Logging In

The lab computers will authenticate with your NetID and password. Please log into the computer you are sitting at.
Course Goals

• The goal of this tutorial is to provide hands-on training basics of using Linux via the command line.
• It addresses people who have no previous experience with Unix-like systems, or who know a few commands but would like to know more.

• Session 1:
  • Using Linux text editors to create documents
  • Commands
  • Manipulating Files
  • Linux Environment
Introduction – Flavors of Linux at GWU

• Debian/Ubuntu
  • User friendly, most popular for workstations and Windows replacement machines
  • Uses APT package manager

• Fedora/Redhat/CentOS
  • Enterprise friendly, most popular for servers and datacenters
  • Uses YUM and RPM package managers

• SuSE
  • Enterprise and user friendly, popular in Europe
  • Uses YaST (Yet another Software Tool) package manager
Introduction – Software vs Operating System

All Linux systems generally contain the following two types of software:

• Operating system
  • For the computer
  • Liaison between computer and user

• Applications
  • Compiled applications - Matlab, Cadence, etc.
  • Programming applications - Python, C, C++, Java, etc.
Introduction – Linux Components

- **Kernel**: The heart of the operating system
  - It interacts with hardware.
  - Memory management, task scheduling and file management.
- **Shell**: The utility that processes your requests. The shell interprets the command and calls the program that you want.
- **Commands and Utilities**: eg: cp, mv, cat and grep etc.
- **Files and Directories**:  
  - All data in UNIX is organized into files.
  - All files are organized into directories.
  - These directories are organized into a tree-like structure called the filesystem.
Introduction – Rules to Live By

• Linux is case sensitive
  • Always try to use lower case
• Enter = execute, be careful what you do!
• ^ = control key
  • Many commands require CTRL+ <another key>
• TAB = the TAB key will autocomplete commands or file paths if possible
• No spaces! NO SPACES!
Commands

• The man command
• nano Text Editor
• Command Structure
• Special Features
Commands – The FIRST command

• man
  • A magical command that will tell you all about other commands
  • usage: man <name of command you want to know about>
  • man will show a description of the command you are looking up and all of the
  switches and options for the command
  • Scroll slowly using down, up arrows
  • Scroll down a page at a time using space bar (down), “b” (up)
  • Return to the beginning by hitting the “g” key, exit using the “q” key
• Exercise:
  • Type: "man cp" (no quotes) and read about the copy command
Command – nano Text Editor

type: nano <filename>.txt

• edit and modify your text file
• use command on the bottom of the window to save and exit
• Exercise:
  • Creating and Opening Files
  • Save and Exit
  • Copying, Cutting and Pasting
Command – nano Text Editor

Copying, Cutting and Pasting

• **Copy, Cut** and **Paste** a single line
  • Alt+6, Ctrl+K and Ctrl+U
  • Move multiple lines
  • Multiple Ctrl+K followed by on Ctrl+U

• **Highlighting**
  • Ctrl+6 or Alt-A at beginning, move to end
Commands – Command Structure

`command -option argument`

**command**
- Name of the program you want to run

**-option**
- not always required
- provides different behavior than the default
- You can have more than one argument and bundle them together
  - `-option1 -option2` or “`-option1option2`”

**argument**
- What the command acts upon, usually a filename
- There can be more than one argument
- Sometimes is not required

All commands are executed when you press the Enter key.
Can combine several commands on one line - separate with semicolons

Example:

```
cd $HOME; mkdir TestDir; cd TestDir; touch testfile.txt
```
Commands – Special Features

The pipe character: |

- usually above the enter key
- Uses the output of one command and inputs it into a second command

Example:

ls $HOME | grep test
Commands – Special Features

> and >>

Redirects output to a file

• > overwrites an existing file or creates a new file
• >> appends an existing file, or creates a new file

Example:

ls > dir.txt

ls >> dir_append.txt (run this command twice)
Run these commands and take note of the output:
  date
  cal 2019
  cal 3 2019
  who
  whoami
  echo This is a test!
  CLEAR
  clear
  history
Files and Directories

- File Structure
- Concepts
- Best Practices
- Manipulation
- Search and Wildcards
Files and Directories - File Structure
Files and Directories - Concepts

Pathname
- A path through the directory system
- `pwd` shows current path

/ - the forward slash
- Represents the very bottom (root) of the file system
- acts as a divider in between directories on the file system
Files and Directories – Getting Around

- `pwd`: Print Working Directory, shows you where you are
- `.` versus `..`: Your current directory versus the directory one level above
- The `~` character: Shortcut your home directory
- `ls`: list current path contents
- `ls -la`: list all details of the current path in long form
- `cd`: change directory

```bash
bash-4.4$ cd /absolute/path
bash-4.4$ cd path/relative/to/where/I/am
```
Files and Directories Exercise

• Use "cd" to move around the file system
• cd to the root of the filesystem
• Type ls
• cd to your home directory. There are two ways to do this. Use both
  
  cd $HOME  

  cd ~
Files and Directories – Best Practices

- When naming a file or folder, no spaces! No Spaces! **NO SPACES!**
- Do not use periods except to identify file extensions
  - good: testfile.txt
  - bad:
    - te.st.fil.e.t.x (This name is valid but will confuse you sooner or later)
    - This is my very special filename I want to remember.txt
- Avoid special characters
  - `\`, ``, `-`, `[`, `]`, `()`
- Use file names that help identify the file.
  - good: myresume.doc
  - bad: 1234.doc
Files and Directories – Manipulation

• Create new directories using “mkdir” command
• Create new files using text editors, output redirection, or the “touch” command

```
nano new_file

ls -al > pwd_contents
touch mytouchtestfile.txt
echo "This is my touch test file text" >> mytouchtestfile.txt
```
Files and Directories - Manipulation

Methods to view text files

`cat filename`

`more filename`

`less filename`
Files and Directories - Manipulation

To move or rename a file or directory, use the “mv” command

Move a file to another directory:

\texttt{mv} \texttt{filename} \texttt{dirname}

Move a directory to another directory:

\texttt{mv} \texttt{src\_dir} \texttt{target\_dir}

Rename a file:

\texttt{mv} \texttt{filename} \texttt{newfilename}

Rename a directory:

\texttt{mv} \texttt{src\_dir} \texttt{target\_dir}
Files and Directories - Manipulation

Copy files from one location to another using the “cp” command

Copy a file:

```
cp filename target_dir
```

Copy multiple files:

```
cp filename1 filename2 target_dir
```

Copy a directory:

```
cp -R dirname target_dir
```

Copy multiple directories:

```
cp -R dirname1 dirname2 target_dir
```
Files and Directories - Manipulation

Delete files/directories using the “rm” command

Delete a file:

```
rm filename
```

Delete multiple files at once:

```
rm filename1 filename2
```

Delete a directory:

```
rm -r dirname
```

Delete multiple directories at once:

```
rm -r dirname1 dirname2
```
Files and Directories - Search and Wildcards

Wildcards can be used to list or find files that meet criteria

* : A **wildcard** is a character that can be used as a substitute for any of a class of characters in a search

Examples:

ls *

ls list *

ls username*

ls *username
Files and Directories - Search and Wildcards

**grep** recognizes patterns in file names or text files and returns files or lines that match the pattern.
`grep <pattern> filename.txt`

`ls -la * | grep <pattern>`
- `grep -Ril <pattern>` will find text pattern inside any file
- Use quotes if looking for a pattern with a space

**find** searches a file path for filenames that match a pattern
- `find ~/ find all files in home directory`
- `find ~/ -name *.html` find all html files in home directory (case sensitive)
Linux Environment

Variables
Configuration Files
Aliases
Linux Environment - Variables

Environment variables hold values about your current Linux environment, like your text color, home directory location, etc.

`printenv`: shows your currently defined environment variables
  - `$PATH` and `$HOME` are two important variables
    - `$PATH`: list of locations with executables visible systemwide
    - Add locations to `$PATH` with "export": `export PATH=$PATH:/path/new`
  - Is `$PWD` a variable?

`unset`: command to remove a variable
  - `unset THISISMYVAR`
Linux Environment - Configuration Files

**.profile**: stores commands and variable definitions you want every time you log into a shell session. Considered a system-wide file.

**.bashrc**: the same as .profile but it only runs when logging into a BASH session. Considered a local file.
Linux Environment - Aliases

Use the “alias” command to create command aliases (shortcuts) for commands that are too long to type repeatedly

- Print a list of aliases: `alias -p`
- Create a new alias: `alias new_ls='ls -las'`
- Remove an alias: `unalias new_ls`

Add aliases to your `.profile` or `.bashrc` to use them every time you login.
Exercise 4. Linux Environment

- Find the value of your $USER variable with `printenv`

Test your `.bashrc` file
- add "ls -la" to your `.bashrc`
- add "export MYTESTVAR=HELLO" to your `.bashrc`
- Close the terminal window and then open a new one
- Type "printenv"
- Confirm you can see your new variable
Questions or Requests