ウィンドウシステムとエディタ

目 次

1　ウィンドウシステムの利用

1.1　ウィンドウシステム

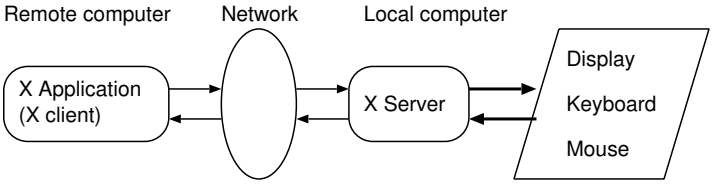
ウィンドウやマウスを使うための仕組みを提供する一連のプログラムをウィンドウシステムと呼びます。UNIX で標準的に用いられるウィンドウシステムは X Window System です。これを単に X と呼んでもかまいません。

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でした。一方，Windows における一般的なアプリケーションソフトウェアと同様に，UNIX には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[1]，X アプリケーションと呼ぶことがあります。

1.2　X の起動

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に絵を描いたり，マウス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[2]。

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，

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

必要があり，そのための準備作業が必要です[3]。2 のための作業をディスプレイの指定といいます。

X アプリケーション利用のための準備手順は「UNIX 実習用コンピュータの利用手順 (X 利用)」[4]を参照してください。このページには遠隔ログインの手順も併せて記載されていますので，遠隔ログインが済んでいる場合には，X アプリケーション利用のための操作のみを行ってください。

---

[1] 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。

[2] X は X サーバと 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 クライアントということもあります。

[3] これらは自動的に行われることもあります。

[4] `http://echoes.hak.hokkyodai.ac.jp/db/?id=15217`

1.3 X アプリケーションと &

1.3.1 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。電卓プログラム (xcalc) 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. `xcalc` と打って，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 xcalc コマンドを入力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 xcalc 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. 電卓ウィンドウに対して `q` を打ち，xcalc 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 `xcalc` 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。

   xcalc を実行すると新しい電卓のウィンドウが現れますが，xcalc 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，xcalc は強制終了します。

1.3.2 & 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. `xcalc &` で xcalc を起動して，起動元の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[5]。この状態では，xcalc の起動元から xcalc を制御 (終了) す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. xcalc は電卓のウィンドウに q を入力して終了できるので，この方法で xcalc を終了しましょう。

[5] 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。

1.3.3　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[6]。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。

2　GNU Emacs を使おう

2.1　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。この授業で作成した通常のファイルはテキストファイルです。これに対し，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[7]。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 では，UNIX 自体の諸設定や各種アプリケーションの動作環境を，原則としてテキストファイルに記述します。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 UNIX の利用において大切です。

[6] 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。

[7] 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 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。

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。これを単にエディタと呼ぶこともあります。ここでは代表的なエディタの一つである GNU Emacs の基本的な使い方を学びます[8]。

2.2 キー表記

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。この資料でもこの表記を用います。

`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。

2.3 起動と終了

2.3.1 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。

   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。

2.3.2 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 -> Exit Emacs
```

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。

[8] 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。特に UNIX の管理を行う必要がある人は，UNIX での標準的なエディタである vi の使い方を習得することが望まれます。

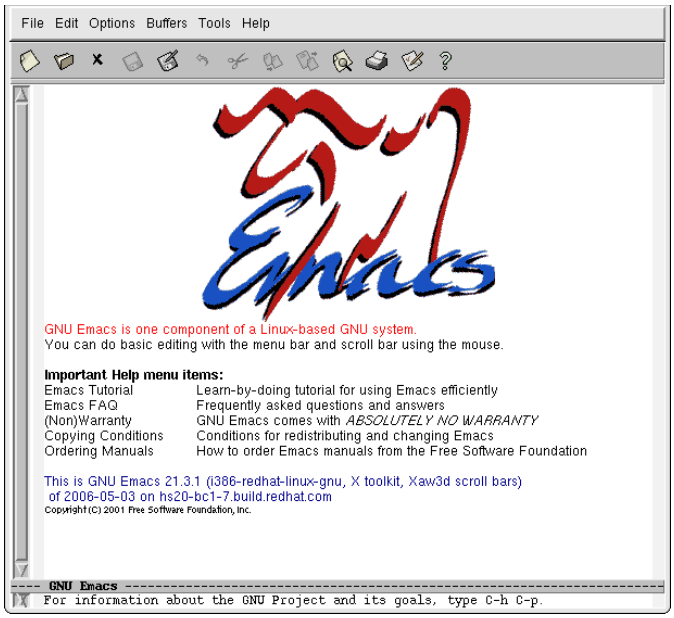


図 1: Emacs の起動画面例

2.3.3 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。

2.4 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[9]。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。

  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 において，様々な文字が表示されている一番広い場所で文書の編集等を行います。

[9] すべての操作がメニューでできる訳ではありません。

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 Unsplit Windows 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。

2.5 困ったときには

2.5.1 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[10]，続いて `2` を押す代りに `C-g` を押すと，操作を取り消すことができます。

[10] 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。

2.5.2 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 Unsplit Window など) をしてください。

3 ファイル編集の基本操作

3.1 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 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，Emacs のメニューバーからマウスで

   File -> Open File...

   を選択します。後でキーボードを使って同じことができるように，キー操作 (File メニューの Open File 項目の括弧内の文字) が，第 4 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ます。
3. 作成するファイルの名前 commands 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 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 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矢印のキーを使うことにします。
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，

`File -> Save (current buffer)`

を選択します。キーボード操作の方法も確認しておきましょう。

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 ができあがりました。

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

3.2 例題 2 — 既存ファイルの編集

第 3.1 節の例題 1 で作成したファイル commands 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 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 を開きます。方法は第 3.1 節 例題 1 でファイルを新規作成したときと同じです。メニューでは `File -> Open File...` ですが，是非，キーボード操作で行ってみましょう。最下行のミニバッファにはファイル名 commands を入力して`<ENTER>` を押してください。

   既存ファイル commands 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 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 は終了しないでください。
6. ファイル commands の内容を cat コマンドで確認しておきましょう。

3.3 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。

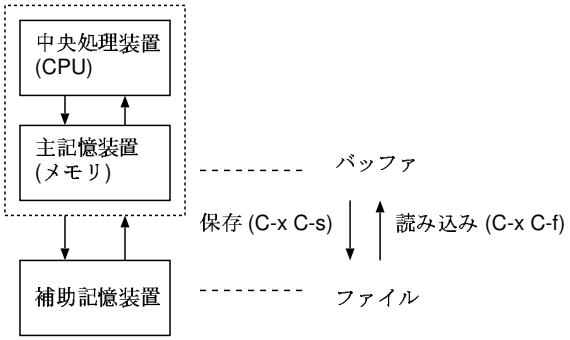


図 2: ファイルとバッファ

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。

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 `File -> Open File...` (`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 3.1 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 3.2 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 `File -> Save (current buffer)` (`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 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 という名のバッファを消してみます。

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

を実行してください。

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 を編集するにはどうしたらいいでしょう?

   ファイル commands 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 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 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 の中身を確認してください。

3.4 まとめ — 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。
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。
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。
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。

3.5 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 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。

3. ホームディレクトリ以外に存在するファイルを Emacs に読み込む操作をしてみましょう。

   (a) ホームディレクトリに unix というディレクトリがあることを，ls コマンドで確認してください。無ければ mkdir コマンドで作成してください。

   (b) mv コマンドを使って emacs_motion を unix に移動してください。

   (c) ホームディレクトリで Emacs を起動し，`C-x C-f` してください。

   (d) ミニバッファを `Find File: ~/unix/emacs_motion` にしてから`<ENTER>` を押してください。

   > ミニバッファでの ~ はホームディレクトリを表していて，~/*file* の代りに~/*subdir*/*file* とすれば，~ の子ディレクトリ (*subdir*) のファイル (*file*) を指定したことになります。これはファイルの読み込み時に限らず，保存時などでも共通です。

   (e) emacs_motion が読み込まれたことを確認したら，Emacs を終了してください。

   (f) ディレクトリ unix の emacs_motion をホームディレクトリに移動してください。

3.6 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 `File -> Save Buffer As...` (`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 と同じ内容を持つファイル commands_file を作ります。

1. ファイル commands 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_file 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_file 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_file 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 ではなく commands_file に対してなされます。

3.7 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[11]。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 3.8 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。

3.8 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[12]。

[11] 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。

[12] 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 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

5 もっと Emacs を使おう

5.1 *scratch* バッファ

コマンド行に `emacs &` と打って，Emacs を起動してください。Emacs 起動後にウィンドウに表示されているバッファの名前は，ウィンドウ下部のモード行に表示されているとおり，*scratch* です[13]。このバッファに何か書き込みを行うか，しばらく時間が経つと

```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
;; then enter the text in that file's own buffer.
```

というメッセージが現われます。メッセージはよく読みましょう ... といっても見慣れない英単語が含まれているので，意味を理解するのが大変かもしれませんね。

> *scratch* バッファは，保存する必要のない一時的なメモを書いたり，Lisp コード[14] を評価するために使います。**新しいファイルを作成したいなら，`C-x C-f` を押して新しいファイル用のバッファを用意し，そこにテキストを書き込んでください。**

しかしながら，これ以降しばらくは *scratch* バッファを使って作業します。保存する必要のない練習用バッファとして使うからだと思ってください[15]。

5.2 変更の取り消し (undo)

undo は誤って行ってしまった操作を取り消す操作です。

5.2.1 一文字削除と取り消し

1. Emacs のウィンドウに `12345` とタイプし，続いて `<BS>` を 5 回押してください。

   タイプした文字が 5 4 3 2 1 の順に削除 (delete) され，全て無くなりましたね。

2. `C-x u` を 1 回押しましょう。

   最後に行った `<BS>` の操作が取り消されて，1 が復活しました。

3. `C-_` を 1 回押しましょう。

   さらに過去に遡って削除の操作が取り消され，2 も復活しました。`C-x u` と `C-_` [16]は，共にテキストに加えた変更を取り消す (undo) ためのコマンドで，どちらを使ってもかまいません[17]。

4. undo 操作で，345 も復活させてください。undo を続けて行うには `C-_` の方が操作しやすいでしょう。

[13] Emacs を起動するときに，編集するファイル名をコマンド行の引数に与えて，`emacs` *file* や `emacs -nw` *file* とすることも可能です。その場合，起動後の Emacs のウィンドウには *file* という名前のバッファが現れます。

[14] Emacs は “Emacs Lisp” というプログラミング言語を内蔵しています。Emacs 上で Lisp を実行したり，Lisp を使って Emacs の動作を変更 (customize) することができます。

[15] Emacs では，バッファの名前に応じて，自動的に適切な編集モードに切り替わります。Emacs の動作はモードによって異なりますので，不適当なモードで Emacs を使うことは好ましくありません。「普通の」バッファ名で Emacs を利用しているときには,「普通の」文章を編集するための Text モードになりますが，*scratch* バッファだと Lisp Interaction モードで動作します。

[16] メニューでは `Edit -> Undo`

[17] `C-/`でも undo できます。

5.2.2 一行消去と取り消し

1. カーソルをバッファの 1 行 3 列目 (;; This buffer ... の T の位置) まで移動し，この行の終りまでを一度の操作で消去してみましょう。

   カーソル位置から行末までを消去するには C-k (kill の頭文字) を使います。

2. カーソルを 2 行下 (;; then enter ... の t の位置) に移動して，3 行目を消去しましょう。

3. 消去した二つの行を undo 操作で復活させましょう。

5.3 消去 (kill) と貼り付け (再入; yank)

ここで紹介するのは，テキストを切り取って貼り付ける操作です。

1. C-k を使って，*scratch* バッファの 1 行目 (;; This buffer ...) すべてを消去しましょう。続いて 3 行目 (;; then enter ...) もすべて消去しましょう。

2. 今回は，カーソルを移動せずに C-y を 1 回押しましょう。

   3 行目が復活しましたね。

3. 再度 C-y を押しましょう。

   undo のときと違って 1 行目は空のままですが，3 行目の内容がカーソル位置に追加されました。

4. さらに何回か C-y を押しましょう。

C-k によって消去された行は，バッファから消えるとともに kill-ring という場所に保存されます[18]。C-y[19] は，最後に kill-ring に保存した内容を，カーソル位置に貼り付ける (再入する; yank[20]) ための操作です。C-y を押しても kill-ring に保存されたものは無くならないので，再度 C-y を押すと，再び同じ内容が貼り付けられます。なお，kill-ring の中身はメニュー Edit -> Select and Paste で確認することができます。また，<BS> 等で 1 文字を削除しても kill-ring には保存されません。

5.4 領域の利用

任意の範囲のテキストを切ったり貼ったりするには，領域 (region) を使います。

5.4.1 領域指定とテキストの移動

1. undo 操作を繰り返し行って，これまでに変更した *scratch* バッファの内容を，すべて元の状態に戻してみてください。操作の途中で混乱してしまって元の状態に戻せなくなったら，一度 Emacs を終了して，emacs & で再度起動してから，どれかキーを一つ押してください。

[18] C-k を連続して押したときの kill-ring の内容については，チュートリアルの「＊挿入と削除」を参照してください。
[19] Edit -> Paste
[20] エディタ vi でも yank という用語を使いますが，Emacs での yank とは用語の使い方が異なります。vi を使う人は注意が必要です。

2. 1 行目の “This” の直後 (“buffer” の直前) の空白位置にカーソルを移動してから C-<SPACE> を押し，ウィンドウ下部のエコー領域に Mark set と表示されるのを確認してください。この操作を「マークを設定する」といいます。

3. カーソルを，1 行目の “buffer” の直後 (“is” の直前) の空白位置に移動してください。

4. C-w を押してください。 “ buffer” が消去されます。

5. カーソルを下の方に動かして，何も書いてない行の先頭に移動してください。続いて C-y を押してください。消去した “ buffer” がカーソル位置に取り出されます。

2. と 3. で行った操作を，領域 (リージョン) の指定といいます。領域とは,「最後にマークを設定した文字」から,「現在カーソルがある位置の直前の文字」までの範囲のことです。先の操作では, “ buffer” を領域指定しました。buffer の前の空白文字も領域に含まれていることに注意してください[21]。

> 「ポイント」という概念を使うと，領域の定義はもっとすっきりします。
>
> ポイントは，カーソルがのっている文字と，その直前の文字との間に存在する**仮想的な**点であって，実際に表示されるものではありません。四角いカーソルの左下の角がポイントだと思ってもいいでしょう。カーソルを移動すれば，それにつれてポイントも移動します。
>
> マークは C-<SPACE> を押したときのポイントに対して設定されます。マークを設定したポイントと，現在のポイントとの間が領域です。

C-w[22] は，指定した領域内の文字を消去するコマンドです。C-k と同様に，C-w で消去したものは kill-ring に保存されますので，C-y を使って取り出せます。

上の操作では，“ buffer” を領域指定しましたので，これが C-w で消去され，C-y で取り出されました。この操作を用いると，任意のテキストを任意の場所に移動することができます。

5.4.2 テキストの複写

M-w[23] は，領域内のテキストを kill-ring に保存するためのコマンドです。C-w とは違い，領域内のテキストは消去されません。C-y と併用すれば，領域内のテキストを他の場所に複写することができます。

1. 2 行目の行頭 (;; If you want ... の ; の位置) にカーソルを移動して，マークを設定してください。

2. 行末の改行文字[24] を含む 2 行目全体を領域指定してください。マーク位置から現在のカーソル位置の直前までが領域ですから，カーソルを次の行 (3 行目) の行頭に移動すればいいです。

3. M-w を押してください。カーソルがマーク位置に移動して，また戻ってきます。見た目には，それ以外の変化はありません。

4. カーソルを下の方にある空白行まで移動して，C-y を押しましょう。領域指定した内容 (2 行目全体) が複写されました。

---

[21] C-<SPACE> で明示的にマークを設定しなくても，他のコマンドを実行したときに，自動的にマークが設定されることがあります。そのために思いがけない動作が起きたら，落ち着いて undo しましょう。

[22] Edit -> Cut

[23] Edit -> Copy

[24] 表示はされませんが，各行の最後には，改行を表す文字が入っています。C-e で行末の改行文字に移動して，C-d (カーソル位置の 1 文字を削除) でそれを削除すれば，どういうことか確認できるでしょう。

5.5 ミニバッファでの編集機能

ウィンドウ下部のエコー領域がユーザーからのキー入力を受け付ける状態になったとき，そこはミニバッファと呼ばれます。例えば，ファイルを読込むとき，C-x C-f を押してからファイル名を入力しますが，この入力はミニバッファに対して行っているのです。

ミニバッファでは，同一行内のカーソル移動コマンド (C-a, C-b, C-f, C-e等) や文字の削除 (<BS>)，行の消去コマンド (C-k) を使って，入力している文字を編集することができます。加えて，次に説明する入力補完機能を利用することができます。

5.5.1 入力補完機能

C-x C-f によるファイルの読み込みを例にとって入力補完の説明をします。入力補完はミニバッファへの入力において一般的に利用できる機能ですので，C-x C-w でのファイル保存等でも使えます。以下の操作を行ってください。

1. ホームディレクトリに存在するファイルの名前を UNIX のコマンドを使って調べて，

   - ファイル名が emacs で始まる既存のファイルが emacs_motion のみであること
   - ファイル名が com で始まる既存のファイルが commands_file と commands の二つのみであること

   を確認してください。これらを満たしていなければ，ファイル名を変更するか，不要なファイルを削除して，この条件を満たすようにしてください。なお，各ファイルの内容は何であっても構いません。

2. C-x C-f を押し，ミニバッファが

   ```
   Find file: ~/
   ```

   となることを確認してください。

3. これからファイル emacs_motion を読み込みますが，ミニバッファには emacs とだけタイプしてください。<ENTER> は**押さない**でください。

4. <TAB> を押してください。ファイル名をすべてタイプしなくても，ミニバッファの表示が

   ```
   Find file: ~/emacs_motion
   ```

   となるはずです。これが入力補完です。

   ホームディレクトリ (~) に存在するファイルのうち，ファイル名が emacs で始まるものは emacs_motion しかありませんから，emacs を入力した時点で，残りの文字列が何であるかは一意に決まります。<TAB> を押すと，その残りの文字列が補完されるのです。

   emacs_m などとタイプしてから <TAB> を押しても，結果は同じです。

5. 続いて <ENTER> を押すと，emacs_motion がバッファに読み込まれます。

6. 次に，commands を読み込むために，C-x C-f を押してから com とタイプして <TAB> を押しましょう。ミニバッファは

```
Find file: ~/commands
```

となります。com で始まるファイルは，commands_file と commands の二つありますから，com だけだとどちらを読み込むのか決められません。<TAB> は，できる限りの範囲で文字列を補完します。

7. さらに _ をタイプして，ミニバッファを

   ```
   Find file: ~/commands_
   ```

   にしてください。commands_ で始まるファイルは commands_file だけですから，この時点で <TAB> を押せば，残りは完全に補完されるはずです。実際にやってみましょう。

8. ファイル名が補完されたら，今回は <ENTER> の代りに C-g を押してください。

   C-g はコマンドの中断です。覚えていましたか？

9. 次に C-x C-f に続き，com? と打ってください。

   ? は補完できる名前の候補を一覧表示します。どんなファイルがあるのか忘れてしまった場合には，この一覧を見ながら，入力することができます[25]。

10. 読み込みの操作を中断してください。

5.5.2 まとめ：ミニバッファでの入力補完コマンド

<TAB>	補完
?	補完候補の一覧表示

5.6 練習

ファイル commands を入力補完機能を利用して Emacs に読み込んでください。ファイルの内容は

```
ls - list directory contents
mv - (略)
cat - (略)
cp - (略)
rm - (略)
```

になっているはずです。このファイルを編集して，コマンドの順番がアルファベット順になるように，行を並べ替えてください[26]。編集が終わったら，同名のファイルに編集結果を保存してください。

[25] 一覧のウィンドウにカーソルを移動して，望みのファイル名のところで <ENTER> を押し，そのファイルを読み込むこともできます。ウィンドウ間のカーソル移動の方法は後述します。

[26] sort コマンドを使うとすぐにできるんですが，今は Emacs の使い方を練習しているので，Emacs でやってください。行の消去または領域の消去と貼り付けを使います。

5.7 バッファとウィンドウ

5.7.1 複数バッファの利用

ここでは，複数のファイルを一度に編集する方法を扱います。まず，ファイル commands の中にある cat の説明の行を，新しいファイル commands_filter の中に複写してみます。以下の操作を行ってください。

1. Emacs が起動中であれば，一度終了してから，再度起動してください。
2. Emacs のバッファにファイル commands を読み込んでください。
3. commands の中の cat の行を，kill-ring に入れてください。

   そのためには，まず，ウィンドウに表示されている cat の行を領域指定する必要があります。領域指定したい行の行頭でマークを設定してから，カーソルを次の行の行頭まで移動すればいいですね。

   さらに，適当な Emacs のコマンドを使って，指定した領域を kill-ring に入れます。消去はしないでください。操作を誤ったら，焦らずに undo しましょう。
4. 新規のファイル commands_filter 用のバッファを用意してください。

   2. で行ったのと同じ Emacs コマンドを使います。ミニバッファに入力する名前は，もちろん，commands_filter です。
5. commands_filter 用のバッファが用意できたら，cat の説明が入っている kill-ring の内容を取り出して (貼り付けて) ください。
6. バッファ commands_filter をファイルに保存してください。

以上でファイル commands_filter が出来上がりました。ところで，commands というバッファはウィンドウから消えてしまいましたが，このバッファは無くなったのでしょうか?

答えは「いいえ」です。バッファ commands は，今使っている Emacs から無くなった訳ではなくて，新しく用意したバッファの陰に隠れてしまっただけです。次に，バッファ commands の内容を再度ウィンドウに表示してみます。

1. `C-x b` を押してください[27]。これは編集中のバッファ (current buffer) を切り替えるコマンドです。
2. エコー領域に

   ```
   Switch to buffer: (default commands_file)
   ```

   と表示されるはずです。ここで (default commands) とあるのは,「特に指定がなければ commands とします」ということなので，そのまま `<ENTER>` を押します。もしも default の後に commands 以外の文字列が現われたら，それを commands に修正してから `<ENTER>` を押します[28]。

   「デフォルト (default)」という言葉は Emacs 以外の場面でもよく使いますので，この機会に覚えておきましょう。

[27] この場面では，`C-x C-f` でも構わないのですが，バッファというものを意識して，バッファ切り替えのコマンドを使ってみましょう。

[28] この操作をメニューで行うには，メニューバーの Buffers をクリックして，commands という項目を選択します。

commands のバッファがウィンドウに現れましたから，このバッファを引き続いて編集することも可能です。

以上の操作では，二つのバッファを同時に使って編集作業を行いましたが，三つ以上のバッファを同時に使うこともできます。

5.7.2 ウィンドウの分割

1. まず，バッファ切り替えのコマンド (C-x b) を使って，*scratch* バッファをウィンドウに表示してください。

   Emacs を使っている間は，バッファ削除の操作をしなければ，*scratch* バッファは常に存在しています。ミニバッファに入力するバッファ名は *scratch* です。<TAB> による入力補完も利用できます。

2. C-x 2 を押しましょう。ウィンドウが上下二つに分割されました。でも表示されている内容は同じですね。

3. a という文字を入力してください。カーソルは上のウィンドウにありますから，文字はそこに入力されたのですが，下のウィンドウにも同じ文字が現われました。*scratch* という名の一つのバッファを，別々のウィンドウ (窓) から覗いて，編集している状態にあるからです。

4. C-x o を押してください。この o は other の頭文字です。別のウィンドウにカーソルが移動しました。もう一度，C-x o を押しましょう。また，別のウィンドウにカーソルが移動して，上に戻りました。

5. さらに 10 個ほど a を入力して[29]，カーソルの位置を確認してから C-x o を押してください。

   別のウィンドウにカーソルが移動しましたが，カーソルの位置が上と下では違いますね[30]。

   一つのバッファを複数のウィンドウで編集しているときには，あるウィンドウで行ったバッファの内容変更 (文字入力) は，他のウィンドウにも反映されます。一方，カーソル位置や表示位置はウィンドウ毎に独立ですので，一つのバッファの異なる場所を，別々のウィンドウに表示して，それぞれで編集することができます。長い文書を扱うときに便利です。

6. C-x 1 を押しましょう。これは，編集中 (カーソルの存在する) ウィンドウをただ一つだけ残して，他のウィンドウを閉じるコマンドでした。

7. C-x 3 を押してください。結果を確認したら，ウィンドウを一つに戻してください。

5.7.3 複数ウィンドウでの複数バッファの操作

第 5.7.2 節では，異なるウィンドウから一つのバッファの内容を眺めてみましたが，もちろん，異なるウィンドウで異なるファイル (バッファ) を編集することもできます。

1. Emacs のウィンドウを上下二つに分割してください。

[29] 10 回 a のキーを押してもいいですが，C-u 1 0 a でもできます。ほとんどの場合，C-u を使えば，その後に指定した数の分だけ，続くコマンド (今の場合は a を入力するという操作) が繰り返されます。ただし，C-u の動作は後続のコマンドに依存しますので，C-u がコマンドの繰り返しを意味しないこともあります。

[30] 上のウィンドウでのカーソル位置をもう一度確認したければ，C-x o を押して上に移って，すぐ下に戻りましょう。

2. 上のウィンドウで，ファイル commands を読み込んでください。

3. カーソルを下のウィンドウに移動させて，ファイル commands を読み込んでください。

ここでは，実際に作業を行うことは略しますが，カーソルを上下のウィンドウに移動することによって，各ウィンドウ内のバッファを編集することが可能です。

これまで使ってきたファイルへの保存の操作 (C-x C-s) では，保存されるバッファは，カーソルが存在するウィンドウ内のもののみです。複数のウィンドウに表示されている異なるバッファをファイルに保存するには，それぞれのウィンドウで C-x C-s を押す必要があります。

一回のコマンド操作で複数のバッファをファイルに保存したければ C-x s を使います。このコマンドを実行すると，元のファイルから内容変更のあったバッファ各々について，保存するか否かを尋ねられますので，保存する場合には y で答えます。

では，C-x s を試してみましょう。今の場合，変更を施したバッファはないはずですから，保存の対象となるバッファはありません。エコー領域に，その旨表示されます。

5.7.4 バッファの一覧と削除

今 Emacs が持っているバッファの一覧を表示するコマンドは C-x C-b[31] です。

1. C-x C-b を実行してください。

   ウィンドウを分割した状態で実行した場合には，カーソルの存在しないウィンドウに *Buffer List* という名前のバッファ一覧が表示されます。一つのウィンドウのみで作業していた場合には，ウィンドウが分割されて，一覧が現われます。

2. *Buffer List* ウィンドウ[32] の *scratch* という語のある行にカーソルを移動して，<ENTER> を押してみましょう。バッファ切り替えのコマンド (C-x b) を使わなくても，*Buffer List* が *scratch* に切り替わります。

次は不要になったバッファを削除してみましょう。そのためのコマンドは C-x k[33]です。これを使って commands バッファを削除してみます。

1. C-x k を押してください。エコー領域に

   ```
   Kill buffer: (default *scratch*)
   ```

   と表示されます。カーソルの存在するウィンドウに commands を表示した状態でコマンドを実行すると，デフォルトのバッファ名が commands になりますが，今は *scratch* になっているはずなので，commands とタイプしてから <ENTER> を押します。<TAB> による入力補完も可能です。

2. バッファ一覧を表示して，バッファが削除されたか確認しましょう。ただし，ここでバッファ一覧を表示するには，C-x C-b を使ってください。バッファ切り替えコマンド (C-x b) でバッファ一覧 (*Buffer List*) を表示しても，一覧に変更は反映されていません[34]。

なお，第 3.3 節で確認したように，バッファを削除しても，ファイルが削除されるわけではありません。ファイルを編集する必要が生じたら，C-x C-f で再度バッファに読込めばよいのです。

[31] Buffers -> List All Buffers
[32] このバッファ内で使えるコマンドは，ファイル編集用のバッファとは異なりますが，カーソル移動は普通にできます。
[33] File -> Close (current buffer)
[34] バッファ一覧を最新にするには，*Buffer List* バッファで g を押します。

5.7.5 複数フレームの利用

オプション –nw をつけずに新しいウィンドウで Emacs を開いたときには，こんなこともできます。ここではメニューを使った操作を紹介します。試してください。

1. `File -> New Frame`
2. `File -> Delete Frame`

注意 特に理由がない限り，コマンド行から emacs & を何度も打って，複数の Emacs を同時に動かすことは避けましょう。計算機の負荷が大きくなってしまいます。Emacs で複数のウィンドウ (Emacs の用語ではフレーム) を利用したければ，ここで紹介した方法を使いましょう。

5.8 文字列の検索と置換

まずは検索から。

1. Emacs のチュートリアルを開きましょう。やり方を忘れていたら第 3.8 節を見てください。
2. `C-s` を打って，エコー領域の表示を確認してください。I-search: と表示されましたね。
3. ウィンドウ内のカーソルの動きに注目しながら，C-s という文字列をそのままタイプしてください。CTRL を押しながら s ではありません。
4. ウィンドウの上部に「＊検索」という単語が現われるまで，`C-s` コマンドを何回か実行しましょう。
5. 見つかったら，`<ENTER>` を押してから，そこの説明を読みましょう。

次に置換です。

1. `M-%` を押してください。メタキー (`<ALT>`) と `<SHIFT>` を使います。
2. エコー領域の Query replace: に対して，emacs とタイプし，`<ENTER>` を押してください。
3. エコー領域の Query replace emacs with: に対して，GNU Emacs とタイプし，`<ENTER>` を押してください。
4. エコー領域に Query replacing emacs with GNU Emacs: (? for help) と表示されます。これは,「emacs を GNU Emacs で置き換えますか?」という意味です。y を押しましょう。
5. さらに y を押し，次に n を押してみましょう。
6. `<ENTER>` を押しましょう。エコー領域に置換した回数が表示されます。

5.9 日本語の入力

Emacs で日本語入力を実現する仕組みは幾つかあります。日本語入力の操作の細部は，使っている日本語入力の仕組みによって異なりますので，ここでは，英数字入力と日本語入力を切り替えるキー操作を紹介するにとどめます。

日本語・英数字入力の切り替え: `C-\`

漢字変換や候補の確定は `<SPACE>` や `<ENTER>` で普通にできますので，試してみてください。