

linux 使おうぜミ (第 3 回)

---

# Emacs 入門

---

Linux 使おうぜ委員会

メモ帳の女神に捧ぐ

# 1 “世の中には二種類のエディタしかない。 Emacs か、それ以下かだ。”

---

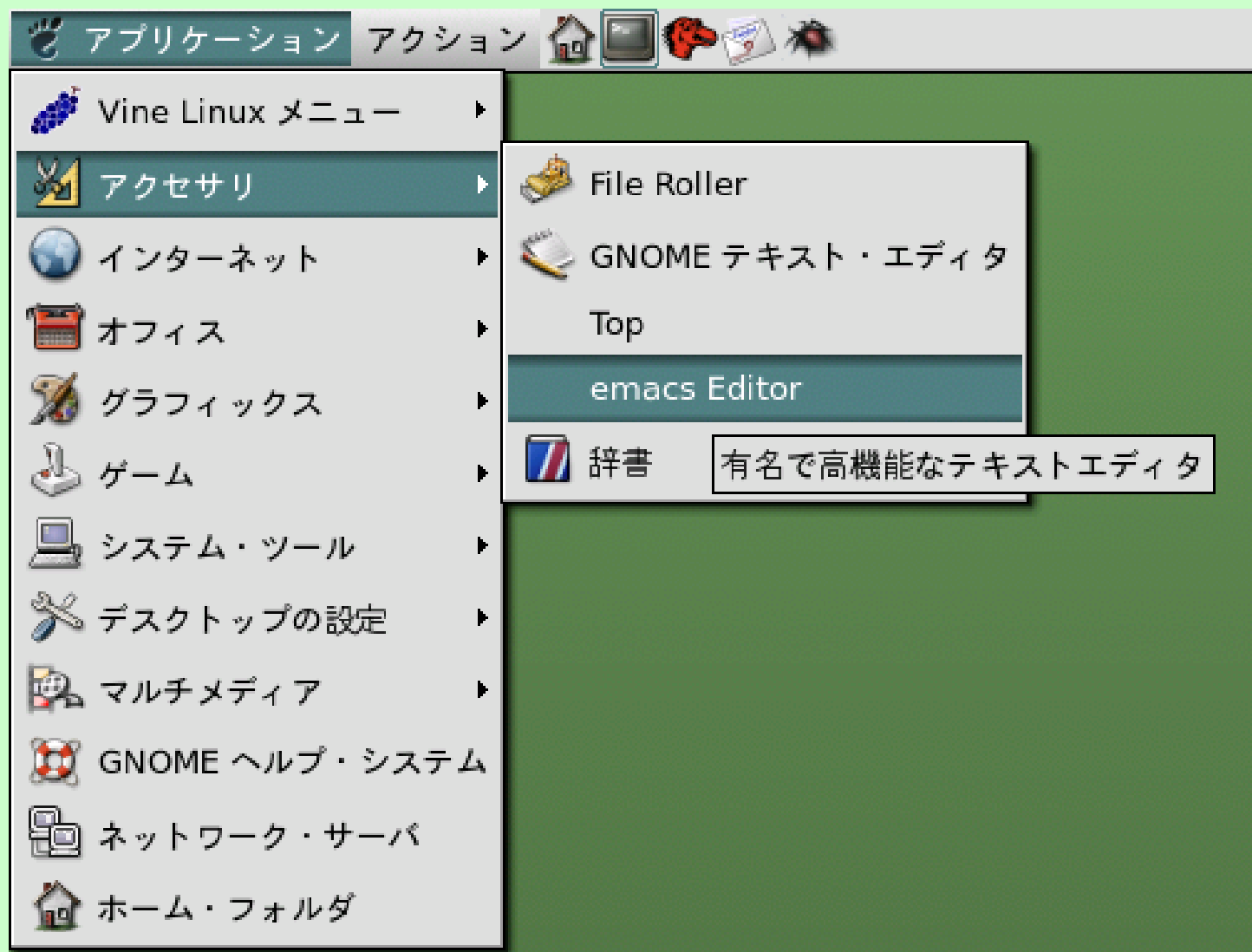
- テキストエディタ・統合環境 → たくさんある ([[参照](#)])
- お手元の環境でも vi, gedit, Emacs などが標準で使える

じゃあ なぜ Emacs か

- ☺ GNU のツール = GPL ライセンス (オープンソース)
  - 多くの人が開発に参加 → 英知が結集
- Mule (多言語処理), XEmacs (グラフィカル要素)
- Lisp 言語によるマクロ (語源: [E](#)ditor [macro](#)s)
  - 支援モード (インデント・色付け・補完)
  - コンパイラとの相性よい → デバッグの手間を軽減
  - カスタマイズに大きな自由度 (.emacs ファイル)

## 2 始め方・逃げ方 (・ 終わり方)

### 始め方



## 2 始め方・逃げ方 (・ 終わり方)

\$ emacs[\_ファイル名]\_& ... 別ウィンドウで起動 (お薦め)

→ 始めてしまえば, マウスを使って 何とか 操作可

- キーバインド (キー組合せ) 覚える → 作業効率が上がる

(別解) \$ emacs\_-nw[\_ファイル名]... コンソール内で起動

### 逃げ方

とりあえず (安全に) Emacs 内の動作を取り止める

- **Ctrl** + **g** ( $\overset{\text{def.}}{\longleftrightarrow}$  **Ctrl** を押しながら **g** を押す)

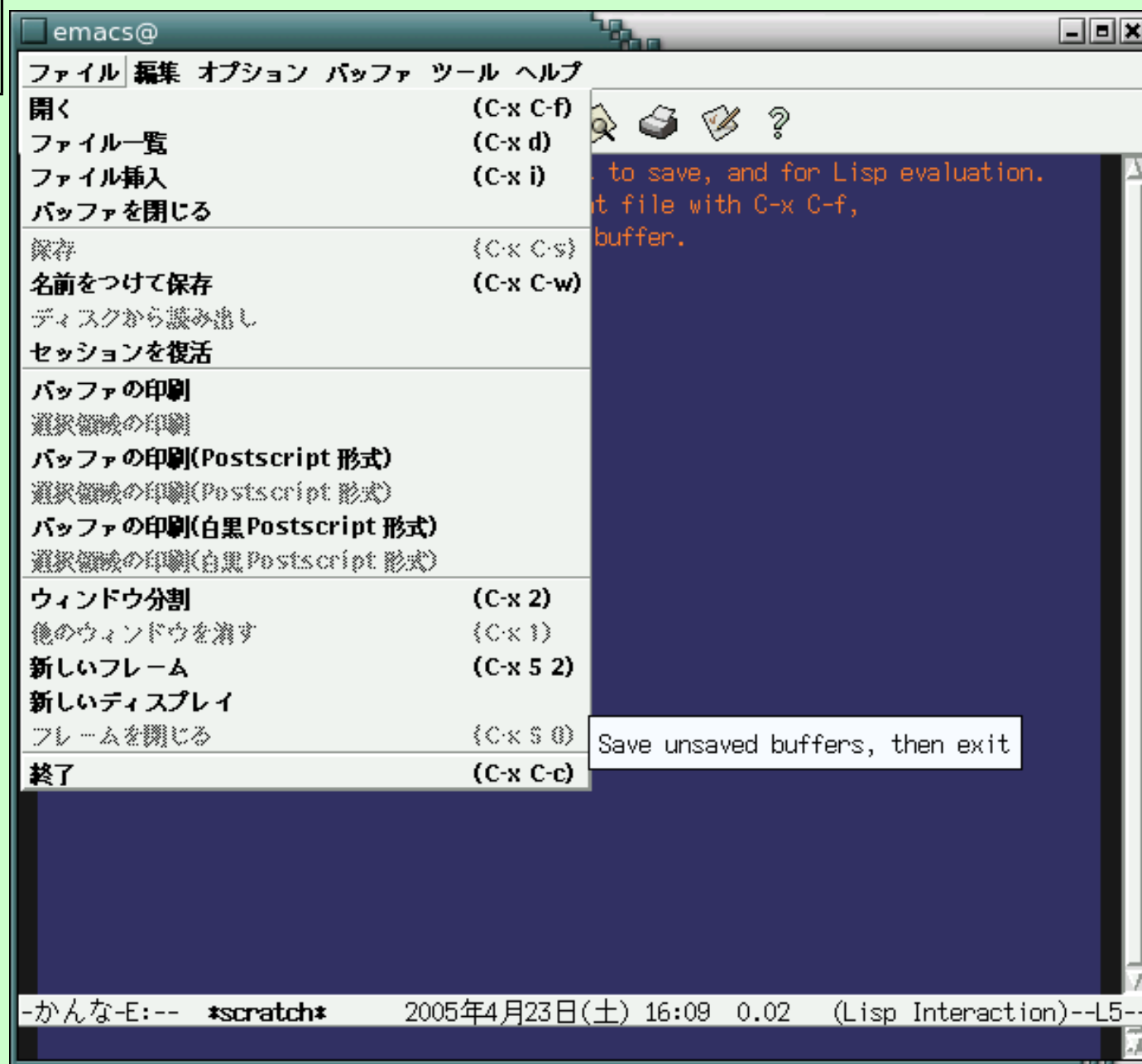
- **Esc** → **Esc** → **Esc**

( $\overset{\text{def.}}{\longleftrightarrow}$  **Esc** を押してから離すを 3 回繰り返す)

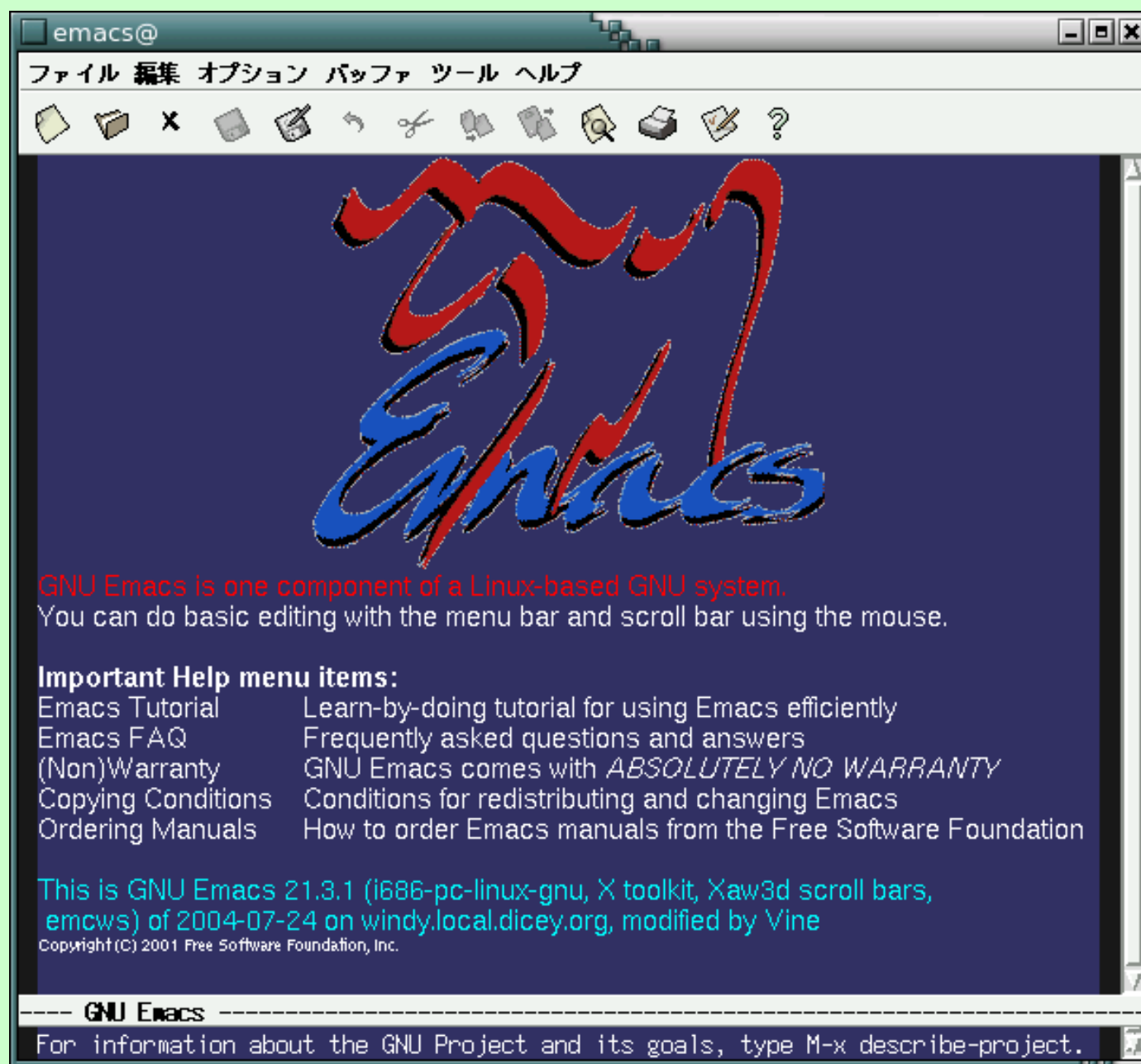
正確には 3 回程度 . 何回押せば止まるかは状況による

## 2 始め方・逃げ方 (・終わり方)

### 終わり方



### 3 ミニバッファと M-, C- 記法



### 3 ミニバッファと M-, C- 記法

#### ミニバッファ

- 表示
  - メッセージ (お叱り) ・ 行われた処理
  - 長い入れ子構造 / 範囲選択 → 括弧対応 / 始めの方
  - 日本語入力 → 候補
- “コマンド” 名の入力に使用

M-, C- 記法 ... “コマンド” 名の表示のために

- $M\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Alt}} + \boxed{key}$  または  $\boxed{\text{Esc}} \rightarrow \boxed{key}$
- $C\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Ctrl}} + \boxed{key}$
- $S\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Shift}} + \boxed{key}$



### 3 ミニバッファと M-, C- 記法

- $M\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Alt}} + \boxed{key}$  または  $\boxed{\text{Esc}} \rightarrow \boxed{key}$
- $C\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Ctrl}} + \boxed{key}$
- $S\text{-}key \xLeftrightarrow{\text{def.}} \boxed{\text{Shift}} + \boxed{key}$



... 多少 嘘が入っている ( $\boxed{key}$  の部分)

例.  $M\text{-}\% \Leftrightarrow \boxed{\text{Alt}} + \boxed{\text{Shift}} + \boxed{5}$  (または  $\boxed{\text{Shift}} + \boxed{\text{Alt}} + \boxed{5}$ )

組み合わせると .....

例.  $C\text{-}u\ 10\ C\text{-}n \Leftrightarrow \boxed{\text{Ctrl}} + \boxed{u} \rightarrow \boxed{1} \rightarrow \boxed{0} \rightarrow \boxed{\text{Ctrl}} + \boxed{n}$   
 $\Leftrightarrow C\text{-}u\ 1\ 0\ C\text{-}n$

## 4 操作 (“コマンド入力”) 手法

- マウスクリック
- *M-x command* (*M-x command* )
  - … *M-x* と打った後, ミニバッファにフォーカスが移る  
→ *command* を入力 (多少の 英語 想像力) → 
- キーバインド
  - まず Emacs Tutorial を読もう！

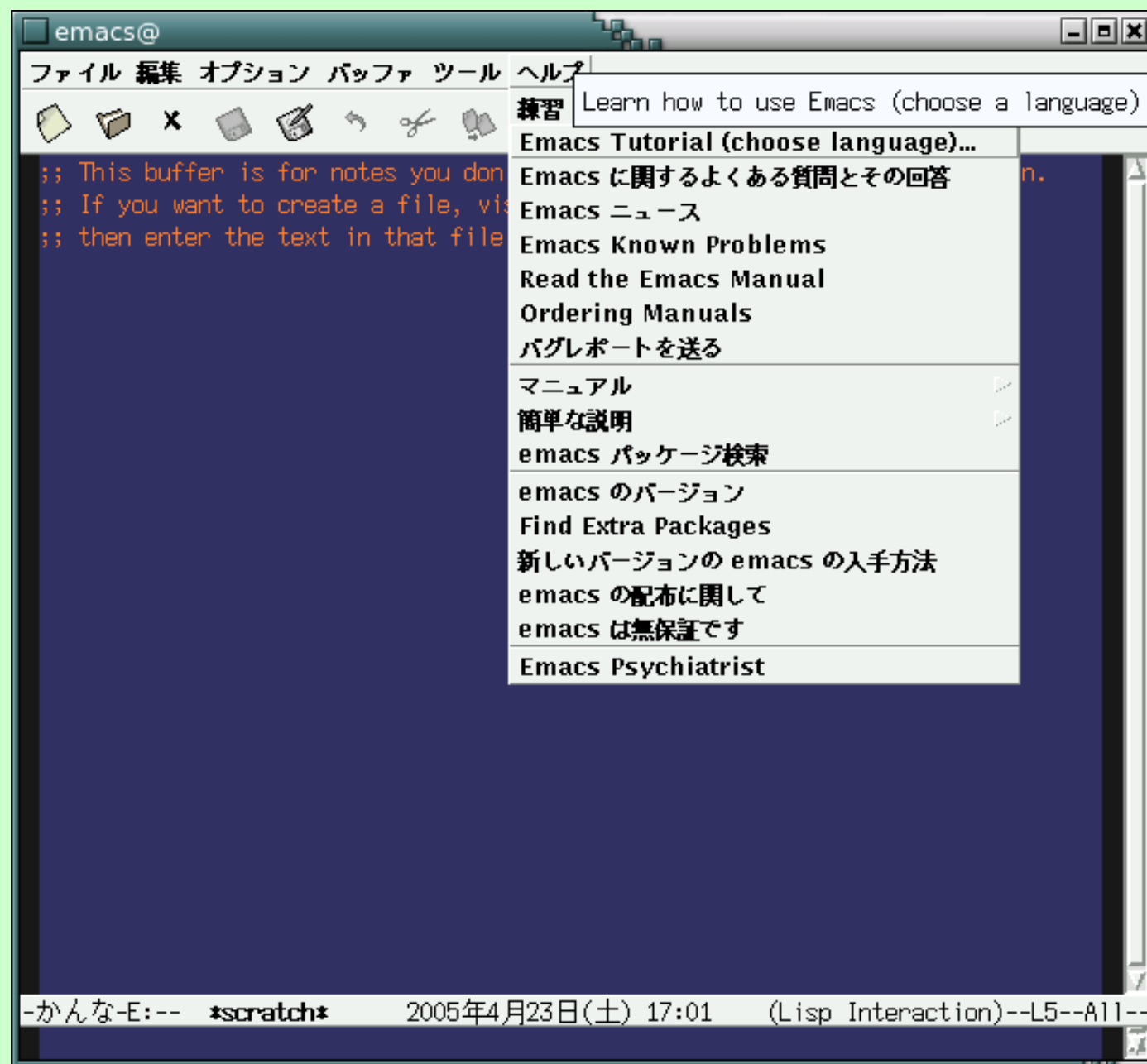
どれを読むの？ どうやって読むの？

### → 早速の復習

- マウスクリック
- *M-x help*

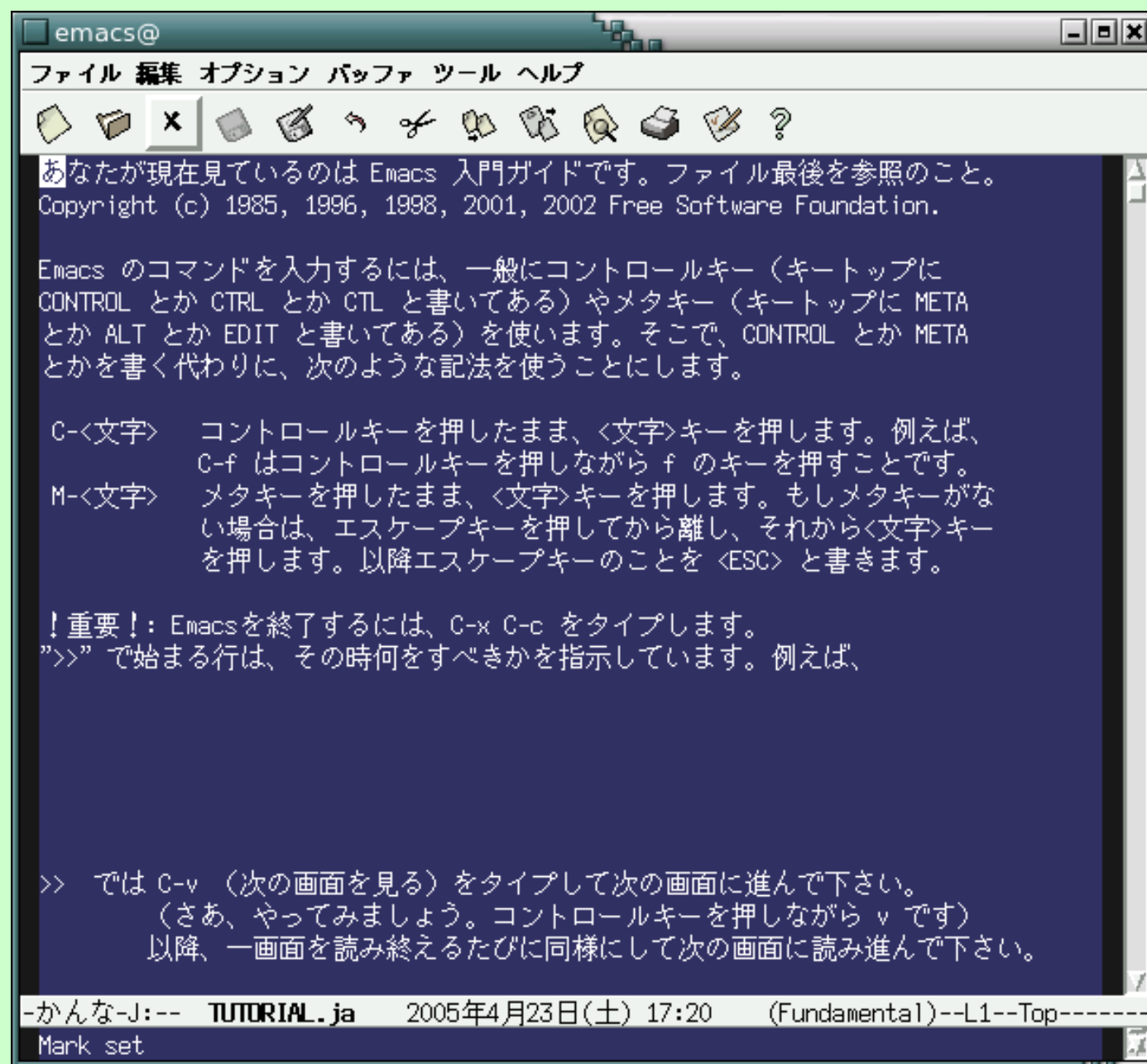
## 5 操作 (“コマンド入力”) 手法 (復習)

### マウスクリック



## 5 操作 (“コマンド入力”) 手法 (復習)

とりあえず消してやる



## 5 操作 (“コマンド入力”) 手法 (復習)

`M-x help`

- … `M-x` と打った後, ミニバッファにフォーカスが移る  
→ `help` を入力 → 
- … ミニバッファに現れる注意して, 次のステップへ  
→ , `Delete` / `Backspace` でスクロールしながら  
tutorial の文字を探す → `t`

キーバインド … ない?

- ∴ *command* にキーバインドを貼っている
- 最低限は覚えて, 困ったときは `M-x command` を使う
  - キーバインドが存在したらお叱りを受ける
  - そこで覚える

## 5 操作 (“コマンド入力”) 手法 (復習)

M-x help

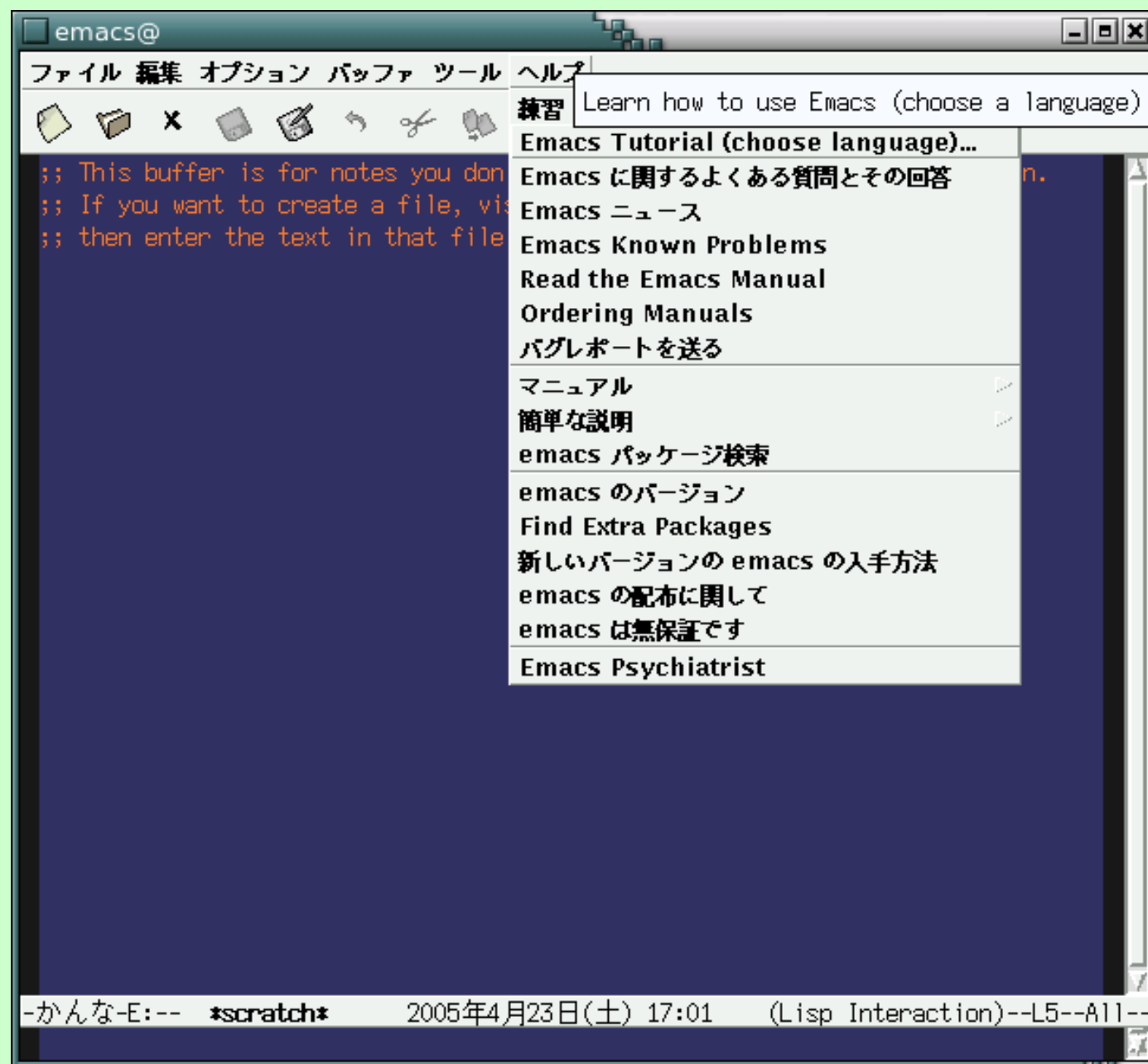
- … M-x と打った後，ミニバッファにフォーカスが移る  
→ help を入力 → 
- … ミニバッファに現れる注意して，次のステップへ  
→ ,  /  でスクロールしながら  
tutorial の文字を探す → 

キーバインド … ない？

- ∴ *command* にキーバインドを貼っている
- 最低限は覚えて，困ったときは M-x *command* を使う
  - キーバインドが存在したらお叱りを受ける
  - そこで覚える (おまえは M か？)

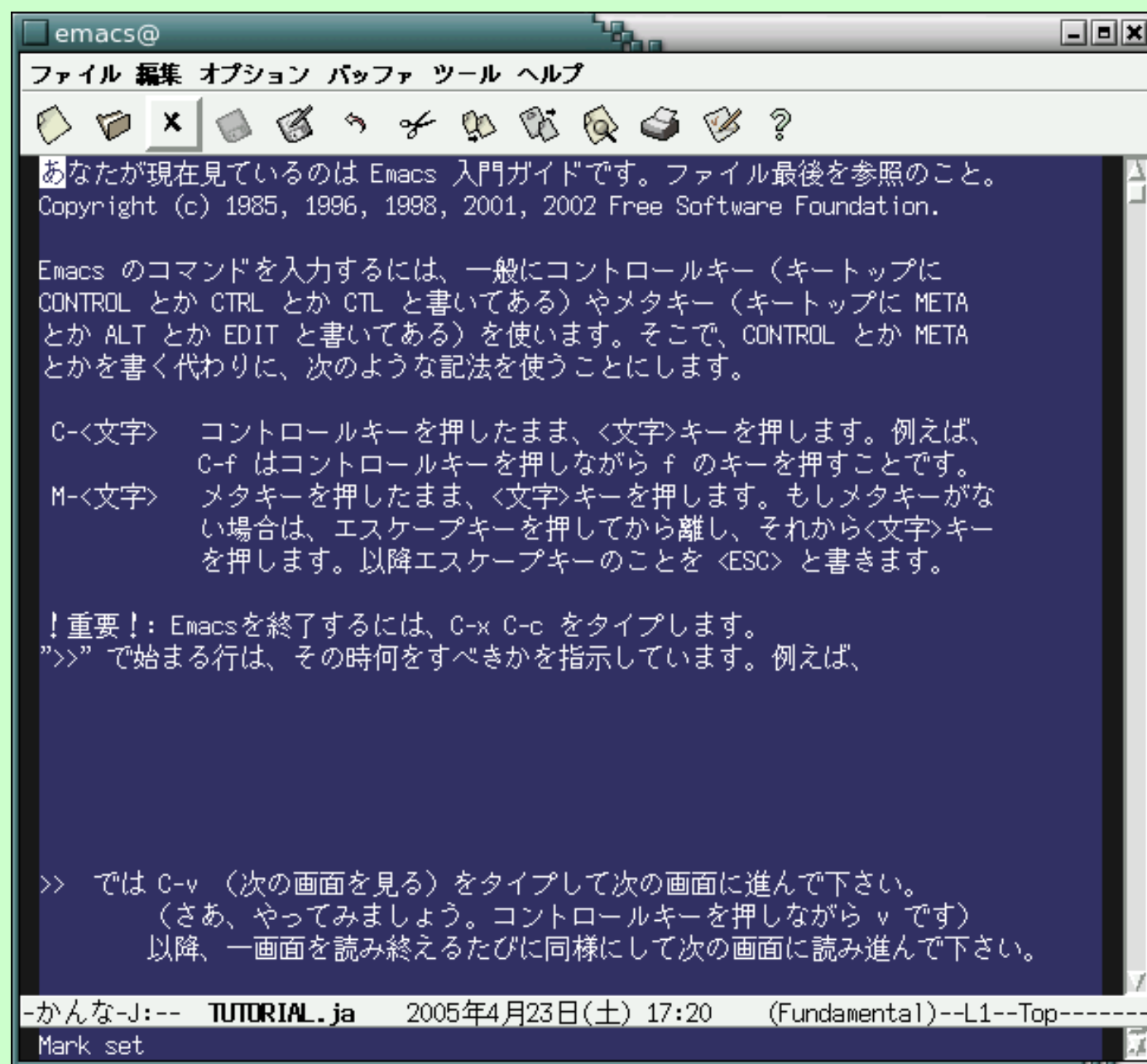
## 6 再録: Tutorial の呼出

### マウスクリック



## 6 再録: Tutorial の消去


とりあえず消してやる








## 6 補足: バッファの概念

「とりあえず消してやる」 $\Leftrightarrow$  バッファを消す

- タブブラウザのタブみたいなもの
- フォーカスが当たっているバッファ は 1 つだけ
  - └(例. 掲示板のフォームで書き込めるタブ)
- 編集するときにはフォーカスを当てる
  - `C-x b` バッファ名 . cf.) バッファ名一覧  $\rightarrow$  `C-x C-b`
- たくさん溜っていても構わない
  - 消したきゃ消せる  / `M-x kill-buffer` / `C-x k`
  - ウィンドウ分割 して見比べ可
    - cf.) `C-x 2` (上下分割), `C-x 3` (左右分割),  
`C-x o` (別のウィンドウへ), `C-x 1` (1 画面に)  
`C-x 0` (自ウィンドウを消して次にフォーカス)

## 6 補足: バッファの概念

---

- 保存しないと変更は反映されない
  - 終了 しようとする
    -  ファイル → 終了
    - M-x kill-emacs
    - C-x C-c
    - フレーム右上の   (環境によっては危険)
- … 保存されているかどうかをチェックしてくれる

## 7 怒涛のごとく・・・波に吞まれず必要性見極めて

「 Emacs で何ができるかわからないから、  
いま使ってるエディタで何ができたら嬉しいか  
って聞かれても困る 」


- 一通り 思いつくものを紹介します
- しかし、全ての機能を使いこなしている人はいない  
∴ 必要な機能だけ使えれば十分くらいの気持ちで


- 一方、最低限さえ覚えれば、
- テキストに関する 多くの機能を 統一的に 扱える
- $\left| \{x \mid \text{言語} \overset{\text{対応}}{\longleftrightarrow} \text{エディタ } x\} \right| \geq 2$  を解消できる！


・・・ メーラー・ブラウザも乗り換え → テキスト処理を楽に！

## 8 ファイル・ウィンドウ・バッファ操作


---

ファイルを  開く `C-x C-f / M-x find-file`


ディレクトリを開く ( ファイル一覧) `C-x d / M-x dired`


 ファイル挿入 `C-x i / M-x insert-file`


バッファを挿入 `M-x insert-buffer`

 保存 (上書き保存) `C-x C-s / M-x save-buffer`

以前あった *file* は, *file~* という名前で保持される

 名前をつけて保存 `C-x C-w / M-x write-file`

 バッファを閉じる `C-x k / M-x kill-buffer`

Emacs を  終了 `C-x C-c / M-x kill-emacs`


## 8 ファイル・ウィンドウ・バッファ操作

---

 ウィンドウを上下に 2 分割 `C-x 2 / M-x split-window`

ウィンドウを左右に 2 分割 `C-x 3 /`

`M-x split-window-horizontally`

 他のウィンドウを消す (ウィンドウを 1 つに) `C-x 1 /`


`M-x delete-other-windows`

自分のウィンドウを消して次のウィンドウへフォーカス

`C-x 0 (零) / M-x delete-windows`

別のウィンドウへ `C-x o / M-x other-window / S-→, ←, ↓, ↑`

別のバッファを呼ぶ `C-x b / M-x switch-to-buffer`

 バッファ一覧 `C-x C-b / M-x list-buffers`

## 9 コピー: 無意識にやっているかも知れませんが

- マウスで操作する場合

カーソルを コピーしたい領域 開始の位置まで動かす  
→ ドラッグして領域選択 (← これだって複合作業)  
→ カーソルを 貼り付けたい領域 開始の位置まで動かす  
→ 中ボタンを押す (または 左右両ボタンを同時に押す)

- キーボードで操作する場合

カーソルを動かす → 選択領域の開始位置を指定  
→ カーソルを動かす → コピー  
→ カーソルを動かす → 貼り付け

## 9 コピー: カーソルを動かす

---

1 文字右 → / C-f / M-x forward-char

1 文字左 ← / C-b / M-x backward-char

約 1 単語右 C-→ / M-f / M-x forward-word

約 1 単語左 C-← / M-b / M-x backward-word

行頭 C-a / M-x beginning-of-line

行末 C-e / M-x end-of-line

1 行下 ↓ / C-n / M-x next-line

1 行上 ↑ / C-p / M-x previous-line

1 スクロール下 PgDn / C-v / M-x scroll-up

1 スクロール上 PgUp / M-v / M-x scroll-down

バッファの先頭 Home / M-< / M-x beginning-of-buffer

バッファの末尾 End / M-> / M-x end-of-buffer

## 9 コピー: 領域を選択 → コピー → 貼りつけ

---

選択領域開始 `C-SPC` (スペース) / `M-x set-mark-command`

カーソルを目的の位置まで移動 矢印など

👉 コピー `M-w`

カーソルを目的の位置まで移動 矢印など

👉 貼りつけ `C-y` / `M-x yank`

切り貼りならば... 「コピー」を「切り取り」に

👉 切り取り `C-w` / `M-x kill-region`

もっとパワフルなコマンドも... (選択領域開始の指定が不要)

‘ここ’ から行末まで切り取り `C-k` / `M-x kill-line`

👉 元に戻す `C-/, C-x u, C-_` / `M-x undo`



## 9 コピー: (発展) 矩形領域編集

---

領域選択を ,

- 1 次元的にとらえるのではなく ,
- 2 次元的 (矩形 (くけい) = 長方形) としてとらえて動作

矩形領域切り取り `C-x r k / M-x kill-rectangle`

矩形領域コピー `C-x r r / M-x copy-rectangle-to-register`

矩形貼りつけ `C-x r y / M-x yank-rectangle`

矩形領域に空白を挿入 `C-x r o / M-x open-rectangle`

矩形領域を新たな文字列で置換 `C-x r t /`

`M-x string-rectangle`

## 10 テキスト操作

---

今日一番覚えていったら得したなぁと思えるコマンド！

バッファに存在する文字列から 補完 `M-/`

Delete `C-d`

Backspace `C-h` (環境によっては自分で設定)

1 文字フリップ `C-t`

単語チェック `M-$ / ispell-word` cf.) `ispell-region`

インクリメンタル・サーチ (= 見つかったらすぐハイライト)

`C-s / M-x isearch-forward` → `C-o` で日本語入力

`C-r / M-x isearch-backward`

置換 `M-% / M-x query-replace`

正規表現 `M-C-s, M-C-r, M-C-% / -regexp` を追加

注) ruby/perl と異なる記法を採用している部分あり

# 11 プログラミング

---

インデント `TAB`

選択領域をインデント `C-M-\` / `M-x indent-region`

領域をコメントアウト・復帰 `C-c` など (言語によって差)

👉 ツール → コンパイル `M-x compile`

次のエラーへ `C-x '` / `M-x next-error`

$n$  行目へ `M-x goto-line n`

👉 ツール → Shell Command `M-!`

結果 → \*Shell Command Output\* バッファに排出

… バッファからの挿入など使うと効率アップか？！

\*shell\* を立ち上げる `M-x shell`

👉 比較 → ファイル `M-x diff` … 色付表示. 差分反映も簡単

## 12 モード・文字コード

---

ディレクトリモード `C-x d / C-x f` で `~` を入力

→ 入った後の操作 (ディレクトリ管理) は Help で

例. ディレクトリ名の変更, ファイルを開く, ...

自動改行モードの on/off `M-x auto-fill-mode`

勝手に改行されては困る (設定ファイル, html など) ときに

いま開いている文書の文字コードを変える

`C-x RET f / M-x set-buffer-file-coding-system`

次に実行する コマンド の文字コードを変える `C-x RET c`

ファイルを開いても文字コードを正しく解釈してくれな

いとき → 上のコマンドで設定してから ファイルを開く

## 13 .emacs, 各種マクロ

---

- ~/.emacs.my.el に自分用の設定を書く

### 例えば

- 選択領域を色づけして見やすく  
;; Visual feedback on selections  
(setq-default transient-mark-mode t)
- 括弧対応を色づけして見やすく  
;; Highlight matching parenthesis  
(when (fboundp 'show-paren-mode) (show-paren-mode t))
- Shift + 矢印 で領域選択  
;; Windows-like mark set  
(load-library "s-region")

注) ウィンドウを移りあうために **S-矢印**は使えなくなる

## 13 .emacs, 各種マクロ・英語 想像力を養う

---

- 世の中にはいろいろなマクロが出回っている
  - メーラー: Mew, Wanderlust,
  - ブラウザ: Emacs-w3m (Wiki など ... 需要増),
  - 各種言語に対応した入力支援 ...

### 英語 想像力を養う

- window, buffer, region, line, word, char,
- exit, kill, split, fill, set, toggle (=on/off 切換),
- next, previous, forward, back など

→ 操作性を上げて幸福な Emacs 生活を！

## 付録 A CapsLock の位置を Ctrl に変える

---

X の設定 (Caps をなくす) `/etc/X11/xorg.conf` の

```
# Option "XkbOptions" "ctrl:nocaps"
```

を有効にして (行頭の # を消す) X を再起動 する.

注 (X の再起動). ログアウト後, Ctrl+Alt+Backspace .

コンソールの設定 (入換) `/etc/sysconfig/keyboard` を

```
#KEYTABLE="jp106"
```

```
KEYTABLE="jp106_Ctrl_CAPS"
```

とする .

(別解) ツール `/usr/sbin/kbdconfig` を使う .

コンソール設定 (Caps をなくす = 設定ファイルを 若干 変更)

`/usr/lib/kbd/keymaps/i386/qwerty` 以下に 適当に  
キーマップファイルを書いて置く → 設定する .

## 付録 B Lisp 言語のいろはの ‘い’

---

\*scratch\* バッファで

```
(+ (- 1 2) 5) C-j
```

```
pi C-j
```

```
(sin (/ pi 4)) C-j
```

- 関数型言語 (入 式も書ける) . 数値・文字列・配列・関数も統一されたフォームで保持される .
- Emacs 本体の大部分 , 各種マクロや .emacs を記述 .
- (講義) 機情 3 年向けのソフトウェア第三で少し扱われた .



## 付録 B Lisp 言語のいろはの ‘い’

---

\*scratch\* バッファで

(+ (- 1 2) 5) C-j

4

pi C-j

3.141593653589793

(sin (/ pi 4)) C-j

0.7071067811865475

- 関数型言語 (入 式も書ける) . 数値・文字列・配列・関数も統一されたフォームで保持される .
- Emacs 本体の大部分 , 各種マクロや .emacs を記述 .
- (講義) 機情 3 年向けのソフトウェア第三で少し扱われた .

来週 (5 月 4 日) は休講 (休日) です .

次回は L<sup>A</sup>T<sub>E</sub>X 入門です .

お楽しみに .

`www.sr3.t.u-tokyo.ac.jp/~okayama/linux-seminar/  
meira.misojiro.t.u-tokyo.ac.jp/mist-laptop/`

[参照]: [oku.edu.mie-u.ac.jp/~okumura/texwiki/?TeX](http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?TeX) 用エディタ