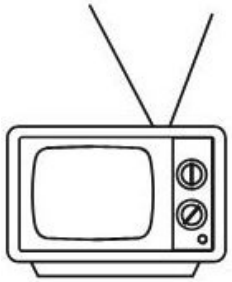


Previously On... CIS4930

- Syllabus review
- A few word on what's Linux and open source
- More than a few words on Command Line Interfaces

Welcome to Day #2 of the “Surprise Course”

Quick Announcements

Module Section in Canvas

- “slides” posted there
- Reading assignments – to supplement slides
- Remember to take the First Week Quiz!
- Final Exam Matrix

TAs are available

- See URL for Amir (we are working on getting a room)
- Canvas message directly Dan NGuyen until specific hours are set

Module

M01
Basic Usage

Menu for this module

CLI Essentials

- Introduction to the Linux Command Line Interface and its Bash shell.
- We will cover the basics of navigating the file system (more about this topic in the module dedicated to the Linux File System),
- as well as managing processes.

Getting Help

- "Give a man a fish and he is fed for a day, teach him to fish...".
- When it comes to Linux, "learning to fish" boils down to learning to RTFM.
- We are going to learn to use the help tools available in any Linux system.

Topic

What is interacting with Bash all about?

https://youtu.be/Y2gvVXG3f-c

Basic Notions

What is Bash?

- Bourne Shell (sh) → Bourne Again Shell (bash)
- There are alternatives…
  - zsh on MacOS
  - Korn shell
  - C shell
  - TENEX csh
  - Friendly Interactive sh

Where is it used?

- Bash used in Linux / UNIX…
- On windows: Cygwin, git bash…
- On MacOS (used to be Bash, now it's Zsh)

What is the shell interpreting exactly? Commands!

"Commands" == Built-in commands

- The shell is the one interpreting them
- echo
- pwd
  - (The shell's prompt shows the current working directory)
- type
  - type pwd
  - type type
  - type date
  - type -a date

Shell	Full Shell Name	# of built-in commands
sh	Bourne Shell	18
ksh	Korn Shell	47
csh	C-Shell	55
bash	Bourne Again Shell	69
tcsh	TENEX C-Shell	87
	FreeBSD Shell	97
zsh	Z Shell	129

“Commands” == Keywords: if while for …

- Also interpreted by the shell itself

“Commands” == Functions: skip until we script

- You guessed it; also interpreted by the shell itself

“Commands” == External Commands

- date
- type date
- type -a date
- Let’s go meta → bash or sh

Topic

Bash Aliases

https://youtu.be/vIGK7A3i6_Q

Let us look at a few aliases

- Look at the alert alias
  - aliases == 1-line scripts
- Look at the ls alias
  - Commands have options
    - One letter (BSD Style)
    - One letter (standard) –a –b –c
    - Full word --help --version
  - Examples
    - ls –l
    - ls –a → Hiding files with dot
- Order does not matter & we can put them together
  - Ls –a –l → ls –l –a → ls –al → ls –la
  - → order does not matter when these are toggles
- Bash is case sensitive
  - ls –a
  - ls –A
- Full word options
  - ls --help –version

Defining our own Aliases

- Aliases can be removed
  - unalias ls
  - Test it to show lack of color!
- They can be defined
  - alias ls='ls --color=auto'
- They go away when closing the shell
  - Close shell
  - Reopen it
  - Check for alias presence
  - → see bash config files later

- What can we (re)define aliases on?
  - → external commands
  - → builtins

    alias type='type -a'

View aliases as pre-processing of the command line string before we determine whether we are going to execute a builtin or an external command.

Side remark: use single quotes for now

alias stuff=‘echo ‘ ‘	The single quote is interpreted as closing the first single quote	>
alias stuff=$‘echo \‘’	$’ is a special notation allowing ANSI escaping inside the single quoted string Escapes the \’ to ’ Alias tries to echo ‘ but this is an unclosed string so PS2 appears	stuff >
alias stuff=‘echo ‘”’”	Concatenating a single quote in double quotes The single quote is added, same problem than above	stuff >
alias stuff=’echo’ “\’”	Concatenating an escape single quote in double quote The alias try to do echo \’ This results in displaying the actual single quote	stuff ‘
alias stuff=“echo \’”	Using double quotes instead works	stuff ‘
alias stuff=“echo ‘”	Again, this would result in the alias expanding to echo ‘ which lacks a closing single quote	stuff >

Quick Announcements

Register on Piazza

- Demo on Canvas
- All announcement about the course will be there instead of in traditional Canvas announcements

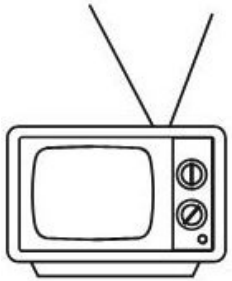
Supplements to address lack of textbook

- Reading assignments in Canvas
- URL of web resources directly in the slides to expand on them
- URL to videos developed for the online / IT version of this course (when appropriate)

Quick Announcements

GQ01 in the queue to release

- Check availability and due date in Canvas syllabus section
- Timed, proctored, 1 attempt only
- Syllabus says textbook is allowed but for this semester this means closed book
- Personal work only



Previously On... CIS4930

- Types of Commands interpreted by the shell
  - Builtins / keywords / functions / external commands
- Syntax for 1-letter and full-word options
- Pre-processing applied to your command string
  - aliases
  - It can get more complicated: alias stuff="echo this ain't easy"

Alias w/ multiple commands in one line

- Semicolon
  - date ; echo “is a nice day”
  - date ; echo “date command worked”

- What if it didn’t work?
  - date –meow ; echo “date command worked”
  - → we want to insert a conditional here…

https://youtu.be/vIGK7A3i6_Q

Conditional Sequential Execution

&&

- Review on shortcut evaluation of boolean expressions in C, Java, Perl…
- Application to shell:
  - date && echo “it worked”
  - date --meow && echo “it worked”

||

- date –meow || echo “it did not work”

How does it know?

- Concept of exit status
- echo $?

Executing commands in subshells

Setup

- TAG="I am the original"
- echo $TAG

Group commands in Subshell

- (echo $TAG)
- (TAG="not sure anymore" ; echo $TAG)
- echo $TAG

Group commands in current shell

- { echo $TAG ; }
- { TAG="what about now“ ; echo $ TAG }
- echo $TAG

Topic

FileSystem Concepts

https://youtu.be/j1l_C0b1ZK4

Absolute vs relative pathnames

- pwd → path name
- Root == origin
  - Not using letter drives like in windows (multiple origins)
  - In Linux / == One root to rule them all
  - Where are all the disks / partitions?
    - mount
- absolute path
  - Refers to folders → ls /home → ls –al /home
  - Refers to files
- Relative path
  - Relative to what? → CWD
  - ls filehere
  - ls folder/filethere
  - . and .. Special folders / notations

Examples

- ls .
- ls ..
- ls ../tux/
- ls /home/tux/../../home/./././tux

Topic

Moving around in the Linux filesystem

https://youtu.be/k3hokNCQwPw

Moving around w/ builtin commands

- pwd

- cd absolute_pathname
- cd relative_pathname

- cd ~

- cd –

Concept of Directory Stack

- type dirs pushd popd
  - Also built-in commands
- Viewing the stack
  - dirs → note that the first entry is always the CWD
  - dirs -l
  - dirs -p
  - dirs -p -l → note that -pl does not work here (builtin)
- Adding to the stack
  - pushd -n /some/where
  - pushd /some/where
  - dirs –p
- Removing from the stack
  - dirs -p
  - popd +0 -n
  - …

Directory Stack access

- dirs –v

- dirs +0
- dirs +1

- dirs -0
- dirs -1

Topic

Creating & removing files & folders

https://youtu.be/xoOAsbnjHJw

Let's start with files

- touch existingFile
  - → modifies date
  - Check it with → `ls -l`
- touch newFile
  - → creates empty file
  - Check it with → `ls -l`
- Removing existing file
  - rm existingFile

What about creating folders?

- Creating empty folder
  - mkdir something
  - mkdir something/else

- mkdir COP3353/m01 → FAILS
  - mkdir --help
  - Check out the –p option

```
mkdir -p COP3353/m01
mkdir -p COP3353/m02/slides/23/
```

- Visualizing hierarchy
  - tree
  - tree -d

What about deleting folders?

Setup

- mkdir COP3353/m01

rmdir COP3353 → warning non empty folder

rm –rf COP3353 → will work

WATCH OUT!!!!

- Uis use TRASHES, CLI does not → watch out
- Not a safe delete either though → forensic / shredding

Topic

Copying, moving & renaming files & folders

https://youtu.be/wc6yR8dbby4

Copy a single file

- cp COP3353/readme.md backup
- Absolute + relative pathnames usable!

Rename one file on the fly

- cp COP3353/readme.md backup/cop3353.md

Copying multiple folders or files

- cp COP3353 COP2512 COP2513 ./backup/
  - → omitting folders
  - cp –r COP3353 COP2512 COP2513 ./backup/
  - → multiple sources and one destination → we cannot rename just copy

What about renaming?

- Use 1 source + 1 destination only
- cp –r COP3353 ./backup/CIS4930

mv

mv something something-else CEN6084/

- Multiple sources + 1 destination

mv something COP4610/something-entirely-different

- 1 source + 1 destination → allows renaming on the fly

Renaming without moving

- mv COP3353 CIS4930

Topic

Bash processes management

https://youtu.be/6N7bNvKCJtM

- So far
  - Execute 1 thing at a time
  - Wait for it to complete
  - Enter the next command

- Let's simulate a process that runs longer than instantly
  - type sleep
  - sleep 4

- If command takes a while, you might want to still be using your shell
  - → execute the command in the BACKGROUND
  - As opposed to FOREGROUND

- sleep 5 &
  - Gives us a job ID then a PID
  - Job is specific to the bash where you started → show 2 shells side by side
  - PID is global to the system

Long-duration processes should be launched in BG

- xeyes → bad
- xeyes & → oh yeah, sweet sweet 90's tek

- Actually, gedit does this automatically for us

Controlling foreground processes w/ shortcuts

- ^C → terminates the process
- ^Z → freezes the process instead of terminating it

Example

- xeyes something ^Z
- sleep 30 &
- jobs
- fg %1 → put back in foreground
- ^Z → ok we froze it again
- bg %1 → ok but how to put it in background?

Topic

Monitoring Processes & Playing Nice

Monitoring Processes / PIDs

Basic syntax:

- ps -e
- ps -f
- ps -eo pi,ppid,ni,comm

Tree views

- ps -ef --forest
- pstree

Live Monitoring

- top
  - forest mode available too: V to enable
  - v to toggle displaying children for a given process
- htop
  - Use F5 for forest view

Prioritizing Processes

- Niceness of a process
  - From -20 to 19, default is 0
  - -20 is highest priority
- Priority of a process
  - PRI = 20 + N in [0:39] or [100:139] for kernel ([1:99] is for real-time)
- Start a process w/ niceness value != 0
  - nice -n 5 run_my_backups.sh
  - nice -n -20 do_this_right_now.sh
- Renice to change it later on, dynamically
  - renice -n -20 –p 70899
  - Try to renice at -20 ☺

https://www.tecmint.com/set-linux-process-priority-using-nice-and-renice-commands/

Topic

Sending signals to processes

https://youtu.be/vSLvhQtAGV4

Sending signals to job IDs or PIDs

- kill → does not KILL processes, sends them SIGNALS
- kill %1 → ok this one kills the process, it's the default
- kill 8834 → works with PID too

Show me the signals!

kill -l

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

The signals **SIGKILL** and **SIGSTOP** cannot be caught, blocked, or ignored.

- kill 8834 → sends SIGTERM #15 by default
  - SIGTERM asks process to terminate politely (process may refuse)
  - SIGKILL #9 is less polite and does not wait for accept
- kill -KILL 8981

- SIGSTOP
  - kill –STOP 9078 → same as ^Z
  - kill –CONT 9078 → same as fg/bg

- Alternative Syntaxes
  - kill –s CONT 9078
  - kill -19 9078 → usual kill -9 5555

Topic

Processes Lineages & Signals Propagation

Celtschk @ https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and

A simple Experiment…

- Launchme.exe &
- jobs
- [kill the terminal window]
- ps -ef |grep launchme.exe [in another terminal]

[note: if we just exit-ed the shell, it would not send SIGHUP to its bg jobs]

Run process in foreground (started from an interactive shell, connected to a terminal

So let's assume you've just typed foo:

https://en.wikipedia.org/wiki/SIGHUP

- fork → The process running foo is created.
- The process inherits stdin, stdout, and stderr from the shell.
  - Therefore, it is also connected to the same terminal.
- If the shell receives a SIGHUP, it also sends a SIGHUP to the process
  - (which normally causes the process to terminate).
- Otherwise, the shell waits (is blocked) until the process terminates or gets stopped.

Run process in background with &

- The process running foo is created.
- The process inherits stdout/stderr from the shell (so it still writes to the terminal).
- The process in principle also inherits stdin, but as soon as it tries to read from stdin, it is halted.
- It is put into the list of background jobs the shell manages, which means especially:
  - It is listed with jobs and can be accessed using %n (where n is the job number).
  - It can be turned into a foreground job using fg, in which case it continues as if you would not have used & on it (and if it was stopped due to trying to read from standard input, it now can proceed to read from the terminal).
  - If the shell received a SIGHUP, it also sends a SIGHUP to the process. Depending on the shell and possibly on options set for the shell, when terminating the shell it will also send a SIGHUP to the process.

Celtschk @ https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and

Disown removes the job from shell's job list

- → all the subpoints above don't apply any more
  - (including the process being sent a SIGHUP by the shell).

- However, it still is connected to the terminal
  - so if the terminal is destroyed, the program will fail as soon as it tries to read from standard input or write to standard output.

Celtschk @ https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and

nohup separates process from the terminal:

- It closes standard input
- It redirects stdout and stderr to file nohup.out
- It prevents the process from receiving a SIGHUP (thus the name).
- Does not remove the process from the shell's job control
- Does not put it in the background

Celtschk @ https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and

Topic

The less pager

https://youtu.be/hM42QDeO7Ic

- less something.txt
  - Up / down / pg up / pg down / enter / space
- h → help page
  - ^ symbol means control
- q → quit
- Searching with / and ?
- Running a shell command with !date
  - ! Is referred to BANG
  - BANG command

Topic

Manpages structure

https://youtu.be/W-keag_QSOo

Getting help

- Just type wrong command
  - mkdir
  - → displays help message (usage)
- Ask for command's help message (usage)
  - ls --help
- help
  - → tells you about all builtin commands
  - help type

manpages

- The “online” manual
  - Online as “right here”
  - Not online as on the web
- man ls
  - Less pager is used to display the page
  - All shortcuts are available
  - Q when done
- When learning a new command, check out the manpage to get a feel of what’s available. Do not memorize the whole thing, this will happen with time automatically

Sections of a manpage

Section	Content
Name	Name & purpose of the command
Synopsis	Syntax of the command
Description	Full description of the command
Environment	Env variables related to the command
Author	Who done it
Files	Files related to the command
See Also	Other manual entries related to this command
Diagnostics	…
Bugs	Known bugs

Topic

Searching through Manpages

https://youtu.be/AS858I02Pzs

Sections of the manpage manual

Section #	Name	Notes
1	Commands	Not including bash built-ins (see help)
2	System calls	fork, execv, kill, …
3	Library Functions	printf, scanf…
4	Special Files	
5	File Formats	
6	Games	☺
7	Miscellaneous Information	
8	System Administration	mount, …

Intro pages & Specifying sections numbers

There are intro pages for each section

- man intro → manpage summarizing section 1 (commands)
- man 1 intro
- man 2 intro → system calls

So…We can specify section numbers!

- man kill → shows section 1 by default
- man 2 kill → shows manpage for system call

whatis

whatis ls

- → gives the short description that appears at top of manpage
- man ls → to show it

whatis kill

- → shows that kill appears in 2 sections of the manual

Does the same as man –f

- man –f ls
- man –f kill

apropos

apropos

- Searches for keywords in the one-line description of the command in the manpage

apropos manual

- Example: we get everything talking about the manual

Same as

- man –k manual

A quick apropos experiment :)

- apropos list directory contents → bunch of sols
- apropos “list directly contents” → only ls and related

- Same when searching with google
- Due to bash using spaces as separators between arguments to its commands already so you need to enclose a space-featuring string in double quotes so that it’s taken as one argument