EE324 – Applications Programming for Embedded Systems

Lecture 01(a)

Review of Unix Basics

1.1 - Introduction to Unix

1.2 - Unix Shells

1.3 - Basic Unix Commands

1.4 - Unix Editors

1.5 - Files & Directories

1.6 - Secturity & Permissions

Unix Accounts

- One must have an “account” to use a Unix computer.
  - To share resources, need to tell users apart.
- Username (public) and password (private).
- You can only access the resources that are specified by your account information.
  - Accounts track, control, and limit user activity.
- There is at least one super user account in a system usually named “root”, who has absolute power over the system. (On Microsoft Windows NT/2000/XP, this account is usually named “administrator”.)

Log out

- When you're done, don't forget to logout!

  obelix > exit

  obelix > logout

Some Basic Commands

- who: Who is using the system.

```
obelix > who
katchab ttyp0      Aug 11 08:47
scott      tty02      Aug 10 11:01
jenny      tty03      Aug 10 07:21
```

- who am i: Who am I.

```
obelix > who am i
katchab ttyp0      Aug 11 08:47
```

Some Basic Commands

- ls: List the files under current directory

```
obelix > ls
readme          cs211.2.ppt  cs211.ppt.gz  notes.zip
cs211.1.ppt     cs211.3.ppt  make/         shell/
```

- cat: Display the content of a file

```
obelix > cat readme
Unix is easy!
obelix >
```

The Unix Philosophy

- The Unix User
  - Wants to use the computer to do things
  - Doesn't want the computer to do things for them
    - They are willing to learn to make it work
    - They don't need their hands held

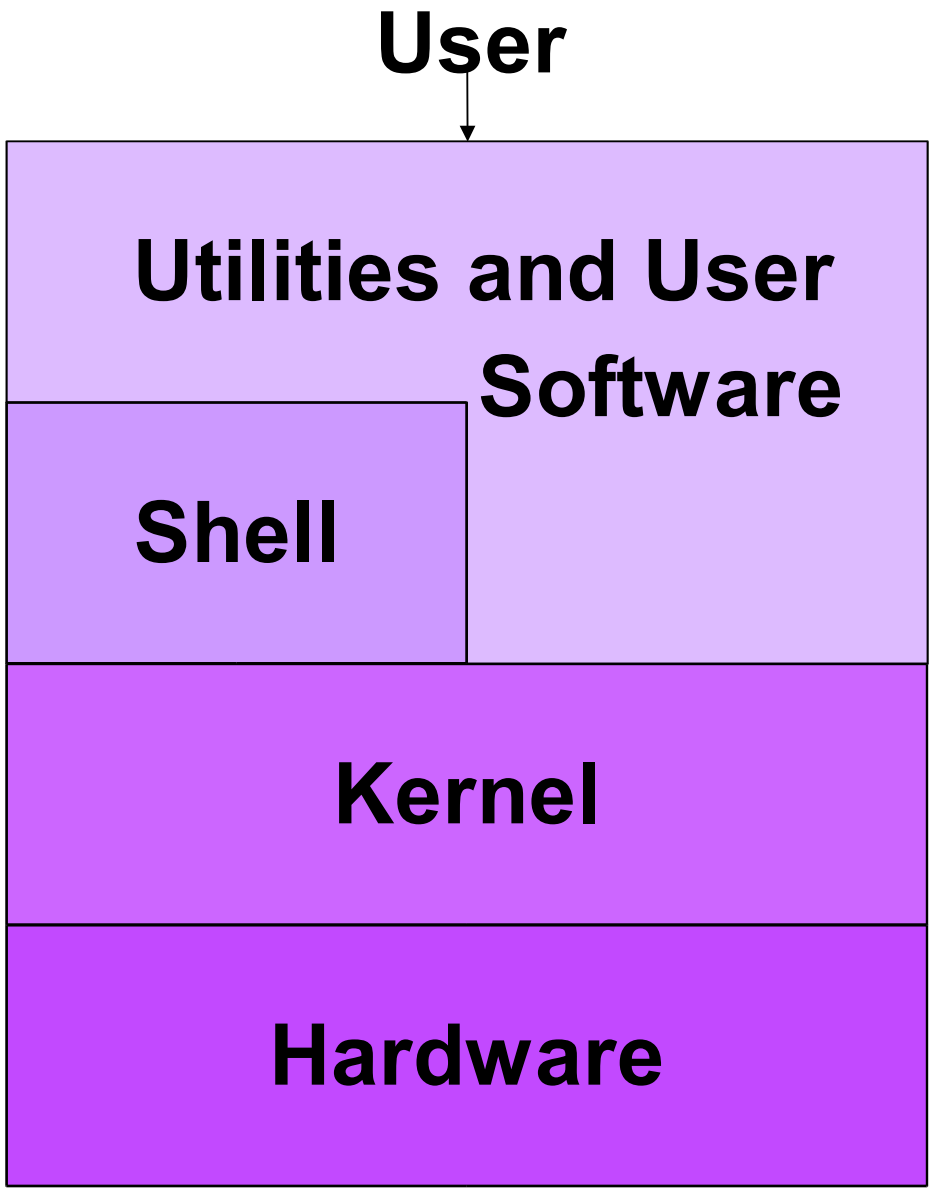
- The Unix Approach
  - Give the users the tools they need
  - They'll get the job done without having to be shown how

The Unix Philosophy

- The Unix Tools
  - Keep each tool simple
  - Have each tool do one thing, and do that one thing really well
  - Keep tools terse and not too talkative
  - More complex tasks can be accomplished by combining tools together in scripts or pipelines

  - Originally, input and output to workstations were slow and tedious, and this approach made things faster and more efficient.

Structure of a Unix System



Parts of a Unix Operating System

- Kernel
  - Manages the processes and resources
  - Controls and hides the hardware
- Shell
  - An interface between users and the kernel
  - A command line interpreter (CLI)
- Utilities are standard tools/applications
  - They are used so often that they become a part of Unix
  - “ftp”, “ssh” and “pine” are Unix utilities, for example

A Word on Linux

- In 1991, Linus Torvalds wrote the Linux kernel
  - As a class project while a student at the University of Helsinki in Finland
- Numerous programmers have worked on it
  - It's a popular Unix-like operating system now
    - Started with hobbyists and at universities
    - Growing popularity in corporations and elsewhere
  - Recently estimated at 20% of PC server market
- Achieved its goal of POSIX compliance

A Word on Linux

- Now available for many architectures
  - x86, PowerPC, SPARC, SGI Indy, HP PA-RISC, DEC Alpha, IA64 (eventually), …
- Growing software base
  - Office suites, desktops, server software, games, ...
- Has become the predominant Unix in the microcomputer world
  - Much more popular than Free/Open/NetBSD, Solaris x86, ...
- Still a "free" operating system
  - Mostly under GNU General Public License (GPL or "Copyleft")
  - Different distributions (Redhat, SUSE, and many others)
  - Check it out at: http://www.linux.org

Unix Shells

Unix Shells

- Command Line Interpreter
  - once logged in, login gives control to a shell
  - it prompts for input, then parses, interprets, finds and executes the commands you type
  - similar to MS-DOS's COMMAND.COM, but more sophisticated and more user friendly
- A High-Level Programming Language
  - shell script is a program contains a series of commands
  - you can let the system perform those commands by typing the file name of the script
  - similar to .BAT batch files under MS-DOS, but again much more sophisticated

Shells vs. Graphical User Interfaces (GUIs)

- GUIs are more friendly to beginners
  - lead you by the hand
  - “point and click” interface requires little experience
- Shells are often better for experienced users
  - shells tend to be faster, more efficient, and flexible
    - fewer steps to do things
    - do not need to change input devices (keyboard vs. mouse and keyboard)
  - but, you must know command names and syntax
- Most modern Unix systems offer both a GUI and a Shell interface
  - often have many choices

Unix Shells

- Many shells to choose from ...
- **sh: The Bourne Shell**
  - the original Unix shell
  - S.R. Bourne designed it at Bell Labs
  - not very "user friendly", but good for programming
  - sh or a reasonable facsimile comes packaged with virtually every Unix system
- **csh: The C-shell**
  - a shell whose syntax is more "C"-like
  - command history and job control
    - make it very popular as a CLI
  - comes with most Unix systems

Unix Shells

- **tcsh: The T C-Shell**
  - updated C-shell with better “line-editing”, access to command history, and command and file name completion
- **bash: The Bourne Again Shell**
  - aimed at providing a public domain version of the Bourne shell
    - close, but there are still some minor differences
  - default shell for Linux
  - implemented as a part of GNU project by public efforts
- **ksh, zsh, tsh, ...**

Changing Your Shell

- Default shell is the shell you are given after you login to the system
- Changing your shell ...
  - Your default shell can be changed using the “chsh” command on Unix.
    - More on this later
  - By typing “sh”, “csh”, “tcsh”, “bash”, etc.
    - Run another type of shell as a “subshell”
    - After you exit from the subshell, you will come back to the old one
    - Your default shell is unchanged

Which Shell(s) Do We Teach?

- For the CLI aspects of the shell, we teach tcsh
- For programming language aspects, we teach sh
- Many Unix users use shells in this way
- Many features of sh or tcsh are shared by other shells
  - e.g. tcsh is really an extension of csh, with some extra features

Basic Unix Commands

Common Commands (1)

- cp for CoPy

  Use: obelix[4] > cp file1 file2

  Action: copy file1 into file2

- rm for ReMove

  Use: obelix[5] > rm file2

  Action: removes or deletes file2

- mv for MoVe

  Use: obelix[6] > mv file1 file3

  Action: renames file1 as file3

  Compare: cp file1 file3; rm file1

Common Commands (2)

- cat for...listing the contents of a file
  - Use: obelix[7] > cat file1
  - Results: display the contents of file1
  - Why "cat"?
    - originally "short" for concatenate
    - can use: obelix[8] > cat file1 file2
    - prints file1 followed by file2
- more for listing the contents of a file, one screen full at a time
  - Use: obelix[9] > more file1
  - Results: display the contents of file1 for a page and pause. Press return for next line. Press space bar to see next page, b to go back one page. Press q to quit.

Common Commands (3)

- date: what date and time is it?

  Use: obelix[10] > date

  Result: print the time and date

- cal: print a calendar

  Use: obelix[11] > cal

  Result: print the calendar of the month

- hostname: what machine am I on?

  Use: obelix[12] > hostname

  Result: print the machine's name

- who: who else is logged onto this computer?

  Use: obelix[13] > who

  Result: a list of users and some info about them

Common Commands (4)

- uptime: how long has the machine been up and running?

  Use: obelix[14] > uptime

  Result: one line with all sorts of neat stuff

- netscape: surf the net

  Use: obelix[15] > netscape

  Result: web surfing software that works only under X-windows

- lynx: surf the net

  Use: obelix[16] > lynx

  Result: web surfing software that is text-only

Common Commands (5)

- echo: print some text

  Use: obelix[17] > echo Unix is easy!

  Result: Unix is easy!

- expr: evaluate an expression

  Use: obelix[18] > expr 1 + 2

  Result: 3

- clear: clear screen

  Use: obelix[19] > clear

  Action: clears the screen

Using man

- man: View manual pages

  Use: obelix[20] > man subject

  Action: Displays the man page for subject

- e.g. "man cat" produces the following:

```
User Commands                                           cat(1)

  I

NAME
     cat - concatenate and display files

SYNOPSIS
     cat  [ -nbsuvet ]  [ file ... ]

DESCRIPTION
     cat  reads each file in sequence and writes it on the  stan-
     dard output. Thus:

     example% cat file
     prints file on your terminal, and:

     example% cat file1 file2 >file3
     concatenates file1 and file2,  and  writes  the  results  in
     file3.  If no input file is given, cat  reads from the stan-
     dard input file.

OPTIONS
     -n        Precede each line output with its line number.
--More--(12%)
```

Important Parts of Manual Pages

- Name
  - The name of the command and brief description
- Synopsis
  - A brief overview on how to use the command
- Description
  - More details of what the command does
- Options and Operands
  - Arguments given to the command
- Examples
- See Also
  - Related commands

More of man

- man -k keyword
  - list all the commands whose brief description (in the "name" field) contains the keyword
  - the apropos command does the same thing as executing man -k
- man man
  - print out the manual of the command man
- man -s n subject
  - prints man page for subject from section n
  - man pages are organized into several sections:
    - Commands, C reference, File formats, …
  - man -l subject will list all of the sections containing the subject … a man -s will then find the man page

Try the Following Commands with man

- cd: change directory to ..
- more: show the content of a file in pages.
- cp: copy a file from .. to ..
- rm: remove a file.
- mkdir: make a directory.
- rmdir: remove a directory.
- mv: move a file or directory to..

Alternatives to man

- xman
  - An X-windows interface to man pages
  - Better browsing and searching facilities
- info
  - A hypertext interface to accessing manuals for GNU software (gcc, emacs, info, …)
- answerbook
  - Newer manual pages from Sun in HTML
  - Also at: http://asterix.gaul.csd.uwo.ca:8888
- http://docs.sun.com/
  - Complete set of manuals from Sun

Unix Editors

Unix Editors

- Editors in Unix come in two general flavours:
  - modal editors have "modes"
    - generally input mode and command mode
      - input mode allows entry of text
      - command mode allows positioning within the file and more sophisticated text modification
    - primary Unix examples: ed and vi
  - modeless editors (or WYSIWYG: What You See Is What You Get) have only one mode
    - positioning and text manipulation are done by special key sequences (like arrow keys and function keys)
      - could also be done by mouse actions or menus
    - primary Unix examples: emacs or pico

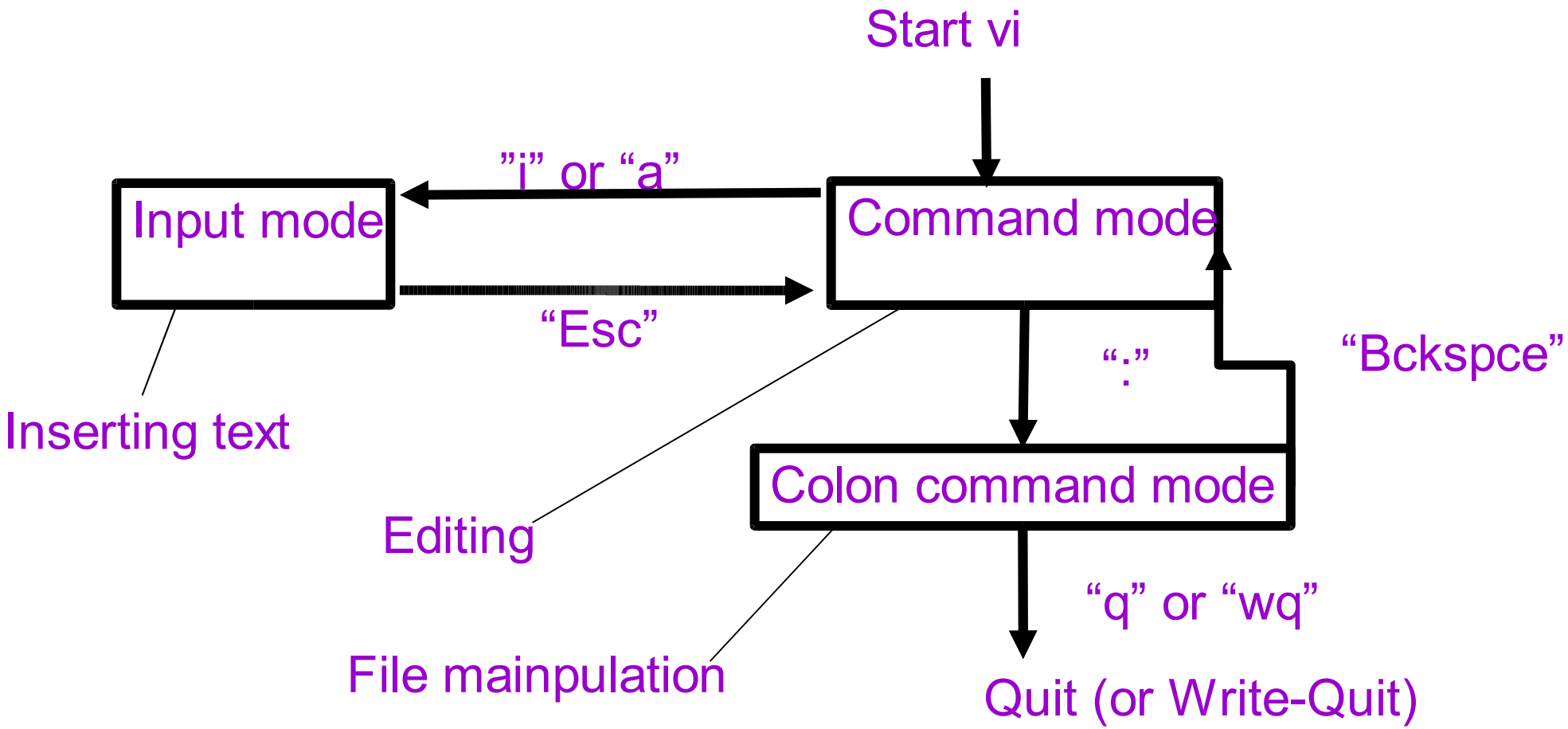
ed

- ed is the original line editor
- Still one of the most powerful editors available
  - Isn't a screen editor so people dislike it
    - It doesn't give you a “local” or screen picture of what is in your file
  - Ability to make massive changes with one command
- We will meet its power base later:

  regular expressions
- Many of its capabilities have been incorporated into newer editors like vi and emacs

vi (1)

- Developed by UCB and comes with all versions of Unix.
- Difficult to use, but very fast for experienced users.
- Has three modes:



vi (2)

- Starting vi

  obelix[21] > vi filename

- Several basic commands
  - arrow keys move the cursor
  - x delete the current character
  - dd delete the current line
  - u undo the change
  - / search for the text following /
  - i change to input mode
  - esc go to command mode
  - :w write to file
  - :wq save and quit
  - :q quit (if no change after the last saving)
  - :q! exit without save

vi (3)

Cursor Movement

- h move one char left
- j move one line down
- k move one line up
- l move one char right
- w move to next word
- e move to end of current word
- b move to beginning of previous word
- 0 move to the beginning of the current line
- $ move to the end of the current line

Search

- / search for something
- ? search backwards for something

Screen Movement

- H move to top of screen
- L move to bottom of screen
- ^F scroll down one page
- ^B scroll up one page
- ^U scroll up one half page
- ^D scroll down one half page

Adding Text

- o (O) opens new line below (above) the current line
- i (I) inserts text before current char (beginning of line)
- a (A) appends text after current char (end of line)

vi (4)

Deletion Commands

- x delete character under cursor
- D delete to end of line
- dd delete entire line
- d$ delete to end of line
- d0 delete to beginning of line
- dw delete the next word
- db delete the previous word

Other Commands

- . redo last modification command
- u undo the last command
- :w file write the buffer to this file
- :r file read this file into the buffer

Change Commands

- s substitute a string for current char (end with ESC)
- r replace current char with another
- R overwrite text (end with ESC)
- C replace to end of current line (end with ESC)
- c0 replace to beginning of current line (end with ESC)
- cw replace the current word (end with ESC)
- cb replace the previous word (end with ESC)

emacs (1)

- Emacs: (Editor MACroS)
  - developed by Richard Stallman and James Gosling amongst many others
  - modeless
  - has versions for Unix, Windows, and other systems
  - menu-driven and mouse-driven under X-windows.
- Emacs uses special keys (ESC and CTRL) to perform editor functions other than input
- This editor can do everything
  - Contains a complete programming language (a LISP interpretter) which can be used to write functions for use in the editor

emacs (2)

- Key combination: a sequence of (special) keys
  - C-x "Control X"
    - Hold down Control key while typing x.
  - C-x C-c
    - Hold down Control key while typing x and c.
    - Or hold down control key while typing x, then release, then hold down control while typing c.
  - C-x u
    - Hold down the Control key, keep it down while typing x. Release the Control key and type u.
  - ESC x "Escape x" or "Meta x"
    - What always works:
      - Type the Escape key. Release. Type x.
    - What sometimes works (and is convenient):
      - Hold down the Alt key and x key at the same time

emacs (3)

- Starting Emacs on a file:

  obelix[23]% emacs myfile

  - myfile is either a new or existing filename.

- The following happen:
  - If the filename you typed was an existing file, you will see the first page of the file on your screen.
  - If you typed a new filename, you will be faced with a blank screen, and you may type the file.
  - The file name will appear at the bottom of the screen.

emacs (4)

- When you encounter problems ...
  - Emacs is a very powerful editor
    - No matter what key combination you press, it probably does something!
    - Sometimes it does something you didn't want!
  - UNDO
    - To undo last operation: Ctrl-_ (Control & underscore)
    - You can also use: Ctrl-x u
    - Can be repeated to keep undoing operations
  - Cancel
    - If you get to a mode which you don't want
      - e.g: you typed Ctrl-x and emacs expects more
    - Type Ctrl-g
      - It will usually back you out of almost anything

emacs (5)

Cursor Movement

- Arrow keys move the cursor around screen.
- Alternatively, use:
  - Ctrl-f Forward a character (Right)
  - Ctrl-b Back a character (Left)
  - Ctrl-n Next line (Down)
  - Ctrl-p Previous line (Up)

Other Movements:

- Ctrl-a Beginning of line.
- Ctrl-e End of line.
- Ctrl-v View next screen.
- ESC v View previous screen.
- ESC < Start of file.
- ESC > End of file.
- ESC f Forward a word.
- ESC b Back a word.
- ESC x goto-line Goes to a given line number.

emacs (6)

Cut and Paste

- To move a block of text

  Move cursor to start of block

  Ctrl-@ Set mark

  Move cursor to end of block.

  Ctrl-w Wipe out (Cut)

  ESC w Copy.

  Move cursor to new location

  Ctrl-y Yank back last thing killed (Paste).

  – The Ctrl-y may be repeated for multiple copies.

Text Deletion

- Backspace
  – Kill character before cursor
- C-k
  – Kill line - deletes to end of line.
- C-d
  – Delete character at cursor
- ESC d
  – Delete next word.
- C-x u Undo last change.
  – Repeat to undo as many changes as you wish.
- ESC x revert-buffer
  – Undo all changes since last save.

emacs (7)

Save / Exit

- Ctrl-x Ctrl-s
  - Save file (over-write original)
- Ctrl-x Ctrl-c
  - Exit from emacs.
- Ctrl-x Ctrl-w
  - Save in different file
  - You are prompted for name

Emacs creates extra files.

- When you save using Ctrl-x Ctrl-s, the old file will be kept as filename~.
- If you exit without saving, the modified unsaved file will be saved as #filename#.

Other Commands

- Check spelling
  - Type ESC $
    - Check spelling of 1 word.
  - ESC x spell-buffer or
    ESC x ispell-buffer
    - Check spelling of file.
- Insert a file
  - Ctrl-x i
    - Insert a file at current cursor position.
- Reformat regions
  - ESC q Reformat paragraph
  - To reformat a region:

    Move cursor to start of block.

    Ctrl-@

    Move cursor to end of block.

    ESC q

emacs (8)

Searching

- Search allows you to search for a string
- Search from the cursor position to the end of file.
- To search for a string, type

  Ctrl-s string
  - Ctrl-s again repeats
  - Ctrl-g to quit

Search and Replace

- Replace all occurrences of one string with another
  - ESC x replace-string
    - you are prompted for the replacement text
- Query-replace asks before replacing each occurrence.
  - Type: ESC %
  - you are prompted for search & replace strings.
- At each occurrence, respond:
  - y/n to replace/not replace.
  - ! to replace all remaining
  - ESC to exit
  - ? for lots more options

pico

- pico is the PIne COmposer
  - the text editor used in the University of Washington's popular pine e-mail program
- pico is a modeless editor like emacs
  - always in "insert" mode
  - command keys available are always listed at the bottom of the screen
  - examples:
    - Ctrl-g Gets help
    - Ctrl-r Reads a file
    - Ctrl-o Writes a file
    - Ctrl-x Exits pico