Introduction to Linux

Jerry Ebalunode
Computational Research and CyberInfrastructure Support Initiative (CRCISI)
University of South Carolina
Columbia, SC
Overview

- Linux
  - What is Linux?
  - Why use Linux?
- Basic Commands
- Working with Files & Folders
- Text Editors
- I/O Redirection & Pipes
- Introduction to BASH
- Open Lab
Log into your accounts
  o Accounts already created for you on the desktop
  o username = “vip01” thru “vip14”
  o Password = vip
The Role of an Operating System (OS)

- software & data that manages computer hardware resources.

- provides a platform for running applications on desktops, servers, clusters.
What is Linux?

Linux is an OS just like Windows or Mac OS X

- Technically speaking, Linux is the kernel: the program in the system that allocates the machine's resources to the other programs that you run. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux

- Under development since 1991, started by Linus Torvalds (Finnish software engineer)

- Why create Linux?
  - Personal computers were becoming popular
  - Needed compatibility with UNIX (IEEE POSIX)
  - Microsoft’s DOS was too limiting
  - Commercial UNIX was expensive
  - Academic-use-only MINIX was restrictive
Today, different versions of the Linux OS are called “distributions,” and there are lots of them (over 100):

Each one offers a unique combination of features and applications to suit needs of different users.
Distrowatch.com provides news, comparisons, popularity ranking of various Linux distributions.

*H.P.D = hits per day
General features of Linux:

- Most distributions are **free**
- Open-source (completely customizable)
- Portable to nearly any hardware platform
- Highly scalable to lots of cores, or lots of memory
- Highly efficient, therefore useful for computation
- Robust and proven security model
- Includes a complete development environment
Linux can be a full-featured, user-friendly OS...
  - i.e. graphical user interfaces (GUIs)

But in High Performance Computing (HPC), the command-line interface (CLI) is the most common way to access & use the OS.

Therefore, knowing how to complete tasks from the command line is critical.
Use your web browser to download intro2linux.zip archive from the tutorial web page to your home directory where you can begin working with it: then

Open up a gnome terminal
- Multiple ways to do this:
  - Click on the gnome-terminal icon on the top panel
  - Or
    - On mouse: right click and select terminal and click

To immediately begin working with tutorial files in your gnome-terminal
- cd “to_my_download_directory”
- unzip intro2linux.zip
- cd intro.linux
Basic Linux Commands

- `pwd` – prints your current working directory
- `whoami` – prints the name of the current user
- `who` – prints a list of all users who are logged-in
- `hostname` – prints the name of the system
- `date` – prints the current date and time
- `ps` – prints snapshot of current shell processes
- `ls` – list the contents of the directory you’re in
- `env` – list all environment variables/settings
- `df` – prints summary of disk usage
Some commands accept “arguments” that change the behavior of the command, or tell the command exactly what to do.

- `df -h` – prints “human readable” disk usage
- `echo $USER` – prints contents of a variable
- `mkdir [name]` – creates a new directory
- `cd [name]` – change directory (move into that directory)
- `cd ..` – back up/out of the directory you’re in
- `cd ../..` – back up 2 levels/directories
- `which [command]` – shows any command’s full path
Here are some commands that are useful for working with files and folders:

- **cp** [file1] [file2]  
  - create a copy of a file
- **mv** [file] [destination]  
  - move (or rename) a file
- **rm** [file]  
  - delete a file (rm -r [dir] for a folder)
- **file** [file]  
  - print the type of file
- **more** [file]  
  - read a text file, one “page” at a time
- **less** [file]  
  - similar to more, but a little better
- **head** -n [file]  
  - print the first n lines of a file
- **tail** -n [file]  
  - print the last n lines of a file
- **cat** [file]  
  - print the contents of a file to the screen
Nearly all commands available for use on a particular system have an accompanying “manual page”:

- `man cp`
- `man ls`
- `man python`
- `man man`
- `apropos [topic]`
  - or `whatis [topic]`

Note: To exit the manual page (man page) viewer
- simply type the letter `Q`.

Use the “up” arrow to scroll through commands you’ve used.

You can view the entire history of commands you have used by executing
- `history`
Nearly all Linux distributions come with a variety of text editors for writing and editing files.

Some of the most common are nano, vi, vim, and emacs.

Using nano

- Example:
  - `nano hello.txt` - opens a file called hello.txt for editing
  - [write something]
  - `CTRL-o (^o)` to save
  - `CTRL-x (^x)` to exit nano
By default, command line programs print to “stdout” (standard out = the computer monitor).

I/O redirection is a way of manipulating the input/output of Linux programs, allowing you to capture the output in a file, or send it to another program.

Get the first 9 words from the dictionary:

- `head -9 dictionary.txt > temp.txt`
- `head -n 9 dictionary.txt > temp.txt`
- `more temp.txt`
- `wc -l temp.txt`

- The “>” character performs a “redirect,” taking the output of the head command and putting it into the file temp.txt.
Use “>>” to append to a file without overwriting:

- `export DATE=`date`
- `echo “Right now it’s $DATE” >> temp.txt`

Another useful technique is to redirect one program's output (stdout) into another program's input (stdin). This is done using a “pipe” character.

- `cat dictionary.txt`
- `cat dictionary.txt | grep ing`
- `cat dictionary.txt | grep ing | grep un`

FYI, `my_linux_command` returns or paste the results of a linux command as the argument rather than the command itself.
More advanced, but very useful commands to try:

- **grep error [file]**
  - searches a file for lines containing “error” and prints them to stdout

- **tar -cvzf [compressed_archive].tar.gz [directory]**
  - “tars” (like “zipping”) a directory into a single compressed file,

useful for file transfers.

- **scp [file] username@server:path_to_destination**

useful for directory or folder transfers.

- **scp -r [dir] username@server:path_to_destination**

**scp => Secure Copy. Used to copy a file or folder or directory to another computer where you have a user account.**

Also,

- `scp username@server:path_to_remote_file path_to_destination_file`
- `scp -r username@server:path_to_remote_dir path_to_destination_dir`
BASH also known as Bourne-again shell

The BASH shell (command line interpreter) is an open-source version of the original UNIX Bourne shell.
  o Allows users to type commands which cause actions
  o Typically run in a text window

Usually the default shell in a Linux environment

Similar to Explorer in Windows, or Finder in Mac OS X

Uses specific syntax (like $ to indicate variable names)

Need to use a different shell? Just run it: /bin/csh

(Type exit or CTRL-D to return to your previous shell)
Multiple commands can be issued in sequence using a script. Create a new file containing these lines and run the file like it’s an executable:

```bash
#!/bin/bash
cd $HOME
tar -cvzf example.tar.gz intro.linux
mkdir dustbin
mv example.tar.gz ./dustbin
cd dustbin
tar -xf example.tar.gz ; mv intro.linux newdir
ls newdir > contents.txt
cd $HOME
```
A simple FOR loop:

```
#!/bin/bash

for i in $(seq 1 10)
do
    echo -n This is iteration $i
    echo -n " and the time is "
    date +%T
done
```

Another way to do same trick

```
#!/bin/bash

for ((i=1; i<=10; i++))
do
    echo -n This is iteration $i
    echo -n " and the time is "
    date +%T
done
```

See file: loop1.sh

See file: loop2.sh
A simple FOR loop:

```bash
#!/bin/bash

for i in $(seq 1 10)
    echo -n This is iteration $i
    echo -n " and the time is "
    date +T
done
```

```bash
#!/bin/bash

for ((i=1; i<=10; i++))
    do
        echo -n This is iteration $i
        echo -n " and the time is "
        date +T
    done
```

Missing "done" command
FOR loop with conditional if:

```bash
#!/bin/bash
for i in $(seq 1 10)
  do
    echo -n This is iteration $i
    if [ $i -eq 5 ]
      then break
    fi
  done
```

Another way to do same trick

```bash
#!/bin/bash
for ((i=1; i<=10; i++))
  do
    echo -n This is iteration $i
test $i -eq 5 && break
  done
```

See file: flow-control-loop1.sh
See file: flow-control-loop2.sh & flow-control-loop3.sh
Every time you log-in, the `.bashrc` script in your home directory is executed.

You can add lines to the bottom of this file to run additional, custom commands when you log-in.

After editing this file, you can execute the commands in this file using the `source ~/.bashrc` command.

An example of a customized `.bashrc` file can be found here:

/`opt/tutorials/intro.linux/bashrc.example`
AWK: a programming language for processing text-based data in files or data streams.

- `ls | awk '{print "mv \" $1 \" \" $1 \".new\" }' | bash`
  - Causes files or directories to be renamed with a “.new” suffix

sed: (“stream editor”) a UNIX utility for parsing text files and implementing textual transformations.

- `sed 's/old/new/g' input.txt > output.txt`

- `ls -l *txt* | awk '{print "mv \"$1\" "$1\"} | sed s/txt/blah/2 | bash`
Control access to files & directories by setting permissions

- `cd intro.linux`
- `ls -l`
  - `-rwxr-xr-x jebalunode public 622783 2010-12-03 09:15 dictionary.txt`
  - `-rwxr-xr-x jebalunode public 8262 2010-12-03 09:15 icb.txt`
  - `-rwxr-xr-x jebalunode public 891777 2010-12-03 09:15 personnel.txt`
  - `-rwxr-xr-x jebalunode public 6599 2010-12-03 09:15 theraven.txt`

Setting permissions using read/write or executable:

- `chmod +r [file]` --makes a file readable
- `chmod +w [file]` --writes to the file are permitted
- `chmod +x [file]` --makes a file executable
- `chmod +rwx [file]` --makes a file executable, writable and readable

For directories you apply the recursive “R”

- `chmod -R +r [dir]` --makes a directory readable
top - will list processes/tasks running on your system
  q or CTRL-c can help you get “unstuck”

**cd** - will return you to your home/login directory

basename - strip directory and suffix from filenames
  basename we_are_young_and_old _and_old
  * output “we_are_young”

**tr** - translate or delete characters
  echo linux | tr [a-z] [A-Z]
  echo linuxx | tr [a-z] [A-Z] | tr -s [X]
  echo linuxx | tr [a-z] [A-Z] | tr -d [E]

**TAB** - completion

“Full path” to a location in the file system (/ vs. ~/)

Change user or group ownership of a file:
  chown [userid] [file]
  chgrp [group name] [file]

Find a file:
  find ./* -name "name_of_my_file.txt"
  locate name_of_my_file.txt

Type **exit** to close your shell.
Requirements

- Have Java installed
  - See do I have java URL
- Have firefox, or microsoft explorer or google chrome installed
- Goto USC VPN site to install VPN
  - For students the url is https://sslvpn.sc.edu/students
For faculty or staff go to https://sslvpn.sc.edu/facstaff
Connecting to USC Linux Systems from Home

1. Enter your network username and password, then click Sign In.

2. Wait while the initial setup begins. Please note that this may take several minutes.

3. Click Start.

4. Wait for the VPN client to launch for the first time.

5. A new window will pop up. Once the status shows Connected you have successfully logged into the VPN client.

6. To end your current VPN session right-click on the Juniper icon in your system tray and select Sign Out.

This is when the juniper vpn client (NC Connect) is installed if not done b4
Take a few minutes to try some of the commands you’ve learned. Perhaps try combining commands to give you very specific results.

If you have not done so already, use your web browser to download intro2linux.zip archive from the tutorial web page to your home directory where you can begin working with it: then
- `cd “to_my_download_directory”`
- `unzip intro2linux.zip`
- `cd intro.linux`

execute the commands you learnt

run the for loops

**Fun Exercise**
- use your “bash kungfu” to rename 50 files
  - `input1.old` – `input50.old`
  - `INPUT1.new` - `INPUT50.new`
Acknowledgements

Computational Research and CyberInfrastructure Support Initiative
- Professor Robert Sharpley
- Dr. Nikolai Sergueev

Research Computing Team
- Andrew Yancey
- Glenn Dufour
- Frank Bhakit
- Steven

Funding:
- USC VPR office
- NSF EPSCoR RII Track-1 & Track-2