Unix Basics

Unix Basics

- Unix directories
- Important Unix file commands
- File and Directory Access Rights through Permission Settings
- Using chmod to change permissions

Unix File Structure

- **Hierarchical file system**
  - Starts at *root*, denoted "/".
  - Abstraction is to navigate through the Unix directory structure relative to the current "working" directory.
  - Slashes separate directory levels.
  - File names cannot have blanks and lower-case is preferred {case-sensitive}.
  - Extensions are just conventions to the file system, but NOT to compilers!

Unix File Notation

. = the current directory

.. = the parent directory

~ = my home directory (i.e., the current directory when I login)

File name wild cards

? = any one character

* = any zero or more characters

Unix Commands

- Basic format:

  Command –option parameters

  e.g. `ls -l labs*`

  e.g. `cp new.c old.c`

- C commands can be cryptic and many are only two characters long, but an important exception is:

man = manual page request

  e.g. **man ls**

Commands: pwd & ls

pwd = print working directory

ls = list file names and attributes.

- -l = long listing
- -d = list directory itself, not contents
- -a = all files (including starting with ".")

```
e.g. ls       [just file names]
e.g. ls -la   [lots of info!]
e.g. ls -la labs* [only info labs]
e.g. ls -d        [just directory names]
```

Commands: mkdir & cd

mkdir = make a new directory

e.g., **mkdir newdir**

cd = change directory

e.g. **cd newdir**

e.g. **cd ../updir**

e.g. **cd** [change to home directory]

Commands: mv & cp

cp = copy file

cp source destination

-p = preserve permissions

e.g. **cp -p new.c old.c**

e.g. **cp prog1.c prog_dir/**

mv = move file

mv source destination

e.g. **mv prog1.c distance.c**

e.g. **mv prog1.c prog_dir/**

For both commands if the destination is an existing directory, the file name stays the same.

File and Directory Permissions

- **Each file or directory has three sets of permissions:**
  - User (i.e. owner)
    - Note - Only the user can change permissions.
  - Group
  - Other (the world!)
- **Each permission set has three permissions:**
  - Read
  - Write
  - Execute

These are visible left to right via:

```
ls -la
```

File and Directory Permissions

- **Read access** = You can read the file contents. You can list the contents of the directory.
- **Write access** = You can write into this file. You can modify this directory.
- **Execute access** = You can run this file as a command. You can use this directory as part of a path.

To access any file, you first need execute permission on all directories from the root to the file.

Command: chmod

chmod = Change mode (permissions)

chmod mode files

mode:

- specify users: **u**, **g**, or **o**
- specify attribute: **r**, **w**, or **x**
- connect with action:
  - **+** = add
  - **-** = delete
  - **=** = set

Command: chmod

- Examples:

```
chmod u+x prog4.cpp
chmod o-r prog4.cpp
chmod u=rwx prog4.cpp
chmod o+r,g+r prog4.cpp
```

- You can also use octal numbers:

```
chmod 700 prog2.c
chmod 750 sample.c
```

Commands: emacs, cat, more

{generic format}

command filename

emacs = edit a file

e.g. **emacs lab1.c**

cat = printout text file

e.g. **cat lab1.c**

more = printout text file (only fill one screen)

e.g. **more lab1.c**

- hit the space bar to see more or q to quit.

Commands: rm, ps, kill

rm = delete a file

e.g. **rm olddat.txt**

ps = print currently active processes

e.g. **ps**

kill = stop one of your running processes

e.g. **kill -9 26814**

Example: ps kill

```
$emacs simple.c
{inside edit of simple.c}
^z
$ps
  PID TTY          TIME CMD
26792 pts/17   00:00:00 tcsh
26814 pts/17   00:00:00 emacs
26815 pts/17   00:00:00 ps
$ kill -9 26814
$
[1]    Killed                        emacs simple.c
```

% *type this to resume*