Linuxの基礎

情報教育システムの Linux/Windows 端末では, 起動時に Linux(CentOS または Ubuntu) と Windows から OS を選択することができる. 本稿では CentOS 環境についての基本的事項を解説する.

情報教育システムの Linux 環境 (CentOS) では, 標準で GNOME と呼ばれるデスクトップ環境を使用する. ただし, Linux へのログイン時にデスクトップとして KDE を選択したり, 使用言語を変更することもできる.[*1] こうしたデスクトップ環境は, Windows や MacOS と同様にグラフィカルユーザインタフェース (GUI, Graphical User Interface) を備えており, マウスを用いたシンプルな操作によってある程度のことは簡単にできる. 従って, 初心者でも簡単に使うことができる反面, より高度で素早い操作には不向きな面もある. Linux をはじめとする UNIX 系 OS の環境では, コマンドを直接キーボードから入力することによって操作するインターフェース (CUI, Command User Interface) によって計算機を操作する方がよい場合がある. それゆえ, UNIX の基本的なコマンドの知識を持っていると非常に役立つ.

第 1 節では, Linux のデスクトップ環境について説明する. ここでは CentOS の標準的なデスクトップ環境である GNOME について説明する. 第 2 節では Linux の基本的なコマンドとその使い方について概説する.

1　デスクトップの使い方

1.1　GNOME デスクトップの構造について

GNOME デスクトップの構造は, 図 1(左) のようになっている.

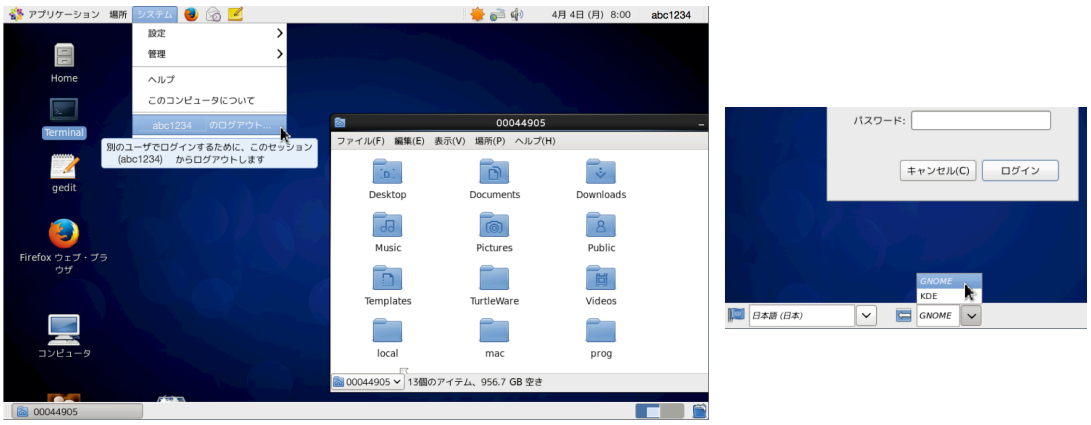

図 1　GNOME デスクトップの例（左）. 画面右上のアイコンをクリックして.「... のログアウト」を選択した状態. 右は, ログイン時のデスクトップ選択パネル.

(1) デスクトップ：画面上の何も置かれていない, 背景のみの部分. アイコン (次項参照) が配置されたり,

[*1] ログインパネルでユーザー ID を入力し, パスワード入力画面に進むと, 画面の下側に図 1(右) のような選択パネルが現れる. 特に理由が無い限り, GNOME を選択する. デスクトップの様子が普段と異なる場合は, 一旦ログアウトして, デスクトップ環境を選択し直してみると良い.

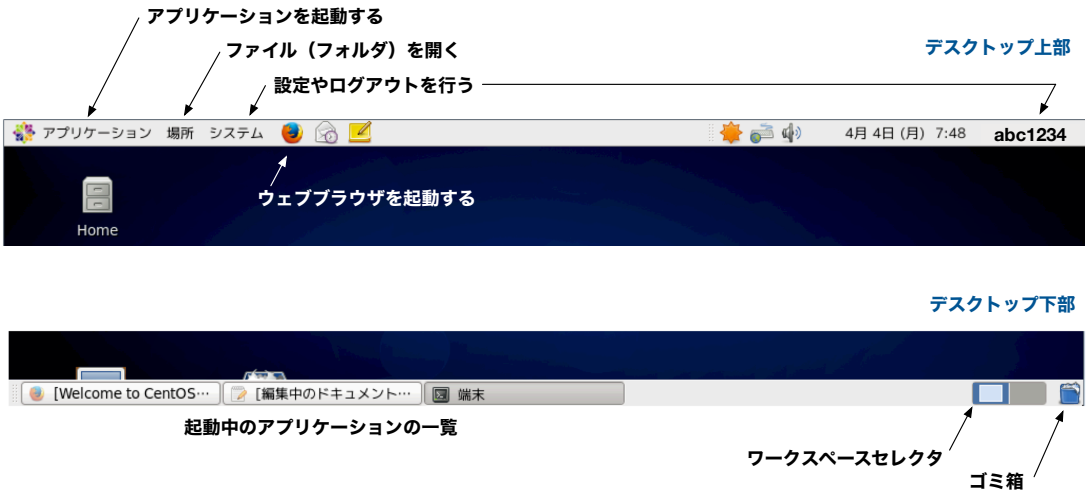

図 2　GNOME デスクトップのタスクバー

ウィンドウ (後述) が開いたりする作業スペースになっている．この領域で右クリックすると，メニューが現れて，背景画像の変更などをすることができる．

(2) アイコン：画面左側には**アイコン** (icon) と呼ばれる絵ボタンが並んでいる．アイコンをダブルクリックすると，そのアイコンに対応するプログラムが起動したり，ファイルが開いたりする．このことから，アイコンのことを対応するプログラムやファイルへの**ショートカット** (shortcut) と呼ぶこともある．設定によって異なるが，およそ次のアイコンが並んでいる．

- **Terminal**
  コンソール（端末）が起動する．コンソールについては次節で説明する．
- **Firefox ウェブブラウザ**
  ウェブブラウザ (Firefox) が起動する．
- **Home**
  ファイルマネージャが起動する．初期ディレクトリはホームディレクトリである．
- **gedit**
  テキストエディタ (gedit) が起動する．

(3) タスクバー：画面上にある．様々なメニューを選択したり，アプリケーションを起動したりする時に使用する．タスクバーの構造は，図 2 のようになっている．各々のアイコンをクリックすると対応するメニューが開いたり，プログラムが起動してウィンドウがデスクトップ上に現れたりする．

- **アプリケーションメニュー**
  アプリケーションを起動したり，ログアウトしたりする時に開く．図 3 は，「オフィス」メニューから「LibreOffice Writer(ワープロ)」を開いた例である．
- **場所メニュー**
  ファイルやディレクトリ（フォルダ）開いたり，操作する．
- **システムメニュー**
  システム設定やログアウトを行う．作業が終了したら，必ずこのメニューを開いて，「... のログアウ

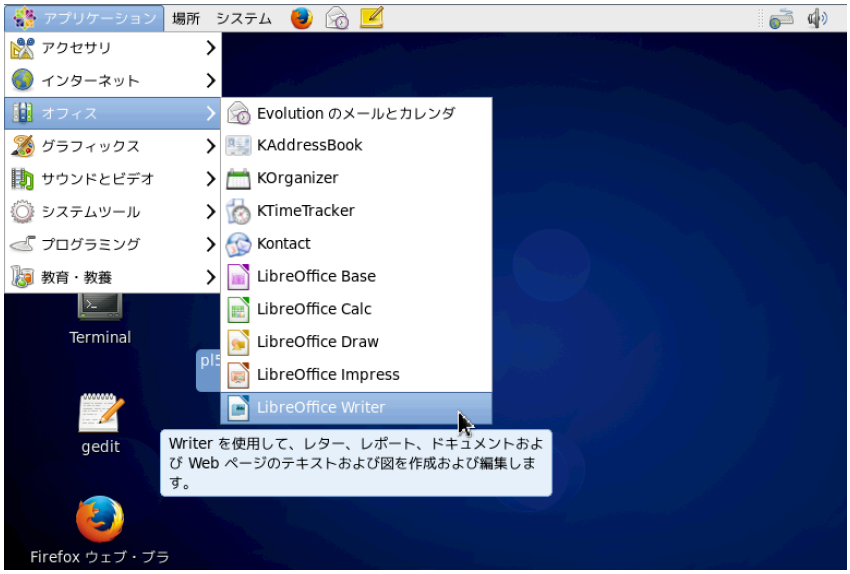

図 3　アプリケーションメニュー

ト」を選択すること．ログアウトしないまま, 端末の電源を切ってはならない．

また, 画面の下部には, 起動中のアプリケーションの一覧や, ワークスペースを切り替えるためのパネルが配置されている (図 2(下))．ウィンドウを最小化 (次節で説明) すると, デスクトップ上からは消えるが, そのアプリケーションは動作を続けている状態にある．そんなとき, 画面下のパネルをクリックすると, 再びデスクトップ上にウィンドウを「呼び出す」ことができる．また, ワークスペースセレクタを使うと, 実質ふたつの画面を切替えて使うことができる．複数のアプリケーションを切替えながら作業する際に便利かもしれない．

1.2　ウィンドウの基本操作

GNOME では, 一部の例外を除いて全てのアプリケーションは**ウィンドウ** (window) と呼ばれる画面上の領域の中で動作する．本項では, コンソールを例にとってウィンドウの操作について説明する[*2]．

コンソールを起動するには, 次のようにする．

(1) 画面左上の「アプリケーション」メニューをクリックした後,「システムツール」にマウスカーソルを合わせる．
(2) 出てきたサブメニューの中から「端末」をクリックする．

デスクトップに「端末 (Terminal)」のアイコンがある場合は, それをダブルクリックしてもよい．すると, デスクトップ上に図 4 のようなウィンドウが現れる．

(1) タイトルバー：そのウィンドウの名前を表示している．たいていはプログラム名や編集中のファイル名などが表示される．

[*2] **コンソール (console)** とは, 計算機と会話する時の窓口になるソフトウェアのことで, **端末 (ターミナル, terminal)** とも呼ばれる．なお, GNOME のコンソールは “Terminal” という名前である．

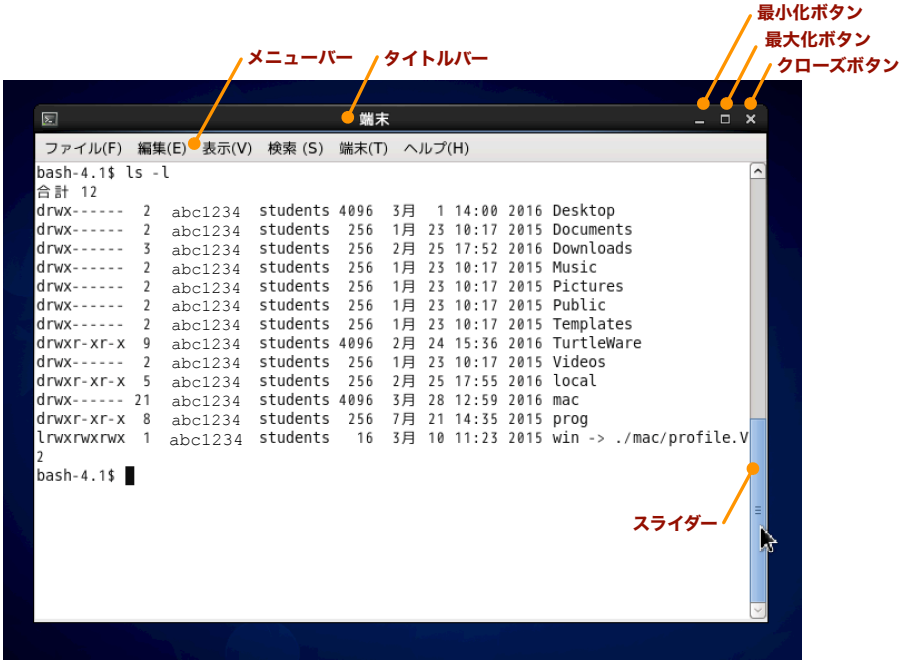

図 4　ウィンドウの例 (ターミナル)

(2) **メニューバー**：クリックするとサブメニューが現れ, そこから希望の機能を選んでクリックすることで, 所定の作業を実行する. メニューの中身は, プログラムによって異なる. プログラムによってはメニューバーとともにアイコンが並ぶこともある.

(3) **スクロールバー**：ウィンドウ本体の大きさが編集対象のファイルに比べて小さいときに現れる. バーをドラッグして, マウスを動かすとそれに対応して画面がスクロールする[*3]. 図 4 では上下方向のスクロールバー（スライダー）が出ているだけだが, ウィンドウの下側に左右方向のスクロールバーが出ることもある.

(4) **最小化ボタン**：クリックすると, ウィンドウ全体がタスクバーの中に折り畳まれる. 折り畳まれたものをクリックすると, また元どおりになる. **このボタンでプログラムが終了するわけではないので, 注意すること**.

(5) **最大化ボタン**：クリックすると, ウィンドウが画面いっぱいに広がる. 最大化された状態では, ウィンドウの大きさを元に戻すボタンとして機能する.

(6) **クローズボタン**：クリックすると, プログラムを終了してウィンドウを閉じる.

■ウィンドウの配置　デスクトップ上に複数のウィンドウを同時に配置させることができる. ウィンドウが重なり合っている部分は, どちらかのウィンドウが下側に隠れる. ウィンドウの一部 (タイトルバーや枠など) を左クリックすると, そのウィンドウが作業対象のウィンドウとして一番上に来る. 作業対象になっているウィンドウは**アクティブウィンドウ** (active window) と呼ばれ, 他のウィンドウが暗い色をしているのに対して,

[*3] 画面上に表示しきれない情報があるとき, 画面上で情報を適宜上下または左右の方向にずらして表示させることを**スクロール**と言う.

明るい色になっているのがわかる.

■ウィンドウの移動・大きさ変更　ウィンドウの枠あるいは四隅付近にマウスカーソルを近づけると, マウスカーソルの形が '↔' のような形に変わる. その状態のときに, ウィンドウの枠 (または隅) をドラッグしてマウスを動かすと, それに応じてウィンドウの大きさが変わる. マウスのボタンを離すと, その時点での大きさになる. ウィンドウのタイトルバーをドラッグしてマウスを動かすと, ウィンドウを動かすことができる.

2　Linux の基本コマンド

本節では, 情報教育システムの Linux 環境におけるコマンドの概要について説明する. 現在では GUI による直感的な操作ができるようになっているが, 本稿の冒頭で述べたようにコマンドによる操作にも慣れておいたほうがいろいろと便利である. なお, ここでは情報教育システムの Linux 環境を意識して説明するが, ほとんどの部分は Linux 以外の UNIX 系 OS でも一般的に通用する.

2.1　コマンドとシェル

ユーザがキーボードなどから直接文字列の形で入力して, 計算機を動作させる命令のことを**コマンド** (command) と呼ぶ. Linux におけるコマンドには, 大きく分けて次の 2 種類がある.

- Linux にはじめから備わっているコマンド.
- シェルが受け付けるコマンド.

シェル (shell) とは, コマンド入力を解釈するソフトウェア (**コマンドインタプリタ**) である. シェルは Linux の中核部分 (**カーネル**, kernel) を包み込み, ユーザからのコマンドを Linux のカーネルへ渡す役割を担う. Linux のカーネルは, このようにしてシェルから伝えられた命令を実行する. Linux のシェルには様々な種類のものがあり, 各ユーザは自分の好みのシェルを選んで使うことができる. 情報教育システムでは, bash と呼ばれるシェルを採用している. シェル上では, **シェルスクリプト**と呼ばれる簡単なプログラムを作成することができる. Linux に本来備わっているコマンドはどのシェル上でも動作する.

どちらのタイプのコマンドも, **コンソール** (console) と呼ばれるウィンドウから, 直接キーボードまたはマウスを用いてコマンド文字列を入力する操作によって使用する. 従って, 普段使っている際には両者の区別をそれほど意識する必要はなく, むしろコマンドを機能別に分類して考えたほうが都合が良い. 本節では, 次のような機能区分に従っていくつかの基本的なコマンドを解説する.

- ファイルシステム関連コマンド.
- 利用環境関連コマンド.
- リダイレクションとパイプ.

■コンソールによるコマンド入力　ここではコンソール (図 5) の使い方を解説する.

`bash-4.1$` と書かれた行は**コマンドライン** (command line) と呼ばれ, `$` 記号の後ろに Linux のコマンドを入力して, 計算機に動作を指示する. コマンド入力は, Enter を押すことにより計算機に伝わる. `bash-4.1$` のような表示は**プロンプト** (prompt) と呼ばれ, ユーザの入力を待っていることを表す[*4]. 画面は自動的にス

[*4] プロンプトのメッセージは, システムやシェルによって異なり, 自由に変更することもできる.

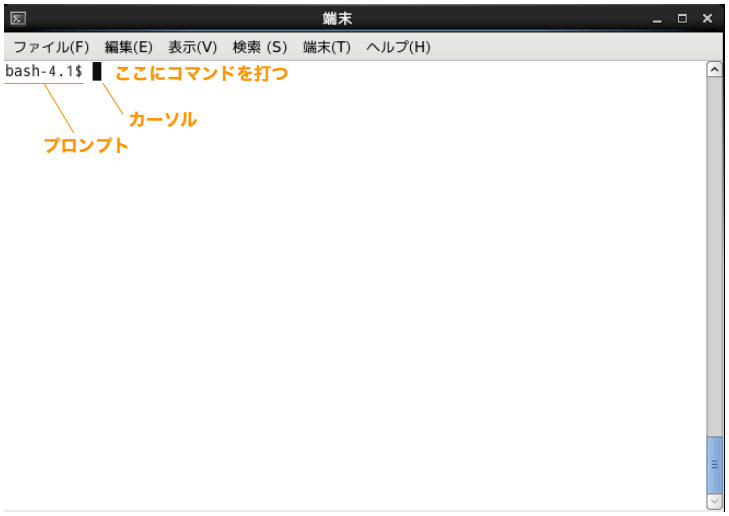

図 5　コンソール (bash)

クロールしていくが, `clear` コマンドで画面をクリアすることもできる. bash を使っている時は, カーソルキー [↑], [↓] で過去に入力したコマンドの履歴をたどることができる (コマンドヒストリー機能). また, bash にはコマンド入力途中の適当な所で [Tab] を押すと, 残りの入力部分を自動的に補完してくれる機能が備わっている.

2.2　ファイルシステム関連コマンド

■ファイル操作に関するコマンド

`cat`　複数のファイルを連結して表示する. (concatenate)

例:　`cat file1`　(ファイル `file1` の中身を表示)

`cat file1 file2`　(ファイル `file1` に `file2` を連結して表示)

`chmod`　Linux ではファイルやディレクトリに保護属性が設定されている. その設定を変えることで, 例えば自分が作ったファイルを他人が勝手に中身を見たり, 書き換えたりできなくなるようにすることができる. `chmod` は, ファイルの保護属性を設定するコマンドである. (change mode)

例:　`chmod o-r file1`　(`file1` の中身を, 所有者以外が参照できないように指定)

具体的な設定方法については, Linux あるいはその他の UNIX 系 OS の参考書を参照のこと. なお, アクセス権限のないファイルに対する変更はできない.

`cmp`　2 つのファイルを比較する. (compare)

例:　`cmp file1 file2`　(`file1` と `file2` を比較)

`cp`　ファイルを複製する. (copy)

例:　`cp file1 file2`　(`file1` の内容を, `file2` として保存)

`less`　ファイルの中身を表示する. `less` コマンドには, 画面の大きさで区切って, カーソルキーで上下にスクロールさせる機能が付いている.

例: `less file1` (`file1` の内容を表示)

`more` ファイルの中身を表示する. `more` には画面の大きさで区切って表示する機能があるが, `less` とは違って上に遡ることはできない.

例: `more file1` (`file1` の内容を表示)

`mv` ファイルの場所を移動する. ファイル名の変更にも使用される. (move)

例 1: `mv file1 dir/` (`file1` を, ディレクトリ `dir` の下に移動)

例 2: `mv file1 dir/file2` (`file1` を, ディレクトリ `dir` の下の `file2` に移動)

例 3: `mv file1 file2` (`file1` を, 同じディレクトリの `file2` に移動)

例 1 では, ファイルがある場所が変わるだけでファイル名の変更はない. 例 2 では, ファイルがある場所が変わるだけでなく, ファイルの名前も変わる. 例 3 は, 同じディレクトリ上での移動であり, 単なるファイル名の変更である.

`rm` ファイルを削除する. (remove) 情報教育システムの Linux 環境では `rm` コマンドでファイルを削除する際, 削除してよいかどうかの確認メッセージが表示される.[*5] 削除してよいなら `y`, 削除をやめるなら `n` と入力して Enter を押す.

例: `rm file1` (`file1` を削除)

`rm file1 file2` (`file1` と `file2` を削除)

`rm file*` (名前が `file` で始まる全てのファイルを削除)

`lpr` ファイルをプリンタへ出力する.

一般形: `lpr -P`*printername filename*

printername の所には, 出力先のプリンタ名を記述する. 情報教育システムにおけるプリンタ名については, オンラインガイドを参照のこと. *filename* の所には, 出力したいファイルの名前を記述する.

■ディレクトリ操作に関するコマンド

`cd` 作業ディレクトリを移動する. (change directory)

例 1: `cd` (ホームディレクトリへ移動)

例 2: `cd ..` (現在の親ディレクトリへ移動)

例 3: `cd kadai` (現在の子ディレクトリ `kadai` へ移動)

例 4: `cd kadai/tex` (現在の孫ディレクトリ `kadai/tex` へ移動)

`ls` ディレクトリの中身 (ファイル名, 子ディレクトリ名) の一覧を表示する. (list)

例 1: `ls` (作業ディレクトリの中身を一覧表示)

例 2: `ls -l` (作業ディレクトリの中身一覧をより詳しく表示)

例 3: `ls ~/temp/latex` (表示するディレクトリを相対パス指定している)

例 4: `ls k*` (作業ディレクトリ中で `k` で始まるファイルを全て表示)

補足: 例 3 で用いた記号 `~/` は, ホームディレクトリを表す.

[*5] 確認メッセージが出ないシステムもある. そのようなときには, `rm -i` のように "`-i`" を付けると確認メッセージが出るようになる.

`mkdir` 新しいディレクトリを作成する (make directory).

例: `mkdir new1` (作業ディレクトリの子ディレクトリ `new1` を生成)

`rmdir` ディレクトリを削除する (remove directory). なお, `rmdir` は削除したいディレクトリの中身を空にしてから使用する.

例: `rmdir new1` (子ディレクトリ `new1` を消去)

`pwd` 現在の作業ディレクトリを絶対パスで表示する. (present working directory)

2.3 利用環境関連コマンド

■利用環境の操作

`&` コマンドをバックグラウンドで並列動作させる.

例: `a.out &` (`a.out &` を実行するプロセスを生成)

`exit` 現在使用中のシェルを終了する.

`kill` プロセス (計算機が処理中の仕事) を強制終了させる. `ps` コマンド (後述) で該当するプロセスのプロセス番号 (ID) を確認して, `kill` でそのプロセスを終了させる.

例: `kill -9 7681` (プロセス ID が 7681 のプロセスを強制終了)

`setenv` 環境変数を設定する.

■利用環境の確認・照会

`cal` カレンダーを出力する.

例: `cal 4 2010` (2010 年 4 月のカレンダーを表示)

`date` 今日の日付を表示する.

`man` 各種コマンドの説明書を表示する.

例: `man ps` (`ps` コマンドの説明書を表示)

`printenv` 環境変数の情報を出力する.

`ps` 稼働中のプロセスに関する情報を出力する.

`time` コマンドの実行時間を計測する.

例: `time a.out` (`a.out` の実行時間を計測)

2.4 リダイレクションとパイプ

Linux では, 入出力装置や記憶装置自体も一つのファイルとして扱われる. 従って, あるコマンドの出力結果を直接ファイルに保存したり, キーボードからデータを入力する代わりに直接ファイルから読み込ませたりすることができる. また, あるコマンドの出力をそのまま別のコマンドの入力へ接続させるといったこともできる.

■リダイレクション 標準では, データの入力先はキーボード, 出力先はディスプレイとなっているが, それを意図的に変更することができる. そのような操作のことを**リダイレクション** (redirection) と言う. 例えば, `ls`

コマンドを実行すると画面にファイルのリストが表示されるが, それを

```
ls > filelist.txt
```

のようにして実行すると, 結果は画面には出力されず, 代わりにファイル filelist.txt へ出力される (filelist.txt が存在しないときは, 自動的に生成される). また, あるプログラム a.out があって, 本来ならばそれを実行するためにキーボードからデータを入力すべきとき, 予め別のファイル data.txt を作っておいて

```
a.out < data.txt
```

として実行すれば, キーボード入力の代わりにファイル data.txt からデータを読み取って動作する (キーボードからの入力は要求されない). このように, リダイレクションでは入力先の変更を <, 出力先の変更を > で指示する.

例 1 `cal 4 2010 > calApr2010.txt`
cal 4 2010 は 2010 年 4 月のカレンダーを, 画面の代わりにファイル calApr2010.txt にする (画面には表示されない).

例 2 `a.out < data.txt >output.txt`
プログラム a.out を, キーボード入力の代わりにデータファイル data.txt からの入力で動作させ, その処理結果を画面の代わりにファイル output.txt へ出力する.

例 3 `cal 4 2005 >> calender2005.txt`
cal コマンドの処理結果を, ファイル calender2005.txt へ「追加」する形で出力する.

■パイプ あるコマンドの出力結果をそのまま別のコマンドの入力に接続させることを**パイプ** (pipe) と言う. 2 つのコマンドをパイプで連結させるには, 縦棒記号 | を使用する. リダイレクションとパイプを組み合わせて使うこともできる.

例 1 `ls -l | grep *.c`
ls -l は作業ディレクトリのファイルリストを生成するが, そのリストはパターン照合コマンド grep の入力として直接渡されます. grep *.c はそのリストの中から, .c で終わるファイルだけを抜き出して画面に表示させる.

例 2 `ls -l | less`
ls -l で生成されるファイルリストが長い場合は, このように less へパイプすることにより, 1 画面ごとに表示させることができる.

例 3 `program1 < data.txt | program2`
program1 はデータファイル data.txt からデータを読み込んで動作する. その処理結果は, 直接 program2 の入力として引き渡される. program2 の処理結果は, 画面に出力される.

例 4 `program1 < data.txt | program2 > output.txt`
途中までは例 3 と同じであるが, program2 の処理結果が画面ではなくファイル output.txt へ出力される.