# UNIX 101

Ryan DeRue, Senior Computational Scientist

PURDUE UNIVERSITY
Rosen Center for Advanced Computing

# Unix 101

## Outline

# What to expect from Unix 101

## Objectives

- What is the Unix operating system (OS) and how is it different from other OSes?
- Develop an understanding of how Unix treats files and their underlying file systems
- Create a toolbox of the most essential commands/utilities in Bash

# Unix 101

## What is Unix?

# What is Unix

## *Brief History of Unix: Computing in the 70s*

- Computing resources are prohibitively expensive and require institutional investment
- A consolidated effort between MIT, GE, and Bell Labs begins to develop a "time-sharing" OS named "Multics"
  - **Multi**plexed **I**nformation and **C**omputing **S**ervice
- After the failure of Multics, Ken Thompson and Dennis Ritchie began work on "Unics"
- Bell Labs licensed Unix to 3rd parties leading to many new variants conforming to the "Unix philosophy"

Dennis Ritchie and Ken Thompson working on a PDP-11 "microcomputer" https://www.bell-labs.com/usr/dmr/www/picture.html

# What is Unix?

## *How is Unix different than other Operating Systems?*

- Unix-family OSes were designed to be used by many users at the same time
- Many of the Unix-family OSes are text-based in contrast to OSes like Windows which present a graphical user interface (GUI)
- Provides a plethora of tools with the "Unix Philosophy" in mind:
  - "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface." - Peter H. Salus, *A Quarter-Century of Unix (1994)*

# Unix 101

## Files in Unix

# Files in Unix

## What is a File in Unix?

- A file is an addressable location that contains some data which can take many forms
  - Text data
  - Binary/Image data
- "Everything is a File" design
- Files have associated meta-data
  - Owner
  - Permissions
  - Timestamps

## File Permissions in Unix

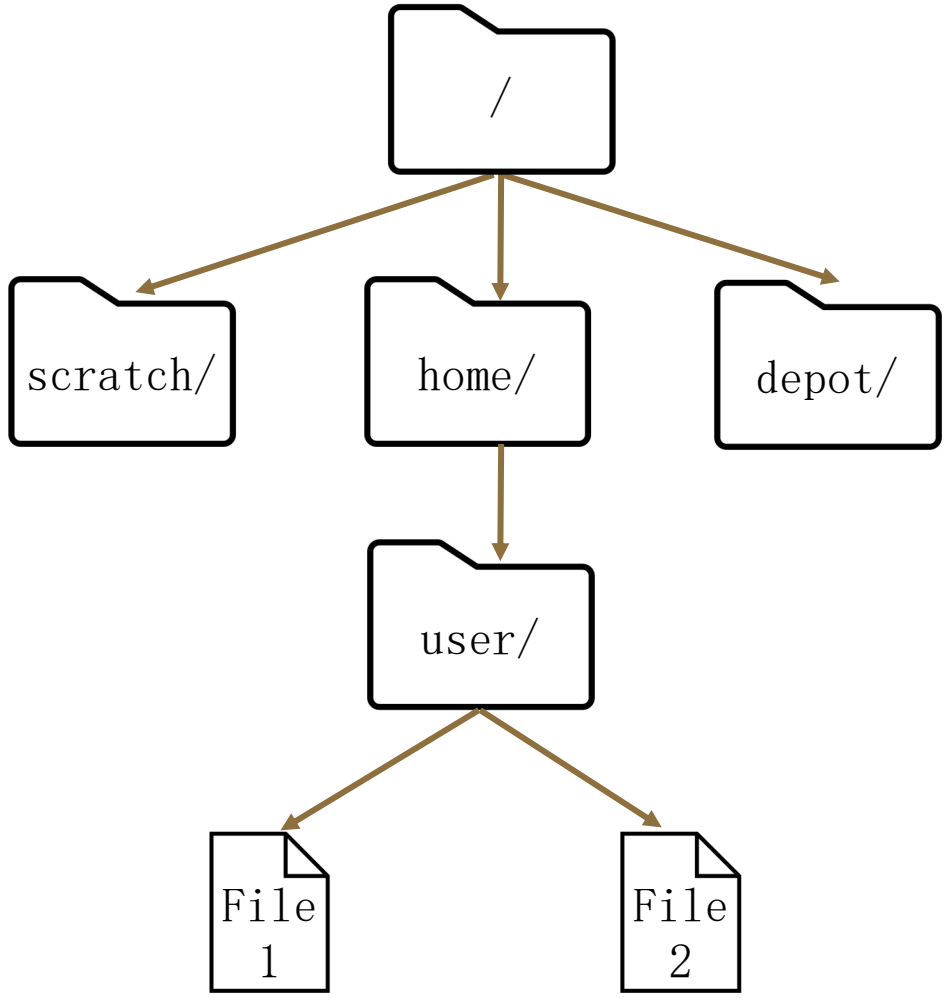
- Types of Permissions
  - Read (r)
  - Write (w)
  - Execute (x)
- How are permissions applied?
  - Owner
  - Group
  - Other

```
-rwxr-xr-x 1 rderue itap    1307 Aug  9 14:08 form.yml.erb
-rw-r--r-- 1 rderue student    4 Jan 18 10:22 LICENSE.txt
-rwxr-xr-x 1 rderue itap     236 Aug  9 14:08 manifest.yml
-rwxr-xr-x 1 rderue itap     462 Aug  9 14:08 submit.yml.erb
drwxr-xr-x 3 rderue itap       5 Aug  9 14:08 template
```

# Files in Unix

## File Organization in Unix

- Files are organized into directories which are analogous to folders in other OSes
- Everything is mounted to the root directory
- Files in Unix are referred to by their location which we call the **path**
  - **Absolute Path (From the root)**
    - `/home/user/File1`
  - **Relative Path (From my location at `/home/`)**
    - `user/File1`
- File extensions are meaningless to Unix
  - Still useful for your own use



# Unix 101

## Interacting with Unix: Shells

# Interacting with Unix: Shells

## What is a shell?

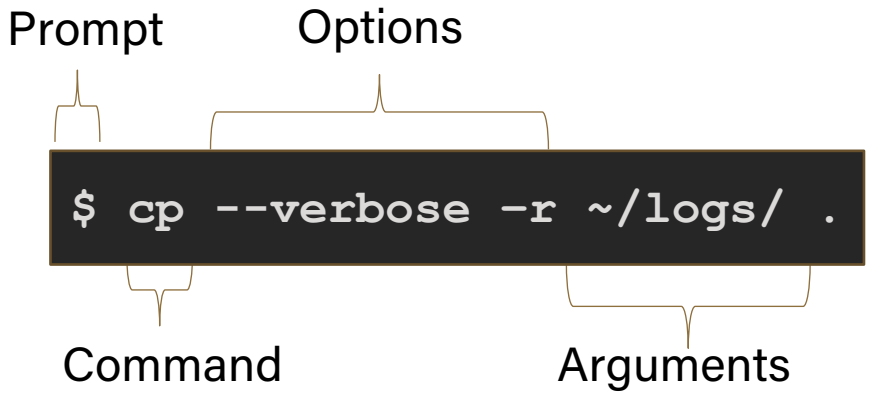
- An interface to the OS that accepts commands
- Many variants of Unix Shells
  - *sh*
  - *bash*
  - *tcsh/csh*
  - *zsh*
- We are going to focus on *bash*



## Introduction to Built-Ins and Utilities

- Built-In Commands vs. Utilities
- Command Structure
  - `COMMAND [-OPTIONS] [ARGUMENTS]`
    - `action [-how] [to what]`
    - `[OPTIONS]` are inherently optional and often take the form of a switch or they set a value (--expression="rcac")
    - `[ARGUMENTS]` can be mandatory or optional and usually denote the "things" you want the command to operate on

### Anatomy of a command



# Interacting with Unix: Command Line Editors

## Editing files within Unix

- Lightweight programs for editing text files within a terminal
- Most common editors
  - vim/emacs
  - nano
- Example usage: nano example.txt
  - If example.txt does not exist already, it will open it

# Unix 101

## Essential Commands

# Essential Commands

## Manual Command

- Purpose: Display the manual page for a given command containing all the available options.
- Usage: `man [section] <command>`
  - Section accepts a number 1-9 which are each over a different topic. This can typically be omitted.

## Example usage of the `man` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ man pwd
```

```
PWD(1)                          User Commands

NAME
       pwd - print name of current/working directory


SYNOPSIS
       pwd [OPTION]...


DESCRIPTION
       Print the full filename of the current working directory.

       -L, --logical
              use PWD from environment, even if it contains symlinks
```

## Present Working Directory Command

- Purpose: Print out the directory that your shell is currently in
- Usage: `pwd [OPTION]`
  - Usually your prompt will display your current working directory

## Example usage of the `pwd` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ pwd
  /home/rderue/teaching/unix101
rderue@gilbreth-fe00:~/teaching/unix101 $
```

# Essential Commands

## List Command

- Purpose: List all the files in the given directory
- Usage: `ls [options] [directory]`
  - If a directory is not supplied as an argument, it will default to your "`pwd`"
- Common Options
  - `-l`: lists files with their most common metadata
  - `-a`: Include hidden files
    - "`.`" and "`..`" are `pwd` and parent directory respectively
  - `-t`: Sort files by their last-modified time
  - `-h`: Print size of files in human readable format

## Example usage of the `ls` command

```
rderue@gilbreth-fe00:~ $ ls -ltha
  total 426M
  drwxr-xr-x 1797 root   root    1.8K Jan 18 02:20 ..
  -rw-------    1 rderue student  26K Jan 14 02:38 .bash_history
  drwx------   55 rderue itap      92 Jan 13 16:47 .
  -rw-------    1 rderue student  16K Jan 13 16:47 .viminfo
  drwxr-xr-x    2 rderue itap       3 Jan 13 16:46 .vim
  drwxr-xr-x    4 rderue student    4 Jan 13 16:26 .matlab
  -rw-r--r--    1 rderue student 1.7K Jan 11 14:59 Untitled4.ipynb
  -rw-r--r--    1 rderue itap    3.5K Jan 11 14:59 Untitled.ipynb
```

## Change Directory Command

- Purpose: Change your `pwd` to another directory
- Usage: `cd [directory]`
  - If a directory is not supplied as an argument, it will default to your home directory (~)
  - Arguments are relative to where you are
- Common Usages
  - "`cd`": Go to my home directory
  - "`cd ..`": Go to the parent directory
  - "`cd ../../`": Go to the parent of the parent directory
  - "`cd -`": Go to your last `pwd`

## Example usage of the `cd` command

```
rderue@gilbreth-fe00:~/teaching $ ls
  unix101
rderue@gilbreth-fe00:~/teaching $ cd unix101
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  hello_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ cd ~lev
rderue@gilbreth-fe00:/home/lev $ pwd
  /home/lev
rderue@gilbreth-fe00:/home/lev $ cd -
  /home/rderue/teaching/unix101
rderue@gilbreth-fe00:~/teaching/unix101 $
```

## Make Directory Command

- Purpose: Creates a new, empty directory with the supplied name
- Usage: `mkdir [options] <directory_name>`
  - A name for the new directory must be given to the command
  - If you do not give a path to the new directory, it's assumed that you want it in your "`pwd`"
- A useful option
  - "`mkdir -p directory/with/many/subdirs`"

## Example usage of the `mkdir` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  hello_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ mkdir test
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  hello_world.sh  logs  test
rderue@gilbreth-fe00:~/teaching/unix101 $ ls -l test
  total 0
rderue@gilbreth-fe00:~/teaching/unix101 $ mkdir two/deep
  mkdir: cannot create directory ‘two/deep’: No such file or directory
rderue@gilbreth-fe00:~/teaching/unix101 $ mkdir -p two/deep
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  hello_world.sh  logs  test  two
rderue@gilbreth-fe00:~/teaching/unix101 $ ls two
  deep
rderue@gilbreth-fe00:~/teaching/unix101 $
```

## Move Command

- Purpose: Move a file to a new location or rename a file
- Usage: `mv [options] <old_file> <new_file>`
  - `old_file` and `new_file` can be file paths which allow you to move a file to a new directory
  - You can use this command to rename a file in the same location by *not* supplying a path
  - The directory you are moving the file to must exist

## Example usage of the `cp` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  hello_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ cp greetings_world.sh
greetings_world.sh.bak
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  greetings_world.sh  greetings_world.sh.bak  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ cp -r logs logs2
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
  greetings_world.sh  greetings_world.sh.bak  logs   logs2
rderue@gilbreth-fe00:~/teaching/unix101 $ ls logs
  log1.txt  log2.txt
rderue@gilbreth-fe00:~/teaching/unix101 $ ls logs2
  log1.txt  log2.txt
```

## Copy Command

- Purpose: Copy a file to a new location
- Usage: `cp [options] <old_file> <new_file>`
  - This command is extremely useful to make back-ups
  - To copy entire directories, use the `-r/--recursive` option
  - The directory you are copying the file to must exist

## Example usage of the `mv` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
hello_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ mv hello_world.sh
greetings_world.sh
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
greetings_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $ mkdir new_dir
rderue@gilbreth-fe00:~/teaching/unix101 $ mv greetings_world.sh
new_dir/greetings_world.sh
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
logs  new_dir
rderue@gilbreth-fe00:~/teaching/unix101 $ ls new_dir
greetings_world.sh
rderue@gilbreth-fe00:~/teaching/unix101 $
```

## Remove Command

- Purpose: Removes files/directories
- Usage: `rm [options] FILE ...`
  - Can delete multiple files by giving a space delimited list of files
- Useful option:
  - "`rm -r`": Deletes *recursively* any file and subdirectories contained within the given directory

## Example usage of the `rm` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
greetings_world.sh  logs  new_dir
rderue@gilbreth-fe00:~/teaching/unix101 $ ls new_dir/
greetings_world.sh
rderue@gilbreth-fe00:~/teaching/unix101 $ rm new_dir/greetings_world.sh
rderue@gilbreth-fe00:~/teaching/unix101 $ rm new_dir
rm: cannot remove 'new_dir': Is a directory
rderue@gilbreth-fe00:~/teaching/unix101 $ rm -r new_dir
rderue@gilbreth-fe00:~/teaching/unix101 $ ls
greetings_world.sh  logs
rderue@gilbreth-fe00:~/teaching/unix101 $
```

# Essential Commands

## Catenate Command

- Purpose: Concatenates the contents of the given file(s) into a single stream
- Usage: `cat [options] FILE ...`
  - Most often used for viewing a single file
- Useful option:
  - "`cat -n`": Prints out the file with line numbers which can be useful when working with scripts

## Example usage of the `cat` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ ll
total 2
-rwxr-xr-x 1 rderue student 95 Jan 18 08:58 greetings_world.sh
drwxr-xr-x 2 rderue student  2 Jan 11 12:43 logs
rderue@gilbreth-fe00:~/teaching/unix101 $ cat greetings_world.sh
#!/bin/bash
# This is just an example file for Unix101


echo "Hello world! Welcome to Unix101"
rderue@gilbreth-fe00:~/teaching/unix101 $ ./greetings_world.sh
Hello world! Welcome to Unix101
rderue@gilbreth-fe00:~/teaching/unix101 $
```

## Difference Command

- Purpose: Compares two different files and marks the line numbers where they are different
- Usage: `diff [options] FILE1 FILE2`
  - Extremely useful for files with similar structure
  - `FILE1` and `FILE2` can be directories
  - Can also be used to compare the output of two different commands using the following syntax:
    - `diff <(COMMAND1) <(COMMAND2)`

## Example usage of the `diff` command

```
rderue@gilbreth-fe00:~/teaching/unix101 $ cat -n greetings_world.sh
     1 #!/bin/bash
     2 # This is just an example file for Unix101
     3
     4 echo "Hello world! Welcome to Unix101"
rderue@gilbreth-fe00:~/teaching/unix101 $ cat -n goodbye_world.sh
     1 #!/bin/bash
     2 # This is just an example file for Unix101
     3
     4 echo "Goodbye Unix101! I hope you learned something!"
rderue@gilbreth-fe00:~/teaching/unix101 $ diff greetings_world.sh
goodbye_world.sh
4c4
< echo "Hello world! Welcome to Unix101"
---
> echo "Goodbye Unix101! I hope you learned something!"
```

# Summary Table of New Commands

| Command Description | Command | Common options |
|---|---|---|
| **Commands for navigation** | | |
| Print my location | `pwd` | |
| List files at my location | `ls` | `-l, -a, -h, -t` |
| Change my location | `cd` | |
| **Commands for arranging files** | | |
| Move/Rename a file | `mv` | `-n, -i` |
| Copy a file | `cp` | `-r, -p` |
| Remove a file | `rm` | `-r, -i, -I` |
| **Commands for using files** | | |
| View a file | `cat` | `-n, -s` |
| Compare files | `diff` | `-r, -s, -y` |
| **Command for referencing** | | |
| Print the manual page | `man` | |

# Unix 101

## What Comes Next?

# What Comes Next?

## Upcoming Seminars

- Unix 102: January 27th
  - Topics:
    - Commands for string searching/manipulation
    - Composing pipelines
    - I/O redirection
    - Introduction to scripting
- Unix 201: February 3rd
- Unix 202: February 10th

# THANK YOU

Feel free to reach out to rderue@purdue.edu with questions.

Slides are posted at:
https://www.rcac.purdue.edu/training/unix101