

Being Productive With Emacs

Part 1



Phil Sung

`sipb-iap-emacs@mit.edu`

`http://stuff.mit.edu/iap/emacs`

Special thanks to Piau Na and Arthur Gleckler

“Emacs is the extensible, customizable,
self-documenting real-time display editor.”

The many faces of Emacs

```

emacs@localhost
;;; ALISTS and related functions

(defvar *month-table*
  '(("Jan" . 1) ("Feb" . 2) ("Mar" . 3) ("Apr" . 4)
    ("May" . 5) ("Jun" . 6) ("Jul" . 7) ("Aug" . 8)
    ("Sep" . 9) ("Oct" . 10) ("Nov" . 11) ("Dec" . 12)))
(defvar *day-table*
  '(("Sun" . 1) ("Mon" . 2) ("Tue" . 3) ("Wed" . 4)
    ("Thu" . 5) ("Fri" . 6) ("Sat" . 7)))
(defun alist-lookup (key alist)
  (cdr (assoc key alist :test #'string=)))
(defun alist-reverse-lookup (value alist)
  (car (rassoc value alist)))

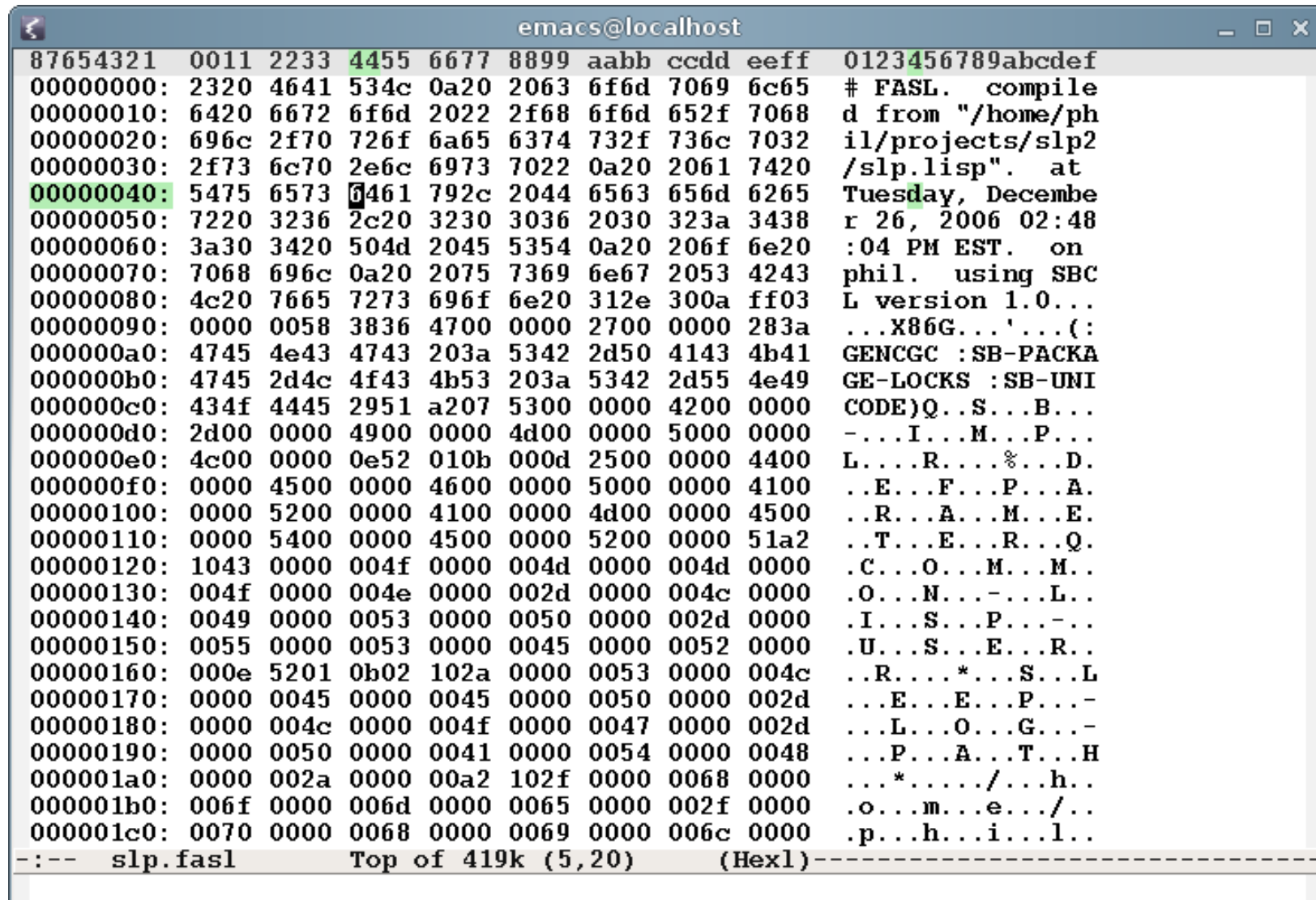
;;; PARSING

(defun read-file-data (file-name)
  "Reads file data and puts it into a list"
  (with-open-file (in-file file-name)
    (when in-file
      (let ((lst '()))
        (loop for line = (read-line in-file nil)
              while line
              do (push line lst))
        (nreverse lst)))))

(defun parse-raw-date (date-str)
  "Parses a date into a list"
  (destructuring-subseq-bind ((day-of-week 0 3)
                             (day 4 6)
                             (month 7 9)
                             (year 10 12)
                             (hour 13 15)
                             (min 16 18)
                             (sec 19 21)
                             (microsec 22 26)
                             (tz 27 31)
                             (dst 32 34)
                             (dst2 35 37)
                             (dst3 38 40)
                             (dst4 41 43)
                             (dst5 44 46)
                             (dst6 47 49)
                             (dst7 50 52)
                             (dst8 53 55)
                             (dst9 56 58)
                             (dst10 59 61)
                             (dst11 62 64)
                             (dst12 65 67)
                             (dst13 68 70)
                             (dst14 71 73)
                             (dst15 74 76)
                             (dst16 77 79)
                             (dst17 80 82)
                             (dst18 83 85)
                             (dst19 86 88)
                             (dst20 89 91)
                             (dst21 92 94)
                             (dst22 95 97)
                             (dst23 98 100)
                             (dst24 101 103)
                             (dst25 104 106)
                             (dst26 107 109)
                             (dst27 110 112)
                             (dst28 113 115)
                             (dst29 116 118)
                             (dst30 119 121)
                             (dst31 122 124)
                             (dst32 125 127)
                             (dst33 128 130)
                             (dst34 131 133)
                             (dst35 134 136)
                             (dst36 137 139)
                             (dst37 140 142)
                             (dst38 143 145)
                             (dst39 146 148)
                             (dst40 149 151)
                             (dst41 152 154)
                             (dst42 155 157)
                             (dst43 158 160)
                             (dst44 161 163)
                             (dst45 164 166)
                             (dst46 167 169)
                             (dst47 170 172)
                             (dst48 173 175)
                             (dst49 176 178)
                             (dst50 179 181)
                             (dst51 182 184)
                             (dst52 185 187)
                             (dst53 188 190)
                             (dst54 191 193)
                             (dst55 194 196)
                             (dst56 197 199)
                             (dst57 200 202)
                             (dst58 203 205)
                             (dst59 206 208)
                             (dst60 209 211)
                             (dst61 212 214)
                             (dst62 215 217)
                             (dst63 218 220)
                             (dst64 221 223)
                             (dst65 224 226)
                             (dst66 227 229)
                             (dst67 230 232)
                             (dst68 233 235)
                             (dst69 236 238)
                             (dst70 239 241)
                             (dst71 242 244)
                             (dst72 245 247)
                             (dst73 248 250)
                             (dst74 251 253)
                             (dst75 254 256)
                             (dst76 257 259)
                             (dst77 260 262)
                             (dst78 263 265)
                             (dst79 266 268)
                             (dst80 269 271)
                             (dst81 272 274)
                             (dst82 275 277)
                             (dst83 278 280)
                             (dst84 281 283)
                             (dst85 284 286)
                             (dst86 287 289)
                             (dst87 290 292)
                             (dst88 293 295)
                             (dst89 296 298)
                             (dst90 299 301)
                             (dst91 302 304)
                             (dst92 305 307)
                             (dst93 308 310)
                             (dst94 311 313)
                             (dst95 314 316)
                             (dst96 317 319)
                             (dst97 320 322)
                             (dst98 323 325)
                             (dst99 326 328)
                             (dst100 329 331)
                             (dst101 332 334)
                             (dst102 335 337)
                             (dst103 338 340)
                             (dst104 341 343)
                             (dst105 344 346)
                             (dst106 347 349)
                             (dst107 350 352)
                             (dst108 353 355)
                             (dst109 356 358)
                             (dst110 359 361)
                             (dst111 362 364)
                             (dst112 365 367)
                             (dst113 368 370)
                             (dst114 371 373)
                             (dst115 374 376)
                             (dst116 377 379)
                             (dst117 380 382)
                             (dst118 383 385)
                             (dst119 386 388)
                             (dst120 389 391)
                             (dst121 392 394)
                             (dst122 395 397)
                             (dst123 398 400)
                             (dst124 401 403)
                             (dst125 404 406)
                             (dst126 407 409)
                             (dst127 410 412)
                             (dst128 413 415)
                             (dst129 416 418)
                             (dst130 419 421)
                             (dst131 422 424)
                             (dst132 425 427)
                             (dst133 428 430)
                             (dst134 431 433)
                             (dst135 434 436)
                             (dst136 437 439)
                             (dst137 440 442)
                             (dst138 443 445)
                             (dst139 446 448)
                             (dst140 449 451)
                             (dst141 452 454)
                             (dst142 455 457)
                             (dst143 458 460)
                             (dst144 461 463)
                             (dst145 464 466)
                             (dst146 467 469)
                             (dst147 470 472)
                             (dst148 473 475)
                             (dst149 476 478)
                             (dst150 479 481)
                             (dst151 482 484)
                             (dst152 485 487)
                             (dst153 488 490)
                             (dst154 491 493)
                             (dst155 494 496)
                             (dst156 497 499)
                             (dst157 500 502)
                             (dst158 503 505)
                             (dst159 506 508)
                             (dst160 509 511)
                             (dst161 512 514)
                             (dst162 515 517)
                             (dst163 518 520)
                             (dst164 521 523)
                             (dst165 524 526)
                             (dst166 527 529)
                             (dst167 530 532)
                             (dst168 533 535)
                             (dst169 536 538)
                             (dst170 539 541)
                             (dst171 542 544)
                             (dst172 545 547)
                             (dst173 548 550)
                             (dst174 551 553)
                             (dst175 554 556)
                             (dst176 557 559)
                             (dst177 560 562)
                             (dst178 563 565)
                             (dst179 566 568)
                             (dst180 569 571)
                             (dst181 572 574)
                             (dst182 575 577)
                             (dst183 578 580)
                             (dst184 581 583)
                             (dst185 584 586)
                             (dst186 587 589)
                             (dst187 590 592)
                             (dst188 593 595)
                             (dst189 596 598)
                             (dst190 599 601)
                             (dst191 602 604)
                             (dst192 605 607)
                             (dst193 608 610)
                             (dst194 611 613)
                             (dst195 614 616)
                             (dst196 617 619)
                             (dst197 620 622)
                             (dst198 623 625)
                             (dst199 626 628)
                             (dst200 629 631)
                             (dst201 632 634)
                             (dst202 635 637)
                             (dst203 638 640)
                             (dst204 641 643)
                             (dst205 644 646)
                             (dst206 647 649)
                             (dst207 650 652)
                             (dst208 653 655)
                             (dst209 656 658)
                             (dst210 659 661)
                             (dst211 662 664)
                             (dst212 665 667)
                             (dst213 668 670)
                             (dst214 671 673)
                             (dst215 674 676)
                             (dst216 677 679)
                             (dst217 680 682)
                             (dst218 683 685)
                             (dst219 686 688)
                             (dst220 689 691)
                             (dst221 692 694)
                             (dst222 695 697)
                             (dst223 698 700)
                             (dst224 701 703)
                             (dst225 704 706)
                             (dst226 707 709)
                             (dst227 710 712)
                             (dst228 713 715)
                             (dst229 716 718)
                             (dst230 719 721)
                             (dst231 722 724)
                             (dst232 725 727)
                             (dst233 728 730)
                             (dst234 731 733)
                             (dst235 734 736)
                             (dst236 737 739)
                             (dst237 740 742)
                             (dst238 743 745)
                             (dst239 746 7
```

Emacs edits source code

The many faces of Emacs



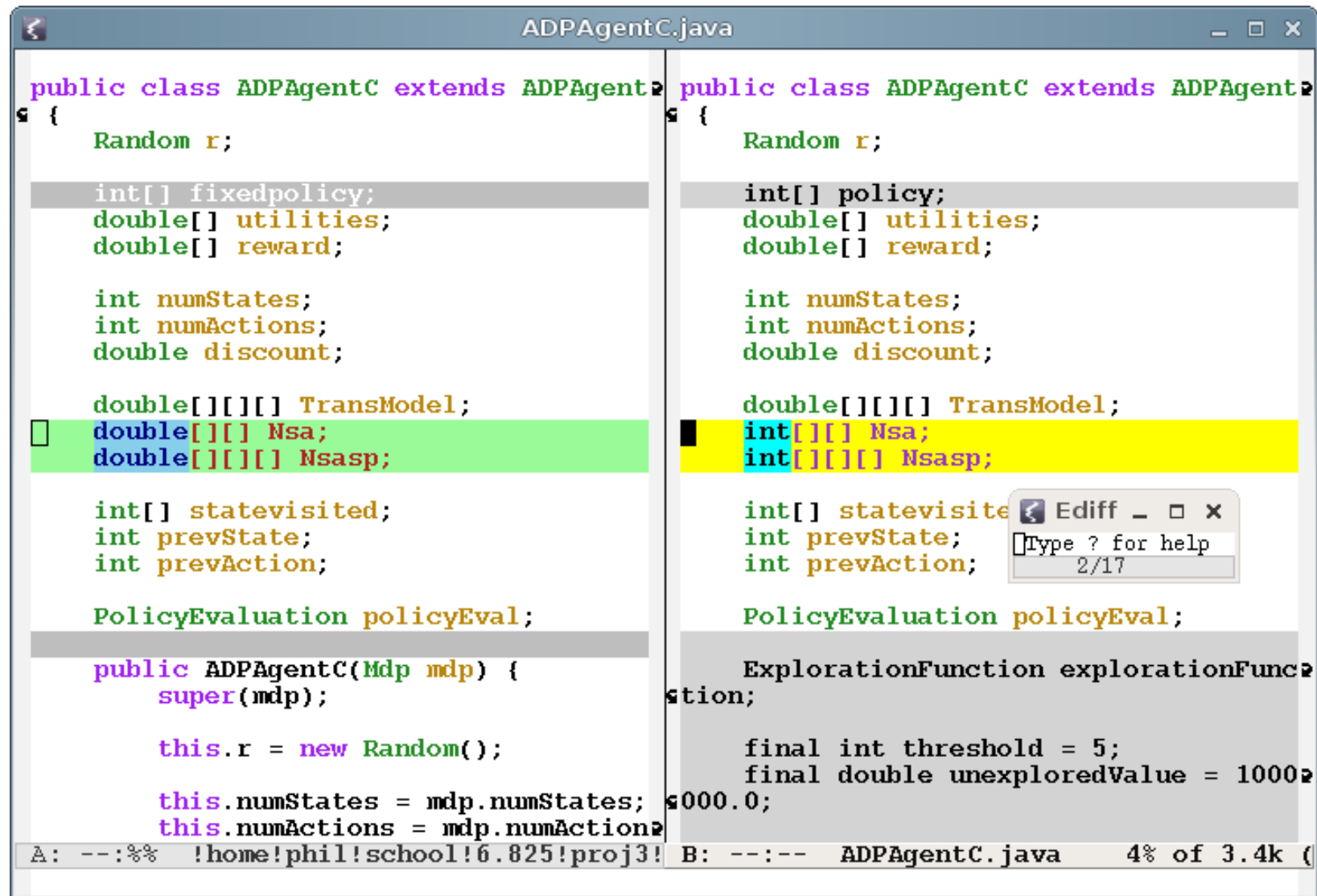
The screenshot shows the Emacs editor window titled 'emacs@localhost'. The main display area is in hex editor mode, showing a file named 'slp.fasl'. The address bar at the top indicates the current position is 'Top of 419k (5,20)' in '(Hexl)' mode. The main display area shows a hex dump of the file, with the first line of the dump being '87654321 0011 2233 4455 6677 8899 aabb ccdd eeff 0123456789abcdef'. The text on the right side of the hex dump is a comment: '# FASL. compile d from "/home/ph il/projects/slp2 /slp.lisp". at Tuesday, Decembe r 26, 2006 02:48 :04 PM EST. on phil. using SBC L version 1.0... ...X86G...'...(: GENCGC :SB-PACKA GE-LOCKS :SB-UNI CODE)Q...S...B... -...I...M...P... L...R...%...D. ..E...F...P...A. ..R...A...M...E. ..T...E...R...Q. .C...O...M...M.. .O...N...-...L.. .I...S...P...-... .U...S...E...R.. ..R...*...S...L ...E...E...P...- ...L...O...G...- ...P...A...T...H ...*.../...h.. .o...m...e.../.. .p...h...i...l..'. The status bar at the bottom shows 'slp.fasl' and 'Top of 419k (5,20) (Hexl)'.

```
87654321 0011 2233 4455 6677 8899 aabb ccdd eeff 0123456789abcdef
00000000: 2320 4641 534c 0a20 2063 6f6d 7069 6c65 # FASL. compile
00000010: 6420 6672 6f6d 2022 2f68 6f6d 652f 7068 d from "/home/ph
00000020: 696c 2f70 726f 6a65 6374 732f 736c 7032 il/projects/slp2
00000030: 2f73 6c70 2e6c 6973 7022 0a20 2061 7420 /slp.lisp". at
00000040: 5475 6573 5461 792c 2044 6563 656d 6265 Tuesday, Decembe
00000050: 7220 3236 2c20 3230 3036 2030 323a 3438 r 26, 2006 02:48
00000060: 3a30 3420 504d 2045 5354 0a20 206f 6e20 :04 PM EST. on
00000070: 7068 696c 0a20 2075 7369 6e67 2053 4243 phil. using SBC
00000080: 4c20 7665 7273 696f 6e20 312e 300a ff03 L version 1.0...
00000090: 0000 0058 3836 4700 0000 2700 0000 283a ...X86G...'...(:
000000a0: 4745 4e43 4743 203a 5342 2d50 4143 4b41 GENCGC :SB-PACKA
000000b0: 4745 2d4c 4f43 4b53 203a 5342 2d55 4e49 GE-LOCKS :SB-UNI
000000c0: 434f 4445 2951 a207 5300 0000 4200 0000 CODE)Q...S...B...
000000d0: 2d00 0000 4900 0000 4d00 0000 5000 0000 -...I...M...P...
000000e0: 4c00 0000 0e52 010b 000d 2500 0000 4400 L...R...%...D.
000000f0: 0000 4500 0000 4600 0000 5000 0000 4100 ..E...F...P...A.
00000100: 0000 5200 0000 4100 0000 4d00 0000 4500 ..R...A...M...E.
00000110: 0000 5400 0000 4500 0000 5200 0000 51a2 ..T...E...R...Q.
00000120: 1043 0000 004f 0000 004d 0000 004d 0000 .C...O...M...M..
00000130: 004f 0000 004e 0000 002d 0000 004c 0000 .O...N...-...L..
00000140: 0049 0000 0053 0000 0050 0000 002d 0000 .I...S...P...-...
00000150: 0055 0000 0053 0000 0045 0000 0052 0000 .U...S...E...R..
00000160: 000e 5201 0b02 102a 0000 0053 0000 004c ..R...*...S...L
00000170: 0000 0045 0000 0045 0000 0050 0000 002d ...E...E...P...-
00000180: 0000 004c 0000 004f 0000 0047 0000 002d ...L...O...G...-
00000190: 0000 0050 0000 0041 0000 0054 0000 0048 ...P...A...T...H
000001a0: 0000 002a 0000 00a2 102f 0000 0068 0000 ...*.../...h..
000001b0: 006f 0000 006d 0000 0065 0000 002f 0000 .o...m...e.../..
000001c0: 0070 0000 0068 0000 0069 0000 006c 0000 .p...h...i...l..
--:-- slp.fasl Top of 419k (5,20) (Hexl)-----
```

Emacs is a hex editor

M-x hexl-find-file

The many faces of Emacs



The screenshot shows the Emacs editor with a file named `ADPAgentC.java` open. The editor is displaying a diff between two versions of the file, with the left pane (A) showing the original code and the right pane (B) showing the modified code. The diff is highlighted in green and yellow. A small `Ediff` window is visible in the bottom right corner of the diff area, showing the command `Type ? for help` and the page number `2/17`. The status bar at the bottom indicates the current buffer is `ADPAgentC.java` and shows the file size as `4% of 3.4k`.

```
public class ADPAgentC extends ADPAgent {
  Random r;

  int[] fixedpolicy;
  double[] utilities;
  double[] reward;

  int numStates;
  int numActions;
  double discount;

  double[][][] TransModel;
  double[][] Nsa;
  double[][][] Nsasp;

  int[] statevisited;
  int prevState;
  int prevAction;

  PolicyEvaluation policyEval;

  public ADPAgentC(Mdp mdp) {
    super(mdp);

    this.r = new Random();

    this.numStates = mdp.numStates;
    this.numActions = mdp.numActions;
  }
}
```

```
public class ADPAgentC extends ADPAgent {
  Random r;

  int[] policy;
  double[] utilities;
  double[] reward;

  int numStates;
  int numActions;
  double discount;

  double[][][] TransModel;
  int[][] Nsa;
  int[][][] Nsasp;

  int[] statevisited;
  int prevState;
  int prevAction;

  PolicyEvaluation policyEval;

  ExplorationFunction explorationFunction;

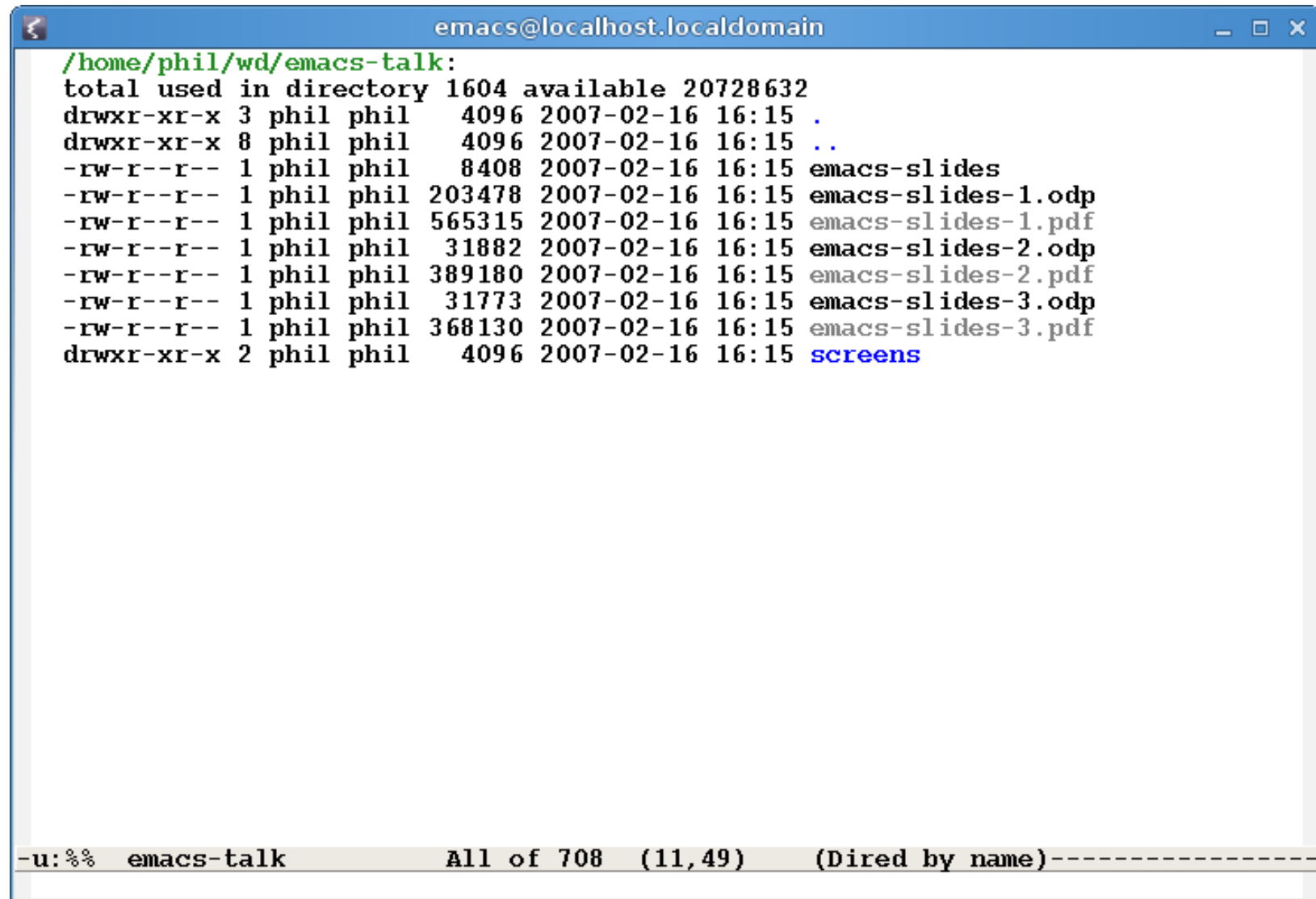
  final int threshold = 5;
  final double unexploredValue = 1000.0;
}
```

A: --:%% !home!phil!school!6.825!proj3! B: --:-- ADPAgentC.java 4% of 3.4k (

Emacs does diffs

M-x ediff-buffers

The many faces of Emacs



The screenshot shows an Emacs window titled 'emacs@localhost.localdomain'. The buffer contains a directory listing for the path '/home/phil/wd/emacs-talk:'. The listing shows the total used and available space, followed by a table of files with their permissions, counts, owners, sizes, dates, and times. The files listed are '.', '..', 'emacs-slides', 'emacs-slides-1.odp', 'emacs-slides-1.pdf', 'emacs-slides-2.odp', 'emacs-slides-2.pdf', 'emacs-slides-3.odp', 'emacs-slides-3.pdf', and 'screens'. The status bar at the bottom shows '-u:%% emacs-talk All of 708 (11,49) (Dired by name)-----'.

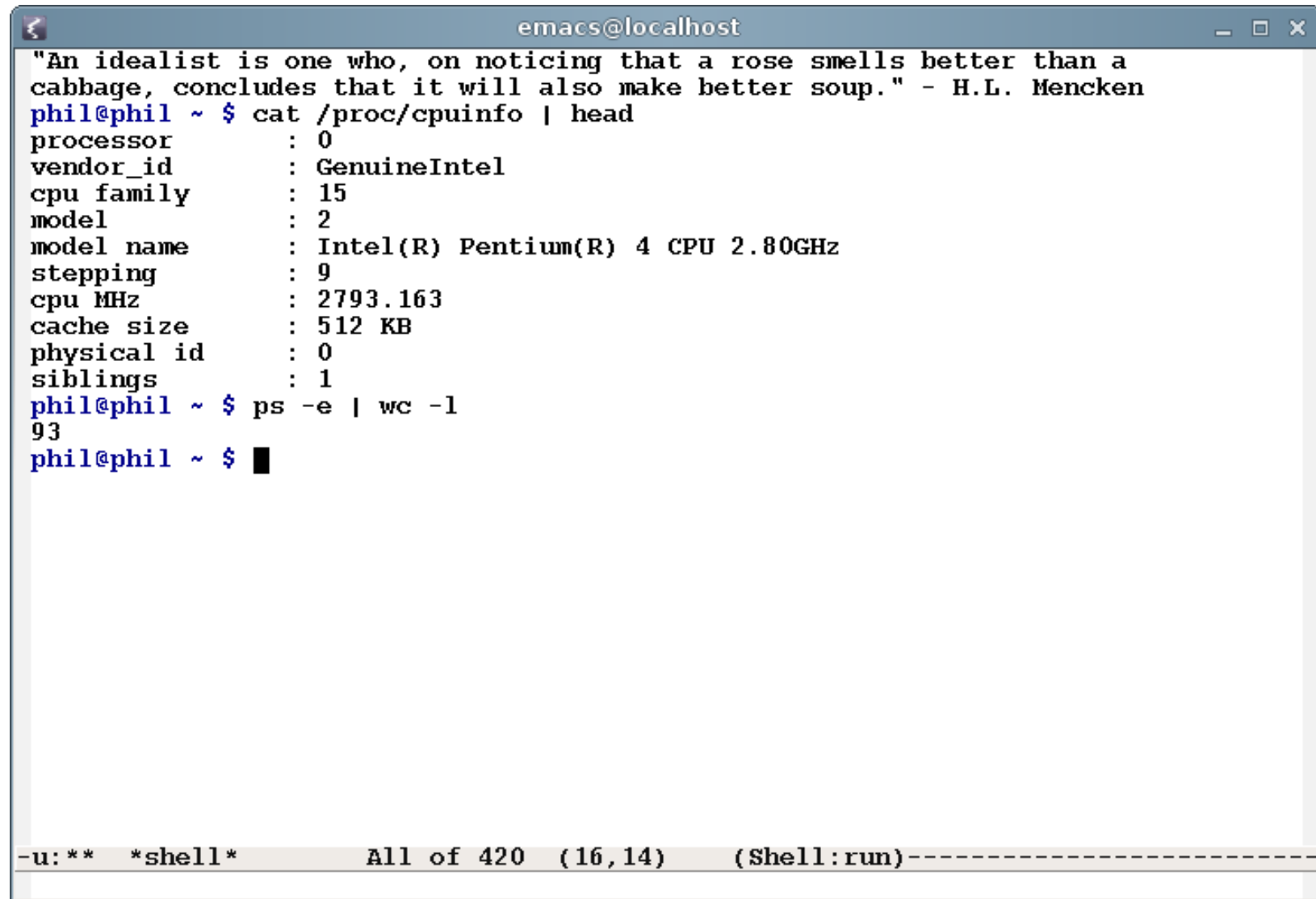
```
emacs@localhost.localdomain
/home/phil/wd/emacs-talk:
total used in directory 1604 available 20728632
drwxr-xr-x 3 phil phil 4096 2007-02-16 16:15 .
drwxr-xr-x 8 phil phil 4096 2007-02-16 16:15 ..
-rw-r--r-- 1 phil phil 8408 2007-02-16 16:15 emacs-slides
-rw-r--r-- 1 phil phil 203478 2007-02-16 16:15 emacs-slides-1.odp
-rw-r--r-- 1 phil phil 565315 2007-02-16 16:15 emacs-slides-1.pdf
-rw-r--r-- 1 phil phil 31882 2007-02-16 16:15 emacs-slides-2.odp
-rw-r--r-- 1 phil phil 389180 2007-02-16 16:15 emacs-slides-2.pdf
-rw-r--r-- 1 phil phil 31773 2007-02-16 16:15 emacs-slides-3.odp
-rw-r--r-- 1 phil phil 368130 2007-02-16 16:15 emacs-slides-3.pdf
drwxr-xr-x 2 phil phil 4096 2007-02-16 16:15 screens

-u:%% emacs-talk All of 708 (11,49) (Dired by name)-----
```

Emacs is a file manager

M-x dired

The many faces of Emacs

A screenshot of an Emacs window titled 'emacs@localhost'. The window displays a shell environment where a user named 'phil' has executed several commands. The first command is a quote by H.L. Mencken. The second command is 'cat /proc/cpuinfo | head', which outputs system information about the processor. The third command is 'ps -e | wc -l', which outputs the number of processes (93). The window has a status bar at the bottom showing '-u:** *shell* All of 420 (16,14) (Shell:run)-----'.

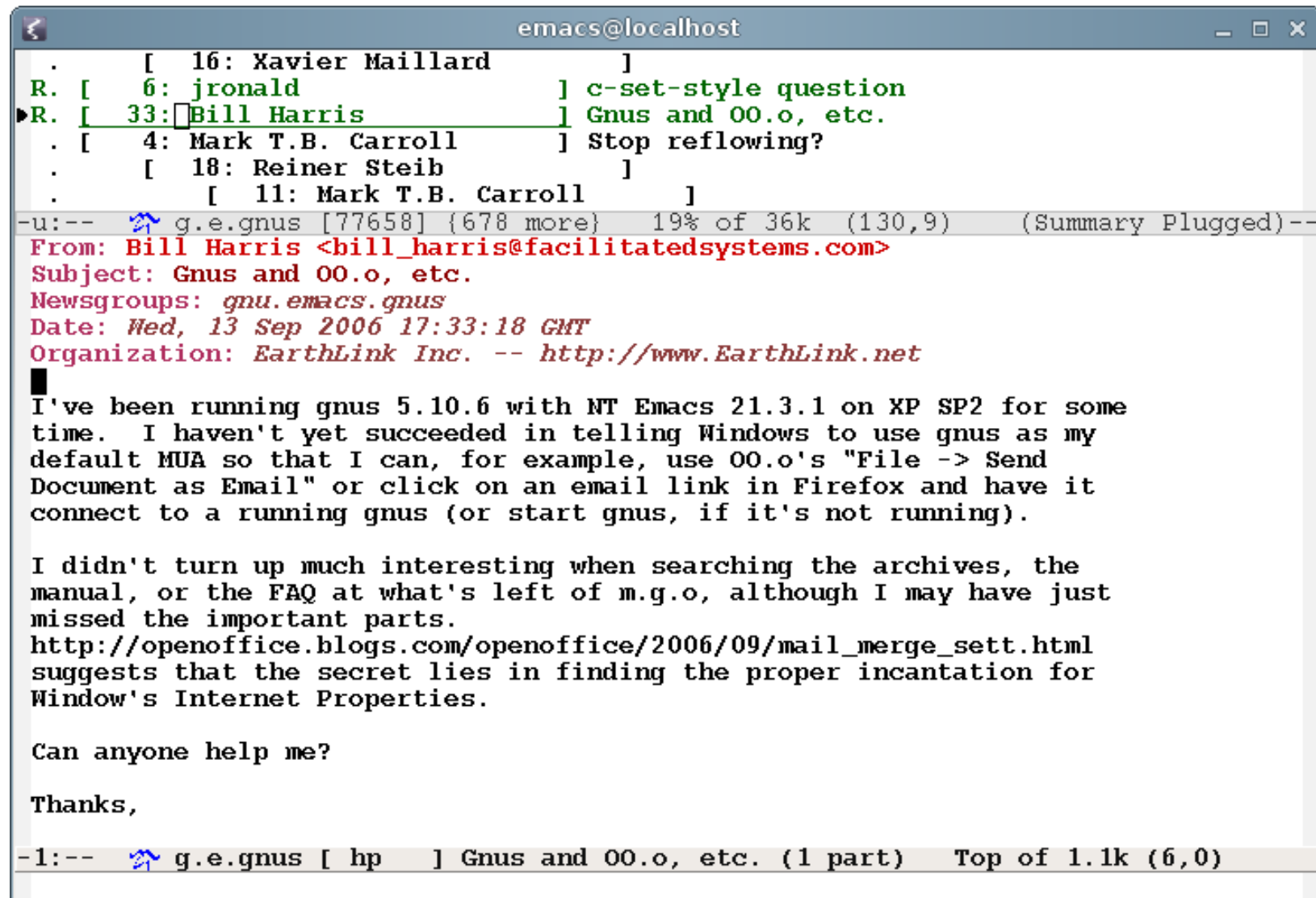
```
emacs@localhost
"An idealist is one who, on noticing that a rose smells better than a
cabbage, concludes that it will also make better soup." - H.L. Mencken
phil@phil ~ $ cat /proc/cpuinfo | head
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 15
model          : 2
model name     : Intel(R) Pentium(R) 4 CPU 2.80GHz
stepping       : 9
cpu MHz        : 2793.163
cache size     : 512 KB
physical id    : 0
siblings       : 1
phil@phil ~ $ ps -e | wc -l
93
phil@phil ~ $ █

-u:** *shell* All of 420 (16,14) (Shell:run)-----
```

Emacs is a shell

M-x shell

The many faces of Emacs



```
emacs@localhost
. [ 16: Xavier Maillard ]
R. [ 6: jronald ] c-set-style question
R. [ 33: Bill Harris ] Gnus and OO.o, etc.
. [ 4: Mark T.B. Carroll ] Stop reflowing?
. [ 18: Reiner Steib ]
. [ 11: Mark T.B. Carroll ]
-u:-- g.e.gnus [77658] {678 more} 19% of 36k (130,9) (Summary Plugged)--
From: Bill Harris <bill_harris@facilitatedsystems.com>
Subject: Gnus and OO.o, etc.
Newsgroups: gnu.emacs.gnus
Date: Wed, 13 Sep 2006 17:33:18 GMT
Organization: EarthLink Inc. -- http://www.EarthLink.net

I've been running gnus 5.10.6 with NT Emacs 21.3.1 on XP SP2 for some
time. I haven't yet succeeded in telling Windows to use gnus as my
default MUA so that I can, for example, use OO.o's "File -> Send
Document as Email" or click on an email link in Firefox and have it
connect to a running gnus (or start gnus, if it's not running).

I didn't turn up much interesting when searching the archives, the
manual, or the FAQ at what's left of m.g.o, although I may have just
missed the important parts.
http://openoffice.blogs.com/openoffice/2006/09/mail_merge_sett.html
suggests that the secret lies in finding the proper incantation for
Window's Internet Properties.

Can anyone help me?

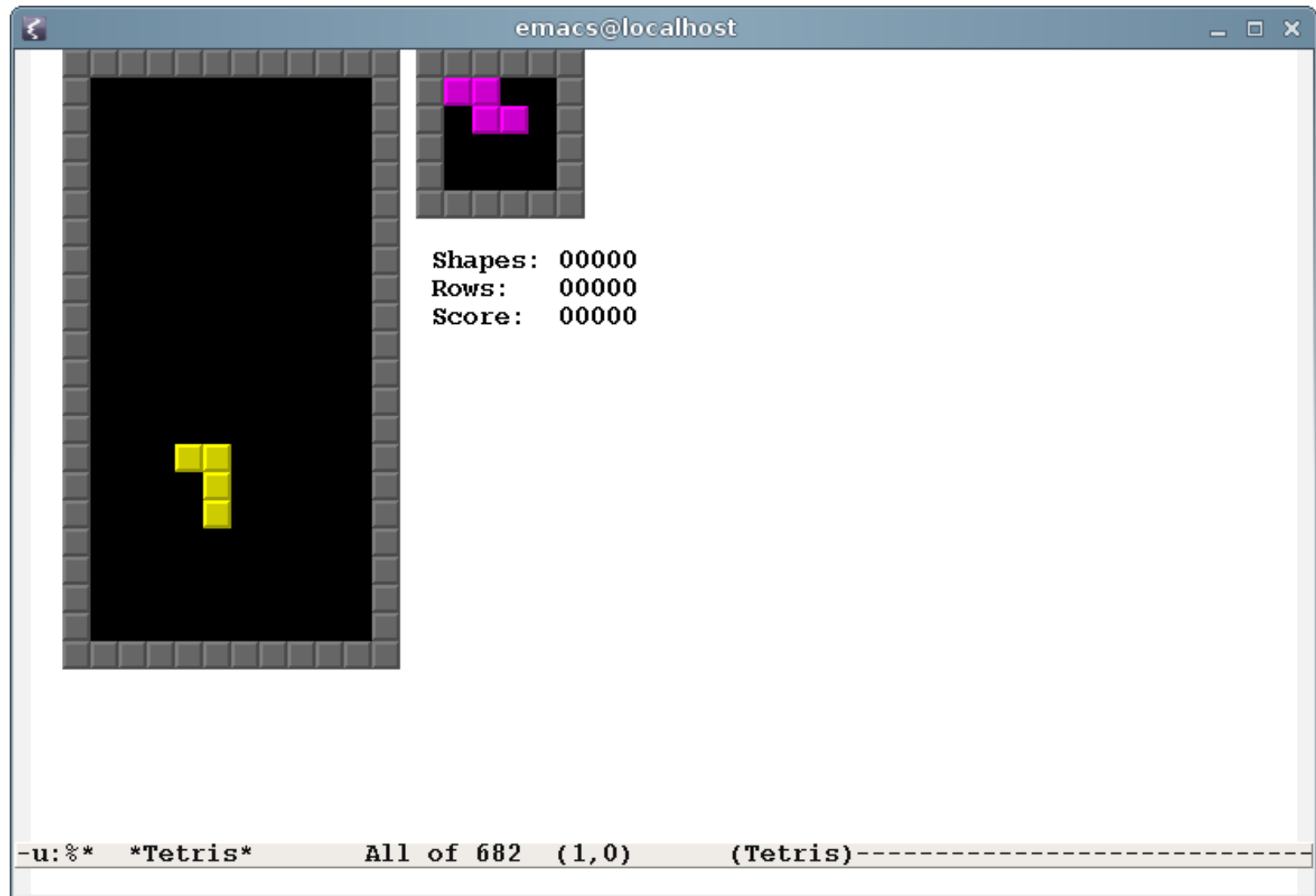
Thanks,
```

-1:-- g.e.gnus [hp] Gnus and OO.o, etc. (1 part) Top of 1.1k (6,0)

Emacs is a mail/news client

M-x gnus

The many faces of Emacs



Emacs plays tetris
M-x tetris

Why Emacs?

- Provides an integrated environment
 - Same editing commands available everywhere
 - Large set of tools available at all times
 - Move text between tasks easily

Why Emacs?

- Easy to extend
 - Lisp for customizing or adding new features
 - Extension code has the full power of Emacs
 - Dynamic environment: no restarting or recompiling
- Portable

■ Today's goal: get the flavor of Emacs

- Getting started with Emacs
- Editing tips
- Demos of useful features
- Common Emacs concepts

Examples based on
GNU Emacs 22

■ Later...

- Advanced customization
- Programming and extending Emacs with Emacs Lisp

Prerequisites (sort of)

- Emacs basic concepts
 - Files, buffers, windows, frames
- Keyboard commands
 - Key commands, prefix keys, `M-x`, the minibuffer
 - "`C-x`" means **Ctrl+x**
 - "`M-x`" means **Meta+x** or **Alt+x**
- Basic tasks
 - Opening and saving files, exiting Emacs

**Take the tutorial
to brush up:
`C-h t`**

It's all about text manipulation

- **Text in files**

- grocery lists, HTML, code, ...

- **Text outside of files**

- shell, debugger, ...

- **Text as a metaphor**

- dired, gnus, ...

Text as a metaphor: dired

M-x wdired-change-to-wdired-mode
after opening any directory

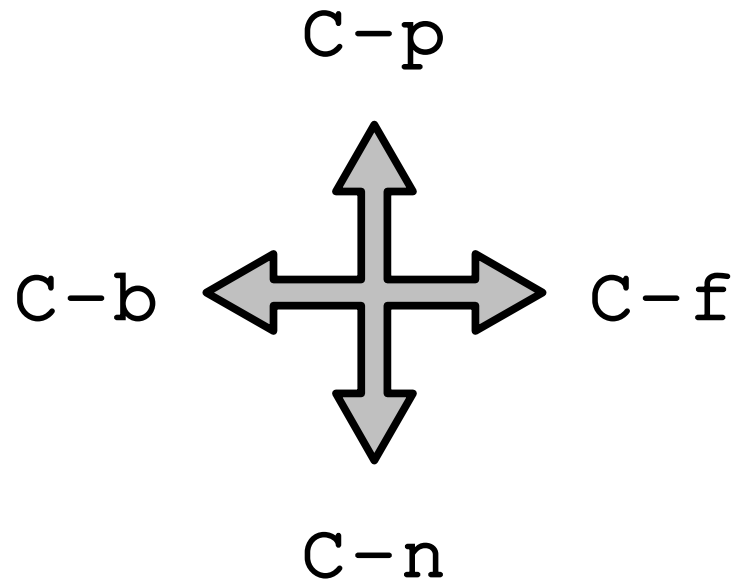
```
emacs@localhost.locald
/home/phil/projects/emacs-talk:
total used in directory 296 available 28869812
drwxr-xr-x  3 phil phil   4096 2007-02-17 17:54 .
drwxr-xr-x 12 phil phil   4096 2007-03-23 15:06 ..
-rw-r--r--  1 phil phil   8408 2007-02-17 17:54 emacs-slides
-rw-r--r--  1 phil phil 199108 2007-03-07 22:29 emacs-slides-1.odp
-rw-r--r--  1 phil phil  33740 2007-02-17 18:37 emacs-slides-2.odp
-rw-r--r--  1 phil phil  33101 2007-02-17 18:37 emacs-slides-3.odp
drwxr-xr-x  2 phil phil   4096 2007-02-17 17:54 screens

-u:--  emacs-talk      All of 516  (7,50)  (Editable Dired,
```

After editing names
in this buffer, C-x
C-s renames the
modified files

Moving around in buffers

- By character or line



Moving around in buffers

- Beginning, end of line
 - C-a, C-e
- By word
 - M-f, M-b
- By sentence
 - M-a, M-e
- By screen
 - C-v, M-v
- Beginning, end of buffer
 - M-<, M->
- Go to line #
 - M-g g

Moving around in buffers

- Move multiple lines forward, backward
 - Example: `C-u 10 C-p` (back 10 lines)
 - `C-u` prefix generalizes to other commands
- Search for text
 - `C-s`, `C-r`
- Exchange point (cursor) and mark
 - `C-x C-x`

Killing ("cutting") text

- Kill line
 - C-k
- Kill many lines
 - C-u 10 C-k (10 lines)
 - C-u C-k (4 lines)
 - C-u C-u C-k (16 lines)

Killing ("cutting") text

- Kill region
 - C-w
- Save without killing
 - M-w
- Kill sentence
 - M-k
- Kill ("zap") to next occurrence of character
 - M-z CHAR

Yanking ("pasting") text

- Yank
 - C-y
- Yank earlier killed text
 - M-y (once or more after C-y)
- The kill ring
 - Almost all commands which delete text save it for possible later retrieval

The mark

- Remembers a previous cursor position
 - `C-x C-x` to swap point (cursor) and mark
- When you...
 - `C-spc`
 - `M-<` or `M->`
 - Search for text
 - Yank text
 - Insert a buffer
- the mark is set to...
 - where you are
 - where you were
 - where you started
 - start of inserted text
 - start of inserted text

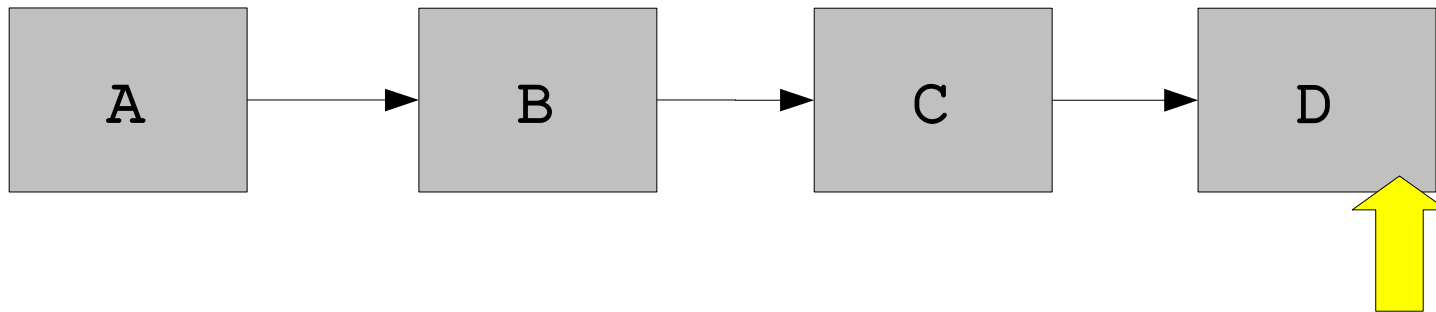
The mark

- The mark ring
 - Move to a previous mark: `C-u` `C-SPC`
- Mark and point are also used to delineate 'the region'
 - Many commands operate on the text in the region
 - Set region by setting mark, then moving point

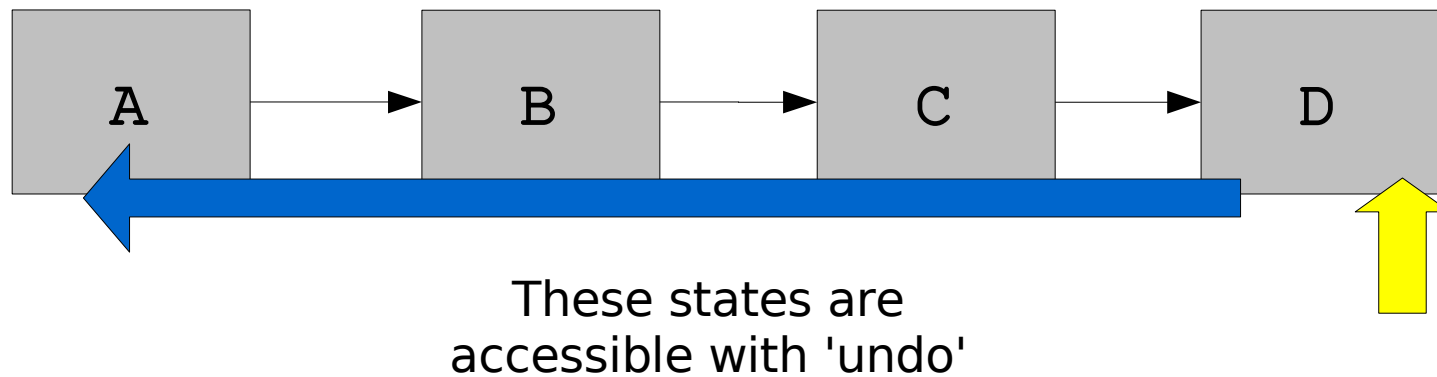
Undo

- Undo previous actions
 - C- / or C- _ or C- x u
- Undo within current region
 - C- u C- /

The undo model, illustrated



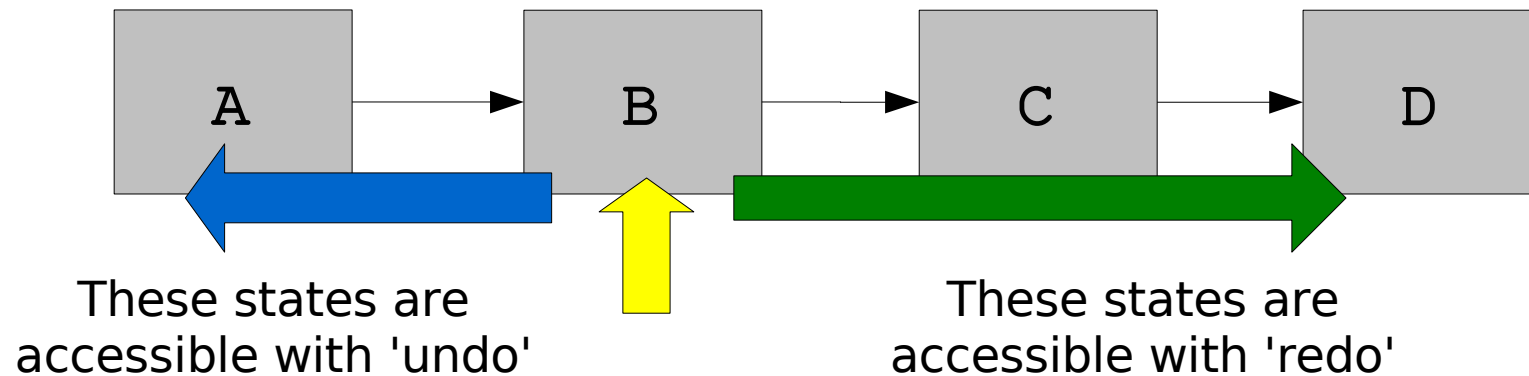
The undo model, illustrated



The undo model, illustrated

Undo some of your actions...

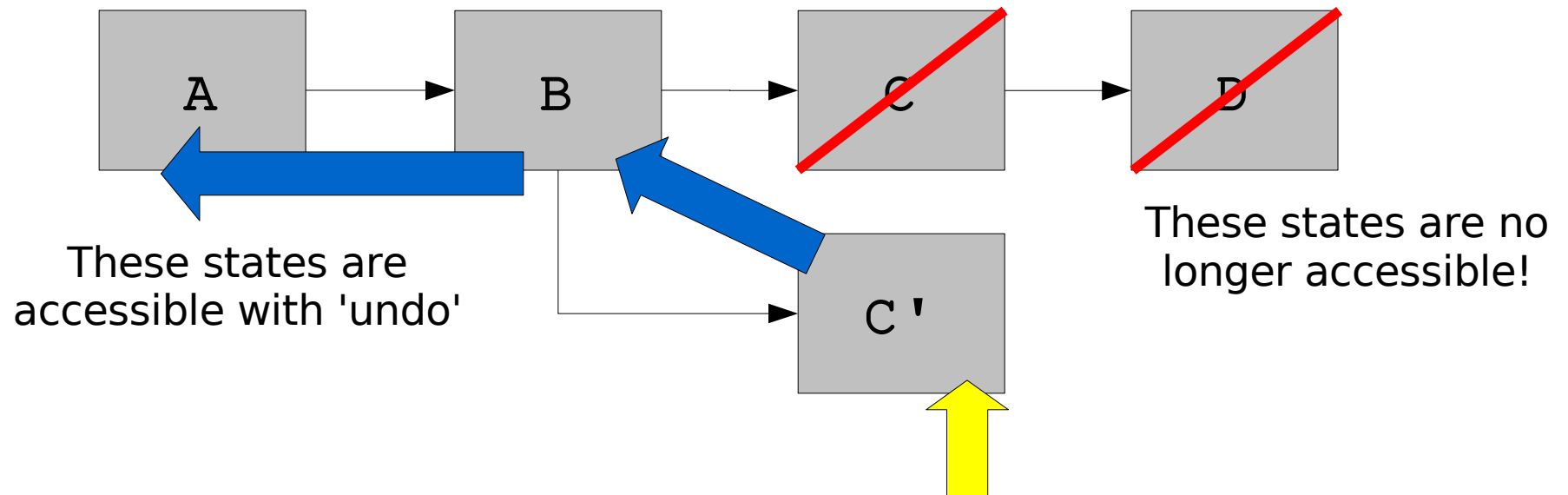
This is how most editors other than emacs work:



The undo model, illustrated

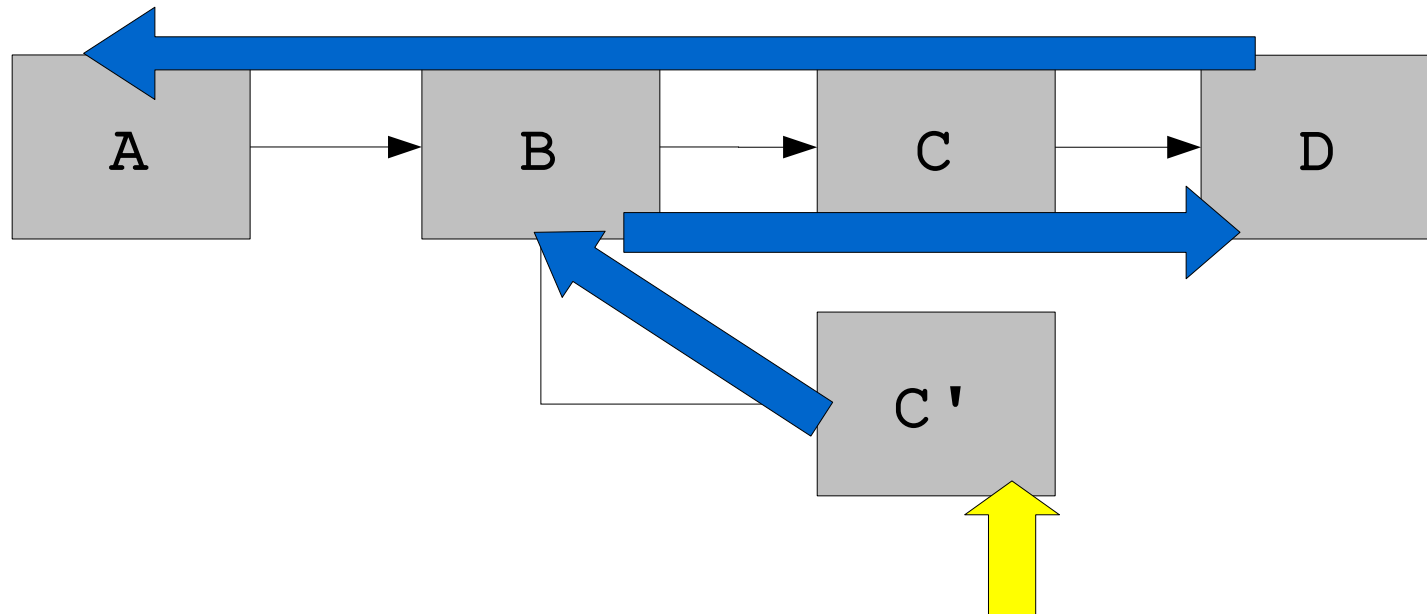
Now do something else...

This is how most editors other than emacs work:



The undo model, illustrated

How emacs handles this situation



The list of states is 'folded' so that all previous actions, including undos, are undoable

Incremental search

- Search for text (like Firefox's "find as you type")
 - `C-s text`
 - ◊ `C-s` again to find next occurrence
 - ◊ `RET` to stop at found occurrence
 - ◊ `C-g` to cancel and go back to start of search
 - `C-r` for reverse search
 - Many options available inside search;
`C-h k C-s` to learn more

Search history

- Search for previously searched string
 - C-s C-s
- Browse and edit previous queries
 - C-s then M-p, M-n

Incremental search

- Search for regular expressions
 - `C-M-s regexp`
 - Regexp describes the form of what to look for
 - Syntax may be slightly different from other REs you may have used
 - Emacs REs are a superset of Perl REs
 - `M-x re-builder` can help you test complex regexps

Searching and replacing

- Search and replace, asking for confirmation
 - `M-%` or `M-x query-replace`
- Display all lines matching RE
 - `M-x occur`

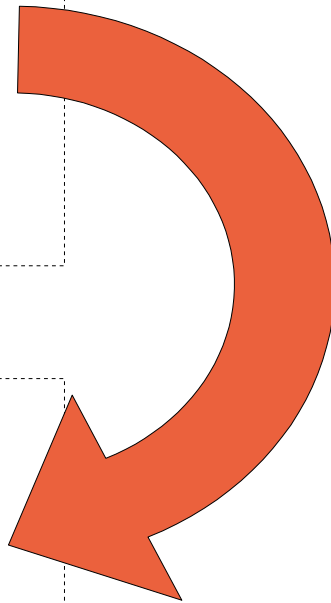
RE search and replacement

- `M-x replace-regexp`
- Replacement text can depend on found text!
- Replacement text gets these substitutions:
 - `\&` (the matched string)
 - `\1`, `\2`, etc. (references to parts of matched string)
 - `\#` (number of matched lines so far)
 - `\?` (prompt user for what to enter)
 - `\, (lisp-expression ...)`

RE replacement example

```
Bill Gates  
Steve Jobs  
Eric Schmidt  
Larry Ellison
```

```
GATES, Bill  
JOBS, Steve  
SCHMIDT, Eric  
ELLISON, Larry
```



```
M-x replace-regexp  
  \(\w+\) \(\w+\)  
      with  
  \, (upcase \2), \1
```

Integration with useful tools

- Shell

- `M-x shell`

- Compile (invoke make)

- `M-x compile`

- Debug

- `M-x gdb`

Integration with useful tools

- Grep

- `M-x grep`, `M-x rgrep`

- Man page reader

- `M-x woman`

- Invoke shell commands

- `M-x shell-command`,
`M-x shell-command-on-region`

...and then some

- Calculator
 - `M-x calc`
- Calendar
 - `M-x calendar`
- Moon calendar
 - `M-x phases-of-moon`

More helpful features

- TRAMP: open remote files over SSH
 - `C-x C-f /user@host:~/remote/file`
- VC: automatically deal with CVS, SVN, etc.
 - `M-x vc-next-action` to commit modified file
 - `M-x vc-diff` to view changes to current file
- etags: name search/completion for source code

Emacs server

- Use a single Emacs session for **all** editing
- Do this once: `M-x server-start`
 - or put `(server-start)` in your `.emacs` file
- To edit a file:
 - `prompt% emacsclient file`
 - File opens in existing Emacs frame
 - `C-x #` when done editing

Macros

- Remembers a fixed sequence of keys for later repetition
- Start recording macro: C-x (
- Stop recording macro: C-x)
- Replay macro: C-x e

Macro example

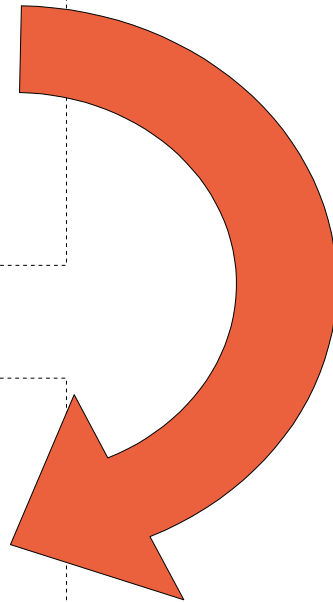
Define macro:

```
M-d C-d M-u , [SPC]  
C-y C-n C-a
```

"Remove first word and space,
uppercase next word, insert comma
and space afterward, reinsert first
word, move to beginning of next line"

```
Bill Gates  
Steve Jobs  
Eric Schmidt  
Larry Ellison
```

```
GATES, Bill  
JOBS, Steve  
SCHMIDT, Eric  
ELLISON, Larry
```



Run macro repeatedly:

```
C-x e e . . .
```

Narrowing

- Restricts view/editing in a buffer to a certain region
 - `C-x n n` or `M-x narrow-to-region` to narrow to region
 - `C-x n w` or `M-x widen` to restore ('widen')

Registers

- Store current window configuration
 - `C-x r w REGISTER`
- Restore window configuration
 - `C-x r j REGISTER`
- Registers can also store positions, text, numbers, file names...

REGISTER may be any letter or number

Prefix arguments

- Sometimes used to indicate repetition
 - `C-u 10 C-f` (forward 10 characters)
 - `C-u C-o` (make 4 new lines)
- Sometimes modify following command
 - `C-/` (undo) vs. `C-u C-/` (undo within region)
 - `M-x shell` vs. `C-u M-x shell`
- A command's documentation (`C-h f` or `C-h k`) describes the effect of the prefix argument, if any

Major modes

- Alters behavior, key bindings, and text display
- Switch mode in existing buffer:
 - `M-x java-mode`
 - `M-x python-mode`
 - `M-x fundamental-mode`
- Or, use another command to create buffer:
 - `M-x shell`
 - `M-x dired`

Language major mode features

- Language-specific indentation, syntax coloring
- Language-specific features:
 - Lisp: commands for manipulating s-expressions
 - Python: commands for (un)indenting blocks
 - HTML: insert/close tags; preview in web browser
 - Modes can define or redefine keys

Minor modes

- Extra functionality you can turn on or off
 - Any number of minor modes may be active at once
 - Independent of major mode functionality
- `M-x auto-fill-mode`
- `M-x flyspell-mode`
- `M-x follow-mode`

Global minor modes

- Offer completions for buffers, commands, etc.
 - `M-x icomplete-mode`
- Show all buffer names on `C-x b:`
 - `M-x iswitchb-mode`

Minibuffer input

- Common features whenever Emacs prompts you to enter something
 - Most buffer editing, movement commands work
 - Browse previous inputs with `M-n`, `M-p`
 - Tab-completion is often available
- `M-x eval-expression`, `M-x find-file`, `M-x switch-to-buffer`, ...

Getting help with Emacs

- Help with key
 - C-h k
- Help with function
 - C-h f
- Help with mode
 - C-h m
- Show key bindings
 - C-h b
- Help about help
 - C-h C-h

Getting help with Emacs

- Apropos (search for command)
 - `C-h a`
- Help with prefix key
 - `C-h` (after prefix key)
- Manuals
 - `M-x info`, then select emacs or efaq

“In the event of an emergency”

- Cancel command
 - C-g
- Undo!
 - C-/ or C-_
- What did I just do?
 - M-x view-lossage

Common problems

- Delete not deleting?
 - `M-x normal-erase-is-backspace-mode`
- Keys with `M-` not working?
 - Use `ESC` instead
 - `ESC x` instead of `M-x`
 - `ESC C-t` instead of `C-M-t`

Migrating to Emacs

- From Windows applications
 - `M-x cua-mode`: recovers `C-z`, `C-x`, `C-c`, `C-v` for their usual purposes
- From vi/vim
 - `M-x viper-mode`

Resources

- Emacs on Athena
 - <http://web.mit.edu/olh/Emacs/>
- Emacs reference card
 - <http://web.mit.edu/olh/Emacs/Refcard.pdf>

Invoking or installing Emacs

- **emacs21 on Athena:** `athena% emacs`
- **emacs22 on Ubuntu/Debian:**
`apt-get install emacs-snapshot-gtk`
- **emacs22 on Gentoo:** `emerge emacs-cvs`
- **emacs on Windows:**
 - `http://ourcomments.org/Emacs/EmacsW32Util.html`

■ **Bonus: Being Unproductive With Emacs**

- `M-x tetris`, `M-x hanoi`, `M-x doctor`, ...

■ **Next time: Emacs lisp**

- Customizing how Emacs works
- Writing new functions, commands, and modes
- Manipulating text programmatically
- Altering behavior of existing commands

Being Productive With Emacs

Part 2



Phil Sung

`sipb-iap-emacs@mit.edu`

`http://stuff.mit.edu/iap/emacs`

Special thanks to Piau Na and Arthur Gleckler

Previously...

- Emacs as an editor
 - Useful features
 - Motifs in emacs
 - Learning more

Previously...

- Acquiring Emacs
 - Already installed on Athena (v.21)
 - Ubuntu: emacs-snapshot-gtk package (v.22)
 - Gentoo: emacs-cvs package (v.22)
 - Windows: EmacsW32 installer

Previously...

- Learning more about emacs
 - Look up an existing function or key
 - C-h f, C-h k
 - Apropos (search for commands)
 - C-h a
 - Help about help facilities
 - C-h C-h

Previously...

- Learning more about Emacs
 - Emacs tutorial
 - `C-h t`
 - Emacs manual
 - `M-x info`, select emacs

Previously...

- If you're stuck...
 - Cancel: C-g
 - Undo: C-/ or C-_

Today

- Customizing Emacs
- Lisp basics
- Defining a new command

Customizing Emacs

- Almost every aspect of the editor can be customized
 - More fine-grained control than major/minor modes
 - In general, use `M-x eval-expression`
 - Show trailing whitespace on lines
`(setq show-trailing-whitespace t)`
 - Show column numbers in mode line
`(column-number-mode t)`

Customizing Emacs

- Customization buffer: `M-x customize`
 - Browse, point and click for customization options
 - No elisp necessary, but capabilities are limited

Review: running elisp code

- Evaluate elisp with `M-x eval-expression`
 - Example function call: `(+ 2 4 6)`
- Get or set variables:
 - `sentence-end-double-space`
 - `(setq inhibit-startup-message t)`
- Sometimes these correspond to commands:
 - `(forward-char)` is same as `C-f` or `M-x forward-char`

Writing elisp code

- Use the `*scratch*` buffer for playing with elisp
 - `C-x C-e` to evaluate
 - Move to `~/ .emacs` for permanent changes

Your .emacs file

- `C-x C-f ~/ .emacs`
- Elisp code here is loaded when Emacs starts
 - Run any valid lisp code here
 - Set variables, keybindings to your liking
 - Define your own commands
- `M-x customize` works by adding code to your .emacs

Why bother with elisp?

- Macros record and play back key sequences
 - Start recording macro: `C-x (`
 - Stop recording macro: `C-x)`
 - Execute last macro: `C-x e`
- Great for automating tedious tasks
 - `C-x e e e ...`
 - `C-u 100 C-x e`

Macro example

```
6.00 12 programming
6.001 15 sicp
6.002 15 circuits
6.003 15 linear-systems
6.004 15 digital
6.011 12 signal-proc
```

```
6.00 programming
6.001 sicp
6.002 circuits
6.003 linear-systems
6.004 digital
6.011 signal-proc
```

Let's remove this
column

M-f M-f M-d C-n C-a repeatedly

Why elisp?

- Macros only repeat canned key sequences
- Sometimes you need:
 - Calculations
 - Control flow
 - User interaction
 - Additional features
 - Maintainability

Elisp is...

- an implementation language
- a customization language
- an extension language

Elisp for implementation

- Example: `M-x calc`
 - `C-h f` to see where `calc` is defined
 - RET on filename in help buffer to view source code

Elisp for customization

- Set variables and options
- Persistent customizations can go in .emacs
- Compare to M-x customize

Elisp for extensions

- Alter behavior of existing commands
- Define your own commands, functions
- Define new modes

Why elisp?

- It's the implementation language
- Dynamic environment
 - No need to recompile/restart emacs
 - Easily override or modify existing behaviors
- Simple one-liners are sufficient to do a lot!

Getting started

- Similar to lisp and scheme
- Use `*scratch*` buffer as a temporary work space
 - or activate `lisp-interaction-mode` anywhere
 - `C-x C-e` after an expression to evaluate it
 - or use `M-x eval-expression (M-:)`
- Example: setting a variable
 - `(setq undo-limit 100000)`

Getting started

- Evaluating an expression can mean
 - Performing some computation/action
 - Displaying the value of a variable
 - Defining a function for later use

Basic elisp

- These are expressions (“atoms”)
 - 15
 - “Error message”
 - best-value
- These are also (“compound”) expressions
 - (+ 1 2)
 - (setq include-all-files t)

Setting variables

- Set variable by evaluating
`(setq undo-limit 100000)`
- i.e. `do M-: (setq ...) [RET]`
- Read variable by evaluating `undo-limit`
- i.e. `do M-: undo-limit [RET]`
- Find out more about any variable with `C-h v`

Common customizations

- Configuration options
- Set your own keybindings

Configuration options

- Some customizations are done by setting variables
 - `(setq undo-limit 100000)`
 - `(setq enable-recursive-minibuffers t)`
 - `(setq fill-column 80)`

Configuration options

- Other options are exposed as their own functions
 - `(menu-bar-mode nil)` (Hide menu bar)
 - `(icomplete-mode)`
(Show completions continuously)
 - `(server-start)` (Start emacs server)

More about variables

- Many variables are boolean
 - Usually a distinction is only made between `nil` and non-`nil` values (e.g. `t`)
- Look in function documentation to see which variables can alter the function's behavior
- `C-h v` to get documentation for a variable

Key bindings

- We've seen two ways to invoke commands
 - `C-x n w` (key invocation)
 - `M-x widen` (M-x invocation, or invoking by name)
- Emacs *binds* each key to a command in a *keymap*
 - A keymap can be specific to a mode or feature
 - Bindings may be changed at any time

Customizing key bindings

- `(global-set-key
 [f2]
 'split-window-horizontally)`
- `(global-set-key "\C-o" 'find-file)`
- `(global-set-key "\C-x\C-\\ "
 'next-line)`

binds to **C-x C-**

Customizing key bindings

- A binding can be set to apply only in a particular mode
 - `(define-key text-mode-map`
 `"\C-c p"`
 `'backward-paragraph)`

binds to **C-c p**

Keybindings

- What keys can you assign?
 - Reserved for users:
 - `C-c [letter]`
 - Reserved for major and minor modes:
 - `C-c C-[anything]`
 - `C-c [punctuation]`
 - `C-c [digit]`

Calling commands

- Any command you use can be invoked programmatically by elisp
 - Often, `M-x my-function` is accessible as `(my-function)`
 - For key commands, look up the full name first
- Use commands as building blocks for more complex behaviors

Hooks

- Specify a custom command to run whenever a particular event occurs, e.g.
 - when a particular mode is entered
 - when any file is loaded or saved
 - when a file is committed to CVS

Hooks

- `(add-hook`
 `'vc-checkin-hook`
 `'(lambda ()`
 `(send-email-to-group)))`

Hooks

- ```
(add-hook 'java-mode-hook
 '(lambda () (setq indent-tabs-mode t)
 (setq tab-width 4)
 (set-fill-column 80)))
```

# Hooks

- General template

- (add-hook 'name-of-hook  
 ' (lambda () (do-this)  
 (do-that)  
 (do-the-other-thing) ) )

# Hooks

- To find available hooks:
  - Every major mode has a hook
  - `M-x apropos-variable` and search for "hook"



# Defining your own functions

- ```
(defun function-name (arg1 arg2 ...)  
  "Description of function"  
  (do-this)  
  (do-that)  
  (do-the-other-thing) )
```
- Invoke with:

```
(function-name one two ...)
```

Strategy for making functions

- Find key commands that would have desired result
- Replace key commands with elisp function calls

A simple function

- ```
(defun capitalize-backwards ()
 "Capitalize last letter of a
word."
 (backward-word)
 (forward-word)
 (backward-char)
 (capitalize-word 1))
```

# Not every function is a command

- Functions need arguments:
  - `(defun square (x) (* x x))`  
`(square 5) ==> 25`
- Commands don't say what arguments to substitute
  - `M-x square ==> ??`
- *Interactive* specification needed to say what arguments to fill in

# A simple command

- ```
(defun capitalize-backwards ()  
  "Capitalize last letter of a  
word."  
  (interactive)  
  (backward-word)  
  (forward-word)  
  (backward-char)  
  (capitalize-word 1) )
```

Problem

- This command moves the cursor
 - This can be distracting if the user isn't expecting it

Restoring the cursor

- ```
(defun capitalize-backwards ()
 "Capitalize last letter of a
word."
 (interactive)
 (save-excursion
 (backward-word)
 (forward-word)
 (backward-char)
 (capitalize-word 1)))
```

# Useful functions

- `(point)`
- `(point-max)`
- `(current-buffer)`
- `(message "This is the answer: %s"`  
          `answer)`



# Local variables

- ```
(let  ( (a new-value)
        (b another-value)
        ... )
      (do-something)
      (do-something-else) )
```

Example: counting word length

- ```
(defun word-length ()
 "Prints the length of a word."
 (interactive)
 (save-excursion
 (backward-word)
 (let ((a (point)))
 (forward-word)
 (let ((b (point)))
 (message "Word is %d letters"
 (- b a))))))
```

# Getting help with elisp

- Manuals
  - `M-x info`, then select elisp or eintr
- Learning by example
  - Function documentation (`C-h f` or `C-h k`)  
always gives a link to the function's source code

# Next week...

- Control flow
- User interaction
- Commands for manipulating text
- Other extension methods

# Being Productive With Emacs

## Part 3



Phil Sung

`sipb-iap-emacs@mit.edu`

`http://stuff.mit.edu/iap/emacs`

Special thanks to Piau Na and Arthur Gleckler

# Previously...

- Customizing emacs
  - Setting variables
  - Key bindings
  - Hooks
- Extending emacs with new elisp procedures
  - Simple text manipulation
  - Interactive specifications

# This time...

- Extending emacs
  - Advising functions
  - Foundations of elisp
  - More about interactive specifications
  - Manipulating text in emacs
  - Creating a major mode

# Advice

- Used to add to any existing function
- Pieces of advice are modular
- Advice vs. hooks
- Advice can be dangerous!



# Advice example: previous line

- When `next-line-at-end` is set to `t`, `next-line` on last line of buffer creates a new line
- Create analagous behavior for `previous-line` at beginning of buffer
  - When on first line of buffer, insert a newline before moving backwards

# Advice example: previous-line

```
(defadvice previous-line
 (before next-line-at-end
 (&optional arg try-vscroll))
 "Insert new line when running previous-line
 at first line of file"
 (if (and next-line-add-newlines
 (save-excursion (beginning-of-line)
 (bobp))
)
 (progn (beginning-of-line)
 (newline))
))
```

# Advice syntax

```
(defadvice function-to-be-modified
 (where
 name-of-advice
 (arguments-to-original-function))
 "Description of advice"
 (do-this)
 (do-that))
```

where can be before, after, or around

# Enabling advice

- `(ad-enable-advice 'previous-line  
                    'before  
                    'next-line-at-end)`
- `(ad-disable-advice 'previous-line  
                      'before  
                      'next-line-at-end)`

# Activating advice

- `(ad-activate 'previous-line)`
  - Do this every time advice is defined, enabled, or disabled
- `(ad-deactivate 'previous-line)`

# Ways to use advice

- `before`: Add code before a command
- `after`: Add code after a command
- `around`: Make a wrapper around invocation of command
  - Useful for executing the command more than once or not at all
  - You can also modify the environment

# Example: around-advice

- ```
(defadvice previous-line  
  (around my-advice)  
    "Conditionally allow previous-line."  
    (if condition1  
        ad-do-it))
```

Foundations of elisp

- Data types in elisp
- Control flow

Data types

- Lisp data types
 - integer, cons, symbol, string, ...
 - Cursor position represented as integer
- Emacs-specific data types
 - buffer, marker, window, frame, overlay, ...

Control flow

- `(progn (do-this)
 (do-something-else))`
- All forms are evaluated, and the result of the last one is returned
 - Useful in e.g. `(if var (do-this) (do-that))` where a single form is required
 - Some control structures like `let` have an *implicit progn*

Control flow

- `(if condition
do-this-if-true
do-this-is-false)`
- `(cond (condition1 result1)
(condition2 result2)
...
(t default-result))`

Control flow

- `or` returns the first non-nil argument, or nil
 - Short-circuit evaluation
 - ```
(defun frob-buffer (buffer)
 "Frob BUFFER (or current buffer if it's nil)"
 (let ((buf (or buffer
 (current-buffer))))
 ...))
```
  - ```
(defun frob-buffer (buffer)
  "Frob BUFFER or prompt the user if it's nil"
  (let ((buf (or buffer
                  (read-buffer "Prompt: "))))
    ...))
```

Control flow

- `and` returns the last argument if all arguments are non-nil
 - Short-circuit evaluation
 - `(and condition1 condition2 (do-this))`
 - equivalent to:
`(if (and condition1 condition2)
 (do-this))`

Control flow

- `(while condition
 (do-this)
 (do-that)
 ...)`

Dynamic scoping

- ```
(defun first (x)
 (second))
(defun second ()
 (message "%d" x))
```
- What does `(first 5)` do?
  - Dynamic scoping: 5
  - Lexical scoping: a global value of `x` is found

# Using dynamic scoping

- Setting variables can alter function behavior
  - No need to pass extra arguments through the chain of function calls
- ```
; text search is case-sensitive  
; when case-fold-search is nil  
(let ((case-fold-search nil))  
  (a-complex-command))
```

 - Any searches done inside `a-complex-command` are altered to be case sensitive

Interactive forms

- Recall: *interactive* tells elisp that your function may be invoked with `M-x`, and specifies what arguments to provide
- The provided arguments may be:
 - The result of prompting the user (e.g. for a buffer)
 - Something in the current state (e.g. the region)

Interactive forms

- Example: find-file (C-x C-f)
 - `(find-file FILENAME)` opens FILENAME in a new buffer
 - `M-x find-file` or `C-x C-f` prompts user for a filename, then calls `(find-file ...)` with it
- Interactive forms make functions more flexible, allowing code reuse

Interactive forms

- Place any of the following at the top of your function
- Pass no arguments
 - `(interactive)`
- Prompt user for a buffer to provide
 - `(interactive "bSelect a buffer: ")`
 - Like how kill-buffer works

Interactive forms

- Prompt user for a file to provide
 - `(interactive "fFile to read: ")`
 - Like how find-file works
- Provide nil
 - `(interactive "i")`

Interactive forms

- Provide position of point
 - `(interactive "d")`
- Provide positions of point and mark, first one first
 - `(interactive "r")`
 - Example: indent-region

Interactive forms

- Provide prefix argument
 - `(interactive "p")`
 - Example: previous-line

Example: interactive forms

- ```
(defun count-words-region (beginning end)
 "Print number of words in the region."
 (interactive "r")
 (save-excursion
 (let ((count 0))
 (goto-char beginning)
 (while
 (and
 (< (point) end)
 (re-search-forward "\\w+\\W*" end t))
 (setq count (1+ count)))
 (message "Region contains %d word%s"
 count
 (if (= 1 count) "" "s"))))))
```

# Interactive forms

- interactive can provide multiple arguments to your function
  - Separate different specifiers with a newline "\n"
  - Example:

```
(interactive
 " bSelect buff er: \n fSelect file : ")
```



# Reading text

- `char-after, char-before`
- `(buffer-substring start end)`
- `(thing-at-point 'word)`  
'line, 'whitespace, etc.

# Locating the cursor

- `point`
- `point-min, point-max`
- `bobp, eobp, bolp, eolp`
- `current-column`

# Moving around in text

- `goto-char`
  - Example: `(goto-char (point-min))`
- All your favorite keyboard-accessible commands (`C-f`, `C-b`, etc.)
- `save-excursion`
  - Saves current buffer, point and mark and restores them after executing arbitrary code

# Modifying text

- `(insert "string")`
- `(insert-buffer buffer)`
- `(newline)`
- `(delete-region start end)`

# Searching text

- `(search-forward "text" LIMIT NOERROR)`
  - LIMIT means only search to specified position
  - When no match is found, nil is returned if NOERROR is t
- `(re-search-forward "regexp"  
LIMIT  
NOERROR)`

# Manipulating buffers

- `get-buffer-create`
  - Retrieves a buffer by name, creating it if necessary
- `current-buffer`
- `set-buffer`
- `kill-buffer`

# Manipulating buffers

- Many functions can either take a buffer object or a string with the buffer name
- For internal-use buffers, use a name which starts with a space

# Getting user input

- `read-buffer`
- `read-file`
- `read-string`
- `etc.`



# Finding the right functions

- Many functions are only intended to be called interactively
  - `M-<` or `beginning-of-buffer` sets the mark and prints a message
  - To move to the beginning of the buffer, use `(goto-char (point-min))` instead
- Function documentation contains warnings about lisp use

# Local variables

- Variables can be either global or local to a buffer
  - Example: `fill-column`
  - `make-local-variable`
- Default values
  - Example: `default-fill-column`

# Defining a new major mode

- A major mode is defined by a procedure which:
  - Sets `'major-mode`
  - Sets a keymap
  - Runs associated hooks
  - Sets local variables
- Lots of code reuse between modes
  - Usually, invoke another mode command first, then tweak keybindings, etc. (e.g. C mode)

# Defining a new major mode

- The `define-derived-mode` macro does most of these things for you
  - Inherits settings from another major mode:
  - ```
(define-derived-mode  
    new-mode  
    parent-mode  
    name-of-mode  
    . . . )
```

Example: major mode

- ```
(define-derived-mode
 sample-mode
 python-mode
 "Sample"
 "Major mode for illustrative purposes."
 (set (make-local-variable
 'require-final-newline)
 mode-require-final-newline))
```
- The macro defines M-x sample-mode
  - It also registers sample-mode-map, sample-mode-syntax-table, etc.

# Example: major mode

- Now we define sample-mode-map:
  - ```
(defvar sample-mode-map
  (let ((map (make-sparse-keymap))
        (define-key map "\C-c\C-c"
                    'some-new-command)
        (define-key map "\C-c\C-v"
                    'some-other-command)
        map)
    "Keymap for `special-mode'.")
```
- Keys defined here take precedence over globally defined keys

Next steps

- Making a new major mode
 - ??-mode-syntax-table
 - *font lock* and font-lock-defaults to control syntax highlighting

Next steps

- Many emacs applications use buffers to interact with the user
 - Use *overlays* or *text properties* to make 'clickable' regions

Learning more about elisp

- Elisp tutorial
 - `M-x info`, select "Emacs Lisp Intro"
- Elisp manual
 - `M-x info`, select "elisp"
- Emacs source code
 - `C-h f` or `C-h k` to view function documentation;
includes link to source code