Linux Command Line

Dr. Gowtham
Director of Research Computing, IT
Adj. Asst. Professor, Physics and ECE
EERC B39 · (906) 487-3593 · g@mtu.edu · http://hpc.mtu.edu

Notations

Definition

Lorem Ipsum is dummy text of the printing and typesetting industry

Trivia

Did you know lorem ipsum has been in use since the 1500s?

Command (to be run in a Terminal; one per line; press ENTER key)

```
echo "Lorem Ipsum, eh?"
```

***Do at home* exercise**

Can you make more lorem ipsum on your own?

Warning

Potential pitfall ahead ... things can go lorem ipsumly wrong

Notations

`john`	Username
`john@mtu.edu`	Email address
http://lmgtfy.com	URL
`colossus.it.mtu.edu`	Server name
`hello_world.cpp`	File name
`hello_world()`	Function name
`# Prints "Hello, World"`	Comment
`print "Hello, World!";`	Code
`rm -rf *`	Command

Replace `john` with your own Michigan Tech ISO username.

Linux

Linux is a Unix-like and mostly POSIX-compliant computer operating system assembled under the model of, and a prime example for concept and practice of, free and open source software development and distribution.

The underlying source code may be used, modified, and distributed – commercially or non-commercially – by anyone under licenses such as the GNU GPL.

Linux, like UNIX, is user friendly but ...

It is picky as to who its friends are, and often very unforgiving of mistakes. It prefers friends to be committed to mindful practice, and be sensitive to case, space, and other weird characters.

Linus Benedict Torvalds (1965 – present): Finnish American software engineer

colossus.it.mtu.edu and guardian.it.mtu.edu

* Intel Xeon X5675 3.07 GHz, 24 CPU cores, 96 GB RAM
* Appropriate for light- to medium-weight computations
* Accessible by all Michigan Tech users from anywhere via SSH

Command to SSH into Colossus or Guardian

```
ssh -Y john@colossus.it.mtu.edu
ssh -Y john@guardian.it.mtu.edu
```

* Linux workstation in a campus lab/office
  * May not be as powerful as `colossus.it` and `guardian.it`
  * May not be directly accessible from off-campus

Replace `john` with your Michigan Tech ISO username.
All machines managed by Michigan Tech IT run Red Hat Enterprise Linux 7.x and will mount your campus home directory.

Commands

Use `man COMMAND` for more information. For e.g., `man mkdir`

Basic

`cat`, `cd`, `clear`, `cp`, `date`, `echo`, `finger`, `grep`, `head`, `history`, `less`, `ls`, `man`, `mkdir`, `more`, `mv`, `pwd`, `rm`, `rmdir`, `tail`, `touch`, `vim`

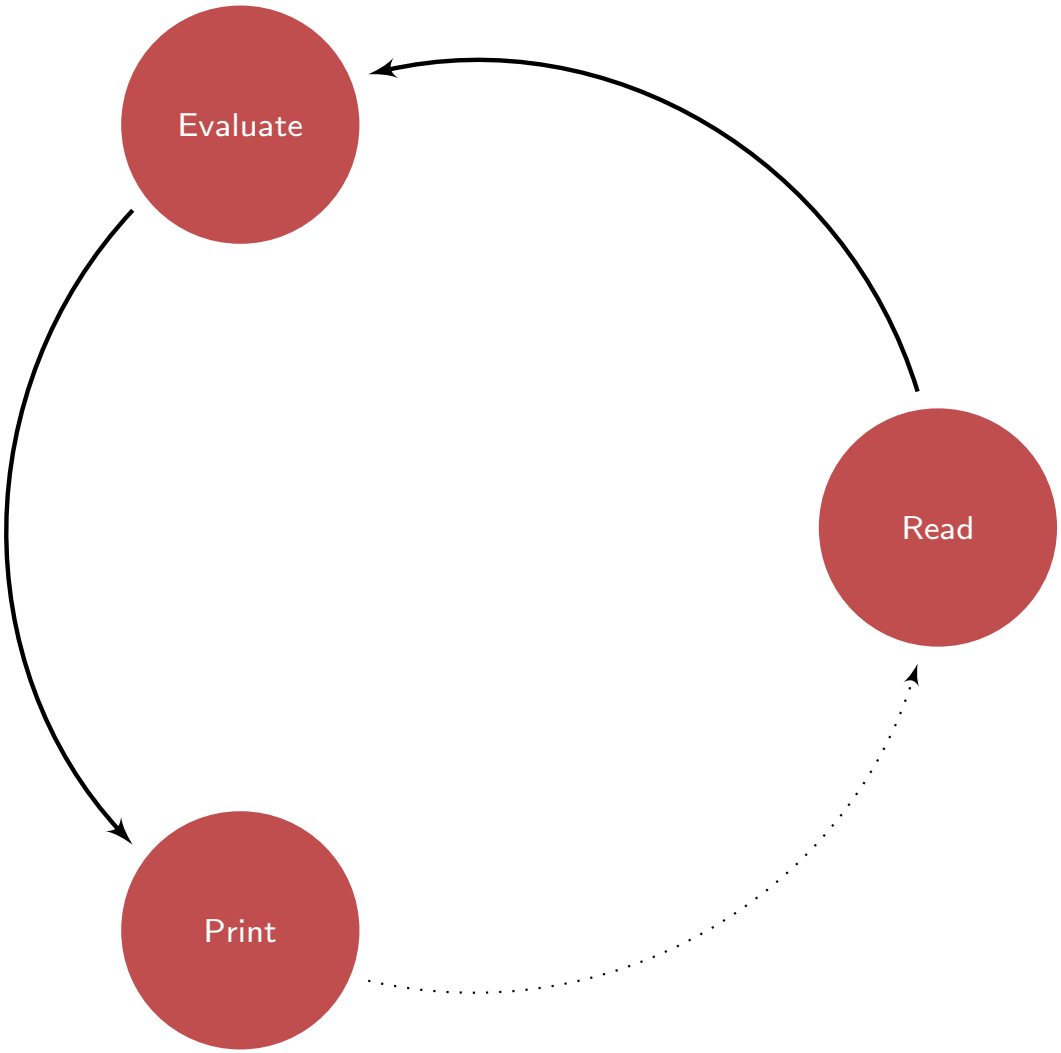
Intermediate

`awk`, `basename`, `bc`, `bzip2`, `chgrp`, `chmod`, `chown`, `comm`, `crontab`, `cut`, `df`, `diff`, `du`, `env`, `expect`, `expr`, `file`, `find`, `free`, `gzip`, `hostname`, `id`, `kill`, `killall`, `ln`, `locate`, `paste`, `ping`, `ps`, `rsync`, `scp`, `sdiff`, `sed`, `seq`, `sleep`, `sort`, `ssh`, `tar`, `time`, `top`, `tr`, `ulimit`, `uniq`, `wc`

Advanced

`groupadd`, `groupmod`, `groupdel`, `ifconfig`, `mount`, `passwd`, `poweroff`, `reboot`, `su`, `uptime`, `umount`, `useradd`, `usermod`, `userdel`

READ, EVALUATE, PRINT loop (REPL) Core of the Command Line Interface (CLI)



The READ, EVALUATE, PRINT loop (REPL) continues until the user decides to log out.

The username

* Uniquely identifies a user in a system
* Contained in the reserved variable, `$USER`
* Same as your Michigan Tech ISO username

Commands to identify the username

```
id -un
whoami
echo $USER
```

The home directory

* Default location when a Terminal is opened
* Contained in the reserved variable, `$HOME`
* Naming convention can vary but almost always includes `$USER`

Commands to identify the home directory

```
echo $HOME
cd ; pwd
```

Returning home (from anywhere in the file system)

```
cd
cd ~/
cd $HOME
```

The shell An interface between humans and the operating system

Why use it?

While using the GUI seems easier, the often repeated shell commands – which seamlessly interface with a plethora of other utilities – can be saved as a script or a function. This not only saves time and effort, and prevents errors, but also to naturally extends the system's capability.

Is there more to the *shell*?

Although most users think of the shell as an interactive command interpreter, it is really a programming language in which each statement runs as a command. Because it must satisfy both the interactive and programming aspects of command execution, it is a strange language, shaped as much by history as by design.

– Brian Kernighan and Robert Pike

The shell

* Contained in the reserved variable, `$SHELL`
* Choice of shell (BASH, TCSH, etc.) depends on personal preference
* BASH is the default shell in most modern linux distributions
* When BASH is invoked as an interactive login shell, (settings in) the following startup files are executed in the following order:

  `/etc/profile` → `$HOME/.bash_profile` → `$HOME/.bash_login` → `$HOME/.profile`

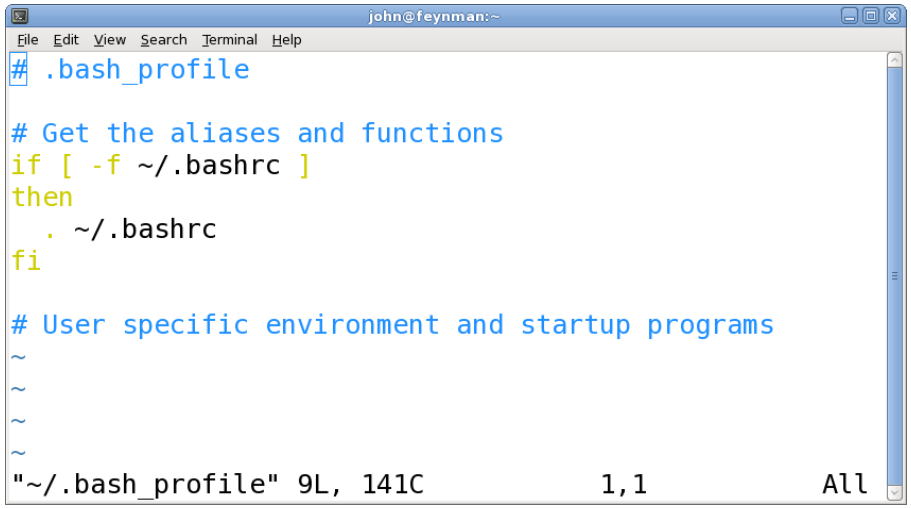
Command to identify the shell

```
echo $SHELL
```

Request Michigan Tech IT to have your default login shell changed to BASH.

Customizing the shell

* Open a Terminal
* Create/Edit `$HOME/.bash_profile` using `vi` (or `gedit`) editor



* Save and close the file

Customizing the shell continued

* Backup `$HOME/.bashrc`, if it exists
* Edit `$HOME/.bashrc` using `vi` (or `gedit`) editor

```
john@feynman:~
File  Edit  View  Search  Terminal  Help
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]
then
  . /etc/bashrc
fi

# User specific aliases and functions
if [ -f ~/.bash_$USER ]
then
  . ~/.bash_$USER
fi
~
"~/.bashrc" 13L, 174C                    2,0-1         All
```

* Save and close the file

Customizing the shell continued

* Create/Edit `$HOME/.bash_$USER` using `vi` (or `gedit`) editor

```
john@feynman:~
File  Edit  View  Search  Terminal  Help
# .bash_$USER
# User specific aliases, variables, functions, environment and startup programs

export HISTCONTROL=ignoreboth
export HISTSIZE=100000
export HISTFILESIZE=100000
export HISTTIMEFORMAT="%b %d %H:%M:%S "
export TMOUT=360000

export PATH="/bin:/usr/bin:/sbin:/usr/sbin:/usr/kerberos/bin"
export PATH="$PATH:/usr/local/bin:$HOME/bin"
export LD_LIBRARY_PATH="/lib64:/usr/lib64:/usr/kerberos/lib64:$HOME/lib"
export MANPATH="/usr/share/man:$HOME/man"

export COLOSSUS="colossus.it.mtu.edu"
export GUARDIAN="guardian.it.mtu.edu"

alias cl="clear"
alias ll="ls -lh"
alias lr="ls -ltrh"
alias la="ls -a"
alias colossus="ssh -Y $COLOSSUS"
alias guardian="ssh -Y $GUARDIAN"
~
~
~
~
"~/.bash_john" 26L, 815C                                  3,0-1         All
```

* Save and close the file

Customizing the shell continued

Reserved shell variables

EDITOR, FUNCNAME, GROUPS, HOME, HOSTNAME, IFS, LD_LIBRARY_PATH, LOGNAME, MACHTYPE, MANPATH, OLDPWD, OSTYPE, PATH, PPID, PS1, PS2, PS3, PS4, PWD, SECONDS, SHELL, TMOUT, TZ, UID, USER, USERNAME

* Carefully redefine a variable or extend its definition
* Add additional customizations (aliases, variables, functions, etc.) to `$HOME/.bash_$USER` as necessary

Command to enforce the changes

```
. $HOME/.bashrc
```

`env` command provides a complete list of variables already in use.
Open a Terminal and create `$HOME/bin`, `$HOME/lib`, `$HOME/man` directories using `mkdir` command.
OS looks for commands, libraries and manual pages in `PATH`, `LD_LIBRARY_PATH` and `MANPATH` respectively.

Files, folders and symbolic links

Commands

```
cd $HOME
touch file_01.txt file_02.txt
mkdir folder_01 folder_02
ln -s file_01.txt file.txt
ln -s folder_02 folder
touch .hidden_file
mkdir .hidden_folder
```

* Entities that start with `.` are hidden
* Hidden entities do not appear in `ls` or `ls -l`
* Use `man ls` to learn how to list all entities

Ownership and permission

```
[john@feynman ~]$ ls -l
total 8
-rw-r--r-- 1 john users    0 Aug 20 14:27 file_01.txt
-rw-r--r-- 1 john users    0 Aug 20 14:28 file_02.txt
lrwxrwxrwx 1 john users   11 Aug 20 14:28 file.txt -> file_01.txt
lrwxrwxrwx 1 john users    9 Aug 20 14:29 folder -> folder_02
drwxr-xr-x 2 john users 4096 Aug 20 14:28 folder_01
drwxr-xr-x 2 john users 4096 Aug 20 14:28 folder_02
[john@feynman ~]$
```

- Entity type: normal file (-), directory (d), link (-), socket (s)
- Ownership levels: user (u), group (g), others (o)
- Permission levels: read (4, r), write (2, w), execute (1, x)

Open a Terminal and type `ls -l`.
Permission level values add up at each ownership level.
Ownership and permission can be changed using `chown` and `chmod` commands respectively.
`file_01.txt` and `file_02.txt` have 644, `folder_01` and `folder_02` have 755, and `folder` and `file.txt` have 777.

Do at home exercise

Changing file/folder permission using alphabet approach

```
chmod u=rwx file_01.txt
chmod g+rw,o-rwx file_02.txt
chmod g+x,o-x folder_01
chmod u-x folder_02
```

Changing file/folder permission using number approach

```
chmod 744 file_01.txt
chmod 660 file_02.txt
chmod 754 folder_01
chmod 655 folder_02
```

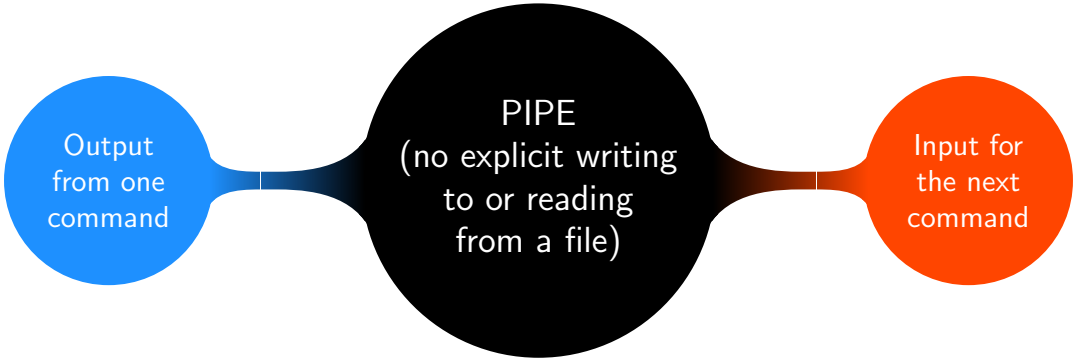
Learn more about the concept of ownership and permissions in Linux OS.
Run `ls -l` after each command to observe the changes. Reset the permission to original values after each approach by using `chmod 644 file_01.txt file_02.txt` and `chmod 755 folder_01 folder_02`.

Running more than one command

Piping

The act of treating the output of one command as the input for a subsequent command without needing to create (and as such, keep track of and later remove) a temporary file.

| represents the pipe character
\ represents the continuation character



It is still a good practice to develop the workflow with explicit writing and reading of intermediate results. Replace them with pipes iff the workflow repeatedly produces the desired result.

Running more than one command

Piping examples

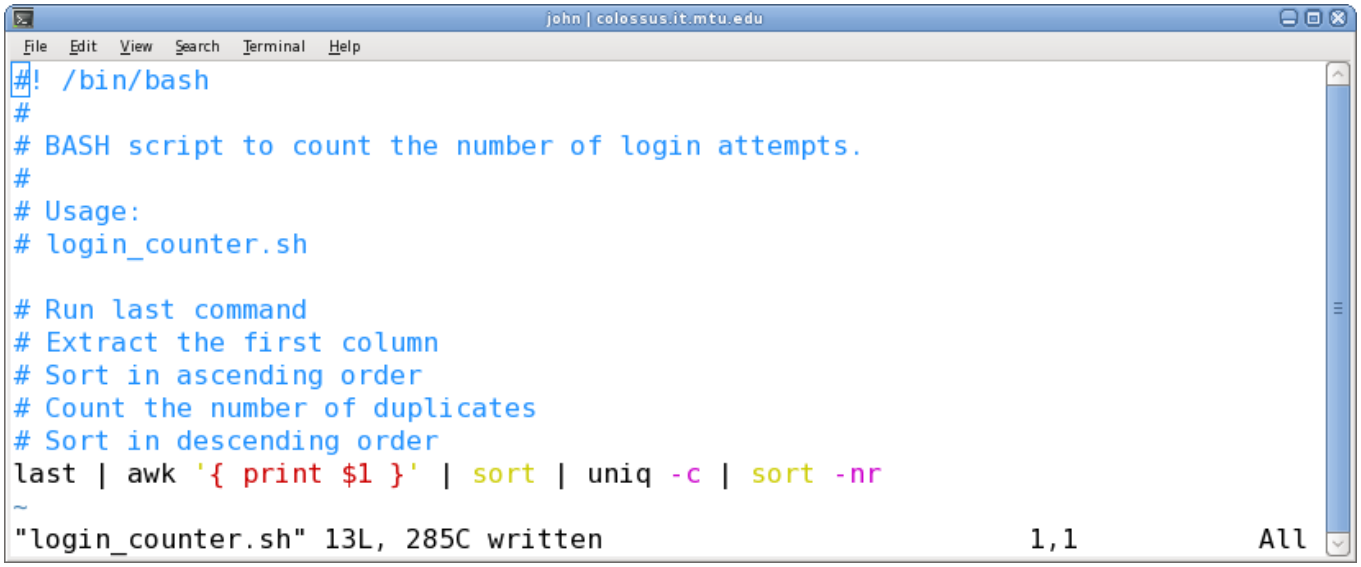
```
du | sort -nr
ls -l | grep '^.....w'
ls -l | tail -n +2 > file.txt
ls -l | sed '1d' >> file.txt
cat file.txt | wc -l
ps aux | grep $USER
echo "scale=15; 4*a(1)" | bc
find .  -type f -iname "*.txt" | xargs ls -l
find .  -maxdepth 2 -mtime +1 -type f | xargs ls -l
seq 1 1 100 | awk '{ sum += $1 } END { print sum }'
last | awk '{ print $1 }' | sort | uniq -c | sort -nr
ls -l | tail -n +2 | sed 's/\s\s*/ /g' | \
 cut -d ' ' -f 3 | sort | uniq -c
```

Run each set of commands incrementally and observe the output.

Remembering every command and its option

Shell script

A set of (piped) commands, to accomplish a given task, saved in a file with meaningful name and comments for easier (repeated) execution. It helps automate the workflow, reduce the chances of errors, and make time for more productive activities.



Save a copy of all shell scripts in `$HOME/bin` folder. A shell script requires 755 (or at least 700) permission to run.

Do at home exercise

Develop a personalized yet consistent file naming convention

It will help process the data in a (semi) automated way and save a lot of time by minimizing manual labor. Preferably, use alphanumeric characters periods and underscores in file/folder names. Parsing other special characters including space can be tricky.

`for` and `while` loops

Suppose that a pattern, say the first occurrence of `MAGFIELD = ##.##`, needs to be extracted from one hundred data files (`filename_001.dat` – `filename_100.dat`) and saved in `summary.txt`.

Write a well-commented BASH script, `extract_magfield.sh`, to accomplish the above task using a `for` or `while` loop. `awk`, `grep`, `ls`, `sed`, `touch`, `truncate` can come in handy.

Do at home exercise

Adding up numbers in a given sequence

`seq` command can generate the number sequence between `A` and `C` in steps of `B`. Common usage of this command is as follows:

```
seq A B C
```

Write a well-commented BASH script, `sum_numbers.sh`, to find the sum of all the numbers in such a sequence for a given combination of `A`, `B` and `C`. Think about (and implement if you can) ways to guard against invalid user input.

What if the numbers in a sequence aren't always integers?

List (and implement if you can) modifications, if any, necessary to accommodate a sequence (or just the step size) of rational numbers?

Do at home exercise

Timing a command or a script

When prefixed with any command or a script, `time` command prints the relevant timing information. Common usage is as follows:

```
time COMMAND
time SCRIPT
```

Time (`time` and `/usr/bin/time`) the following commands and scripts:

```
whoami
ls -l | sed '1d' | wc -l
sum_numbers.sh
login_counter.sh
extract_magfield.sh
```

`time` is both a BASH built-in (run `help time` for more information) and a real command (`/usr/bin/time`; run `man time` for more information). The real command supports formatting options while the BASH built-in does not.

Additional references

* Linux | The Linux Command Line | The Command Line Crash Course
* BASH Introduction/Scripting/Programming
  Introduction | Basic scripting | Advanced scripting
* Linux, BASH scripting and Gnuplot Tips
* Handy one liners: `awk` | `sed`
* Vi(m) editor: Interactive Tutorial | Reference
* Twitter: @CLIMagic | @Linux | @LinuxFoundation | @Linux_Tips
  @MasteringVim | @RegExTip | @UNIXToolTip | @UseVim
  @VimLinks | @VimTips

A really good and effective way to learn Linux command line quickly is imposing upon yourself to use it for accomplishing as many, if not all, tasks every single day until it starts becoming second nature.

Need help?

Contact Dr. Gowtham to schedule an appointment

EERC B39 · (906) 487-3593 · g@mtu.edu