

# CSCI 4061: Unix Basics

Chris Kauffman

*Last Updated:*

*Thu Jan 24 09:20:42 CST 2019*

# Logistics

## Reading Stevens and Rago

- ▶ Ch 1: Unix Overview
- ▶ Ch 2: Unix Standards (skim)
- ▶ Ch 7: Processes (Next Week)

## Goals Today

- ▶ Warm-up C program
- ▶ Course Mechanics
- ▶ Unix Basics

# Access to Unix Machines

- ▶ CSE Labs
  - ▶ Via SSH
  - ▶ Via <http://vole.cse.umn.edu>
- ▶ Mac OS X: Terminal, development tools
- ▶ Windows
  - ▶ [Install Linux Subsystem for Windows 10](#) (recommended)
  - ▶ [Install Cygwin](#) (not recommended)
- ▶ Any: Install [VirtualBox](#) to host a Unix you like
- ▶ Install Native Linux or BSD: *"Now you're playing with power!"*

## Exercise: Quick Review

- ▶ Name two major functions of all operating systems
- ▶ Give some major components/abstractions that Unix provides as part of its interfaces
- ▶ What is a system call? How does it work?

# Unix Standards: POSIX

POSIX defines what you can plausibly expect on unix-like systems.

Includes

- ▶ C libraries for system calls, standard libraries
- ▶ Basic layout of file system and naming conventions
- ▶ Some Devices such as `/dev/null`

## Distinction: C Standard vs Unix Library

- ▶ Lots of systems have a C compiler which has the C standard library: `printf()`, `fopen()`, `exp()` etc.
- ▶ Unix systems have additional, separate libraries for Unix-specific stuff like `read()`, `fork()`, `select()`, `kill()`
- ▶ Some branches of Unix have their own special, special versions of these like Linux `clone()`

# Command Line: Basic File System Navigation

Command	Effect
pwd	print the current directory
cd folder	change directory / folder
ls	list files in directory
cd ~	change to home directory

```
> pwd
/home/kauffman
> ls
1103-F2017  aurs      Downloads  Hello.class  Hello.java~  PathClassLoader.txt
4061-F2017  Desktop  Dropbox    Hello.java   misc          public_html
> cd 4061-F2017
> ls
exams  lectures  Makefile~  projects      schedule.html~  schedule.org~  textbook
labs  Makefile  misc      schedule.html  schedule.org    syllabus
> pwd
/home/kauffman/4061-F2017
> cd lectures
> pwd
/home/kauffman/4061-F2017/lectures
> ls
00-course-mechanics.org  00-course-mechanics.tex  01-introduction.org  01-introduction.tex
00-course-mechanics.org~  01-introduction-code     01-introduction.org~  02-unix-basic.c
00-course-mechanics.pdf  01-introduction-code.zip  01-introduction.pdf  02-unix-basics.org
> cd ~
> pwd
/home/kauffman
> ls
1103-F2017  aurs      Downloads  Hello.class  Hello.java~  PathClassLoader.txt
4061-F2017  Desktop  Dropbox    Hello.java   misc          public_html
```

# Determining File Types

Command	Effect
<code>file something.ext</code>	try to determine the type of given file

```
> file xxx
xxx: UTF-8 Unicode text, with very long lines
> file test.txt
test.txt: ASCII text
> file www
www: directory
> file 4061-F2017
4061-F2017: symbolic link to /home/kauffman/Dropbox/teaching/4061-F2017
> file 4061-F2017/
4061-F2017/: directory
> cd 4061-F2017/lectures/
> file 01-introduction-code.zip
01-introduction-code.zip: Zip archive data, at least v1.0 to extract

> file 02-unix-basics-code/no_interruptions.c
02-unix-basics-code/no_interruptions.c: C source, ASCII text

> file 02-unix-basics-code/no_interruptions.o
02-unix-basics-code/no_interruptions.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped

> file 02-unix-basics-code/a.out
02-unix-basics-code/a.out: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=fffb87934737b0e48b891d27573ae8a2e5687c46a, not stripped
>
```

# Searching and Manipulating Text

Command	Effect
<code>cat file.txt</code>	show contents of file in terminal
<code>less file.txt</code>	"page" text file, press "q" to quit
<code>grep 'expression' file.txt</code>	show lines matching expression in file
<code>grep 'expression' *.txt</code>	search every .txt file for lines
<code>find .</code>	show all files recursively from current
<code>find . -name '*.c'</code>	find all C source files recursively

These may be covered in a future lab.



# Editing Files

Command	Effect
vi	modal editing, terse, powerful, ALWAYS present
emacs	modes for editing, extensible, mostly available, ♥
nano	simple, podunky, usually available

- ▶ Learn some vi or emacs
- ▶ Comes in real handy when you need to edit but there is no graphical login

## Permissions on Files

Command	Effect
<code>ls -l</code>	long listing of files
<code>chmod u+x file.abc</code>	make file executable by user
<code>chmod o-rwx file.abc</code>	remove permissions from other users
<code>chmod 777 file.abc</code>	everyone can do anything to file

```
> ls
a.out no_interruptions.c no_interruptions.c~^Ino_interruptions.o
> ls -l
total 40K
-rwxrwx--- 1 kauffman kauffman 8.5K Sep  7 09:55 a.out
-rw-r--r-- 1 kauffman kauffman  955 Sep  7 09:55 no_interruptions.c
-rw-r--r-- 1 kauffman kauffman  883 Sep  7 09:54 no_interruptions.c~
-rw-rw---- 1 kauffman kauffman 2.4K Sep  7 11:59 no_interruptions.o
> chmod u-x a.out
> ls -l
total 40K
-rw-rwx--- 1 kauffman kauffman 8.5K Sep  7 09:55 a.out
-rw-r--r-- 1 kauffman kauffman  955 Sep  7 09:55 no_interruptions.c
-rw-r--r-- 1 kauffman kauffman  883 Sep  7 09:54 no_interruptions.c~
-rw-rw---- 1 kauffman kauffman 2.4K Sep  7 11:59 no_interruptions.o
> ./a.out
bash: ./a.out: Permission denied
> chmod u+x a.out
> ./a.out
Ma-na na-na!
```

# Permissions / Modes

- ▶ Unix enforces file security via *modes*: permissions as to who can read / write / execute each file
- ▶ See permissions/modes with `ls -l`
- ▶ Look for series of 9 permissions

```
> ls -l
total 140K
-rwx--x--- 2 kauffman faculty 8.6K Oct  2 17:39 a.out
-rw-r--r-- 1 kauffman devel  1.1K Sep 28 13:52 files.txt
-rw-rw---- 1 kauffman faculty 1.5K Sep 26 10:58 gettysburg.txt
-rwx--x--- 2 kauffman faculty 8.6K Oct  2 17:39 my_exec
----- 1 kauffman kauffman 128 Oct  2 17:39 unreadable.txt
-rw-rw-r-x 1 root      root    1.2K Sep 26 12:21 scripty.sh
  U  G  O      O      G      S      M  T      N
  S  R  T      W      R      I      O  I      A
  E  O  H      N      O      Z      D  M      M
  R  U  E      E      U      E      E      E
      P  R      R      P
~~~~~
```

PERMISSIONS

- ▶ Every file has permissions set from somewhere on creation

# Manual Pages

Command	Effect
man ls	Bring up the manual page for command ls

```
> man ls | cat
```

```
LS(1)
```

```
User Commands
```

```
LS(1)
```

```
NAME
```

```
ls - list directory contents
```

```
SYNOPSIS
```

```
ls [OPTION]... [FILE]...
```

```
DESCRIPTION
```

List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

```
-a, --all
```

```
do not ignore entries starting with .
```

```
-A, --almost-all
```

```
do not list implied . and ..
```

```
...
```

# Program Search PATH

Command	Effect
<code>echo \$PATH</code>	show where shell looks for programs
<code>PATH=\$PATH:/home/kauffman/bin</code>	also look in my bin directory
<code>PATH=\$PATH:.</code>	also look in current directory
<code>PATH=.</code>	ONLY look in the current directory

```
> echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/lib/jvm/default/bin:
/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/home/kauffman/bin:
/home/kauffman/Dropbox/bin:/home/kauffman/code/bin:/home/kauffman/code/utils:.
```

Search directories are separated by colons in Unix

## Exercise: Compilation

- ▶ What command is typically used to compile C programs?
- ▶ What is the default name of a compiled program on Unix and how can it be changed?
- ▶ What function does a *runnable* C file need to have to make a program?
- ▶ How can you compile a C file *without* that special function?

## Answers: Compilation

- ▶ What command is typically used to compile C programs?
  - > gcc myprog.c
- ▶ What is the default name of a compiled program on Unix and how can it be changed?
  - > ./a.out
  - > gcc -o mprog mprog.c
  - > ./myprog
- ▶ What function does a *runnable* C file need to have to make a program?
  - ▶ main() must be present in at least one C file to make program
- ▶ How can you compile a C file *without* that special function?
  - > gcc -c funcs1.c # produces funcs1.o
  - > gcc -c funcs2.c # produces funcs2.o
  - > gcc -o prog funcs1.o funcs2.o # link object files
  - > ./prog

## make and Makefiles

- ▶ Example of a **build system**
- ▶ Very old system, many newer ones but a good starting point
- ▶ Will be discussed in Lab01 which will go up over the weekend
- ▶ Make sure to attend your first lab, *the one you are registered for*



# How make and Makefile Works

Build up dependencies recursively

- ▶ A tree-like structure (actually a DAG)
- ▶ Run commands for the lowest level
- ▶ Then go up a level
- ▶ Then up another ...
- ▶ Can recurse to subdirectories to use other Makefiles as well
- ▶ Makefile describes dependencies between source/program files and commands to generate/compile

## Makefile Format

```
target1 : dependency1 dependency2
    do command 1
    then do command 2

target2 : target1 dependency3
    do command X
    then do command Y
```

# Showing and Murdering Running Processes

Command	Effect
ps	show running processes associated with terminal
ps a	show ALL running processes
ps u	show all processes for me
kill 1234	send process 1234 the TERM signal
kill -9 1234	send process 1234 the KILL signal
pkill a.out	send process named a.out the TERM signal
pkill -9 a.out	send process named a.out the KILL signal

```
> ps
  PID TTY          TIME CMD
 8050 pts/1    00:00:00 bash
 8061 pts/1    00:00:00 ssh
11033 pts/1    00:00:00 ps
```

```
> ps u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
kauffman  724  0.0  0.0 201092  5520 tty2    Ssl+  Sep06   0:00 /usr/lib/gdm/gdm-x-session --run-script o
kauffman  726  0.1  0.5 691872 94388 tty2    Rl+   Sep06   2:08 /usr/lib/xorg-server/Xorg vt2 -displayfd
kauffman  737  0.0  0.3 603020 49496 tty2    Sl+   Sep06   0:00 cinnamon-session --session cinnamon
kauffman  784  0.0  0.1 565264 23008 tty2    Sl+   Sep06   0:00 /usr/lib/cinnamon-settings-daemon/csd-or
...
```