ウィンドウシステムとエディタ

目 次

1　テキストファイルとエディタ

cat コマンドや more コマンド等で内容を表示できる，文字の書かれたファイルを**テキストファイル** (text file) といいます。一方で，cat コマンド等では中身を読めない通常のファイルもあります。これをバイナリファイル (binary file) と呼びます[1]。例えば，画像や音声の入ったファイルの多くや，Windows 上のワープロ等で “普通に” 保存したファイルの多くはバイナリファイルです。

UNIX では文書等のデータを基本的にテキストファイルとして保存します。その理由にテキストファイルの持つ汎用性が挙げられます。テキストファイルは，多くのコマンドやワープロ等のアプリケーションで共通に使えるファイル形式ですので，テキストファイルに対しては各種コマンドやアプリケーションによる様々な処理を施せます。

また UNIX 系の OS では，OS 自体の諸設定や各種アプリケーションの動作環境を，テキストファイルに記述することが多いです。コンピュータプログラミングをするにもテキストファイルを作成しなければなりません。そのため，テキストファイルを作成したり編集する道具を使えることは大切です。

さて，テキストファイルを作成したり編集するためのプログラムを**テキストエディタ** (text editor) といいます。単にエディタと呼ぶことも多いです。UNIX にはさまざまなエディタがありますが，この資料の後の章では UNIX の標準的なテキストエディタである vi と GNU Emacs の基本的な使い方を紹介します。

2　vi エディタを使ってみよう

2.1　vi の基本操作

2.1.1　vi の起動法

vi を起動するには，コマンド行で

`vi` *file*

を実行します[2]。ここで *file* には，新規に作成するファイルの名前や，編集したい既存のファイル名を与えます[3]。

2.1.2　vi の編集モード

vi はモードという概念を持つエディタです。vi では，現在どのモードになっているかにより，使用可能な操作が異なります。

コマンド入力モード vi 起動直後には，このモードになっています。カーソル移動や，文字の消去，ファイルへの保存等の操作は，コマンド入力モードにおいて，vi のコマンドを入力することにより行います。vi のコマンドの一部を第 2.4 節に挙げますので，参考にしてください。

---

[1] より正確にいうと，文字コードと若干の制御コードのみを含むファイルがテキストファイルであり，それ以外のファイルがバイナリファイルです。cat コマンドを使うと，ファイル内の文字コードが文字に変換されて表示されますが，od というコマンドを使うとファイル内の文字コードそのものを見ることができます。

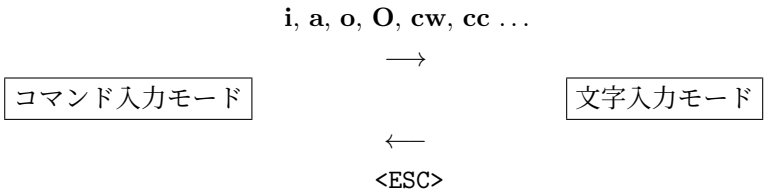
[2] 最近のシステムでは vi というコマンドを実行すると，実際には vi の機能拡張版である vim が起動するものが多いです。また，vi コマンドで vi が起動しない場合，代わりに vim と打ってみてください。

[3] *file* を指定せずに vi を起動してから，既存のファイルを読み込んだりもできますが，ここでは必ずファイル名を指定して起動することにします。また，ファイル名を複数指定して編集する機能も扱いません。

コマンド入力モードでは文字の入力はできません。**i** や **a** 等の vi のコマンドを入力し，次に紹介する文字入力モードに移行してから，文字入力をします。

文字入力モード このモードは文字を入力するためのモードであり，基本的にそれ以外の操作はできません。カーソル移動等のためには，<ESC> を押して，コマンド入力モードに戻る必要があります[4]。

ただし，<ENTER> や <ESC> を押す前であれば，入力したばかりの文字は <BS> (バックスペースキー) または CTRL-h で削除できます[5]。<BS> を続けて押すと，さらにその前に入力した文字を削除できます。なお，<BS> を押しても，カーソルが左に動くだけで元の文字は表示されたままになっているかもしれませんが，これらは無視してください。<ESC> を押せばちゃんと消えます。



vi では，これらのモードを行ったり来たりしながら，文書を編集します[6]。現在，どちらのモードなのかが分からなくなったら，<ESC> を 1 回または 2 回押して，コマンド入力モードに移行してください。

2.2 例題

2.2.1 例題 1 —ファイルの新規作成

vi を使ってファイルの中身が

```
cal date who
```

である commands という名前のファイルを作成してみます。

1. vi の起動法に従って，コマンド行に `vi commands` と入力して，vi を起動してください。
2. **i** を押してください。

   この操作によって「コマンド入力モード」から「文字入力モード」に移行しました。これ以降，カーソル位置に文字を入力できます。
3. ファイルに書き込む内容 (cal<ENTER>date<ENTER>who) を入力してください。<ENTER> (エンターキー) は改行するためのキーです。タイプミスをしてしまっても，ここでは修正しないでおきましょう。
4. <ESC> (エスケープキー) を押してください。

   この操作によって,「文字入力モード」から「コマンド入力モード」に移行しました。

[4] 文字入力モードでもカーソルキー（矢印キー）でカーソル移動が可能な場合があります。
[5] これらと違うキーを使うシステムもあるかもしれません。
[6] より多くのモードを考えるのがよい場合もありますが，ここでは簡略化してモードは二つとします。

5. **:x** と打って <ENTER> を押してください。この vi のコマンドによって，3. で入力した内容がファイル commands に保存され，vi が終了してプロンプトが現れます。うまくいかない場合，再度 4. と 5. を行ってください。

以上を終えたら，commands という名のファイルが出来上がったことを，ls コマンドを使って確認してください。また，cat コマンドでファイルの内容を確認してください。

2.2.2 例題 2 —既存ファイルの編集

ファイル commands の内容を

```
cal    displays a calender
date   set date and time
```

に変更します。

1. vi の起動法に従って，コマンド行に `vi commands` と入力して vi を起動してください。既存のファイル commands の内容が表示されます。

   もし，画面に commands の内容が表示されなければ，**:x** <ENTER> を打って一旦 vi を終了し，ファイル名を確認してから，再度 vi を起動してください。

2. 文字を追加すべき位置 (cal の l の位置) までカーソルを移動します。カーソル移動の方法は，第 2.4 節を参照してください。**l** (エル) の利用がお勧めです[7]。

3. カーソルの右側に文字を追加するために，**a** を打ちます。

   これにより，文字入力モードに移行しました。コマンド i と a の違いを第 2.4 節で確認してください。

4. 追加すべき文字「`  displays a calender`」(4 個の空白に続いて display...) をタイプします。入力ミスは <BS> で修正できます。

5. <ESC> を押してコマンド入力モードに戻り，date の行の行末までカーソルを移動します。

6. **a** を押して文字入力モードに入り，追加すべき文字を入力します。入力を終えたら <ESC> でコマンド入力モードに戻ります。

7. カーソルを who の行の行頭まで移動させてください。

   これから who を削除しますが，<BS> では文字入力モードで打ち込んだばかりの文字しか消せませんので，ここでは文字削除のコマンドを使います。まず **x** を押してカーソルの乗っている文字 w を消してみましょう。

   一行すべてを削除するコマンドもあります。**dd** と打って，カーソルの乗っている 3 行目を消してください。

8. 編集を終えたら **:x** <ENTER> を打ちます。変更した内容が元のファイルに保存され，vi が終了します。

[7] vi では，右手ホームポジション付近のキーだけでカーソル移動ができます。慣れると快適です。

2.3 練習

1. ファイル commands の内容を

   ```
   cal    displays a calender
   date   display or set date and time
   ```

   に変更してください。

   この練習では，ファイルの内容を，上記の通りにきれいに整えましょう。コマンド説明の出だしが揃っていなければ，揃えてください。不要な空白行があれば，行削除の vi コマンドで削除してください。

   注意: vi は行単位でテキストを編集するエディタなので，改行文字を消すことで複数の行を一行にまとめたりはできません。その代わりにコマンド **J** を使います。

2. vi におけるカーソル移動 (**j** や **k** など) やテキスト検索 (/ や **n** など) の操作は，man でマニュアルを閲覧しているときのページ操作と共通です。特に検索の方法は知っておくと便利です。

   まず ls と ls -t を実行して -t の効果を予想しましょう。次に man ls を実行してオプション -t の意味を調べてください。その際には / を使って -t を検索してください。ls のマニュアルには -t の記述が複数ありますので **n** を使って検索を繰り返してみてください。

3. 実習用のコンピュータで vi を起動すると，実際には vi の拡張版である vim が起動します。vim には操作を学ぶためのチュートリアルがあります。vimtutor というコマンドで起動できますので，時間があれば試してください。

2.4 vi の主要コマンド

機能	コマンド	コメント
テキストを入力する	**i** *text* <ESC>	カーソルの位置に *text* を挿入する (insert)
	a *text* <ESC>	カーソルの右に *text* を追加する (append)
	o *text* <ESC>	今いる（カーソルがある）行の下に *text* を入れる
	O *text* <ESC>	今いる行の上に *text* を入れる
ファイルを保存する	**:w** <ENTER>	編集中のファイルを元の名前のまま保存する
	:w *file* <ENTER>	編集中のファイルを *file* として保存する
vi を出る	**:x** <ENTER> (**ZZ**)	ファイルを保存し vi を出る
	:q! <ENTER>	ファイルを保存せず vi を出る
カーソルを動かす	**h** (←)	左隣に移動する
	l エル (<SPACE> または →)	右隣に移動する
	k (↑)	上に移動する
	j (<ENTER> または ↓)	下に移動する
	0 ゼロ	行頭に移動する
	$	行末に移動する
	CTRL-f　CTRL-b	次ページ / 前ページに移動する
	G	最後の行に移動する
	n **G**	第 *n* 行に移動する
テキストを削除する	**x**	今いる文字（カーソルが乗っている文字）を削除する
	X	カーソルの左隣の文字を削除する
	dw	今いる単語を削除する
	dd	今いる行を削除する
行を連結する	**J**	今いる行に次の行を連結する
テキストを検索する	/ *string* <ENTER>	*string* が最初に現れる位置にカーソルを移動する
	n	一番最近行った検索を繰り返す
テキストを置換する	**r** *character*	今いる文字を *character* で置換する
	cw *text* <ESC>	今いる単語を *text* で置換する
	cc *text* <ESC>	今いる行を *text* で置換する
変更を繰り返す	**.**	直前のコマンドによる変更を繰り返す
変更を取り消す	**u**	直前のコマンドによる変更を取り消す (undo)
テキストをコピーする	**yy**	今いる行を名前なしバッファにコピー（ヤンク）する
	p	名前なしバッファ中のテキストを挿入（プット）する

括弧 () は当該操作を他のコマンドで行えることを表します。そのコマンドを括弧内に併記しています。
斜体字 (*italic*) の箇所は具体的なものに置き換えて記述します。例えば，*text* には入力テキストを，*file* には
ファイル名を書きます。

3　ウィンドウシステム

3.1　X Window System

ウィンドウやマウスを使うための仕組みを提供する一連のプログラムを**ウィンドウシステム**と呼びます。UNIX で標準的に用いられるウィンドウシステムは X Window System です。これを単に X と呼ぶこともあります。

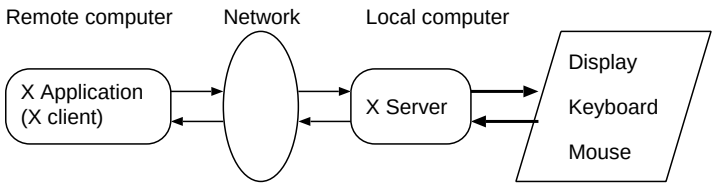
これまでに紹介したコマンドは，コマンドを実行したウィンドウ内に出力が文字で表示されるものばかりでしたが，UNIX にも，Windows で動く一般的なアプリケーションソフトウェアと同様に，新しいウィンドウが開いてそこに絵が表示されたり，マウスを使って操作できるコマンドが存在します。これらは実行時に X を必要とするので[8]，X アプリケーションと呼ばれることがあります。

3.2　X の利用

X アプリケーションは，画面に描いたり，マウスからの入力を受け取る機能を，自分では持っていません。そのため，X サーバーと呼ばれる別のプログラムにこれらの処理を依頼します。X サーバーは X アプリケーションからの依頼に応じて，自身が動いているコンピュータの画面 (display) に何かを出力し，マウス (mouse) やキーボード (keyboard) からの入力を受け取って X アプリケーションに渡します[9]。

遠隔ログインして UNIX 機を利用しているときには，X アプリケーションが動作する remote のコンピュータ (ログイン先の UNIX 機) と，表示や入力に使っている local のコンピュータ (Windows 機) が異なることに注意してください。このような環境で X アプリケーションを使うには，次のことが必要です。

1. local のコンピュータで X サーバーが動いている。
2. remote のコンピュータで動作する X アプリケーションが，どのコンピュータで動いている X サーバーに入出力を依頼すべきかを知っている。



函館校内の教育用計算機から X アプリケーションを利用するための準備手順は

`https://system.hak.hokkyodai.ac.jp/appserver/teraterm_ssh.html`
（学内からのみ閲覧可）

に掲載されています。

[8] X が使えない環境では X アプリケーションも使えません。一方，マウスが不要のコマンドは X が無くても使えます。
[9] なお，X は X サーバーと X アプリケーション間のクライアント／サーバー型の通信プロトコルでもあります。そのため X アプリケーションを X クライアントと呼ぶこともあります。

3.3 X アプリケーションと &

3.3.1 X アプリケーションの特徴

X アプリケーションの起動は，これまでと同様にコマンド入力で行えます。関数電卓プログラム (gcalctool) を例に，X アプリケーションとこれまでのコマンドとの違いを確認しましょう。

1. `gcalctool` と打って，関数電卓プログラムを起動し，マウスを使って何か計算してみましょう。また，キーボードから数値や `+`，`-`，`*`，`/`，`=` などを電卓に入力して計算してみましょう。

2. 次に gcalctool コマンドを入力した (コマンドをタイプした) ウィンドウ内の様子を観察してください。

   wc コマンドを引数なしで実行したときと同様に，新しいプロンプトが現れていません。実行中の gcalctool が実行元のウィンドウを占有しているためです。

   X アプリケーションは，通常，終了の操作をしないと止まりませんので，このままだと別のコマンドを実行できません。

3. 関数電卓ウィンドウの「電卓」メニューから gcalctool を終了させましょう。

   新しいプロンプトが現れます。

4. 再度 `gcalctool` を実行してから，コマンドを入力したウィンドウで CTRL-c を押しましょう。

   gcalctool を実行すると新しい電卓のウィンドウが現れますが，gcalctool というコマンド自身はコマンドの入力元で動いていると考えてください。そこに対して CTRL-c を入力すると，gcalctool は強制終了します。

3.3.2 & の利用

コマンド行の終りに & を付けて X アプリケーションを起動すると，X アプリケーションを動かしながら，別のコマンドを実行できます。

1. `gcalctool &` と打って関数電卓を起動して，起動元の (コマンドをタイプした) ウィンドウに着目しましょう。

   プロンプトが現れましたね。

2. プロンプトに対して，CTRL-c を押しましょう。続いて ls と打ちましょう。

   & を付けてコマンドを実行すると，そのコマンドは起動元で裏側に隠れて動き出します[10]。この状態では，gcalctool を起動元のウィンドウから制御 (終了) させることはできません。その代わりに別のコマンドを実行することが可能になります。

3. gcalctool を終了しましょう。

[10] これをバックグラウンドジョブ (background job) といいます。& を付けないのはフォアグラウンドジョブ (foreground job) です。実行中のフォアグラウンドジョブをバックグラウンドジョブに変更するには，プロンプトが表示されていない起動元のウィンドウで CTRL-z を押してから `bg` を実行します。

注意 1　& をつけてコマンドを実行するのは，X アプリケーション (起動時に新規ウィンドウを生成するコマンド) だけにしてください。それ以外のコマンドに & を付けて実行し,「中断」や Suspended 等と表示されたら,

```
fg
```

というコマンドを実行してください。それでよくわからない状態になったら，CTRL-d や CTRL-c を打ってみてください。

注意 2　& は，リダイレクトの際の < 等と同じくシェルへの指示であり，コマンドの引数ではありません。

3.3.3　練習

1. xclock コマンドを & を付けずに実行して終了しましょう。次に，& を付けて実行してみましょう。なお，資料作成者の知る限りにおいて，時計のウィンドウにキー入力して xclock を終了する方法はありません。

2. `man man &` を実行するとどうなるか，試してください。続いて，今実行した man コマンドを正常に終了させてください。

3. xterm はコマンド入力用のウィンドウを新たに作るコマンドです[11]。xterm コマンドに & を付けて実行し，新しく現れた xterm のウィンドウで適当なコマンドを実行してみましょう。

   xterm を終了させるには，xterm のウィンドウ内で `exit` と打ちます。

[11] この種のプログラムを端末エミュレータ (terminal emulator) といいます。

4 GNU Emacs を使おう

UNIX 系の OS で利用可能なエディタは vi を始めとして，いろいろとあります。その中でも，vi と並んで代表的なエディタとして GNU Emacs があります。本章では，この GNU Emacs の使い方の基本を紹介します[12]。

4.1 キー表記法

GNU Emacs の説明文書では，次のキー表記を使うのが一般的です。

C-文字 コントロールキー (<CTRL>) を押したまま，文字を押します。例えば、C-f はコントロールキーを押したままで f のキーを押すことです。

M-文字 メタキー (普通は <ALT> がメタキーです) を押したまま，文字を押します。または，<ESC> を押して離してから文字を押します。

4.2 起動と終了

4.2.1 起動方法

GNU Emacs (以下，単に Emacs という) を起動するためのコマンドは emacs です。X Window System が動作している環境では，コマンド行オプションの有無によって，二通りの動作形態があります。

1. emacs (オプション無しで起動)：起動時に新規ウィンドウを生成して動作する。

   Windows のアプリケーションを操作するのと同様の感覚で，マウスを使って Emacs を操作できます。

2. emacs -nw または emacs --no-windows：コマンドを入力したウィンドウ内で動作する。

   X Window System を用いずに Emacs の操作をすべてキーボードで行うことになります。

X が動作していなければ，いずれの起動法を使っても，Emacs はコマンドを入力したウィンドウ内で動作します。ただし，この資料は X が動作していることを前提として記述しています。

これから，Emacs の使い方を学びますが，できるだけマウスを使わないで操作する方法までを習得することが望ましいです。それは，マウス操作よりもキーボード操作の方が作業効率が良いことが多いし，マウスはいつも使えるとは限らないからです。

4.2.2 終了方法

マウスを使って Emacs を終了するには，メニューから

```
File -> Quit
```

を選択します。

キーボード操作による場合は

```
C-x C-c
```

です。

[12]GNU Emacs は UNIX の標準コマンドではありませんので，必ずしもすべての UNIX で利用できるとは限りません。

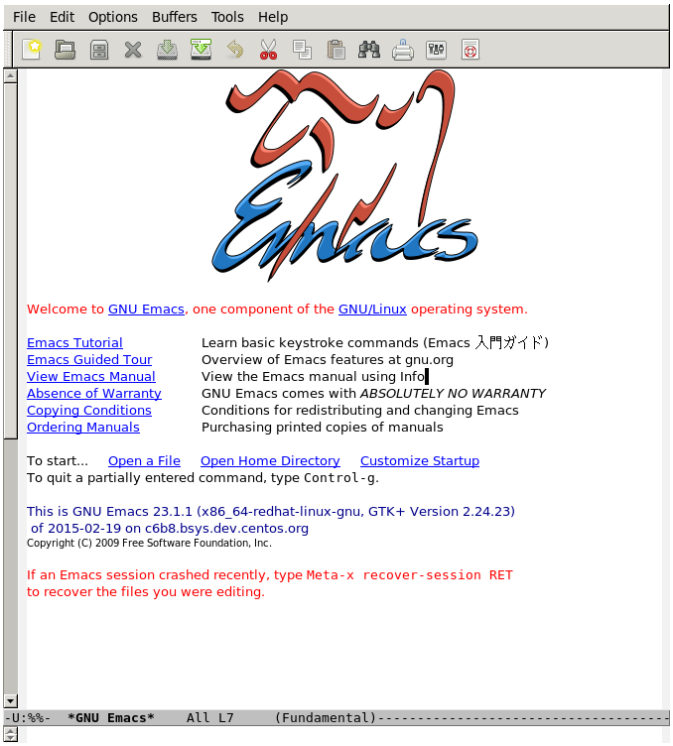


図 1: Emacs の起動画面例

4.2.3 練習

1. コマンド行に `emacs` と打って Emacs を起動しましょう。起動元のウィンドウに新しいプロンプトが現れないことと，Emacs の起動画面を観察したら終了しましょう。

2. コマンド行に `emacs -nw` と打って Emacs を起動しましょう。起動画面を観察したら，キーボード操作で終了しましょう。

4.3 画面構成

コマンド行に `emacs &` と打って，Emacs を起動してください。新しいウィンドウに図 1 のような起動画面が表示されましたね。& をつけて Emacs を起動したので，起動元のウィンドウに新しいプロンプトが現れていることも確認してください。

- 最上行の File Edit Options ... 等と表示されている部分をメニューバー (menu bar) と呼びます。ここをマウスでクリックして，メニューから Emacs の操作を選択することができます[13]。すぐ下の行には，マウス操作のためのボタンが並んでいます。

- 最下行の For information about ... と表示されている行をエコー領域 (echo area) と呼びます。ここに Emacs からの様々なメッセージが表示されます。

[13] すべての操作がメニューでできる訳ではありません。

この行に，Emacs で編集したいファイル名等を入力することもあります。この行は，キーボードからの入力を受け付ける状態になったとき，ミニバッファ (minibuffer) と呼ばれます。

- 残る部分を (Emacs における) ウィンドウと呼びます。図 1 では，様々な文字が表示されている一番広い場所がウィンドウです。ここで文書の編集等を行います。

  なお，X Window System における “ウィンドウ” (これまでウィンドウと呼んできたもの) のことを Emacs ではフレーム (frame) と呼びます。

- ウィンドウの最下行 (エコー領域を含めて数えると下から 2 行目) をモード行 (mode line) といいます。ここにはウィンドウに関する様々な情報が表示されます。

ここで少し Emacs を操作してみましょう。

1. キーボードから何か文字を打ってください。

   Emacs のウィンドウに文字を入力できましたか。

2. マウスを使い，メニューバーの File の項目から Split Window を選択してください。

   ウィンドウが分割 (split) されて二つになります。Emacs では複数のウィンドウを使ってファイルを編集できます。

3. 同じく File メニューの Remove Splits を選択してください。

   予想どおりになりましたか。

次に，ウィンドウ分割をキーボード操作で行ってみます。

1. まず，File メニューをクリックしてプルダウンメニューを表示し，メニュー項目をよく観察してみましょう。

   項目の右側に，括弧で囲まれた記号があります。この記号は，当該操作をキーボードで行うためのキーを表しています。

2. Split Window の項目の右には (C-x 2) とあるのを確認したら，この表示に従って，キーボードから `C-x 2` をタイプしてください。`C-x 2` は，コントロールキーを押しながら x を押し，続いて (空白は打たずに) 2 だけを押すことを意味します。

3. キーボード操作でウィンドウを一つにしましょう。File メニューをマウスで開き，キー操作による方法を確認したら，実行してください。

4.4 困ったときには

4.4.1 操作の中断

Emacs を使っていて良く分からない状況に陥ったときのために，次の Emacs のコマンド (キー操作) を是非覚えておいてください。

Emacs におけるコマンドの取り消し: `C-g`

1 回押しただけでうまくいかなければ 2 回押します。

たとえば，キー操作でウィンドウを二つに分割しようと思ったけど，途中で止めたくなったとします。ウィンドウ分割のためにまず C-x を押しますが[14]，続いて 2 を押す代りに C-g を押すと，その操作を取り消すことができます。

4.4.2 分割された Window を一つにする

Emacs では，操作の途中で自動的にウィンドウが二つになることがあります。その場合には，ウィンドウ内に表示された指示に従って対処するか，よく分からなければウィンドウを一つにする操作 (C-x 1 やメニューの Remove Split など) をしてください。

4.5 Emacs におけるファイル編集の基本操作

4.5.1 例題 1 — ファイルの新規作成

ファイルの中身が

```
ls - list directory contents
mv - move files
```

である commands_file という名前のファイルを Emacs を使って作成します。以下の操作を行ってください。

1. Emacs が起動していなければ emacs & で起動してください。
2. ファイルを新しく作成するには，C-x C-f を押します。このコマンドが第 4.10 章の Emacs コマンド一覧のどこに載っているかを確認しておきましょう。

   最下行のエコー領域に Find file: ~/ というメッセージが現れます。カーソルもエコー領域に移動しましたので，ここにキーボードから文字を入力できます。エコー領域は，文字入力できる状態になったとき，ミニバッファと呼ばれるのでしたね。。

3. 作成するファイルの名前 commands_file をミニバッファに入力して <ENTER> を押しましょう。入力を誤ったら <ENTER> を押す前に <BS> を使って修正するか，C-g を押して 2. からやり直してください。

   ウィンドウ下部のモード行には，これから作成するファイルの名前 commands_file が表示されます。カーソルは上部のウィンドウに戻ります。

4. 作成するファイル commands_file の中身

   ```
   ls - list directory contents
   mv - move files
   ```

   を Emacs のウィンドウに入力してください。

   改行するには <ENTER> を押します。文字の修正には <BS> が使えます。カーソルの移動には，ここでは矢印のキーを使うことにします。

[14] この状態で少し待って，エコー領域に入力したキー (C-x-) が表示されることを確認しておきましょう。

5. Emacs に入力したテキスト (文書) をファイルに保存するには `C-x C-s` を押します。

   エコー領域 (最下行) に Wrote ... というメッセージが表示され，保存に成功したことがわかります。これでファイル commands_file ができあがりました。

6. 一度 Emacs を終了し，ファイル commands_file の内容を cat コマンドで確認してください。

4.5.2 例題 2 — 既存ファイルの編集

第 4.5.1 節の例題 1 で作成したファイル commands_file の内容を

```
ls - list directory contents
mv - move files
cat - print files
```

に変更します。また，Emacs でのファイル編集作業と併行して UNIX のコマンドも使ってみます。

1. `emacs &` と打って Emacs を起動してください。
2. ファイル commands_file が存在することを， emacs コマンドを入力したウィンドウで ls コマンドを実行して確認してください。ファイル名が違っていたら，正しいものに変更してください。
3. 編集したいファイル commands_file を開きます。方法は第 4.5.1 節 例題 1 でファイルを新規作成したときと同じです。`C-x C-f` を押してください。最下行のミニバッファにはファイル名 commands_file を入力して `<ENTER>` を押してください。

   既存ファイル commands_file の内容が Emacs のウィンドウに表示され (Emacs に読み込まれ)，モード行 (下から 2 行目) には commands_file と表示されます。

4. ファイルに追加すべき内容 (`cat - print files`) を Emacs のウィンドウに入力しましょう。
5. 例題 1 と同じ方法で，Emacs のウィンドウに表示されている内容をファイルに保存してください。Emacs はまだ終了しないでください。
6. ファイル commands_file の内容を cat コマンドで確認しておきましょう。

4.6 ファイルとバッファ

図 2 は，コンピュータの構造を，これまでに行ったことを理解するために必要な部分に限って，ごく簡単に表現したものです。

CPU (central processing unit) はコンピュータの頭脳にあたる役割をし，コンピュータの動作を制御したり，様々な計算を行います。

主記憶装置 (main memory; 以下，単にメモリと呼ぶ) は，実行中のプログラムやデータを格納する場所です。CPU は，処理に必要なデータをメモリから読み取り，処理結果をメモリに書き込みます。

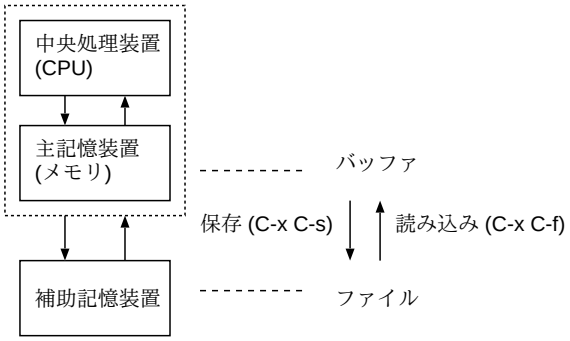


図 2: ファイルとバッファ

Emacs が動いているときには，Emacs のプログラムや編集中のテキスト (文書) はメモリにあります。Emacs がテキストを保持するために使うメモリ内の領域を，Emacs の用語で**バッファ** (buffer) といいます。

メモリに対するデータの読み書きは高速に行えますが，メモリの容量は比較的小さく，メモリ内のデータはコンピュータを停止すると消えてしまいます。そのため，メモリはデータの永続的な保存には使えません。

補助記憶装置 はメモリに比べて動作が遅い反面，記憶容量が大きく，中身はコンピュータを停止しても消えません。**ファイル**は補助記憶装置にあります。

さて，Emacs のウィンドウに表示されるテキストは，ファイルではなく，バッファの内容です。したがって，Emacs でファイルを作成・編集するには，まず，そのためのバッファを用意する必要があります。これを行うのが `C-x C-f` です。この操作ではファイル名を入力しますが，その名のファイルが存在しなければ，同名の空のバッファが用意されます (第 4.5.1 節 ファイルの新規作成)。ファイルが存在すれば，ファイルは同名のバッファに読み込まれます (第 4.5.2 節 既存ファイルの編集)。なお，バッファに読み込まれるのはファイルの内容のコピーですから，ファイルを読み込んでも元のファイルが無くなる訳ではありません。

Emacs を終了するとバッファは消えるので，保存したいバッファの内容はファイルに入れる必要があります。それをするのが `C-x C-s` です。

編集中のバッファが保存済かどうかは，モード行 (下から 2 行目) におけるバッファ名の左側の表示でわかります。ここが `--` から `**` に変わったら，バッファとファイルの内容に違いが生じたこと，すなわち，バッファの内容を変更したのに未保存であることを表しています。

では，上記のことを確認するために，以下の操作を行いましょう。

1. Emacs が起動していなければ，`emacs &` と打って起動してください。commands_file をバッファに読み込んでいなければ，読み込む操作をしてください。

2. Emacs のウィンドウに表示されている，commands_file という名のバッファを消してみます。メニューから

   ```
   File -> Close (current buffer)
   ```

   を選んでください。

3. ls コマンドを使って，消したバッファと同じ名前のファイルが存在することを確認してください。

   バッファを消してもファイルは消えませんね。

4. Emacs を終了せずに，引き続き commands_file を編集するにはどうしたらいいでしょう?

   ファイル commands_file を再度バッファに読み込んでください。

5. Emacs のモード行 (下から 2 行目) に表示されているバッファ名 commands_file のすぐ左側の表示が -- であることを確認してから，バッファ (Emacs のウィンドウ) 3 行目の cat の説明を

   ```
   cat - concatenate and print files
   ```

   に変更してください。

   モード行の表示が ** に変わりましたね。

6. UNIX のコマンドでファイル commands_file の内容を確認してください。

   バッファの内容を変更しただけでは，ファイルは変わりません。

7. バッファの内容をファイルに保存してください。

   モード行の表示が -- になりましたね。バッファとファイルの内容が同じになりました。

8. ファイル commands_file の中身を cat コマンドで確認してください。

4.7 まとめと練習

4.7.1 まとめ—Emacs によるファイル編集の流れ

Emacs を用いてファイルを編集するときの流れは次のとおりです。

1. Emacs を起動する。— `emacs &` (X ウィンドウシステムが使えないときには & を付けない)
2. 新規ファイル用のバッファを用意する。または，編集したいファイルをバッファに読み込む。— `C-x C-f`
3. バッファへの文字の追加・削除・変更などの編集作業を行う。
4. バッファの内容をファイルに保存する。— `C-x C-s`
5. 必要に応じて 2 から 4 の作業を繰り返す。複数の文書を編集するときでも Emacs を終了して再度起動する必要はない。
6. すべての作業が終わったら Emacs を終了する。 — `C-x C-c`

4.7.2 練習

1. 次の内容を持つファイル emacs_motion を Emacs で作成してください。

```
           backward forward
character  C-b      C-f
line       C-p      C-n
```

   ファイル作成後も Emacs を終了しないでください。

2. 起動中の Emacs を使って，既存のファイル commands_file に，これまでと同じ形式で，cp と rm の説明

```
cp - copy files and directories
rm - remove files or directories
```

   を追加してください。作業を終えたら Emacs を終了してください。

4.8 その他の操作

4.8.1 ファイル名を指定してファイルに保存

編集中のバッファの内容を，ファイル名を指定して保存するには C-x C-w を使います。既存のファイルをバッファに読み込んでからこのコマンドを使うと，元のファイルを異なるファイル名で保存することができますので，UNIX の cp コマンドでファイルを複写するのと同様のこともできます。

以下では，ファイル commands_file と同じ内容を持つファイル commands_copy を作ります。

1. ファイル commands_file をバッファに読み込んでください。
2. C-x C-w を押しましょう。

   エコー領域 (ミニバッファ) にメッセージ Write file: ~/ が現われます。
3. commands_copy と打って，<ENTER>を押しましょう。

この操作で新しいファイル commands_copy が作成されましたが，モード行に表示されているバッファ名も commands_copy に変ったことに注意してください。C-x C-w を実行すると，編集中のバッファの内容が別のファイルに書き込まれるだけではなく，バッファ自体の名前も変わります。引き続きこのバッファで編集作業をすれば，それは元の commands_file ではなく commands_copy に対してなされます。

4.8.2 Emacs 終了時のファイル保存

バッファの内容を変更したにもかかわらず，それをファイルに保存しないまま Emacs を終了しようとすると，バッファの内容をファイルに保存するかどうかを尋ねるメッセージが，ウィンドウ

下部のエコー領域に表示されます[15]。その場合には，保存の必要性を判断して，y, n, yes, no 等で答えてください。

例えば，第 4.9 節で紹介するチュートリアルの文書に，何か書き込みをしてから Emacs を終了しようとすると，このメッセージが現れます。普通はチュートリアルを保存する必要がないので，Save file ... ? に対しては n を，... exit anyway? に対しては yes を打ってください。

4.9 Emacs Tutorial

Emacs 起動時のメッセージに

```
Important Help menu items:
Emacs Tutorial          Learn-by-doing tutorial for using Emacs efficiently.
```

と記されているとおり，Emacs をキーボード操作で効率よく使う方法を学ぶチュートリアルがあります。Help メニューには “Emacs Tutorial (C-h t)” という項目がありますので，マウスでこれを選択するか，CTRL-h に続いて t を押せばチュートリアルを始めることができます[16]。

[15] Emacs には，ファイル編集以外の用途に用いられるバッファ (例えば *scratch*) が存在します。 それらの内容を変更しても，終了時に保存のメッセージは現れません。

[16] Emacs の標準設定では `C-h` にヘルプ表示の機能が割り当てられているのですが，`C-h` が他の機能に変更されているシステムがあるかもしれません。その場合，`M-x help <ENTER>` を打てばヘルプを表示できます。ヘルプを介さずにチュートリアルを始めるには `M-x help-with-tutorial <ENTER>` とします。

4.10 主な Emacs コマンドの一覧

C-x C-c	終了
C-g	コマンドの取り消し
C-x C-f	ファイルを開く (バッファへのファイル読込み)
C-x C-s	現在のバッファをファイルに保存
C-x C-w	現在のバッファに名前をつけて保存
C-x s	編集中のバッファをすべてファイルに保存
C-h	ヘルプ (M-x help <ENTER>)
C-h t	チュートリアル (M-x help <ENTER> t)
C-x u	変更の取り消し (undo)
C-_	変更の取り消し (undo)
C-f	カーソルを 1 文字右 (forward) に移動
C-b	カーソルを 1 文字左 (backward) に移動
C-a	カーソルを行頭に移動
C-e	カーソルを行末に移動
C-p	カーソルを前 (previous) の行に移動
C-n	カーソルを次 (next) の行に移動
C-v	次の画面を見る
M-v	前の画面を見る
<BS>	カーソル左の 1 文字を削除
C-d	カーソル位置の 1 文字を削除
C-k	行末まで消去 (kill)
C-y	最後に保存した kill-ring の内容取りだし (yank)
C-<SPACE>	マーク (領域の始点) の設定
C-w	領域消去 (kill-ring に保存)
M-w	領域を kill-ring に保存
C-x b	バッファの切り替え
C-x C-b	バッファの一覧表示
C-x k	バッファの削除
C-x o	他のウィンドウに移動
C-x 0	現在のウィンドウを削除
C-x 1	現在のウィンドウのみ残す
C-x 2	現在のウィンドウを上下に分割
C-x 3	現在のウィンドウを左右に分割
C-s	検索
M-%	置換