# Lab 1

## Environment Setup and Basic Unix

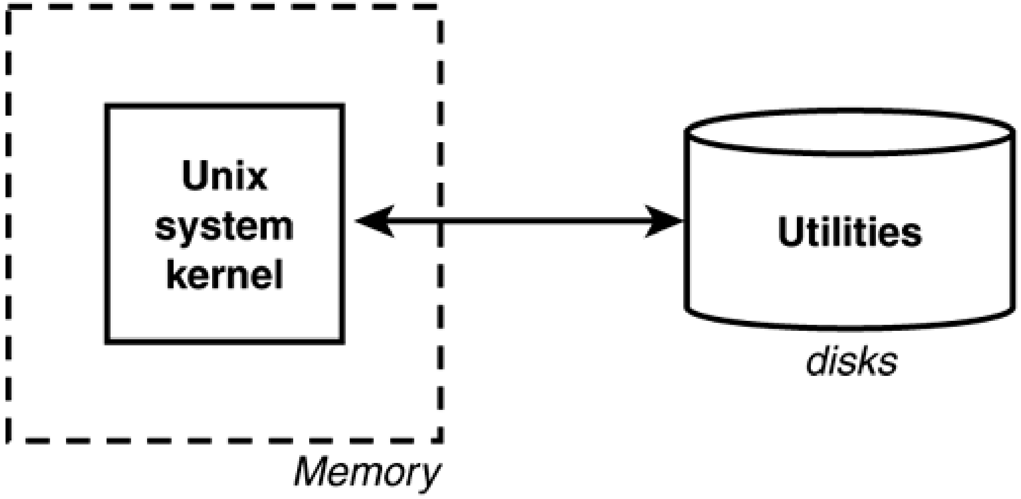
CSCI-2500 - Fall 2015

## Lab Objectives

- Get to know your lab TA and undergraduate mentors
- TA will lecture for 20-30 minutes on basic Linux commands and the file system
- Make sure your computer is setup to run Linux and/or Cygwin and gcc 4.8.2 or later
  - Mac OS X (XCode Command Line Tools + gcc)
  - Windows - Cygwin
  - Mac/Windows - Virtual Box + Ubuntu Desktop Linux 14.04
- To get credit for the lab you need to demonstrate knowledge of basic Linux/UNIX commands and the file system and show your computing environment is setup correctly

# Introduction to Linux/Unix

- A Unix system is itself logically divided into two pieces:
  - kernel
  - utilities



- Kernel is the “heart” of the system and resides in memory
- Utilities are loaded from the disk into memory and executed

## Introduction to Linux/Unix

- All Unix systems require an account:
  - username and password (case sensitive)
  - userid and groupid (unique numbers, a user can have multiple groupids)
  - home directory and shell (command line interpreter)
- Popular Unix systems shells include:
  - Bourne Shell (sh)
  - C Shell (csh)
  - Bourne Again Shell (bash)
  - TENEX C Shell (tcsh)
  - Korn Shell (ksh)
- You interact with the shell (prompts and waits for your commands)

## Basic Unix Commands

- The `date` command tells the system to print the date and time:

```
bash-3.2$ date
Sun Jan 25 13:49:38 EST 2015
```

- If you need help with a specific command you can check the man pages for that command:

```
bash-3.2$ man date
```

## Basic Unix Commands

- The echo command prints or echo at the terminal whatever else you happen to type on the line (there are exceptions to this):

```
bash-3.2$ echo this is a test
this is a test
bash-3.2$ echo why not print a longer line with echo?
why not print a longer line with echo?
bash-3.2$ echo

bash-3.2$ echo one        two      three   four  five
one two three four five
bash-3.2$ echo "one        two      three   four  five"
one        two      three   four  five
```

- Notice echo squeezes out the extra blanks between words.  Words are more important by default on a Unix system (more on this in the next lab)

# Unix File System Commands

- The `ls` command shows what files are listed in your directory:

```
bash-3.2$ ls
feb86    jan5.89    jan87   memo10
jan12.89   jan85    jan88   memo2
jan19.89   jan85.89   mar88   memo2.sv
jan26.89   jan86    memo1
```

- Unix systems recognize only three basic types of files: ordinary files, directories, special files
  - ordinary files - contain data (e.g. text, program instructions)
  - directories - used to group ordinary files together (folders)
  - special files - a file with special meaning to the system (e.g. symbolic links)
- Maximum length of file names is system dependent

## Unix File System Commands

- The `cat` command lets you examine the contents of a file:

```
bash-3.2$ cat names
Susan
Jeff
Henry
Allan
Ken
```

- The `wc` command gives you a count of the total number of lines, words, and characters contained in a file:

```
bash-3.2$ wc names
      5       5      27 names
```

command output: lines, words, characters, file name

## Unix Commands Options

- Most Unix commands allow the specifications of options at the time that a command is executed. These options generally follow the same format:

  `-letter`

- A command option is a minus sign followed immediately by a single letter (or word). Examples with the `wc` command:

```
bash-3.2$ wc -l names
       5 names
bash-3.2$ wc -c names
      27 names
bash-3.2$ wc -w names
       5 names
```

-l gets # of lines in file

-l gets # of characters in file

-l gets # of words in file

## Unix File System Commands

- The `cp` command makes a copy of a file:

```
bash-3.2$ cp names saved_names
bash-3.2$
```

- The first argument is the source file while the second file is the destination (or copy) file

- The `mv` command moves or renames a file:

```
bash-3.2$ mv saved_names hold_it
bash-3.2$
```

- Warning: For both commands above, the Unix system will overwrite an existing file without warning (see the man pages on these commands for safety options)

## Unix File System Commands

- The `rm` command is used to delete files:

```
bash-3.2$ rm hold_it
bash-3.2$
```

- It is possible to delete multiple files at once:

```
bash-3.2$ rm wb collect mon
bash-3.2$
```

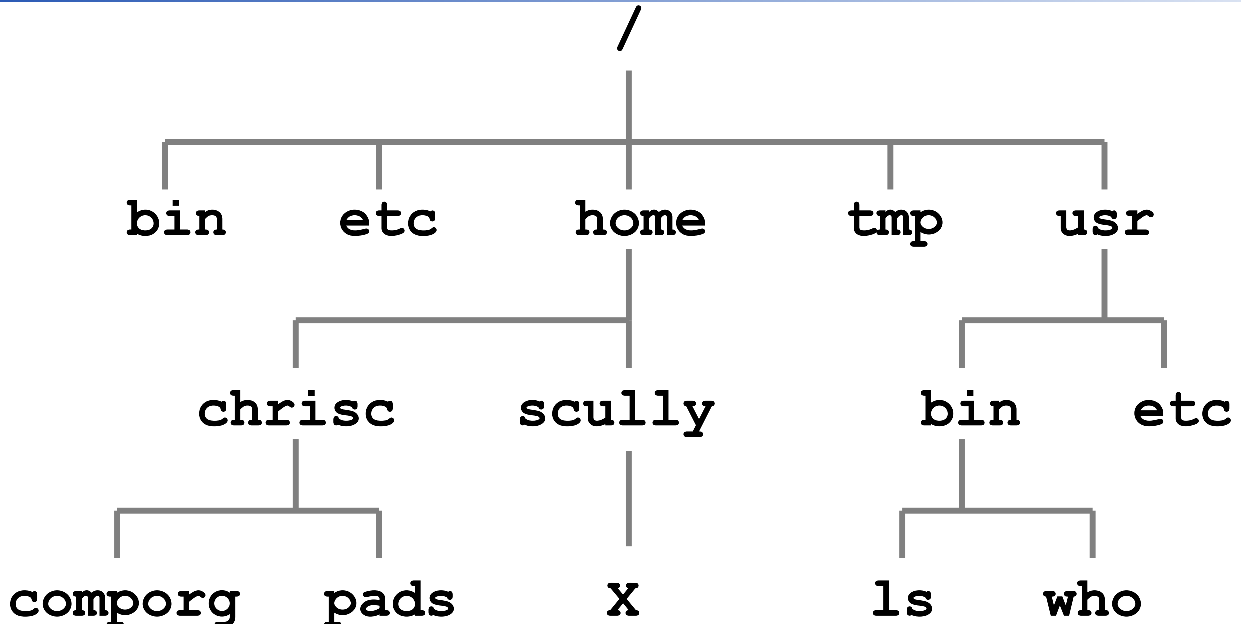
## Unix File System: Directories

- A directory is a special kind of file - Unix uses a directory to hold information about other files
- We often think of a directory as a container that holds other files (or directories)
- Mac and Windows: A directory is the same idea as a folder
- Folders are used as a GUI interface to directories and not unique to Unix/Linux/ FreeBSD

## Unix File System: Files

- Every file has a name
- Each file in the same directory must have a unique name
- Files that are in different directories can have the same name

# Unix File System



- The filesystem is a hierarchical system of organizing files and directories
- The top level in the hierarchy is called the "root" and holds all files and directories.
- The name of the root directory is /
- Example pathname: /home/chrisc/comporg/pads

## Unix File System Commands

- The pwd command tells you the name of your current working directory ("get your bearings"):

```
bash-3.2$ pwd
/home/calonge
```

- The cd command changes the current working directory:

```
bash-3.2$ cd test
bash-3.2$ pwd
/home/calonge/test
```

- If you type the cd command without any arguments it will return you to your home directory:

```
bash-3.2$ cd
bash-3.2$ pwd
/Users/calonge
```

## Unix File System Commands

- The easiest way to get one level up in a directory is to issue the command:

```
bash-3.2$ cd ..
bash-3.2$ pwd
/home
```

- You don’t have to change directories one level at a time:

```
bash-3.2$ cd calonge/test
bash-3.2$ pwd
/home/calonge/test
```

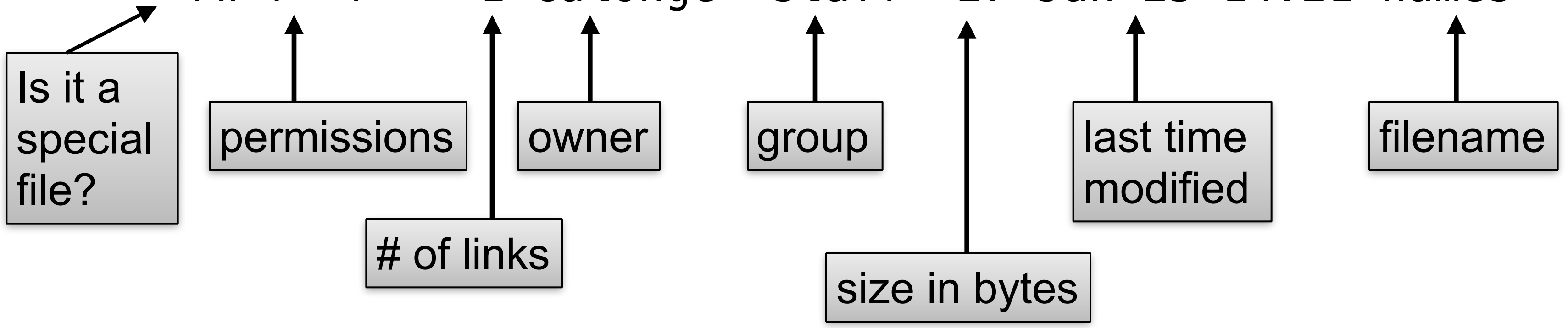
- To go back up two levels:

```
bash-3.2$ cd ../..
bash-3.2$ pwd
/home
```

## Unix File System Commands

- The `ls -l` (long format) command can also show detailed information on the files in a directory:

```
bash-3.2$ ls -l
total 16
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:01 data
-rw-r--r--  1 calonge  staff  54 Jan 25 15:00 memo1
-rw-r--r--  1 calonge  staff  13 Jan 25 14:07 memo10
-rw-r--r--  1 calonge  staff  34 Jan 25 14:07 memo2
-rw-r--r--  1 calonge  staff  12 Jan 25 14:07 memo2.sv
-rw-r--r--  1 calonge  staff  27 Jan 25 14:21 names
```



# Unix File System: Permissions

`ls -l` and permissions

```
-rwxrwxrwx
```

Owner Group Others

**Type of file:**

**- means plain file**

**d means directory**

- You can change file and directory permissions with the `chmod` command

## Unix File System Commands

- The `mkdir` command creates a new directory:

```
bash-3.2$ ls -l
total 16
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:01 data
-rw-r--r--  1 calonge  staff  54 Jan 25 15:00 memo1
-rw-r--r--  1 calonge  staff  13 Jan 25 14:07 memo10
-rw-r--r--  1 calonge  staff  34 Jan 25 14:07 memo2
-rw-r--r--  1 calonge  staff  12 Jan 25 14:07 memo2.sv
-rw-r--r--  1 calonge  staff  27 Jan 25 14:21 names
bash-3.2$ mkdir metadata
bash-3.2$ ls -l
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:01 data
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:11 metadata
-rw-r--r--  1 calonge  staff  54 Jan 25 15:00 memo1
-rw-r--r--  1 calonge  staff  13 Jan 25 14:07 memo10
-rw-r--r--  1 calonge  staff  34 Jan 25 14:07 memo2
-rw-r--r--  1 calonge  staff  12 Jan 25 14:07 memo2.sv
-rw-r--r--  1 calonge  staff  27 Jan 25 14:21 names
```

## Unix File System Commands

- The `rmdir` command deletes a new directory:

```
bash-3.2$ ls -l
total 16
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:01 data
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:11 metadata
-rw-r--r--  1 calonge  staff  54 Jan 25 15:00 memo1
-rw-r--r--  1 calonge  staff  13 Jan 25 14:07 memo10
-rw-r--r--  1 calonge  staff  34 Jan 25 14:07 memo2
-rw-r--r--  1 calonge  staff  12 Jan 25 14:07 memo2.sv
-rw-r--r--  1 calonge  staff  27 Jan 25 14:21 names
bash-3.2$ rmdir metadata
bash-3.2$ ls -l
drwxr-xr-x  2 calonge  staff  68 Jan 25 15:01 data
-rw-r--r--  1 calonge  staff  54 Jan 25 15:00 memo1
-rw-r--r--  1 calonge  staff  13 Jan 25 14:07 memo10
-rw-r--r--  1 calonge  staff  34 Jan 25 14:07 memo2
-rw-r--r--  1 calonge  staff  12 Jan 25 14:07 memo2.sv
-rw-r--r--  1 calonge  staff  27 Jan 25 14:21 names
```

- Note: only works if the directory is empty!

## Unix File System Commands

- The `touch` command changes the file timestamp and can also create blank files:

```
bash-3.2$ ls -l
total 8
-rw-r--r--  1 calonge  staff  27 Jan 25 15:46 myfile
bash-3.2$ touch myfile
bash-3.2$ ls -l
total 8
-rw-r--r--  1 calonge  staff  27 Jan 25 15:56 myfile
bash-3.2$ touch newfile
bash-3.2$ ls -l
total 8
-rw-r--r--  1 calonge  staff  27 Jan 25 15:56 myfile
-rw-r--r--  1 calonge  staff   0 Jan 25 15:57 newfile
bash-3.2$
```

## Unix File System Commands

- The ln command allows you to “link” files together (i.e. give more than one name to a file)

```
bash-3.2$ ls -l
total 8
-rw-r--r--  1 calonge  staff  27 Jan 25 15:35 names
bash-3.2$ ln names namelist
bash-3.2$ ls -l
total 16
-rw-r--r--  2 calonge  staff  27 Jan 25 15:35 namelist
-rw-r--r--  2 calonge  staff  27 Jan 25 15:35 names
bash-3.2$ cat names
Susan
Jeff
Henry
bash-3.2$ cat namelist
Susan
Jeff
Henry
bash-3.2$
```

## gcc: Hello World!

- We will discuss C programming starting in about 2 weeks (Lecture 5). For now try simply create the following C program in a file called hello.c (pick any text editor):

```
/* Hello World! Program */

#include<stdio.h>

int main()
{
    printf(“Hello World!\n”);
}
```

- Compile and run it using the following command:

```
bash-3.2$ gcc hello.c; ./a.out
```

## Steps for Lab Credit

- Demonstrate the following to your lab TA or the undergraduate mentors present (make sure you get checked off):
  - Change directories and list files
  - Copy a file from one directory to another
  - Move a file from one directory to another
  - Create and delete a file
  - Change file permissions (hint: you will need to use the `chmod` command — check the man pages for guidance)
  - Output the current date and time in the following format:
    YYYY-MO-DD HH:MM:SS
    where,
    YYYY = year, MO = month, DD = day
    HH = hour, MM = minutes, and SS = seconds
  - Compile and run the “Hello World!” C program