Unix for non-programmers

- Obtaining an account

- Logging in (get to the login prompt; usually by using the telnet utility to sign on to a computer)

- Upon logging on you will see a prompt (usually $ or %) which is displayed by a special kind of program called a SHELL program.

- To set your password, use the passwd command

- Three popular shells: Bourne Shell, Korn Shell, C Shell
  All have common core functionality; some differences.

- Each shell has its own programming language. (Shell programming).

Running Utilities

- To run a utility, simply type the name of the utility after the prompt.

- Some utilities: date, man, clear, stty, passwd

```
Utility: date [yymmddhhmm[.ss]]
Utility: clear
Utility: man [-s section] word
         man -k keyword
```

- To logout, enter CTRL-D

Special Characters

- meta characters can be listed using the stty -a command

- erase: CTRL-H, werase: CTRL-W, kill: CTRL-U, rprnt: CTRL-R
  intr: CTRL-C, suspend: CTRL-Z, stop: CTRL-S/CTRL-Q, eof: CTRL-D

Some common UNIX utilities

- pwd
- cat, more, page, head, tail
- ls, cd
- mv, cp, rm
- mkdir, rmdir
- file, wc, lp
- vi, pico, emacs
- groups, chgrp, chmod

```
Pathnames: /home/raj/x.c
           /usr/oracle/m01/app/oracle/product/8.0.4/bin/sqlplus
           . refers to current directory (example: ./a.out)
           .. refers to parent directory (example: ../p2.cpp)
```

```
$ cat > letter
Hi Mom,

Please send me money!

David
^D
$ ls -l letter
-rw-r--r--   1 raj      other          38 May 28 11:20 letter
```

Some common Unix commands

```
$ ls -algFsdR <file-spec>
directory listing; a: hidden files, l: long listing, g: group
                   F: *, @, /, = after file name indicating
                      executable, link, directory, socket
                   s: num. disk blocks; d: dir details not contents
                   R: recursive listing

$ cat file-spec-list
list contents of file(s) on screen without pause

$ more file-spec-list
same as cat; pauses after each screen -more-
space bar takes you to next screen; q quit; enter next line
h: help key for more commands
```

```
$ page <file-spec-list>
same as more; clears screen before each page. quicker

$ head -n file-spec
display n lines from front of file

$ tail -n file-spec
display n lines from end of file

$ mv -i old new
$ mv -i file-spec dir
$ mv -i dir dir
rename files (simply change labels in the file hierarchy)
```

```
$ mkdir dname
create a directory

$ cd dir
shell built-in; moves shell to a different directory

$ pwd
print working directory

$ cp -i old new
$ cp -i file-spec dir
$ cp -ir file--spec dir
copy files
```

```
$ rmdir
delete directory

$ rm -fir file-spec
delete file (f: inhibits all prompts/messages; i: inquire; r: recursive)

$ lp -d dest -n copies file-spec
$ lpstat dest
$ cancel request-id
line printing commands

$ wc -lwc file-spec
```

File Attributes

```
46 -rw-r--r--   1 raj      other      23055 Nov 30  1998 paper.tex

46            : num. of physical blocks
-rw-r--r--    : file permissions;
                file type (- regular, d dir, b buffered file disk drive,
                           c unbuffered file terminal, l link, p pipe,
                           s socket)
1             : hard link count
raj           : file owner
other         : file's group
23055         : file size in bytes
Nov 30  1998 : file modification date
paper.tex     : file name

$ file fname
ascertains the type of file (ascii/binary etc.)
```

When a process executes, it has

- a real user ID
- an effective user ID
- a real group ID
- an effective group ID

When you log in, your login shell process has its

- real/effective user id set to your user id
- real/effective group id set to your group id

When a process runs, the file permissions apply as follows:

- If process' effective user id = owner of file; User Privilege applies
- If process' effective user id <> owner of file
  but its effective group id = filesgroup id; Group Privilege applies
- Otherwise Other Privilege applies

real user id is used for accounting purposes only

Note: Process access rights depends on who executes the process NOT
who owns the executable.

This is undesirable sometimes; ex. game of rogue; file of best scores
needs to be modified ONLY by rogue program process when player
ends game; NOT by players shells.

Special feature: 2 special file permissions

- set user ID (s instead of x)
- set group ID

When an executable with set user ID is executed, the effective user ID
of the process becomes that of the executable NOT of the players
shell (Thereby allowing process to modify the best scores file)

```
$ groups userid
lists the groups a particular user belongs to

$ chgrp -R gp-name file-spec
change the group a file belongs to

$ chmod -R XXX file-spec
$ chmod -R g+w file-spec
$ chmod -R u-rw file-spec
$ chmod -R u+w,g-r file-spec
change file permissions; u,g,o,a  r,w,e,s

$ chown new-owner file-spec
change owner of file (only super user can do this)

$ newgrp group-name
creates subshell with effective group id = group-name
```

Vi

```
- Command mode vs Text Entry Mode
- To enter text entry mode: use the following commands:
    i,I,a,A,o,O,R
  to get back to command mode use ESC
- line ranges: 1,$   1,.    .,$   .,.-2   5,12   .,.+12
```

```
Cursor Movement:

  uparrow, downarrow leftarrow rightarrow
  ^  $
  w  b
  CTRL-D  CTRL-F  CTRL-U  CTRL-B
  :nn

Deleting Text:

  x   dw  dd   D   :<range>d

Replacing text:

  r
  cw
  cc  (entire line)
```

Pasting text:

```
:<range>y     (yank lines)
p
P
:nnp   (after line nn)
```

Searching:

```
/sss/
?sss?
n
N  (opposite direction)
```

Searching/Replacing:

```
:<range>s/sss/ttt/
:<range>s/sss/ttt/g
```

Saving/Loading:

```
:w <fname>
:w
:<range>w <fname>
:e <fname>          edit new file
```

Misc:

```
CTRL-L
:!comand
:q
:q!
```