D を押す前に Return キーを押して改行し、次のコマンドを実行します。

  ~c address(es)

  この方法によって 1 人または複数の受信者にカーボンコピーを送信する場合は、次の例のように各受信者のアドレスをスペースで区切ります。次に例を示します。

```
~c hank@fretful george@lonesome stormy@snoozer
```

- `Cc:` プロンプトは、`~h` コマンドでも表示できます。`~h` コマンドは、メッセージのすべての見出し行を表示します。つまり、`To:`、`Subject:`、、`Cc:`、`Bcc:` (ブラインドカーボンコピー) というプロンプトがすべて 1 行ずつ表示されます。その際に、各空白行には入力することができ、入力した行を修正することもできます。ほかのチルドコマンドと同様に、メールの本文作成中は常に改行してから `~h` コマンドを実行してください。

注 – `~c`、`~h` などのチルドコマンドについての詳細は、この章で後述の 107 ページの「チルドコマンド」を参照してください。

メッセージやファイルのコピーを挿入する

メールボックス内にある任意のメッセージのコピーを現在作成中のメッセージに挿入することができます。同様に、任意のテキストファイルのコピーも挿入できます。

メッセージの挿入

メッセージの本文作成中に他のメッセージを挿入するには、次の形式のコマンドを使います。

~m *number*

number は、挿入するメッセージの番号です。たとえば、メールボックスの 3 番目のメッセージのコピーを取り込んで新規のメッセージを作成し、それをほかのユーザに送信する場合は、次の手順を実行します。

1. メールの本文作成中に改行してから ~m 3 というコマンドを入力し、Return キーを押します。

   mailx により、次のメッセージが表示されます。

   「メッセージ挿入」 3 (continue)
2. 送信者の画面には 3 番目のメールの内容は表示されませんが、受信者の画面には表示されます。(続く) のあとに続けてメッセージを入力するか、あるいは取り込んだコピーをそのまま送信します。
3. 取り込んだコピーも含めたメッセージ全体を表示するには、~p コマンドを入力します。

ファイルの挿入

メッセージの本文作成中に任意のテキストファイルのコピーを挿入することもできます。この場合、次の形式のコマンドを入力します。

~r *filename*

たとえば、現在作成中のメッセージに outline というファイルを挿入するには、次のように入力します。

```
?~r outline
```

メッセージへ応答する

メールに返事を出すには、mailx プロンプトに対して次のコマンドを入力します。

r *number*

メッセージの番号を入力しないと、現在のメッセージへの応答となります。

たとえば、2 番目のメッセージの送信者に応答するには、次のコマンドを入力します。

```
? r 2
```

mailx は自動的に応答先のアドレスを指定し、Re: Subject:という行を表示します。この行には応答するメッセージの Subject: 行の内容が表示されます。この応答メッセージは、通常のメッセージと同様に送信できます。

R コマンドは r コマンドの変形で、送信者だけでなく、応答するメッセージの受信者全員に対しても応答を送ります。不必要なメールが作成されるのを防ぐため、このコマンドは本当に必要な場合にだけ使うようにしてください。

注 – 前節で説明したのと同様の方法で、応答にもメッセージのコピーを挿入できます。応答するメッセージのコピーを挿入するには、メッセージの番号を指定しないで ~m コマンドを実行します。

メールの保存と読み取り

メッセージの送受信のほかに、メッセージを保存しておき、そのメッセージをあとで読むことができます。mailx では、通常のテキストファイルにメッセージを追加することによってメッセージを保存できます。また、フォルダと呼ばれる特殊ファイルにメッセージを追加することもできます。これらの方法について次に説明します。

mailx では、メッセージの保存とメッセージのコピーは区別されます。メッセージを保存すると、そのメッセージはメールボックスから削除されてファイルやフォルダに追加されます。一方、メッセージをコピーすると、メッセージはメールボックス内に残り、そのコピーがファイルやフォルダに追加されます

メッセージをファイルに保存およびコピーする

メッセージをファイルに保存するには、mailx プロンプトに対して次の形式のコマンドを実行します。

s *number filename*

number は保存するメッセージの番号、*filename* はメッセージが保存されるファイル名です。たとえば、3 番目のメッセージを ~/notes/finance というファイルに保存するには、次のように入力します。

```
? s 3 ~/notes/finance
```

パス名中の ~ は、各ユーザのホームディレクトリを示します。

複数のメッセージを一度に同じファイルに保存することもできます。たとえば、3 、5 、6 、7 、8 番目のメッセージを~/notes/finance に保存するには、次のように入力します。

```
? s 3 5-8 ~/notes/finance
```

指定したファイルが存在しない場合は、mailx によって作成されます。ファイルが存在する場合は、保存するメッセージはファイルの終わりに追加されます。

メッセージをファイルに保存すると、そのメールはメールボックスから削除されます。この場合は、保存されたメッセージの見出しにアスタリスク (*) が表示されます。

3 番目のメールをメールボックスに残したまま別のファイルに追加 (コピー) する場合は、次のように copy コマンドを入力します。

```
? c 3 ~/notes/finance
```

メッセージをフォルダに保存およびコピーする

メールフォルダにメッセージを保存またはコピーする場合は、ファイルの絶対パス名を指定する必要はありません。フォルダは、フォルダディレクトリに保管されている特殊ファイルです。

フォルダにメッセージを保存またはコピーすると、同じディレクトリ内にメッセージが自動的に保管され、長いパス名を入力しなくてもそのディレクトリに簡単にアクセスできます。

フォルダディレクトリの設定

フォルダを使うには、まず最初にフォルダディレクトリ (フォルダ用のディレクトリ) を設定する必要があります。この設定には次の作業を行います。

1. mkdir コマンドを使ってディレクトリを作成します。

   たとえば、Messages というフォルダディレクトリを設定するには、次のようなディレクトリを作成します。

   ```
   $ mkdir Messages
   ```

2. フォルダディレクトリのパスを設定するために、テキストエディタを使ってホームディレクトリ内の .mailrc ファイル (mailx オプションが入っている) を編集します。

   次の例のように、新しく作成したフォルダディレクトリの絶対パス名を指定するために、folder 変数を編集します。次に例を示します。

   ```
   set folder=/home/austin/rose/Messages
   ```

   C シェルの短縮名である ~ を使ってホームディレクトリを指定することもできます。

   ```
   set folder=~/Messages
   ```

これでフォルダディレクトリは、フォルダに保存されるメッセージを受け取るよう設定されます。.mailrc ファイルの変更内容は、次回 mailx を起動したときに有効になります。

フォルダの指定

フォルダにメッセージを保存またはコピーするときは、ファイルの場合と同じコマンドを使います。ただし、フォルダ名の前にはパス名の代わりにプラス記号 (+) を付けます。+ は、フォルダをフォルダディレクトリ (ここでは Messages) に保存するよう mailx に指示します。

たとえば、3 番目のメッセージを projects というフォルダに保存するには、次のように入力します。

```
$s 3 +projects
```

mailx は、このコマンドを「3 番目のメールを ~/Messages/projects に保存する」という意味に解釈します。指定のフォルダが存在しない場合は、mailx によって作成されます。

メッセージをフォルダにコピーする場合は、次のように入力します。

```
$c 3 +projects
```

ファイルやフォルダにメッセージを直接送信する方法

メッセージのコピーは、任意のファイルやフォルダに直接送信できます。フォルダにコピーを送信する場合は、Cc: プロンプトまたは Bcc: プロンプトに対してフォルダ名 (たとえば、projects) を入力するだけです。ファイルにコピーを送信する場合も同様ですが、ファイルの絶対パス名を指定する必要があります。

ファイルやフォルダ内のメッセージを読み取る

ファイル内に保存されたメッセージを読み取るには、次の形式のコマンドを使います。

mailx -f *filename*

上記の例では、~/memos/finance ファイル内のメッセージを読み取るには、次のように入力します。

```
$ mailx -f ~/memos/finance
```

フォルダに保存されたメッセージを読み取る場合も同じコマンドを使います。その場合は、絶対パス名の代わりに + 記号を使います。たとえば、projects フォルダ内のメッセージを読み取るには、次のように入力します。

```
$ mailx -f +projects
```

このコマンドは、指定されたファイルやフォルダに対してmailx を起動し、それらのファイルやフォルダ内に存在するメッセージの見出しを表示します。読み取るメッセージを選択するには、mailx プロンプトに対してメッセージの番号を入力し、Return キーを押します。

mailx プログラム内からもフォルダを操作できます。フォルダのリストを参照するには、mailx プロンプトに対して次のコマンドを入力します。

```
? folders
```

メールボックスからフォルダに切り換えるには、次の形式のコマンドを使います。

? *folder +foldername*

メールボックスに戻るには、mailx プロンプトに対して次のコマンドを入力します。

```
? %
```

mailx を起動したなかで最後に使用したフォルダに戻るには、次のように入力します。

```
? #
```

mailx での vi の使い方

mailx の実行中に vi テキストエディタを使ってメッセージを作成できます。このエディタにより、メッセージを送信する前にテキストの追加や削除、誤りの修正などができます。vi に慣れていない場合は、第 6 章を参照してください。

mailx プログラムでは、vi の標準的なコマンドを使ってテキストの挿入、削除、変更ができます。

vi を使ってメッセージを作成するには、次の手順に従います。

1. **mailx** プロンプト **(?)** で **m** の後にアドレスを指定するか、コマンドプロンプトでアドレスを指定した **mailx** コマンドを入力します。
2. **Subject:** 行に表題を入力して、**Return** キーを押します。
3. 改行してから **~v** コマンドを入力して **vi** を起動します。
   vi の画面に /tmp ディレクトリ内の空ファイルが表示されます。
4. **vi** のコマンドを使ってメッセージの本文を入力し編集します。
5. 編集が完了したところで、**:wq Return** キーまたは **ZZ** コマンドで **vi** を終了します。
   vi を終了すると、mailx によって (続く) : というメッセージが表示されます。この時点で、(vi を使用せずに) メッセージに何か書き加えることも、Ctrl-D を押し

てメッセージを送信することもできます。

メールエイリアス

メールエイリアスとは、単一の名前によってまとめられたユーザ名の集合です。

メッセージを同一のユーザグループに繰り返し送信する場合は、メールエイリアスを使うと便利です。たとえば、hank@fretful、george@lonesome、sally@dakota に頻繁に送信する場合は、amigos というメールエイリアスを作成できます。それ以降 amigos 宛てにメールを送信すると、この 3 ユーザがすべてそのメールを受信することになります。

メールエイリアスは、次の2 箇所で設定できます。

- .mailrc ファイル
- /etc/aliases ファイル

.mailrc ファイルで設定されたメールエイリアスと /etc/aliases ファイルで設定されたメールエイリアスとでは、その働きが異なります。それらの相違点は、この節の終わりにある表 7–1 にまとめてあります。

.mailrc ファイルでメールエイリアスを設定する

.mailrc ファイルでメールエイリアスを設定する場合は、次の点に注意してください。

- .mailrc 内のメールエイリアスはユーザ専用です。つまり、そのエイリアスを使えるのは、それを設定したユーザ自身に限られます。たとえば、あるユーザが .mailrc ファイル内で設定した amigos というメールエイリアスに対してほかのユーザがメールを送信しようとすると、「unknown user」というエラーメッセージが出力されます。
- メールが送信されると、.mailrc 内のメールエイリアスはそれが示す全ユーザ名に自動的に展開されます。たとえば、amigos にメールを送信すると、そのメールは amigos 内の全ユーザ名を受信者として指定した場合とまったく同様に送信されます。受信者側からは、メールの送信にメールエイリアスが使われたかどうかはわかりません。

.mailrc ファイルは、各ユーザのホームディレクトリ内にあります。このファイルには、mailx とメールツールの動作を制御する設定がいくつか入っています。

.mailrc にメールエイリアスを追加するには、次のように入力します。

```
$ vi ~/.mailrc
```

注 – .mailrc ファイルの編集には任意のテキストエディタを使えます。上記の例は、vi エディタを使って編集する方法を示しています。vi に慣れていない場合は、第 6 章を参照してください。

各メールエイリアスは、.mailrc ファイル内で 1 行ごとに記述されています。つまり、複数の行に渡っていても、途中にキャリッジリターンが入っていることはありません。各メールエイリアスには、以下の情報をスペースで区切って記述しなければなりません。

- 「alias」というワード
- メールエイリアスの名前 (1 ワードのみ)
- メールエイリアスに含まれる受信者 (ログイン名とマシン名) で、それぞれスペースで区切る

次の例は、2 つのメールエイリアスを示しています。最初のエイリアス (amigos) には 3 ユーザ、2 番目のエイリアス (softball) には 8 ユーザが含まれています。softball の行では、名前が画面上では複数行に渡って表示されていますが、Return キーが入力されていない限り問題はありません。

```
alias amigos hank@fretful george@lonesome sally@dakota
alias softball earl@woofer tex@twister elmer@farmhouse
jane@freeway hank@fretful jj@walker sally@dakota steve@hardway
```

.mailrc ファイル内のエイリアスに含まれるユーザにメールを送信する場合は、そのメールエイリアスをアドレスに指定し、マシン名は指定しないようにします。たとえば、次のメッセージを送信したとします。

```
$ mail amigos
Subject: Let's eat

Hey Compadres. How about
getting together for lunch on Friday?
Anyone interested?
```

受信者は、次のようなメールを受け取ります。To: の行が展開されていることに注意してください。

```
To: hank@fretful george@lonesome sally@dakota
Subject: Let's eat

Hey Compadres. How about getting together for lunch on Friday?
Anyone interested?
```

/etc/aliases でメールエイリアスを設定する

/etc/aliases ファイルでメールエイリアスを設定する場合は、次の点に注意してください。

- /etc/aliases 内のメールエイリアスは公共用です。つまり、/etc/aliases 内で softball というメールエイリアスを設定すれば、全ユーザがこのエイリアスを利用して softball@*your-machinename* 宛てにメールを送信できます。
- メールが送信されても、/etc/aliases 内のメールエイリアスは展開されません。たとえば、softball@*machinename* にメールを送信すると、エイリアスは展開されずそのままの形で受信されます。受信者側には、メールエイリアスの名前は分かりますが、そのエイリアスに含まれるほかのユーザ名はわかりません。

/etc/aliases に作成されるメールエイリアスの形式は、.mailrc の形式とは若干異なります。 /etc/aliases 内の各エイリアスは、次の形式でなければなりません。

- メールエイリアスの名前とそのあとのコロン (:)
- コンマで区切った受信者 (ログイン名とマシン名)。メールエイリアスの指定は複数行に渡ってもかまいません。

/etc/aliases ファイルを変更するには、最初に root ユーザになる必要があります。root がパスワードで保護されている場合は、root のパスワードが必要です。

次のコマンドを入力してシステムの root ユーザになります。

```
$ su
Password:
#
```

root ユーザになるとコマンドプロンプトが変わることに注意してください。

次の例は、メールエイリアス softball@texas をデフォルトの /etc/aliases ファイルに追加する方法を示しています。

```
# vi /etc/aliases
##
#Aliases can have any mix of upper and lower case on the left-
#hand side,
#but the right-hand side should be proper case (usually lower)
#
#>>>>>>>>>>The program "newaliases" will need to be run after
#>> NOTE>>this file is updated for any changes to
#>>>>>>>>>>show through to sendmail.
#
#@(#)aliases 1.10 89/01/20 SMI
##
# Following alias is required by the mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's
mail problems.
Postmaster: root

# Alias for mailer daemon; returned messages from our MAILER-
DAEMON
# should be routed to our local Postmaster.
MAILER-DAEMON: postmaster

# Aliases to handle mail to programs or files, eg news or vacation
```

```
# decode: "|/usr/bin/uudecode"
nobody: /dev/null

# Sample aliases:
# Alias for distribution list, members specified here:
#staff:wnj,mosher,sam,ecc,mckusick,sklower,olson,rwh@ernie

# Alias for distribution list, members specified elsewhere:
#keyboards: :include:/usr/jfarrell/keyboards.list

# Alias for a person, so they can receive mail by several names:
#epa:eric

########################
# Local aliases below #
########################
softball@texas: earl@woofer
tex@twister elmer@farmhouse
jane@freeway hank@fretful jj@walker sally@dakota steve@hardway
:wq          (vi を終了して /etc/aliases ファイルを保存する)
# exit       (root ユーザを終了する)
$
```

/etc/aliases ファイルの編集には任意のテキストエディタを使えます。上記の例は、vi エディタを使って編集する方法を示しています。vi に慣れていない場合は、第 6 章を参照してください。

/etc/aliase ファイル内のポンド記号 (#) は、そのあとのテキストやエイリアスの例が注釈であることを示しています。先頭にポンド記号が付いた行の情報は、実際のエイリアスとしては処理されません。

意図的にエイリアスを使わないようにする場合以外は、このファイルに追加するエイリアスの前にはポンド記号を付けないでください。

/etc/aliases 内のエイリアスに含まれるユーザに対してメールを送信する場合は、そのエイリアスとマシン名をメールのアドレスとして指定します。たとえば、次のメールを送信したとします。

```
$ mail softball@texas
Subject: Practice Today

Let's meet at the diamond
behind Building 4 after work tonight.
Goodness knows we can use the practice for Saturday's game! Be
there as early as you can.
```

受信者は、次のようなメールを受け取ります。

```
To: softball@texas
Subject: Practice Today

Let's meet at the diamond behind Building 4 after work tonight.
Goodness knows we can use the practice for Saturday's game! Be
there as early as you can.
```

To: の行が展開されていないことに注意してください。

/etc/aliases で設定されたメールエイリアスを使ってメールを送信する場合は、そのエイリアスが設定されているマシン名を必ず指定します。たとえば、マシン freeway 上で riders というメールエイリアスを設定した場合は、riders@freeway と指定してメールを送信しなければなりません。

.mailrc で設定されたメールエイリアスと /etc/aliases で設定されたメールエイリアスとの相違点を表 7-1 にまとめます。

表 **7–1** .mailrc と /etc/aliases のメールエイリアス比較

	.mailrc	/etc/aliases
変更するために root になる必要性	なし	あり
メッセージを送信するときのアドレス	*alias*	*alias@machinename*
To: 行と Cc: 行で受信者の一覧がつくか	付く	付かない
名前をコンマで区切るか	区切らない	区切る
すべての名前を 1 行内に記述するか	する	しない
他のユーザがメールエイリアスを使えるか	使えない	使える

メールエイリアスについての詳細は、システムプロンプトに対して man aliases または man addresses と入力してマニュアルページを参照してください。

チルドコマンド

メッセージの作成中に、チルドコマンドを使ってさまざまな機能を実行できます。通常、チルドコマンドはチルド文字 (~) のあとに 1 文字を付けたものです。次の表 7–2 では、便利なチルドコマンドをいくつか説明しています。これらのコマンドの一部はすでにこの章で紹介したものです。

注 – メッセージにチルド文字そのものを使いたい場合は、チルド文字を 2 つ入力します。これによりチルド文字が 1 つだけ表示されます。

表 **7–2** チルドコマンド (mailx)

コマンド名	説明
~!*command*	シェルコマンドを実行する
~.	ファイルの終わりを示すCtrl-D と同じ機能
~?	チルドコマンドの概要を表示する
~b *username*	ブラインドカーボンコピー (Bcc:) のリストにユーザ名を追加する
~c *username*	カーボンコピー (Cc:) のリストにユーザ名を追加する
~d	dead.letter ファイルの内容を現在作成中のメッセージに読み込む
~f *number*	指定されたメッセージを送信する。メールの読み取り中にメッセージを送信する場合のみ有効
~h	見出し行のプロンプト (Subject:、To:、Cc:、Bcc:) を表示する
~m *number*	現在作成中のメッセージに、指定されたメッセージを挿入する。メールの読み取り中にメッセージを送信する場合のみ有効
~p	入力中のメッセージを画面に表示する
~q	Ctrl-C を 2 回押すのと同じ機能。作成中のメッセージ本文が空でない場合は、その内容が dead.letter ファイルに保存される
~r *filename*	指定されたファイルを読み込む
~s *string*	表題の行 (Subject:) を *string* に変更する
~t *name*	指定された名前を To: リストに追加する
~w *filename*	現在作成中のメッセージから見出し行を取り除いたものを指定されたファイルに書き込む
~x	mailx を終了する。メッセージが dead.letter ファイルに保存されない点以外は ~q と同じ

ヘルプコマンド: その他の mailx コマンド

mailx には、コマンドとその機能のリストを表示するヘルプコマンドが 2 種類あります。コマンドモードで使われるコマンドのリストを参照するには、mailx プロンプト (?) に対して ? を入力します。同様に、入力モード (メッセージの作成中など) では、同等のコマンドである ~? を入力すれば、チルドコマンド (チルドエスケープとも呼ばれる) のリストを参照できます。

マニュアルページでは、mailx コマンドをより技術的に詳しく説明しています。mailx のマニュアルページを参照するには、次のコマンドを入力します。

```
$ man mailx
```

詳細は、『*manpages section 1 : User Commands*』を参照してください。

第 8 章

プリンタの使い方

LP (ラインプリンタ用サブシステム) プリントサービスは、Solaris オペレーティング環境で使用できる印刷ツールを提供します。LP プリントサービスには多種多様の機能がありますが、このマニュアルではその一部分だけを取り扱います。この章では、LP プリントサービスを使って、次に示す基本的な印刷作業を実行するために必要な手順を説明します。

LP プリントサービスについての詳細は、『*Solaris* のシステム管理 *(*上級編*)*』を参照してください。

印刷要求の実行依頼

コマンドプロンプトからファイルを印刷するには、そのファイルを印刷するプリンタに対して `lp` コマンドを使って要求を送信します。要求が送信されると、LP プリントサービスはその要求を印刷待ち行列に登録し、要求 ID 番号を表示したあと、シェルプロンプトを再表示します。

デフォルトプリンタへの印刷要求の実行

システム管理者がデフォルトプリンタを指定して LP 印刷サービスを設定している場合、次のコマンドによりプリンタ名を入力することなく印刷要求を送信できます。

```
$ lp filename
```

filename は、印刷するファイル名です。

指定されたファイルはデフォルトプリンタの印刷待ち行列に登録され、要求 ID 番号が表示されます。

たとえば、/etc/profile ファイルを印刷するには、次のコマンドを入力します。

```
$ lp /etc/profile

request id is jetprint-1 (1 file)

$
```

デフォルトプリンタの指定方法については、『*Solaris* システム管理 (上級編)』を参照してください。

プリンタ名を指定した印刷要求

システム上でのデフォルトプリンタの指定の有無にかかわらず、接続されている任意のプリンタに対して印刷要求を実行できます。特定のプリンタに印刷要求を送るには、次のコマンドを入力します。

```
$ lp -d printername filename
```

printername は特定のプリンタの名前、*filename* は印刷するファイルの名前です。

指定されたファイルは宛先プリンタの印刷待ち行列に登録され、要求 ID 番号が表示されます。

たとえば、/etc/profile ファイルを fastprint というプリンタで印刷するには、次のコマンドを入力します。

```
$ lp -d fastprint /etc/profile

request id is fastprint-1 (1 file(s))

$
```

システム上で接続されていないプリンタ (ここでは thorn) に対して印刷要求を行うと、次の例に示すようなメッセージが表示されます。

```
$ lp -d newprint /etc/profile

newprint: unknown printer

$
```

プリンタの接続方法については、『*Solaris* のシステム管理 (上級編)』を参照してください。また、システムで利用できるプリンタの確認方法については、114 ページの「プリンタの状態を確認する」を参照してください。

印刷完了通知の要求

大きいファイルの印刷要求を行う場合は、印刷処理が完了したときに LP プリントサービスから完了通知があると便利です。LP プリントサービスに対しては、次の 2 つの形式の完了通知を要求できます。

- 電子メール (email) メッセージの送信
- コンソールウィンドウに対するメッセージの書き込み

電子メールによる通知を要求するには、印刷要求時に `-m` オプションを指定します。

```
$ lp -m filename
```

コンソールウィンドウに対するメッセージの書き込みを要求する場合は、印刷要求時に `-w` オプションを指定します。

```
$ lp -w filename
```

filename は、印刷するファイル名です。

複数部数の印刷

`lp` コマンドに `-n` オプションを指定することにより、1 つのファイルを複数部印刷できます。

複数部の印刷を要求するには、次の形式のコマンドを入力します。

```
$ lp -n number filename
```

number は印刷する部数、*filename* は印刷するファイル名です。このような印刷要求は 1 つのプリントジョブと見なされ、ヘッダページは 1 ページだけ印刷されます。

たとえば、ファイル `/etc/profile` を 4 部印刷するには、次のコマンドを入力します。

```
$ lp -n4 /etc/profile

request id is jetprint-5 (1 file)

$
```

lp コマンドのオプション

`lp` コマンドにオプションを指定することにより、印刷要求をカスタマイズできます。`lp` コマンドで頻繁に使われるオプションを表 8–1 にまとめてあります。これらのオプションは、コマンド行で 1 つだけ指定しても任意の順序で組み合わせて指定してもかまいません。 複数のオプションを組み合わせる場合は、各オプションの前にハイフン (-) を付け、各オプション間はスペースで区切って指定します。

たとえば、プリンタの指定と電子メールによる通知の要求を行い、さらにファイルの印刷部数を 6 部と指定する場合は、次のコマンドを入力します。

```
$ lp -d printername -m -n6 filename
```

printername は印刷するプリンタの名前、*filename* は印刷するファイル名です。

表 **8–1** 頻繁に使われる `lp` オプション

オプション	説明
`-d`	印刷宛先。宛先のプリンタ名を指定する
`-m`	メール。ファイルの印刷が正常終了したときに、印刷要求元に電子メールを送信する
`-n`	部数。印刷の出力部数を指定する
`-t`	表題。印刷要求の表題 (バナーページ上にだけ印刷される) を指定する
`-o nobanner`	オプション。個別の印刷要求に関するバナーページの印刷を抑止する
`-w`	メッセージ書き込み。ファイルの印刷が正常終了したときに、コンソールウィンドウにメッセージを書き込む.

`lp` コマンドオプションの詳細なリストについては、『*man pages section 1: User Commands*』を参照してください。

プリンタの状態を確認する

LP プリントサービスの状態を調べるには、`lpstat` コマンドを使います。`lpstat` を使って、印刷待ち行列内のジョブ状態の検査、利用可能なプリンタの確認、ジョブを取り消すためのリクエスト ID の確認などを行うことができます。

印刷要求の状態を確認する

印刷要求の状態を調べるには、次のコマンドを入力します。

```
$ lpstat
```

印刷要求を実行したファイルのリストが表示されます。

次の例では、プリンタ `jetprint` で印刷するファイルが 1 つ待ち行列に登録されています。

```
$ lpstat

jetprint-1              user2            11466   Nov 01 15:10 on jetprint

$
```

lpstat コマンドは、印刷ジョブごとに 1 行ずつ要求 ID、その要求をスプールしたユーザ、バイト単位の出力サイズ、印刷要求を実行したシステムと日付と時刻に関する情報を表示します。

利用可能なプリンタを確認する

システムに接続されているプリンタを確認するには、次のコマンドを入力します。

```
$ lpstat -s
```

最初にスケジューラの状態が表示され、その後デフォルトの宛先プリンタ、利用可能なシステムとプリンタのリストが表示されます。

次の例ではスケジューラが実行中であり、デフォルトプリンタは jetprint です。プリンタ jetprint と fastprint の印刷サーバは prtsrv1 です。

```
$ lpstat -s

scheduler is running

system default destination: jetprint

system for jetprint: prtsrv1

system for fastprint: prtsrv1

$
```

状態情報をすべて表示する

lpstat コマンドに -t オプションをつけると、LP プリントサービスの全状態を要約したリストが表示できます。

このリストを表示するには、次のコマンドを入力します。

```
$ lpstat -t
```

利用可能な状態情報がすべて出力されます。

次の例では、印刷待ち行列内にジョブが存在しません。ファイルが印刷のためにスプールされると、それらの印刷要求の状態も表示されます。

```
$ lpstat -t

scheduler is running
```

```
system default destination: jetprint

system for jetprint: prtsrv1

system for fastprint: prtsrv1


jetprint accepting requests since Wed Nov  1 15:09:29 MST 2000

fastprint accepting requests since Wed Nov  1 15:09:47 MST 2000

printer fastprint is idle. enabled since Wed Nov  1 15:09:46 MST 2000.

jetprint-1              user2            11466   Nov 01 15:10 on jetprint

$
```

プリンタの状態を表示する

lpstat コマンドに -p オプションを指定することにより、個々のプリンタについての状態情報を要求できます。このオプションは、プリンタの活動状況 (アクティブまたはアイドル)、使用可能または不可能になった時間、印刷要求の受付の可否などを表示します。

システム上の全プリンタに関する状態情報を表示するには、次のコマンドを入力します。

```
$ lpstat -p
```

次の例は、2 つのプリンタがアイドル状態、使用可能、印刷要求受付可能であることを示しています。

```
$ lpstat -p

printer jetprint is idle. enabled since Wed Nov  1 15:09:28 MST 2000.

   available.

printer fastprint is idle. enabled since Wed Nov  1 15:09:46 MST 2000.

   available.

$
```

これらのプリンタの印刷待ち行列にジョブが存在すれば、そのジョブ情報も表示されます。

個々のプリンタに関する状態情報を表示するには、次のコマンドを入力します。

```
$ lpstat -p printername
```

printername は、特定のプリンタの名前です。

lpstat コマンドのオプション

lpstat コマンドを使うと、プリンタの状態に関するさまざまな情報を表示できます。lpstat コマンドで頻繁に使われるオプションを表 8–2 にまとめてあります。これらのオプションは、コマンド行で 1 つだけ指定しても、任意の順序で組み合わせて指定してもかまいません。 複数のオプションを組み合わせる場合は、各オプションの前にハイフン (-) を付け、各オプション間はスペースで区切って指定します。

表 **8–2** 頻繁に使われる lpstat オプション

オプション	説明
-a	受付状態。印刷宛先プリンタの要求受付状況を表示する
-c	クラス。プリンタクラスとそのメンバを表示する
-d	印刷宛先。デフォルトの印刷宛先プリンタを表示する
-f	書式。印刷書式を表示する
-o	出力。印刷要求の状態を表示する
-p [*list*] [-D] [-l}	プリンタ指定・説明・詳細リスト。プリンタの状態を表示する
-r	要求。要求スケジューラの状態を表示する
-R	印刷待ち行列内のジョブの位置を表示する
-s	状態情報。要約した状態情報を表示する
-S	文字セット。文字セットを表示する
-u [*username*]	ユーザ。要求の状態をユーザごとに表示する
-v	プリンタのデバイス名を表示する

lpstat コマンドオプションの詳細なリストについては、『*man pages section 1: User Commands*』を参照してください。

印刷要求の取り消し

印刷要求が待ち行列内にあるときや印刷実行中に要求を取り消すには、cancel コマンドを使います。要求を取り消す場合は、印刷の要求 ID が必要になります。要求 ID は、必ずプリンタ名、ハイフン、印刷要求番号から構成されています。印刷要求を実行すると、その要求 ID が表示されます。この要求 ID を忘れた場合は、lpstat -o コマンドを入力して Return キーを押すと確認できます。印刷要求を取り消せるのは、その要求を実行したユーザ、あるいは root または lp としてログインしたユーザだけです。

ID 番号による印刷要求の取り消し

指定の ID 番号を使って印刷要求を取り消すには、次のコマンドを入力します。

```
$ cancel request_id
```

request_id 指定する印刷要求のID 番号(*printername-number* から成る)です。

要求が取り消されることを知らせるメッセージが表示され、待ち行列内の次のジョブの印刷が開始されます。

次の例では、2 つの印刷要求が取り消されています。

```
$ cancel jetprint-6 fastprint-5

jetprint-6: cancelled

fastprint-5: cancelled

$
```

プリンタ名による印刷要求の取り消し

リクエスト ID の代わりにプリンタ名を入力しても、現在印刷中のジョブ (自分で実行依頼したものに限る) を取り消すことができます。

```
$ cancel printername
```

printername は、要求の送信先となったプリンタの名前です。

要求が取り消されることを知らせるメッセージが表示され、待ち行列内の次のジョブの印刷が開始されます。

次の例では、現在の印刷要求が取り消されています。

```
$ cancel jetprint

jetprint7: cancelled

$
```

システム管理者は root または lp としてログインし、プリンタ名を `cancel` コマンドの引数に指定することによって、現在印刷中の任意の要求を取り消すことができます。

第 9 章

ネットワークの使い方

ネットワークとは、相互に通信できるように設定されたコンピュータのグループです。使っているマシンがネットワークの一部である場合は、自分のマシン (ローカルマシン) にログインしたままでネットワーク上のほかのマシン (リモートマシン) の資源を利用できます。ユーザは、ほかのマシンに影響を及ぼすリモートコマンドを自分のマシンから実行したり、ほかのマシンにログインしたりすることができます。

この章の内容は次のとおりです。

使っているマシンがネットワークに接続されていない場合は、この章で説明する内容は現在の環境には適用できません。ただし、この章を一通り読めば、ネットワークの利点を大まかに理解することができます。

ネットワークの一般概念

マシン間をネットワークで接続することによって、相互に情報を伝送できます。一般的なネットワークとして次のようなものがあります。

- ローカルエリアネットワーク (LAN) — 小さな領域 (一般に数千フィート未満) に渡るネットワーク
- 広域ネットワーク (WAN) — 何千マイルもの範囲に渡ることがある大きなネットワーク
- キャンパスエリアネットワーク (CAN) — 中規模サイズのネットワーク

いくつかのネットワークが結びついて構成されているネットワークは、インターネットワークと呼ばれます。たとえば、ビル内部のローカルなネットワークの一部であるマシンが、同時にそのローカルなネットワークと国中にまたがる同種類のネットワークとを接続するインターネットワークの一部になることもあります。通常、ネットワークとインターネットワークとの違いはユーザの目には触れないため、このマニュアルでは「ネットワーク」という用語をネットワークとインターネットワークの両方の意味で使います。

ネットワークに接続されたマシンは、ネットワークプロトコル (共通のネットワーク言語) を使って通信を行います。ネットワークプロトコルにより、情報はネットワーク上の適切な場所に確実に転送されます。リレーとも呼ばれるインターネットワークプロトコルは、ネットワークを相互にリンクするために使われます。

リモートログイン (rlogin)

rlogin コマンドを使って、ネットワーク上のほかの UNIX マシンにログインできます。

現在使用しているマシン以外にログイン (リモートログイン) を行うには、次のコマンドを入力します。

```
$ rlogin machinename
```

machinename は、リモートマシンの名前です。

パスワードの入力を促すプロンプトが表示されたら、リモートマシン上でのパスワードを入力して Return キーを押します。使用しているマシン名がリモートマシン上の /etc/hosts.equiv ファイル内に記述されていれば、そのリモートマシンはパスワードの入力を要求しません。

```
venus$ rlogin starbug -l user2

Password:

Last login: Wed Nov  1 13:08:36 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

venus$ pwd

/home/user2

venus$ logout

Connection closed.

venus$
```

ホームディレクトリなしの rlogin

上記の例では、pwd コマンドで示されているように、user2 というユーザがリモートマシン starbug にログインし、そのマシン上の /home/user2 ディレクトリに移動しています。自分のアカウントが存在しないマシンにログインした場合は、そのリモートマシン上にホームディレクトリがないというメッセージが rlogin によって表示され、そのマシンのルートディレクトリ (/) に移動します。

```
venus$ earth -l user2

Password:

No directory! Logging in with home=/

Last login: Thu Nov  2 12:51:57 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

earth$ pwd

/

earth$ logout

Connection closed.

earth$
```

現在のログイン名以外での rlogin

rlogin コマンドの -l オプションを使うと、ほかのユーザとしてリモートマシンにログインできます。このオプションは、ほかのユーザのマシンでその人のユーザ名を使って作業をしているときに、自分のマシンに自分のログイン名でログインする場合などに便利です。

rlogin コマンドの -l オプションには、次のコマンド構文を使用します。

```
# rlogin machinename -l username
```

次の例は、マシン venus のユーザ user2 がマシン starbug に user1 としてログインする方法を示しています。

```
venus$ rlogin starbug -l user1

Password:

Last login: Thu Nov  2 12:51:57 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

starbug$ pwd
```

```
/home/user1

starbug$ logout

Connection closed.

starbug$
```

現在のログイン名以外でリモートマシンにログインすると、指定したユーザのホームディレクトリに移動します。

未登録のマシンに対する rlogin

名前が認識できないリモートマシンにログインしようとすると、rlogin はホスト名データベースの検索に失敗し、次のようなメッセージを表示します。

```
$ rlogin stranger

stranger: unknown host

$
```

rlogin セッションの中止

通常は作業セッションの最後に logout と入力して rlogin セッションを終了します。この方法でセッションを終了できない場合は、チルド文字とピリオド (~.) を入力することによって rlogin セッションを中止できます。これによりリモートマシンとのログインセッションは中止され、ユーザは自分のマシンに戻ります。

あるリモートマシンからそれ以外のリモートマシンにアクセスすることによって、複数のマシンにログインしている状態で、~. を使ってそれらのマシンのどれかとのセッションを中止した場合も、最初にログインした自分のマシンに戻ります。

```
venus$ rlogin starbug -l user2

Password:

Last login: Thu Nov  2 15:13:10 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

starbug$ ~.  (You may not see the ~ on the
 screen.)

Closed connection.

venus$
```

1 つ前の中間セッションに戻りたい場合は、2 つのチルド文字のあとにピリオドを付けたコマンド (~~.) を使います。

```
venus$ rlogin starbug -l user2

Password:

Last login: Thu Nov  2 15:14:58 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

starbug$ rlogin earth -l user2

Password:

Last login: Thu Nov  2 15:24:23 from starbug

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

earth$ ~~.  (You may not see the ~~ on the screen.)

Closed connection.

starbug$
```

rlogin セッションの中断

rlogin セッションを一時的に中断してあとでまた戻りたい場合は、チルド文字 (~) のあとに Ctrl-Z を入力します。rlogin セッションは中断されたプロセスになり、ユーザはログイン元のマシンに戻ります。

rlogin セッションを再開するには、fg と入力します。また、パーセント記号 (%) と、中断されたプロセスのプロセス番号を入力して再開することも可能です。プロセス番号を指定しないと、最後に中断されたプロセスが % によってアクティブ化されます。

```
venus$ rlogin goddess -l user2

Password:

Last login: Thu Aug 31 14:31:42 from venus

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002

goddess$ pwd

/home/user2

goddess$~^Z


Stopped (user)

venus$ pwd
```

```
/home/user2/veggies

venus$fg

rlogin goddess

goddess$ logout

venus$
```

また、チルド文字 2 つを入力して Ctrl-Z を押すことによっても、現在の `rlogin` セッションが中断され、中間の `rlogin` セッションに戻ります。

`rlogin(1)` コマンドについての詳細は、『*man pages section 1: User Commands*』を参照してください。

ログイン状態の確認 (who am i)

さまざまなリモートマシンにログインしたあとは、現在の場所を確認しなければならない場合があります。その場合、`who am i` と入力することによって、現在ログインしているマシン名と現在のユーザ名を表示できます。

ファイルのリモートコピー (rcp)

`rcp` コマンドを使って、異なるマシン間でファイルをコピーできます。`rcp` は、リモートマシン上の `/etc/hosts.equiv` ファイルと `/etc/passwd` ファイルを参照して、ユーザがアクセス権を持っているかどうか確認します。`rcp` の構文は、`cp` のコマンド構文に似ています。

注 – 異なるマシン間でサブディレクトリとその内容をコピーするには、 `rcp -r` を使います。

リモートマシンからのファイルのコピー

リモートマシンから自分のマシンにファイルをコピーするには、次の構文で `rcp` コマンドを実行します。

```
$ rcp machinename:source destination
```

machinename はリモートマシン名、*source* はコピーするファイル名、*destination* はコピーしたファイルが格納されるローカルマシン上のパス名です。

次の例は、リモートマシン starbug のファイル /etc/hosts を、ローカルマシン venus の /tmp ディレクトリにコピーする方法を示しています。

```
venus$ rcp starbug:/etc/hosts /tmp
```

rcp では、さまざまな省略名と構文を組み合わせて使うことができます。たとえば、リモートマシン fretful 上のユーザ hank のホームディレクトリにある .doc で終わる全ファイルを、ローカルマシン venus の現在のディレクトリにコピーするには、次のコマンドを入力します。

```
venus$ rcp fretful:~hank/*.doc .

venus$
```

ローカルマシンからリモートマシンへのファイルコピー

ローカルマシンからリモートマシンにファイルをコピーするには、次のコマンド構文を使用します。

```
$ rcp source machinename:destination
```

source はコピーするファイル名、*machinename* はリモートマシン名、*destination* はコピーしたファイルが格納されるリモートマシン上のパス名です。

次の例は、/tmp ディレクトリのファイル newhosts をリモートマシン starbug の /tmp ディレクトリにコピーする方法を示しています。

```
venus$ cd /tmp

venus$ rcp newhosts starbug:/tmp

venus$
```

rcp(1) コマンドとそのオプションについての詳細は、『*man pages section 1: User Commands*』を参照してください。

コマンドのリモート実行 (rsh)

rsh コマンド (リモートシェル 用) を使うと、リモートマシンにログインしなくてもそのマシンに対して特定のコマンドを実行できます。リモートマシンに対して 1 つのコマンドを実行するだけでよい場合は、rsh によって時間を節約できます。

リモートマシンに対して 1 つのコマンドを実行するには、次のコマンド構文を使用します。

rsh *machinename command*

次の例は、リモートマシン starbug 上のディレクトリ /etc/skel の内容を表示する方法を示しています。

```
venus$ rsh starbug ls /etc/skel*

local.cshrc

local.login

local.profile

venus$
```

rlogin や rcp と同様に、rsh は、リモートマシンの /etc/hosts.equiv ファイルと /etc/passwd ファイルを参照して、ユーザにリモートマシンに対するアクセス権があるかどうかを確認します。

rsh(1) コマンドとそのオプションについての詳細は、『*man pages section 1: User Commands*』を参照してください。

ユーザ情報の参照 (rusers)

rusers コマンド (リモートユーザ) を使って、ネットワーク上のほかのマシンにログインしているユーザの情報を表示できます。rusers コマンドを引数なしで入力すると、ネットワーク上の各マシン名とそのマシンにログインしているユーザの情報が表示されます。

```
$ rusers

starbug          user2 user1 root

earth            user1

venus          user3
```

マシン starbug には現在 3 人のユーザがログインしています。

特定のリモートマシンに関する情報を表示するには、次のように rusers コマンドのあとにマシン名を指定します。

```
$ rusers starbug

starbug          user2 user1 root

$
```

rusers で -l オプションを指定すると、次のような情報が表示されます。

- ユーザ名
- マシン名と端末名
- 各ユーザがログインした時点
- ユーザがアイドル状態にある期間 (1 分を超している場合)
- 各ユーザがログインに使用したマシン名

```
$ rusers -l starbug

root       starbug:console       Oct 31 11:19          (:0)

user2      starbug:pts/7         Oct 31 11:20   40:05 (starbug)

user1      starbug:pts/13        Nov  1 14:42   17:18 (starbug)

$
```

`-l` オプションは、マシン名を指定しなくても使えます。

`rusers(1)` コマンドとそのオプションについての詳細は、『*man pages section 1: User Commands*』を参照してください。

ネットワークアプリケーションの実行

通常、システム上のすべてのアプリケーションは、ローカルマシンで実行されるプログラムです。ただし、使用しているワークステーションがネットワークの一部であれば、リモートマシン上でアプリケーションを実行してその結果をローカルマシンの画面に表示できます。ほかのマシンでアプリケーションを実行すれば、ローカルマシン上の資源を節約できる上、ネットワーク上のすべてのアプリケーションにアクセスできます。

この節では、リモートマシン上でアプリケーションを実行してその結果をローカルマシンの画面に表示する方法を紹介します。ネットワークアプリケーションの実行に関連した補足情報については、129 ページの「セキュリティに関する詳細」を参照してください。

注 – ネットワークアプリケーションを実行するための処理は、コンピュータ環境によって異なる可能性があります。

`rlogin` を使用してネットワークアプリケーションを実行する

以下の手順に従ってリモートマシン上のアプリケーションを実行するには、次の要件を満たしている必要があります。

- リモートマシンに対するアクセス権限を持っている
- NFS を使用してホームディレクトリをリモートマシン上にマウントできる
- アプリケーションと適切なライブラリがリモートマシン上にインストールされている

これらの要件を満たしているかどうか不明な場合は、システム管理者に問い合わせてください。

リモートマシンに存在するネットワークアプリケーションを実行するには、次のように環境変数を設定する必要があります。

- リモートマシン上のシェルの DISPLAY 環境変数を、ローカルマシンの画面に設定します。
- ローカルマシンでアプリケーションプログラムが表示されるようにリモートマシンを設定します。このためには、次のように xhost コマンドを指定します。

```
$ xhost +remote_machine_name
```

次に、rlogin コマンドを使用してリモートマシン上でアプリケーションを実行するための手順を示します。

▼ リモートマシンでアプリケーションを実行する方法

1. リモートマシンの表示ができるように、ローカルマシン上で **xhost** コマンドを実行します。

```
starbug$ xhost + venus
```

2. リモートマシンにログインします。

```
starbug$ rlogin venus -l user2

Password:

Last login: Wed Nov  1 16:06:21 from starbug

Sun Microsystems Inc.   SunOS 5.9       Generic February 2002
```

3. **DISPLAY** 変数をローカルマシンに設定します。

```
venus$ DISPLAY=starbug:0.0
```

4. **DISPLAY** 変数をローカルマシンにエクスポートします。

```
venus$ export DISPLAY
```

5. アプリケーションを実行します。

```
venus$ bigprogram &
```

このアプリケーションとの対話は画面上のほかのアプリケーションとまったく同様に行われますが、このアプリケーションの実行はリモートマシンで行われます。

このような方法でアプリケーションを実行すると、ローカルマシンにインストールされているアプリケーションよりもコンピュータ資源の消費を小さく抑えられるという利点があります。アクセスが認められているリモートアプリケーションを実行する場合はこの方法を利用できます。

セキュリティに関する詳細

この節では、ネットワーク上でアプリケーションを実行するときに必要と思われるネットワークセキュリティの基本事項について説明します。具体的には次のような事項です。

- ユーザベースおよびホストベースのアクセス制御機構
- 認証プロトコルの `MIT-MAGIC-COOKIE-1` と `SUN-DES-1`
- サーバに対するアクセス制御を変更する方法とその時期
- アプリケーションをリモートで実行 (別のユーザとしてローカルで実行) する方法

対象読者

リモートサーバでアプリケーションを実行する場合以外は、少なくとも Solaris オペレーティング環境のデフォルトのセキュリティ構成を変更する必要はありません。

アクセス制御機構

アクセス制御機構は、どのクライアントまたはアプリケーションが X11 サーバにアクセスできるかを制御します。アクセスが許可されたクライアントだけがサーバに接続できます。アクセスが許可されていないその他のクライアントはすべて次のエラーメッセージを表示して終了します。

```
Xlib: connection to hostname refused by server

Xlib: Client is not authorized to connect to server
```

接続の結果は、次のようにサーバコンソールに表示されます。

```
AUDIT: <Date Time Year>: X: client 6 rejected from IP 129.144.152.193

   port 3485     Auth name: MIT-MAGIC-COOKIE-1
```

アクセス制御機構には、 ユーザベースとホストベースの 2 種類があります。ユーザベースの機構は特定のユーザのアカウントに対するアクセスを許可し、ホストベースの機構は特定のホスト (マシン) に対するアクセスを許可します。`Xsun` に `-noauth` オプションを指定しないと、これらのアクセス制御機構はどちらもアクティブ状態になります。詳細は、132 ページの「サーバへのアクセスを制御する」を参照してください。

ユーザベース

ユーザベースすなわち認証ベースのアクセス制御機構では、あるユーザに対し、任意のホストマシン上の特定のユーザに対するアクセスが明示的に許可されます。ユーザのクライアントは、サーバに認証データを渡します。そのデータがサーバの認証データと一致すれば、そのホストのすべてのユーザはサーバのアクセスを許可されます。

ホストベース

ホストベースのアクセス制御機構には、汎用性があります。この機構では、特定のホストへのアクセスを許可し、そのホスト上のすべてのユーザがサーバに接続できるようにします。これは、ゆるやかなアクセス制御方式です。ホストにサーバに対するアクセス権が与えられていれば、そのホスト上のすべてのユーザがサーバに接続できます。

Solaris 環境では、ホストベースの制御機構は、下位互換性を目的として使用されます。

注 – `Xlib` または `libcps` の旧バージョンを使ってリンクされたクライアントはリンクし直してください。これにより、新しいユーザベースのアクセス制御機構の下でサーバに接続できるようになります。

認証プロトコル

Solaris オペレーティング環境は、以下の2 種類の認証プロトコルをサポートしています。 `MIT-MAGIC-COOKIE-1` と `SUN-DES-1` です。これらの 2 種類のプロトコルでは、使用される認証デー タの形式は異なりますが、アクセス制御機構は似ています。Solaris オペレーティング環境のデフォルトは、ユーザベース機構を使用した `MIT-MAGIC-COOKIE-1` プロトコルです。

MIT-MAGIC-COOKIE-1

`MIT-MAGIC-COOKIE-1` 認証プロトコルは、マサチューセッツ工科大学で開発されました。サーバの起動時に、サーバとシステムを起動したユーザについてマジッククッキー (*magic cookie*) が作成されます。接続を要求するたびに、ユーザのクライアントは、接続パケットの一部としてマジッククッキーをサーバに送信します。このマジッククッキーは、サーバのマジッククッキーと比較されます。マジッククッキーが一致すれば接続は許可され、一致しなければ拒否されます。

SUN-DES-1

米国 Sun Microsystems, Inc. が開発した SUN-DES-1 認証プロトコルは、機密保護 RPC (遠隔手続き呼び出し) にもとづいており、DES (データ暗号化規格) を必要とします。認証情報としては、マシンに依存しないユーザの netname (ネットワーク名) が使われます。この認証情報は暗号化され、接続パケットの一部としてサーバに送信されます。サーバは暗号化された認証情報を復号化して、netname が登録されているものであれば接続を許可します。

SUN-DES-1 プロトコルは、MIT-MAGIC-COOKIE-1 よりも高水準のセキュリティを提供します。あるユーザがほかのユーザの netname を使ってそのユーザのサーバにアクセスする方法はありませんが、マジッククッキーを使うと可能な場合もあります。

このプロトコルは、Solaris 1.1 環境および Solaris 1.1 互換環境に入っているライブラリでしか使用できません。静的ライブラリで構築されたアプリケーションは、この認証プロトコルを使用できません。

134 ページの「SUN-DES-1 を使用する場合のアクセス許可」では、ほかのユーザの netname を自分のサーバのアクセスリストに追加することによって、自分のサーバに対するそのユーザのアクセスを許可する方法について説明しています。

デフォルトの認証プロトコルの変更

デフォルトの認証プロトコル MIT-MAGIC-COOKIE-1 は、システムでサポートされるもう 1 つの認証プロトコル SUN-DES-1 に変更できます。また、ユーザベースのアクセス制御機構をまったく使わないようにすることもできます。デフォルト認証プロトコルを変更するには、 /usr/dt/config/Xservers ファイルの Xsun という行を編集します。たとえば、デフォルトを MIT-MAGIC-COOKIE-1 から SUN-DES-1 に変更するには、/usr/dt/config/Xservers ファイルの次の行を編集して Xsun コマンドに -auth sun-des オプションを加えます。

```
:0  Local local_uid@console root /usr/openwin/bin/Xsun :0 -nobanner -auth sun-des
```

ユーザベースのアクセス制御機構を使わないで Solaris オペレーティング環境を実行する必要がある場合は、/usr/dt/config/Xservers ファイル内の次の行を編集して Xsun コマンドに -noauth オプションを加えます。

```
:0  Local local_uid@console root /usr/openwin/bin/Xsun :0 -nobanner -noauth
```

注意 – -noauth オプションを指定した場合、セキュリティの水準は低下します。この設定は、OpenWindows をホストベースのアクセス制御機構だけで実行した場合と同じです。サーバはユーザベースのアクセス制御機構を無効にします。あるマシンでアプリケーションを実行できるユーザであれば、そのマシンのサーバに対するアクセスを許可されることになります。

サーバへのアクセスを制御する

Xsun を -noauth オプションとともに使用していない場合は (131 ページの「デフォルトの認証プロトコルの変更」を参照してください) 、ユーザベースとホストベースのアクセス制御機構が両方ともアクティブ状態になります。サーバは最初にユーザベースの制御機構をチェックした後、ホストベースの制御機構をチェックします。デフォルトのセキュリティ構成では、MIT-MAGIC-COOKIE-1 がユーザベースの制御機構として使われ、空のアクセスリストがホストベースの制御機構として使われます。ホストベースのアクセスリストは空であるため、実質的にはユーザベースの制御機構だけが有効になります。-noauth オプションを指定すると、サーバはユーザベースのアクセス制御機構を無効にし、ホストベースのアクセスリストにローカルホスト名を追加してリストを初期化します。

以下の 2 つのプログラムのどちらかを使って、サーバのアクセス制御機構を変更できます。 xhost または xauthです。 詳細は、マニュアルページを参照してください。これらのプログラムは、認証プロトコルによって作成される 2 つのバイナリファイルにアクセスします。これらのファイルには、各セッション固有の認証データが入っています。一方のファイルは、サーバから内部的に使用するためのもので、もう一方のファイルは下記のファイル名でユーザの $HOME ディレクトリにあります。

- .Xauthority
- （クライアント認証ファイル）

サーバ上にあるホストベースのアクセスリストを変更するには、xhost プログラムを使用します。これにより、アクセスリストに対するホストの追加と削除ができます。デフォルトの構成 (空のホストベースのアクセスリスト) で OpenWindows を起動した場合は、xhost を使ってアクセスリストにマシン名を追加すると、セキュリテレベルを低下させます。つまりサーバは、デフォルトの認証プロトコルで指定されるユーザのほかに、リストに追加されたホストに対してもアクセスを許可します。ホストベースのアクセス制御機構のセキュリテレベルが低いと見なされる理由については、130 ページの「ホストベース」を参照してください。

xauth プログラムは、.Xauthority クライアントのファイル内の認証プロトコルデータにアクセスします。アクセスを操作する側のユーザは自分の .Xauthority ファイルからデータを抽出して、ほかのユーザがそのデータを自分の .Xauthority ファイルにマージできるようにします。これにより、操作側ユーザのサーバまたはそのユーザが接続しているサーバに対してほかのユーザがアクセスできるようになります。

xhost と xauth の使用例は、133 ページの「MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可」を参照してください。

クライアント認証ファイル

クライアント用の許可ファイル .Xauthority には、次の形式のエントリがあります。

connection-protocol *auth-protocol* *auth-data*

デフォルト設定では、.Xauthority ファイルに *auth-protocol* として MIT-MAGIC-COOKIE-1 が格納され、ローカル表示に関するエントリが *connection-protocol* および *auth-data* としてのみ格納されます。たとえば、ホスト *anyhost* 上の .Xauthority ファイルに次のエントリが格納されている場合が考えられます。

anyhost:0 MIT-MAGIC-COOKIE-1 *82744f2c4850b03fce7ae47176e75*

localhost:0 MIT-MAGIC-COOKIE-1 *82744f2c4850b03fce7ae47176e75*

anyhost/unix:0 MIT-MAGIC-COOKIE-1 *82744f2c4850b03fce7ae47176e75*

クライアントの起動時に、*connection-protocol* に対応するエントリが .Xauthority から読み取られ、*auth-protocol* および *auth-data* が接続パケットの一部としてサーバに送られます。デフォルトの構成では、xhost によって、空のホストベースのアクセスリストが戻され、認証が実行可能な状態であることを示します。

認証プロトコルをデフォルトの MIT-MAGIC-COOKIE-1 から SUN-DES-1 に変更した場合、.Xauthority ファイルのエントリには、*auth-protocol* として SUN-DES-1 が格納され、*auth-data* としてユーザのネットワーク名 (netname) が格納されます。netname の形式は次のとおりです。

unix.*userid@NISdomainname*

たとえば、ホスト *anyhost* 上で .Xauthority ファイルに次のエントリが格納されている場合が考えられます。この unix.*15339@EBB.Eng.Sun.COM* は、マシンに依存しないユーザの netname です。

anyhost:0 SUN-DES-1 "unix.15339@EBB.Sun.COM"

localhost:0 SUN-DES-1 "unix.15339@EBB.Sun.COM"

anyhost/unix:0 SUN-DES-1 "unix.15339@EBB.Sun.COM"

注 – 自分のネットワーク名、またはマシンに依存しないネットワーク名が不明な場合は、システム管理者に問い合わせてください。

MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可

MIT-MAGIC-COOKIE-1 認証プロトコルを使用している場合は、次の手順により自分のサーバにほかのユーザがアクセスできるようにすることができます。

1. サーバを実行しているマシン上で xauth を実行し、*hostname:0* に対応するエントリを抽出して、ファイルに入れます。

   例として、*hostname* が *anyhost*、ファイルが *xauth.info* の場合を示します。

   ```
   myhost% /usr/openwin/bin/xauth nextract - anyhost:0> $HOME/xauth.info
   ```

2. アクセスを要求しているユーザに、手順 1 で作成したエントリが入っているファイルを送ります。

この送信には、電子メールツールや rcp コマンドなどのファイル転送プログラムを使用します。

注 – 認証情報が入っているファイルの転送には、rcp よりもメールを使用するほうがより安全です。rcp を使用する場合は、ほかのユーザが簡単にアクセスできるディレクトリにそのファイルを入れないようにしてください。

---

3. ほかのユーザは、エントリを自分の .Xautority ファイルにマージしなければなりません。

   例として、*userhost* が *xauth.info* を自分の .Xauthority ファイルにマージする場合を示します。

   ```
   userhost% /usr/openwin/bin/xauth nmerge - < xauth.info
   ```

注 – *auth-data* は特定のセクションに対応します。したがって、サーバが再起動されるまでの間有効です。

SUN-DES-1 を使用する場合のアクセス許可

SUN-DES-1 認証プロトコルを使用している場合は、次の手順により自分のサーバにほかのユーザがアクセスできるようにすることができます。

1. サーバを実行しているマシン上で xhost を実行しサーバに新規ユーザを通知します。

   例として、新規ユーザ *somebody* に *myhost* 上での実行を許可する場合を示します。

   ```
   myhost% xhost + somebody@
   ```

2. 新規ユーザは、xauth コマンドを使用して自分の .Xauthority ファイルにエントリを追加します。

   例として、新規ユーザの、マシンに依存しないネット名が unix.15339@EBB.Sun.COM の場合を示します。下記のコマンドは、途中に Return キーなどを入れずに 1 行に入力しなければならないことに注意してください。パイプ記号のあとに、空白と続いてコマンドの残りの部分を入力します。

   ```
   userhost% echo 'add myhost:0 SUN-DES-1 "unix.15339@EBB.Sun.COM"' |

   $OPENWINHOME/bin/xauth
   ```

クライアントをリモートで実行する場合とローカルで実行する場合

X クライアントは、環境変数 DISPLAY の値を使用して接続するサーバ名を取り出します。

クライアントをリモートで実行するか、または他のユーザとしてローカルで実行するには、次の手順を実行します。

1. サーバを実行しているマシン上で、ほかのユーザのアクセスを許可します。

   使用する認証プロトコルに応じて、133 ページの「MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可」または 134 ページの「SUN-DES-1 を使用する場合のアクセス許可」で説明した手順を実行します。

2. 環境変数 DISPLAY の値として、サーバを実行しているホスト名に設定します。

   例として、ホスト名が *remotehost* の場合を示します。

   ```
   myhost% setenv DISPLAY remotehost:0
   ```

3. クライアントプログラムを実行します。

   ```
   myhost% client_program&
   ```

   クライアントは、リモートマシン *remotehost* 上に表示されます。

第 10 章

動作環境のカスタマイズ

システムの初期設定ファイル内の環境変数を変更することによって、動作環境におけるさまざまな状態を制御や調整できます。ユーザがログインすると、初期設定ファイルが読み取られ、その中の環境変数を使ってシステムが構成されます。環境変数を適切に設定することによって、独自の作業をより効率的で簡単に実行できるようにシステムをカスタマイズできます。

この章では、次の作業について説明します。

初期化ファイルの変更

システムの構成に使われる初期設定ファイルは、システムのインストール時にシステム管理者がデフォルトシェルとして指定したシェルの種類によって異なります。Solaris オペレーティング環境のデフォルトシェルは Bourne シェルですが、C シェルや Korn シェルも利用できます。これらのシェルには、独自の初期設定ファイル (複数の場合もある) があります。

デフォルトシェル (ログインシェル) を確認するには、次のように行います。

1. **echo $SHELL** と入力します。

```
$ echo $SHELL

/bin/sh
```

2. このコマンドの結果から、次のようにデフォルトシェルを判断します。

- /bin/sh – Bourne シェル
- /bin/bash – Bourne Again シェル
- /bin/csh – C シェル
- /bin/ksh – Korn シェル
- /bin/tcsh – TC シェル
- /bin/zsh – Z シェル

最初にシステムにログインすると、使っているシェルの種類にかかわらず、通常はシステムプロファイルの /etc/profile ファイルが実行されます。このファイルの所有者は通常はシステム管理者であり、その他のユーザは読み取りだけ許可されています (書き込みはできません)。

システムプロファイルの実行が完了すると、次にユーザプロファイルが実行されます。ユーザプロファイルは、各ユーザ独自の動作環境を定義する 1 つ以上の初期設定ファイルです。たとえば、CDE 環境では、新しい端末またはウィンドウを開始するたびに、これらのファイルのチェックまたは設定が行われます。

デフォルトシェルとして設定されているシェルの種類によって、ユーザプロファイルは次のどれかになります。

- .profile (Bourne シェルまたは Korn シェルの場合)
- .bash_profile (Bourne Again シェルの場合)
- .login と .cshrc (C シェルの場合)
- .tcshrc と .cshrc (TC シェルの場合)
- .zlogin と .zshrc (Z シェルの場合)

ユーザプロファイルは各ユーザのホームディレクトリ内にあり、独自の動作環境を構成するために使います。

環境変数の設定

システム環境は、初期設定ファイルで定義された複数の環境変数を使って構成されます。現在の作業環境を一時的に変更する場合は、コマンドプロンプトから直接コマンドを入力します。ただし、動作環境の変更を一時的にではなく、常時有効にしたい場合は、適切なユーザプロファイルファイル内に永続的な環境変数を設定できます。

システムに現在設定されている環境変数を表示するには、env コマンドを使用します。

- **env** コマンドを入力して **Return** キーを押します。

```
$ env

HOME=/home/user2

PATH=/usr/bin:
```

```
LOGNAME=user2

HZ=100

TERM=dtterm

TZ=US/Mountain

SHELL=/bin/csh

MAIL=/var/mail/user2

PWD=/home/user2

USER=user2

$
```

注 – env コマンドを使って、ログインシェルの確認も行えます。ログインシェルは、SHELL 環境変数で指定されます。上記の例では、シェルは /bin/csh (C シェル) に設定されています。

ユーザプロファイル

この節では、一般的によく使われる環境変数について説明します。これらの環境変数の大部分は、すでにユーザプロファイルに入っています。すでに述べたように、ユーザプロファイルのファイルは、各ユーザのホームディレクトリにあります。

注 – 隠しファイル (ドットファイル) は、ls コマンドに -la オプションを指定することで表示できます。

次のリストは、ユーザプロファイルで使える環境変数の一部です。環境変数を定義する構文は、現在のシェルによって異なります。

- CDPATH – 絶対パス名を指定しないで一意のディレクトリ名を入力したときに検索されるディレクトリを指定します。
- HISTORY – history コマンドで利用できるコマンドの数を指定します。
- HOME – 各ユーザのホームディレクトリの絶対パス名を指定します。cd コマンドを引数なしで入力したときに移動先となるディレクトリは、この情報にもとづいて決められます。
- LANG – ロケールの言語を指定します。有効な値は、日本語、ドイツ語、フランス語、スウェーデン語、イタリア語などです。

- LOGNAME – ユーザのログイン名を指定します。LOGNAME 変数のデフォルト値は、passwd データベースで指定されたログイン名となるようログイン処理の過程で自動的に設定されます。passwd データベースの詳細は、『*Solaris* のシステム管理 *(*基本編*)*』を参照してください。
- LPDEST – デフォルトプリンタを指定します。
- MAIL – メールボックスのパス名を指定します。通常、メールボックスは /var/mail/*username* ディレクトリにあります (*username* は各ユーザのログイン名)。メールボックスについての詳細は、第 7 章を参照してください。
- MANSECTS – オンラインのマニュアルページで利用できるセクションを設定します。
- PATH – コマンドを入力したときに、実行可能プログラムが検索されるディレクトリを順番に指定します。適切なディレクトリが検索パスにない場合は、それを PATH 変数に追加するか、コマンドを入力するときに絶対パス名を指定する必要があります。

  この変数のデフォルト値は、ユーザプロファイルファイルで指定されたとおりになるようにログイン処理の一環として自動的に設定されます。
- PS1 – コマンドプロンプトを指定します。Bourne シェル、Bourne Again シェル、Korn シェルのデフォルトプロンプトは、ドル記号 ($) です。C シェル、TC シェル、Z シェルのデフォルトプロンプトは、パーセント記号 (%) です。root のデフォルトプロンプトは、どのシェルでもポンド記号 (#) です。
- SHELL – vi やほかのツールで使われるシェルを指定します。
- TERMINFO – terminfo データベースに追加された、デフォルトでない端末のパス名を指定します。terminfo データベース内のデフォルト端末については、この変数を設定する必要はありません。terminfo データベースの詳細は、『*Solaris* のシステム管理 *(*上級編*)*』を参照してください。
- TERM – 現在使っている端末を指定します。エディタを実行するときは、TERM 変数で定義された名前と同じ名前のファイルが検索されます。その場合、まず最初に TERMINFO 変数で指定されるパスが検索され (TERMINFO が定義されている場合)、次にデフォルトディレクトリの /usr/share/lib/terminfo が検索されて、端末の特性が決定されます。TERM 変数で定義されたファイルが見つからない場合は、その端末はダム端末と認識されます。
- TZ – システムクロックのタイムゾーンを指定します。

PATH 環境変数の設定

PATH 環境変数は、SunOS ディレクトリ階層内でコマンドを検索するために使われます。PATH 環境変数を設定すると、コマンド名を入力したときに特定のディレクトリが常に検索されるようになります。

たとえば、PATH 変数を設定していない場合にファイルをコピーする場合、/usr/bin/cp のように、コマンドの絶対パス名を入力しなければなりません。これに対して、/usr/bin ディレクトリが PATH 変数に含まれるよう設定しておけば、cp

と入力するだけでコマンドを実行できます。これは、PATH 変数で指定された各ディレクトリ内で cp コマンドが検索され、見つかった時点で実行されるためです。頻繁に使われる SunOS コマンドのディレクトリを PATH 変数に設定しておけば、作業効率を大幅に向上させることができます。

Bourne シェル、Bourne Again シェル、および Korn シェルについては、次の構文を使ってホームディレクトリのユーザプロファイルファイル内で PATH 変数を設定できます。

```
PATH=.:/usr/bin:/home/bin
```

home は、ホームディレクトリのパス名です。

C シェル、TC シェル、および Z シェルについては、次の構文を使ってホームディレクトリのユーザプロファイルファイル内で PATH 変数を設定できます。

```
set path=(/usr/bin home/bin .)
```

home はホームディレクトリのパス名です。

注 – C シェル、Korn シェル、TC シェル、Bourne Again シェル、および Z シェルでは、短縮名の ~ を使ってホームディレクトリのパス名を表すことができます。

C シェル、TC シェル、または Z シェルの環境で PATH 変数を変更した場合は、ログアウトしなくても source コマンドを実行すれば、現在のウィンドウ内で PATH の設定を有効にできます。

```
example% source user-profile-file
```

Bourne シェル、Bourne Again シェル、または Korn シェルを使っている場合は、ログアウトしなくても次のコマンドを実行すれば、現在のウィンドウ内で PATH の設定を有効にできます。

```
$ . user-profile-file
```

コマンドの別名

コマンドの別名は、頻繁に入力するコマンドに利用できる便利な短縮名です。たとえば、削除コマンド (rm) のデフォルト設定では、ファイルを削除する前に確認を求められませんが、入力を誤って必要なファイルを削除してしまう危険性があるため、この設定では不都合な場合があります。alias 変数を使うと、ユーザプロファイルファイルを編集してこの設定を変更できます。

C シェルと TC シェルでは、ユーザプロファイルファイルに次の行を追加します。

```
alias rm  'rm -i'
```

Bourne Again シェル、Korn シェル、および Z シェルでは、ユーザプロファイル ファイルに次の行を追加します。

```
alias rm='rm -i'
```

ユーザプロファイルファイルにこの行があれば、rm と入力するだけで rm -i (対話形式の rm コマンド) と入力したのと同じことになります。したがって、ファイルが削除される前に常に確認を求められるようになります。上記の例で、rm -i の両側の引用符は、rm と -i の間に空白を挿入するために必要です。 この引用符がないと、シェルは空白のあとのテキストを正しく解釈できません。

ユーザプロファイルファイルに対して行なった変更を、現在のウィンドウ内でただちに有効にするには、別のコマンドを入力する必要があります。C シェルと TC シェルでは、次のコマンドを使用します。

```
example% source user-profile-file
```

source コマンドを実行すると、現在のユーザプロファイルファイルが読み取られてこのファイル内のコマンドが実行されます。

Bourne Again シェル、Korn シェル、および Z シェルでは、次のコマンドを入力すると別名がただちに有効になります。

```
$ . user-profile-file
```

Bourne Again シェル、Korn シェル、および Z シェルでは、C シェルおよび TC シェルにおける source コマンドと同じ役割を果たす (.) コマンドを使用します。

注 – alias コマンドを使用して作成されるコマンドエイリアスは、現在のセッションにしか適用されません。

コマンドプロンプトの変更

コマンドプロンプトの変更に使う構文は、使用しているシェルによって異なります。

Bourne、Bourne Again、Korn、Z シェルの場合

Bourne シェル、Bourne Again シェル、Korn シェル、および Z シェルでは、PS1 コマンドを使ってコマンドプロンプトを定義し直すことができます。次の 3 つの例を参照してください。

- コロンのあとに空白が挿入されたプロンプトを設定するには、次のコマンドを入力します。

  ```
  PS1=": "
  ```

- マシン名の後にコロンとスペースが挿入されたプロンプトを設定するには、次のコマンドを入力します。

```
PS1="`hostname`: "
```

- マシン名のあとに中括弧 { } で囲んだログイン名、コロンと空白が挿入されたプロンプトを設定するには、次のコマンドを入力します。

```
PS1="`hostname`{`logname`}: "
```

上記の例のいずれかを入力して現在のコマンドプロンプトを変更してみてください。この変更は、コマンドプロンプトを再度変更するかログアウトするまで継続します。

変更を持続させるには、上記の例の 1 つ (または自分で作成したプロンプト) を自分のユーザプロファイルファイルに追加します。指定するプロンプトが、ログインしたり、新しいシェルを起動するときに表示されます。

C シェルと TC シェル

C シェルと TC シェルでは、`set prompt` コマンドを使ってコマンドプロンプトを変更できます。次の 3 つの例を参照してください。

- パーセント記号のあとに空白が挿入されたプロンプトを設定するには、次のコマンドを入力します。

```
example% set prompt="% "
```

- マシン名のあとにコマンドの履歴番号 (*hostname*1、*hostname*2、*hostname*3 など)、コロンが挿入されたプロンプトを作成するには、次のコマンドを入力します。

```
example% set prompt="`hostname`\!: "
```

- マシン名のあとに中括弧で囲んだログイン名、コロン と空白が挿入されたプロンプトを作成するには、次のコマンドを入力します。

```
example% set prompt="`hostname`{`logname`}: "
```

上記の例のいずれかを入力して現在のコマンドプロンプトを変更してみてください。この変更は、コマンドプロンプトを再度変更するかログアウトするまで継続します。

変更を持続させるには、上記の例の 1 つ (または自分で作成したプロンプト) を自分のユーザプロファイルファイルに追加します。指定するプロンプトが、ログインしたり、新しいシェルを起動するときに表示されます。

その他の便利な変数

ユーザプロファイルファイルには、上記以外にも多数の変数を設定できます。変数についての詳細は、『*manpages section 1 : User Commands*』を参照してください。次節以降では、比較的よく使われるオプションをいくつか説明します。

noclobber 変数

cp コマンドを使ってファイルをコピーするときに、誤ってファイルを上書きするのを防ぐには、set noclobber を使います。この変数は、Bourne Again シェル、C シェル、Korn シェル、および TC シェルに適用できます。ユーザプロファイルファイル内に次の行を入力します。

```
set noclobber
```

history 変数

history 変数を使用すると、履歴リストに保存するコマンドの数を設定できます。history コマンドは、以前に入力したコマンドを参照する場合に便利です。履歴リストを使って、以前のコマンドを繰り返すこともできます。次の行を .cshrc ファイルか .tcshrc ファイルに入力します。

```
set history=100
```

ユーザプロファイルファイルに次の行を入力すれば、Bourne シェル、Bourne Again シェル、Korn シェル、Z シェルについても同様の設定ができます。

```
HISTORY=100
```

デフォルトのファイルアクセス権の設定方法

umask コマンドを使って、新規に作成される全ファイルと全ディレクトリに対してデフォルトのファイルアクセス権を設定できます。たとえば、セキュリティを重視するため、グループのメンバとその他の全ユーザに対しては、自分の所有するディレクトリに関する読み取り権や実行権 (-rwxr-xr-x) 、ファイルの場合は (-rw-r--r--) だけを付与したい場合があります。そのような場合は、新規に作成されるファイルやディレクトリがこのアクセス権で保護されるようユーザプロファイルファイル内で umask を使えます。

chmod コマンドと同様に、umask では数値コードを使ってファイルの絶対アクセス権を表します。ただし、umask で使うコードは、chmod のコードとは異なる方法で計算します。

たとえば、umask が 000 に設定されている場合は、新規に作成される全ファイルのアクセス権は次のとおりです (読み取り権と書き込み権があり、実行権なし)。

rw-rw-rw- (mode 666)

新規に作成される全ディレクトリのアクセス権は次のとおりです (読み取り権、書き込み権、実行権の全アクセス権がある) 。

rwxrwxrwx (モード 777)

umask で使う値を決定するには、ファイルに割り当てられている現在のデフォルトアクセス権の値から、設定したいアクセス権の値 (chmod コマンドで指定する値と同じ) を差し引きます。この差し引き後の値を umask コマンドで使います。

たとえば、ファイルのデフォルトモードを 666 (rw-rw-rw-)から 644 (rw-r--r--) に変更する場合は、666 から 644 を差し引いた値の 022 をumask コマンドで指定します。

```
$ umask 022
```

chmod コマンドの数値コードと同様に、umask では次の 3 桁の数字を使います。

- 1 桁目は所有者のアクセス権を制御する。
- 2 桁目の数字はグループのアクセス権を制御する。
- 3 桁目の数字はその他の全ユーザのアクセス権を制御する。

umask コマンドの数値コードの各桁で設定されるファイルアクセス権を表 10-1 に示します。

表 **10–1** umask のアクセス権

umask コード	設定するアクセス権
0	rwx
1	rw-
2	r-x
3	r--
4	-wx
5	-w-
6	--x
7	---(アクセス権なし)

umask コマンドについての詳細は、『*manpages section 1 : User Commands* を参照してください。

付録 A

キーボードの設定変更

この付録では、キーボードの特殊キーのオプションをマッピングし直す方法や、Compose キーを有効または無効にする方法などについて説明します。

マウスボタンの再マッピングの詳細 (左ききの人がマウスを操作しやすいようにする方法など) は、使用しているデスクトップ環境のユーザズガイドを参照してください。

Compose キーの設定解除と設定

x86 のみ – Compose キーは、x86 システムでは Ctrl-Shift-F1 キーに割り当てられています 。

Compose キーを使用しない場合は、そのキー操作を無効にすることができます。まず Multi_key のキーコードを探します。

```
$ xmodmap -pk | grep Multi_key
```

次のような行が表示されます。

nn 0xff20 (Multi_key)

行の先頭にある 2 桁のキーコード番号 (*nn* で示される) を使用して、 .xinitrc ファイル内に次の行を作成します。

```
xmodmap -e 'keycode nn = NoSymbol'
```

Compose キーを再設定するには、 .xinitrc ファイル内の上記の行をコメントに変更したあとで OpenWindows を再起動します。

SPARC: 左きき用のキーの再マッピング

この節で示すキーの再マッピングスクリプト (Type-4 と Type-5 のキーボード用) を使って、キーボードの左右のパネル上にある大部分の特殊キー (つまり、キーボードの左側と右側にあるキーパッド) を再マッピングします。

SPARC のみ – この節の以降の説明はすべて、SPARC ベースのマシンだけに適用されるので注意してください。

SPARC: 再マッピングスクリプトの使い方

キーボードの左右のパネル上にある特殊キーをマッピングし直すには次のようにします。

1. 任意のテキストエディタを使って、**`lefty.data`** というスクリプトファイルを作成します。

   このファイルは、どのディレクトリにあってもかまいません。ただし 手順 4 は、このファイルを作成したのと同じディレクトリで実行しなければなりません。

2. 後述の**148** ページの「**`lefty.data`** スクリプト」 に示すスクリプトを入力します。

   先頭に感嘆符 (!) が付いた行はコメント行で、実際の操作は実行しません。

3. ファイルを保存してエディタを終了します。

4. プロンプトに対して、次のコマンドを入力します。

   ```
   $ xmodmap lefty.data
   ```

   注 – 上記のコマンドは、スクリプトファイル `lefty.data` と同じディレクトリ内で入力してください。

   ---

5. ワークスペース内でマウスボタンをクリックして、スクリプトの内容を有効にします。

   以上の作業で、左ききの人用にマッピングされたキーボードを使えるようになります。

`lefty.data` スクリプト

手順 2 で説明したように、次のスクリプトを `lefty.data` ファイルに入力します。

```
!
! lefty.data
!
! Data for xmodmap to set up the left and right function keys
! for left-handed use on Sun Type-4 keyboard. To use this data,
! type the following where <filename> is the name of the file
! (i.e., lefty.data).
!
! xmodmap <filename>
!
! The comments below correspond to the keycode assignments
! following immediately thereafter.
!
! swap L2 (Again) with R1 (Pause)
! swap L3 (Props) with R6 (KP_Multiply)
! swap L4 (Undo) with R4 (KP_Equal)
! swap L5 (Front) with R9 (KP_9)
! swap L6 (Copy) with R7 (KP_7)
! swap L7 (Open) with R12 (KP_6)
! swap L8 (Paste) with R10 (KP_Left)
! swap L9 (Find) with R15 (KP_3)
! swap L10 (Cut) with R13 (KP_1)
!
! chng R3 (Break) to L1 (Stop)
! chng R2 (Print) to R10 (Left)
! chng R5 (KP_Divide) to R12 (Right)
!
! chng Linefeed to Control-R
```

```
!
keycode 10 = R1 R1 Pause
keycode 28 = L2 L2 SunAgain
keycode 32 = R6 R6 KP_Multiply
keycode 54 = L3 L3 SunProps
keycode 33 = R4 R4 KP_Equal
keycode 52 = L4 L4 SunUndo
keycode 56 = R9 R9 KP_9 Prior
keycode 77 = L5 L5 SunFront
keycode 58 = R7 R7 KP_7 Home


keycode 75 = L6 L6 SunCopy
keycode 79 = Right R12 KP_6
keycode 100 = L7 L7 SunOpen
keycode 80 = Left R10 KP_4
keycode 98 = L8 L8 SunPaste
keycode 102 = R15 R15 KP_3 Next
keycode 121 = L9 L9 SunFind
keycode 104 = R13 R13 KP_1 End
keycode 119 = L10 L10 SunCut
keycode 30 = L1 L1 SunStop
keycode 29 = Left R10 KP_4
keycode 53 = Right R12 KP_6
keycode 118 = Control_R
add control = Control_R
```

SPARC: キーボードの再マッピングの取り消し

次の方法で、キーを元の設定に戻すことができます。

- Solaris オペレーティング環境を終了して起動し直す
- 別のスクリプトを作成し、元の状態に戻したい時点でそのスクリプトを実行する

再設定用のスクリプトを作成するには、次の手順を実行します。

1. テキストエディタを使って、**nolefty.data** というスクリプトファイルを作成します。
2. 後述の**151** ページの「**nolefty.data** スクリプト」 に示すスクリプトを入力します。

   先頭に感嘆符 (!) が付いた行はコメント行で、実際の操作は実行しません。
3. **lefty.data** スクリプトと同じディレクトリに **nolefty.data** スクリプトを保存し、エディタを終了します。
4. プロンプトで次のコマンドを入力します。

```
$ xmodmap nolefty.data
```

注 – nolefty.data ファイルを有効にするには、上記のコマンドを nolefty.data が存在するディレクトリで実行してください。

---

nolefty.data スクリプト

```
!

! nolefty.data

!

! Data for xmodmap to reset the left and right function keys

! after being set for left-handed use on the Sun type-4 keyboard.

! To use this data, type the following where <filename> is the name

! of the file.

!

! xmodmap <filename>

!

! Reassign standard values to left function keys.
```

```
!
keycode 10 = L2 L2 SunAgain
keycode 32 = L3 L3 SunProps
keycode 33 = L4 L4 SunUndo
keycode 56 = L5 L5 SunFront
keycode 58 = L6 L6 SunCopy
keycode 79 = L7 L7 SunOpen
keycode 80 = L8 L8 SunPaste
keycode 102 = L9 L9 SunFind
keycode 104 = L10 L10 SunCut
!
! Reassign standard values to right function keys.
!
keycode 28 = R1 R1 Pause
keycode 29 = R2 R2 Print
keycode 30 = R3 R3 Scroll_Lock Break
keycode 52 = R4 R4 KP_Equal
keycode 53 = R5 R5 KP_Divide
keycode 54 = R6 R6 KP_Multiply
keycode 75 = R7 R7 KP_7 Home
keycode 77 = R9 R9 KP_9 Prior
keycode 98 = Left R10 KP_4
keycode 100 = Right R12 KP_6
keycode 119 = R13 R13 KP_1 End
keycode 121 = R15 R15 KP_3 Next
!
! Reassign the Linefeed key as such and remove from control map.
!
```

```
remove control = Control_R

keycode 118 = Linefeed
```

x86: ファンクションキーとコントロールキーの再マッピング

x86 マシンのファンクションキーを再マッピングして、SPARC キーボード上にある Help、Cut、Copy、Paste、Undo、および Front などの各キーとして機能させることができます。また右側のコントロールキーを再マッピングして、メタキーにすることもできます。

---

x86 のみ – この節の以降の説明は、x86マシンだけに適用されるので注意してください。キーを再マッピングした場合、キーボードの再マッピングを取り消してからでないと `kdmconfig` を使って設定情報やビデオ情報を変更することはできません。

---

x86: 再マッピングスクリプトの使い方

再マッピングスクリプトを作成し使用する手順は次のとおりです。

1. 任意のテキストエディタを使って、ユーザのホームディレクトリ上に **`fkeys`** というスクリプトファイルを作成します。
2. 後述の**154** ページの「**`fkeys`** スクリプト」 に示すスクリプトを入力します。
3. ファイルを保存してエディタを終了します。
4. プロンプトで次のコマンドを入力します。

   ```
   $ xmodmap fkeys
   ```

   注 – 上記のコマンドは、スクリプトファイル `fkeys` と同じディレクトリ内で入力してください。

   ---

5. ワークスペース内でマウスボタンをクリックして、スクリプトの内容を有効にします。

   上の手順を実行すると、ファンクションキーを Help キー、Cut キー、Copy キー、Paste キー、Undo キー、および Front キーとして使用することができます。

fkeys スクリプト

```
!

keysym F2 = L10

keysym F3 = L6

keysym F4 = L8

keysym F5 = L9


keysym F8 = L4

keysym F9 = L5


remove control = Control_R

keycode 0x47 = Meta_R

add mod1 = Meta_R
```

x86: キーボードの再マッピングの取り消し

キーボードを元の設定に戻すには、次に示す 2 つの方法があります。

- Solaris オペレーティング環境を終了して起動し直す
- 別のスクリプトを作成し、元の状態に戻したい時点でそのスクリプトを実行する

再設定用のスクリプトを作成するには、次の手順を実行します。

1. テキストエディタを使って、**normal** というファイルを作成します。
2. 後述の **155** ページの「**normal** スクリプト」に示すスクリプトを入力します。
3. **fkeys** スクリプトと同じディレクトリに **normal** ファイルを保存し、エディタを終了します。
4. プロンプトに対して、次のコマンドを入力します。

   ```
   $ xmodmap normal
   ```

注 – 上記のコマンドは、スクリプトファイル normal と同じディレクトリ内で入力してください。

---

normal スクリプト

手順 1 で説明したように、次のスクリプトを normal ファイルに入力します。

```
keycode 8 = grave asciitilde

keycode 9 = 1 exclam

keycode 10 = 2 at

keycode 11 = 3 numbersign

keycode 12 = 4 dollar

keycode 13 = 5 percent

keycode 14 = 6 asciicircum

keycode 15 = 7 ampersand

keycode 16 = 8 asterisk

keycode 17 = 9 parenleft

keycode 18 = 0 parenright


keycode 19 = minus underscore

keycode 20 = equal plus

keycode 21 =

keycode 22 = BackSpace

keycode 23 = Tab

keycode 24 = Q

keycode 25 = W

keycode 26 = E

keycode 27 = R

keycode 28 = T

keycode 29 = Y

keycode 30 = U

keycode 31 = I

keycode 32 = O
```

```
keycode 33 = P

keycode 34 = bracketleft braceleft

keycode 35 = bracketright braceright

keycode 36 = backslash bar brokenbar

keycode 37 = Caps_Lock


keycode 38 = A

keycode 39 = S

keycode 40 = D

keycode 41 = F

keycode 42 = G

keycode 43 = H

keycode 44 = J

keycode 45 = K

keycode 46 = L

keycode 47 = semicolon colon

keycode 48 = apostrophe quotedbl

keycode 49 =

keycode 50 = Return

keycode 51 = Shift_L

keycode 52 =

keycode 53 = Z

keycode 54 = X

keycode 55 = C

keycode 56 = V

keycode 57 = B

keycode 58 = N
```

```
keycode 59 = M
keycode 60 = comma less
keycode 61 = period greater
keycode 62 = slash question
keycode 63 =
keycode 64 = Shift_R
keycode 65 = Control_L
keycode 66 =
keycode 67 = Alt_L
keycode 68 = space
keycode 69 = Alt_R
keycode 70 =
keycode 71 = Control_R
keycode 72 =
keycode 73 =
keycode 74 =
keycode 75 =
keycode 76 =
keycode 77 =
keycode 78 =
keycode 79 =
keycode 80 =
keycode 81 =
keycode 82 = Insert
keycode 83 = Delete
keycode 84 =
keycode 85 =
keycode 86 = Left
```

```
keycode 87 = Home

keycode 88 = End

keycode 89 =

keycode 90 = Up

keycode 91 = Down

keycode 92 = Prior

keycode 93 = Next

keycode 94 =

keycode 95 =

keycode 96 = Right

keycode 97 = Num_Lock

keycode 98 = Home KP_7 KP_7

keycode 99 = Left KP_4 KP_4

keycode 100 = End KP_1 KP_1

keycode 101 =

keycode 102 = KP_Divide

keycode 103 = Up KP_8 KP_8

keycode 104 = KP_5 KP_5 KP_5

keycode 105 = Down KP_2 KP_2

keycode 106 = KP_Insert KP_0 KP_0

keycode 107 = KP_Multiply

keycode 108 = Prior KP_9 KP_9

keycode 109 = Right KP_6 KP_6

keycode 110 = Next KP_3 KP_3

keycode 111 = Delete KP_Decimal KP_Decimal

keycode 112 = KP_Subtract

keycode 113 = KP_Add
```

```
keycode 114 =

keycode 115 = KP_Enter

keycode 116 =

keycode 117 = Escape

keycode 118 =

keycode 119 = F1

keycode 120 = F2

keycode 121 = F3

keycode 122 = F4

keycode 123 = F5

keycode 124 = F6

keycode 125 = F7

keycode 126 = F8

keycode 127 = F9

keycode 128 = F10

keycode 129 = SunF36

keycode 130 = SunF37

keycode 131 = Print SunSys_Req

keycode 132 = Scroll_Lock

keycode 133 = Pause Break

keycode 134 =

keycode 135 = Multi_key

keycode 136 = Mode_switch
```

索引

C

N

O

P

Q

R

S

T

お

か

き

く

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

わ