User Linux Course

web: abdolhosseini.iut.ac.ir/content/prerequisites-computational-projects

Instructor : Paddy Doyle (TCHPC)

Getting started with the shell

Logging out/closing the shell

ctrl-d   exit

changing password

passwd

pwgen ← examples of reasonably good passwords

First Commands

ls ← list

LS ← case sensitive

Username and user ID

whoami

user and Group IDs

id

who else is logged in

who

who logged on most recently

last

clearing the screen

ctrl-l

Root ← The superuser

Command-line Arguments

last joe

command-line options → hyphen character (-)

last -10

last -u   u ← user

Combining Command-Line options

ls -l -a ← shows hidden files

ls -a -l ← shows long listing of files

ls -la

ls -al

short form   Long form

ls -l --all

1

Command-line Completion

```
ls E<TAB>
ls Examples
```

Command History

history ←

↑
↓ ← allow you to move up and down through your history

Searching the command history

ctrl-r

ctrl-c & ctrl-g ← exit this mode

Selected Useful Editing Features

move to

left-arrow	left
right-arrow	right
ctrl-a	beginning of line
ctrl-e	end of line
up-arrow	pervious command
down-arrow	next command

Delete:

Backspace	previous character
ctrl-w	previous word
ctrl-d	next character
ctrl-k	rest of line

Getting Help

Command-line Help

--help ← you need a -i or -n switch,

id --help

id [option] .... [USERNAME] ← short and long Form options in message

Getting Help : man

man ls ← standard help command

Type q to exit man

Searching the manual pages

man ← Exact name of the command

man password X

man -k ITEM

man -k password ← search for keywords in the man pages

Directory Management

Viewing File Listings with ls

ls -l ← long list

ls -a ← hidden file

ls -t ← sort most recent first

ls -r ← reverse the current sorting mechanism

ls -tr

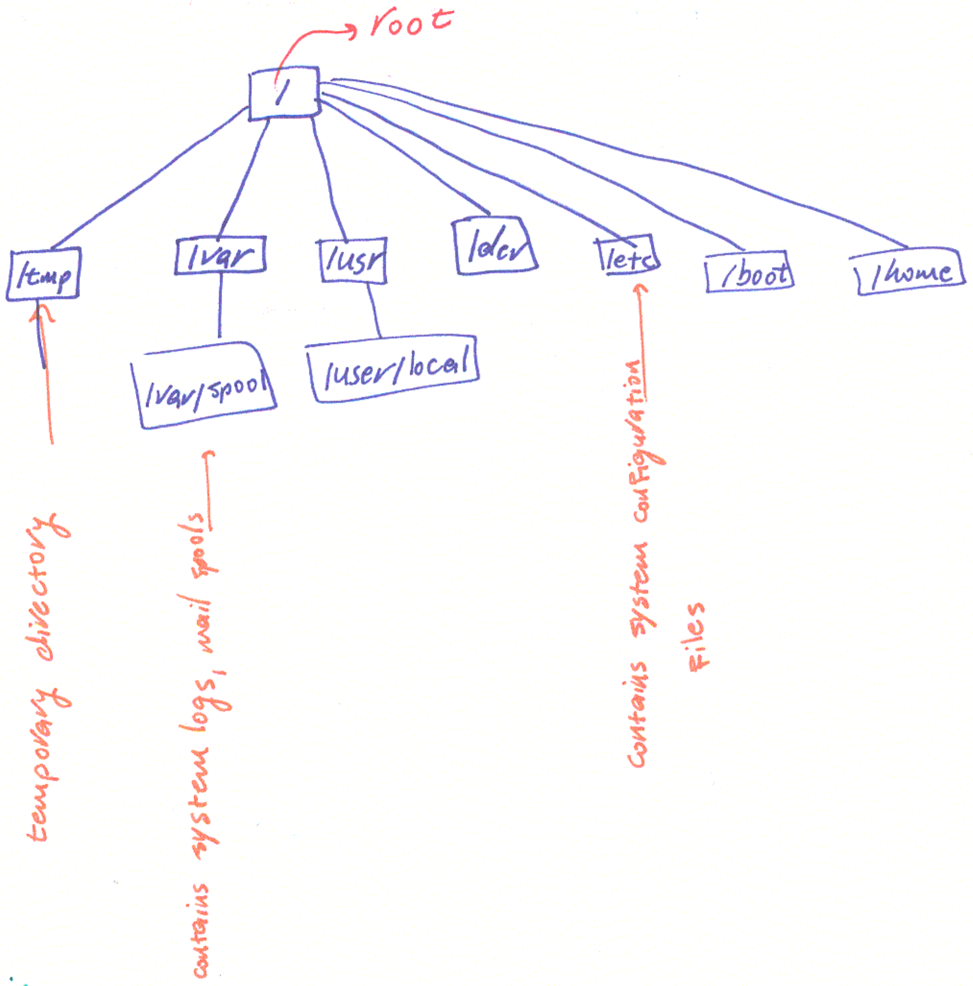
ls -ltr

which directory you are in

pwd

Typical Linux Filesystem Organisation

Removing or renaming any of them will cause your system to behave incorectly or halt



/bin
/sbin
/usr/bin
/usr/sbin

/usr/local/bin
/usr/local/sbin

← These location contain the programs that the user may want to use.

unique id is known as the path to the file

/home/sarsari

cd (change directory) ← go home

cd <path> ← go <path>

cd . ← refers to the current directory itself

cd .. ← refers to the paren directory (or one level up)

cd ~ ← tilda = home

cd = cd ~

in the / directory leaves you in / ← cd..

~ ← is translated by the shell into "the users home directory"

Case sensitivity

- most linux filesystem are case sensitive
- not all windows or mac filesystem are case sensitive.

↓

General rule : don't rely on case sensitivity when naming files or directories

- creating Directories

mkdir

mkdir -p archives/2008/oct ← creating multiple levels of Directories

- Removing Directories

rmdir

rm -rf

checking disk usage for files/directories with

du [option]... [FILE]...

du -h ← outputs file sizes in human readable form

du -s ← summarises the output, only showing the total at the end

du -k ← prints the output in kilobytes

du -m ← prints the output in megabytes

du -hs

Getting information about file systems

df ← displays the amount of disk space available on all
file systems on the current system.

df -h → human readable form

Viewing Text Files

Viewing file contents with

cat [files] ← displays the contents of a file (or list of files) to you terminal

(no edit)

[concatenate] ⟶ cat

is perfect for viewing short files

↓

long files ?

cat F₁

cat F₂

cat F₁ F₂

Less ⟨File⟩ ← can be used to view long files

Type q to exit less

less Commands

up-arrow, down-arrow ← move up & down the output line by line

pageUp, pageDown ← " " page by page

/ Forward Search

? Reverse search

n Finds the next match

ctrl-g Display information at the bottom of the terminal window about the file and your location within it

q quit the paper

man pages as similar as less pages

Files & Directories

Regular Files ← are mostly what are on your machine

Directories ← contain a list of file names

Creating Regular Files

touch ← Empty new files maybe created
↖ will also update timestamp of files

Deleting Regular Files

rm ← There is no "are you sure" promp! ▲
← There is no recycle bin! ▲

Deleting Files with a promp

rm -i File1

type n or y

rm testdir/tdfile2 ← It takes path arguments

Recursive Directory Removal with rm

rmdir ← delet empty directories

rm -rf <directory name> ← remove directories containing files

▲ Never use rm -rf *

Moving Files & Directories

mv

file → file ✓ ← rename

file → directory ✓

directory → file ✗

directory → directory ✓

mv —⚠→ will overwrite files without warning

mv -i file1 file2 ← moving with prompt

mv file1 dir1
- → dir1 does not exist → a file called dir1 is created
- → dir1 does exist → will be overwritten unless write protected

mv dir1 dir2
- → dir2 does not exist → rename
- → dir2 does exist → moving

Copying Files & Directories

cp

cp file1 file2

cp -i file1 file2 ← Copying with a promp

cp file1 dir1

cp dir1 dir2 ✗

cp -r dir1 dir2 ✓

cp -r dir1 dir2 $\xrightarrow{\text{dir2 exists}}$ $ ls dir2
dir1

cp dir1 file1 ✗

cp -r dir1 file1 ✗

mv and cp ← Can take multiple arguments if the last argument is a directory.

← accept Complex path

Getting information about Files

ls -l

The link count　Group　Date and Last modification time

```
drwxr-xr-x  3  joe  tchpc  4096  Aug 25  19:31  Directory 1
```

File permissions　owner　Size of file in bytes　Name of file.

ls -a ← hidden files

.bashrc
.bash_profile

Many application place configuration information in hidden files

Ownership and Access Permissions

r w x	r w x	r w x
User	Group	other

Changing File permissions
Directory

chmod [ugo][+-][rwx]

chmod u+rw,g-w file1 ← No space after comma

permissions for Directories [search bit]

Executive permission ← in order to list or open individual files by name

ls /usr/bin/gpg → requires that the execute bit be set for the ~~directories~~ directories /, /user/ and /usr/bin

```
chmod u-w .; ls -ld
rm file1
rm: cannot remove fil1 : permission denied
```

suggested Directory permissions

```
chmod go-rwx ~
```

Recursion:

```
chmod -R
```
← permissions on directories ~~for other~~ and all their files and subdirectories

```
chmod -R o-r
```

Octal permissions

```
chmod 777 file1
```

$\boxed{7}$ → 4+2+1 rwx

←

4	r
2	w
1	x

7	rwx
5	r-x
6	rw-

1	2	3	4	5	6	7
x	w	wx	r	rx	rw	rwx

umask and new files

umask ← set when you log in

value of mask is octal and mirrors the octal number associated with permissions

$$\begin{array}{r} 0777 \\ 0002 \\ \hline 0775 \end{array}$$

umask ←

↙ default

changin file ownership

```
chown <usr>:<group> file1
chown mary:group2 file1
```

Files

regular files

directories

symbolic links ← Unix provides for shortcuts or pointers to files and folders, through the use of symbolic links (also called soft links)

```
ln -s file1.txt linktofile1

ls -l linktofile1
```

- single program to be linked in various directories
- single copy of the file exist
- we can link to directories as well

Text and Binary files

text files ← containing ASCII characters

binary files ← contains non ASCII " ← executable files resulting from compiling code.

file ← Determining file type

pattern

ls foo*	matches any string.
ls ?oo	matches any single character.
ls f[aeiou]o	matches any one of the enclosed characters.
ls f[abcd]o	
ls f[a-d]o	
ls [[:lower:]]*	
ls [[:upper:]]*	

12

Text editors

gedit ← a graphical editor similar to Notepad

vim ← a powerful Command_line text editor

Vim stands for Vi IMproved

Vi stands for VIsual editor

$$\begin{tikzcd}
 & \text{Insert mode} \arrow[d, bend right, "\text{Esc}"'] \\
\text{vim} \arrow[ur] \arrow[r] & \text{Command mode} \arrow[u, bend right, "\text{click a or i}"']
\end{tikzcd}$$

Essential vim Commands

Esc	Exit out of Insert mode, into Command Mode
u	Undo
ctrl-r	Redo
i	Enter into insert mode
:w ⟨RETURN⟩	Save
:q ⟨RETURN⟩	Quit (exit)
:wq ⟨RETURN⟩	Save and quit
ZZ	Save and quit

Cursor Movement

j	move the cursor down one line
k	〃 up one line
h	〃 left by one character
l	〃 right 〃
b	〃 left by one word
w	〃 right by one word
gg	move to the first line in the file
G	〃 last 〃
^	move to the start of the current line
$	〃 end 〃 〃

Searching

/term <RETURN>	Search forward for term
? term <RETURN>	Search backward for term
n	Repeat last search operating
*	Search forward for the word under the cursor
:noh <RETURN>	Turn off syntax highlighting (can get annoying)
%	Go to matching parentheses {} () []

Editing

x	Delete the character under the cursor
dw	Delete the current word
cw	Delete the current word and enter into Insert Mode
dd	Delete the current line
cc	〃 and enter into Insert Mode
yy	Coppy the current line
p	past the text in the buffer

Edit/Replace

:s/t_1/t_2/ <RETURN>	Replace first "t_1" with "t_2" on current line
:s/t_1/t_2/g <RETURN>	Replace every "t_1" with "t_2" on current line
:%s/t_1/t_2/g <RETURN>	Replace every "t_1" with "t_2" on every line

Repetition and muliplication

vim editing commands can be repeated by hitting the "." key (the period) when in command mode

vim editing command can be multiplied by ~~ty~~ typing a number before the command 10dd

Viewing Differences between File Edits

diff file1 file2 | more

diff -R ← rove compare directories

The Process Environment

* Each running program has a list of environment settings

name = value

PATH = /bin:/usr/bin:/usr/local/bin,

Common Environment Variables

variable	Example	purpose
HOME	/home/joe	the home directory of the current user
PATH	/bin:/usr/bin:/usr/local/bin	Defines the set of directories that the shell should search to find an executable file called without a full path-name
SHELL	/bin/bash	The current shell
LOGNAME	joe	The current login name
TERM	xterm	The current terminal type

/bin:/usr/bin:/usr/local/bin

To find the ls executable, it searches each of these directories from left to right to find /bin/ls

```
$ which find        ← which tells where the path is
  /usr/bin/find
```

Setting Variables

bash makes it possible to assign values to variables, using the syntax

```
$ name = value
$ myvar ='a string'
$ echo $myvar
$ a string
```

Setting Environment Variables export

export ← To convert a shell variable into an environment variable, use the export command

Environment variables | normal shell variables

upper case | are passed on child processes

printenv ← prints the environment it inherits

```
:
TERM: xterm
SHELL: /bin/bash
```

Setting the PATH

In terminal → This is often needed when new executable are installed in a non standard directory

```
export PATH : /bin/: /sbin/: /usr/bin: /usr/sbin
```
← ✗ have corrupted your path

correct way: `export PATH = $HOME/bin : $PATH`

In .bashrc

add in .bashrc → `export PATH = $PATH : /user/support/bin`

source .bashrc

or

```
export PATH = $PATH : $HOME/bin: /home/users/sarsari
        /application/yambo-3.2.4-r.855/bin
```

bash Aliases

creating Aliases

```
sarsari $ alias Lt = 'last -10'
```

we can add this line in .basrc

source .bashrc

Running processes and capturing output

ps ← Viewing Running Processes

ps aux(ww) ← Display every process on the machine with wide output

ww → wide output

ps aux

ps auxwwf

ps code	STAT
D	uninterruptible sleep (usually IO)
R	Running or runnable (on run queue)
S	Interruptible sleep (waiting for an event to complete)
T	stopped, either by a job control signal or because it is being traced
W	paging (not valid since the 2.6.xx kernel)
X	dead (should never be seen)
Z	Defunct ("zombie") process, terminated but not reaped by its parent

A Dynamic, Real Time View of Running processes

top

command	Argument	Purpose
h, ?	← none	shows help information
u	<username>	limit the processes displayed to those whose effective uid is <username>
M	none	sort by % memory usage
P	none	sort by % cpu usage (the default)
B	none	toggle bold font display

Stopping a process

ctrl - c

kill

The kill command

```
kill [-signal | -s signal] pid ...
```

The kill command (ctd) ...

SIGNAL	numeric value	purpose
KILL	9	this signal may not be blocked, and terminates the processs
TERM	15	usually used to tell the process that it is about to be terminated giving the process a chance to write out to disk before it is terminated
STOP		This signal may not be blocked, this is usually used to "pause" a running process
CONT		tell the process to continue if stopped otherwise ignore
HUP	1	usually used to tell the process to reload its configuration files and continue running or restart itself.

```
kill -KILL 4321
```

Command Redirection

```
ls > manifest.txt
```

Appending output

```
ls >> manifest.txt
```

Error Redirection

The Bit Bucket

Using Input Redirection

```
gnuplot < input.gnuplot
```

this creats a new PostScript file which we can view with the gv command

Using Pipes to Combine Commands

output of one command | (pipe) input of another

```
ps auxww | less
ls -al /etc | less
history | less
last | less
```

A Filtering Text with grep

```
grep <searchstring> <filename>
grep Hamlet Hamlet.txt
```

```
ps auxww > ps.txt ]
grep joe ps.txt   ]  ps auxww | grep joe
```

More grep pipe Examples

```
du -h ~ | grep conf

history | grep Example
```

Multiple pipes

```
du -h ~ | grep conf | less

ps auxww | grep joe | grep bash
```

Running jobs in the Background (The ampersand. &

```
gedit myFile.txt &
```

The jobs Command

jobs ← is a bash command which prints all jobs running in the background in the current terminal window

```
[1] + Running gedit myFile &
```

Foreground ← not in the background

```
$ gedit myFile
  ctrl-z
  jobs
  bg %2
```

Command	Argument	purpose
ctrl-z	none	suspend current foreground job
jobs	none	list jobs running/suspe in current terminal
bg	job number: bg %n	place job n running in the background
fg	job number: fg %n	place job n running in the foreground
kill	job number: kill %n	kill job n

The grep Command $\longrightarrow$ Case sensitive Matching

grep ← very useful command for filtering text

```
grep 'include' main.c
```

Case Insensitive grep

```
grep -i
```

Getting Numbers for Matching Lines

```
grep -n
```

Omitting what you Don't want

```
grep -v
ls | grep foo
ls | grep -v foo
ps auxww | grep -v root
who | grep -v joe
```

Searching for strings in multiple files

```
grep 'include' *.c
```

Recursive grep

```
grep -r <searching string> <directory>
grep -r foo .
```

← to find all files containing a string in the current directory and in all subdirectories

Show Only The Files which Contain a Match

grep -l sleep *

Show a Context For the Matched pattern

grep -c <num> <searching string> <File> ...

grep -C3 caeser Hamlet.txt ← ..to see 3 lines before and after the match..

Using Basic Regular Expressions with grep

grep

ps auxww | grep ^root — ^ match a pattern only at the start of the line

ps " | grep bash$ — $ match a pattern only at the end of the line

. Matches any single character

^$

[...]

\w matches any single alphnumeric character

? The preceding item is optional and matched at most one

* " will be matched zero or more times

+ " " one or mor times

{n} " is matched exactly n times

{n,} " " n or more times

{n,m} " " at least n times, but not more than m times

Counting Words / Line with wc

word count
↑
```
wc  Hamlet.txt

wc -l  Hamlet.txt
```
← How many lines
```
ps auxww | grep peabody | wc -l
```
← How many processes
```
ls -l | grep ^d | wc -l
```
← How many subdirectories

The sort Command

```
who | sort

wc -l  Hamlet.txt

sort -u  Hamlet.txt | wc -l
```
Remove duplicate lines

```
sort  Hamlet.txt | uniq -c
```
← How many times each line is repeated

Sorting Numerically
```
sort  Hamlet.txt | uniq -c | sort

sort  Hamlet.txt | uniq -c | sort -n
```
← -n to tell it to sort numerically
```
du * | sort -n
```

The tail Command

```
du * | sort -n | tail

du * | sort -n | tail -5
```

The head Command

```
head -50  Hamlet.txt
```

Offline Searching with locate

locate foo

Realtime Search with find

man find

find <starting directory> -nam <file>

find . -name 'foo.txt'

find / -user joe — all files on system directory owned by joe

find ~ -size +1M — all files in your home directory bigger than 1 MB

find ~ -size -1M — 〃 smaller 〃 〃

find / -amin -1 — 〃 that were accessed less than 1 minute ago

find / -amin +1

find / -type d -name 'etc' → find all directories named 'etc' on system

find / -type d -name 'etc' -printo

find / -type d

The xargs command

xargs ← can be used with any command whose output is directed to its ~~standard~~ input / It is often used with find

find . -name 'f*' | xargs ls -l ← { preform a long listing on all files starting with 'f' below current directory

find . -name 'f*' | xargs grep 'conten' — find all files begining with 'f' which contain the string 'conten'

find ~ -name '*.txt' | xargs wc -l — find all files in your home directory ending in ".txt", and count how many lines each one contains

find ~ -name '*.txt' $\boxed{\text{-printo}}$ | xargs -0 wc -l

-printo ↙ to be seper when filenames contained spaces

-0 → pipe the output of xargs

Compression Utilities

gzip

bzip

tar

gzip ← Regular files can be compressed with this program.

gzip foo.txt ⟶ foo.txt.gz

gunzip foo.txt.gz

bzip2 ← offers better compression than gzip

bzip2 foo.txt ⟶ foo.txt.bz2

bunzip2 foo.txt.bz2

tar

tar COMMAND [OPTIONS] archive file.tar [FILE_OR_DIR] ...

tar Command	Description
c	creat an new archive
x	Extract files from an existing archive
t	List the contents of an existing archive
f	specify to work with a file, rather than a tape drive
v	Use verbose output – i.e., print all filenames to the terminal when creating/extracting

```
tar cvf Examples.tar Example
gzip Examples.tar
tar czvf Examples.tar.gz Example
tar tzvf Example.tar.gz
tar tvf  Example.tar
tar xzvf Example.tar.gz
tar xvf  Example.tar
tar xjvf mysoftware.tar.bz2
```

tar archive by using the t

z replace with j