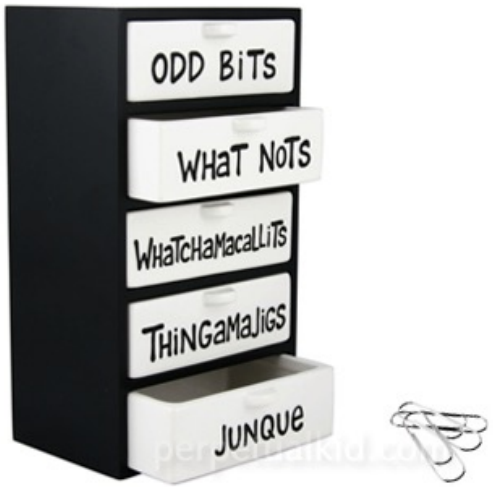
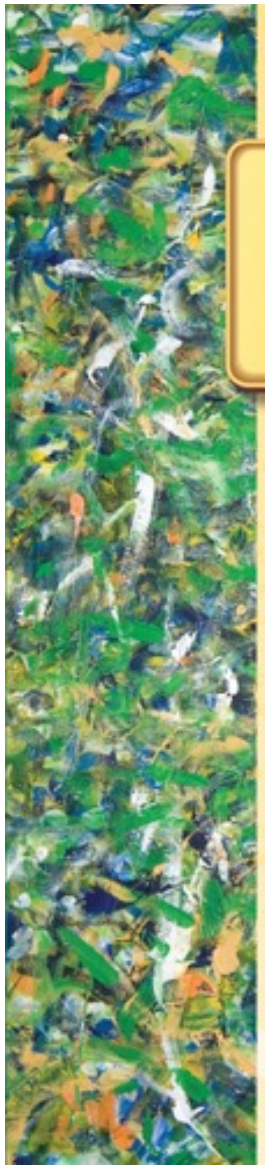
Quick Announcements



- Feedback on Virtual Box
  - cut & paste → Guest Additions
  - folder sharing → Extension pack
  - Come to office hours (TAs or mine) for help with this

    https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox
- Feedback on exam:
  - Some stuff unclear → Review solutions together (end of session)
- Feedback on the slides
  - Level of details insufficient → working on it
  - Book recommendations? → see next slide



HARLEY HAHN'S GUIDE TO UNIX AND LINUX

HARLEY HAHN

STUDENTS AROUND THE WORLD ARE LEARNING UNIX FROM HARLEY HAHN

Harley Hahn, one of the most respected names in the Unix community, demystifies Unix for students, professionals, system administrators, and everyday users.

HARLEY HAHN'S GUIDE TO UNIX AND LINUX is written in Hahn's crystal-clear, engaging style with more than 1,000 hands-on examples.

Major topics include:

- What Is Unix? What is Linux?
- The Unix Work Environment
- The Online Unix Manual and the Info System
- Command Syntax
- The Shell (covers Bash, Korn Shell, Tcsh, C-Shell)
- Standard I/O, Redirection, Pipes
- Filters
- Regular Expressions
- Displaying Files
- The vi Text Editor
- The Unix Filesystem
- Directories and Files
- Processes and Job Control

Harley Hahn is an internationally recognized, best-selling author, having sold more than 2,000,000 books. Hahn has an undergraduate degree in Mathematics and Computer Science from the University of Waterloo (Canada), and a graduate degree in Computer Science from the University of California at San Diego. This is his 32nd book and 7th Unix book.

Visit Harley at his Web site:
www.harley.com

Visit his cat Little Weedly at:
www.weedly.com

"From novice to expert in a single volume."
— Sanjiv Bhatia, University of Missouri, St. Louis

"Even after 18 years of using Unix, I was able to learn from this book."
— Michael Schuster, Solaris Kernel Engineer, Sun Microsystems

"Thorough, understandable explanations. The Unix book you've been looking for!"
— Ron Thomson, Central Michigan University

Thanks to Harley's books, I was able to become fluent in Unix. Hahn's style is so entertaining, you'll forget you are learning. Nobody explains Unix —or anything— better than Harley Hahn.
— Alan Colmes, Host of "Hannity and Colmes", Fox News & "The Alan Colmes Show", Fox News Radio

"Thorough, complete, witty, and relevant, the perfect text for your Unix or Linux course!"
— Don Southwell, Delta College, Michigan

"Detailed, authoritative, and easy to read. We have been teaching from Hahn's Unix books for over a decade. Students will use Harley Hahn's Guide to Unix and Linux as a reference long after graduation."
— Ivan Bajic, San Diego State University

"Accessible to the beginner, yet thorough enough that even experienced Unix users will find things they didn't know."
— James Hellotis, Rochester Institute of Technology

The McGraw-Hill Companies

McGraw-Hill Higher Education

ISBN 978-0-07-313361-4
MHID 0-07-313361-2

www.mhhe.com

Topic

M2T3
Bash Initialization Files

https://youtu.be/DDYw1b29wk8

Before we get started….

Whose shell are we running?

→ you can have multiple bash shells running for any given user

Where are the configuration files located?

- `~/` → specific to a given user
- `/etc/` → affect all users

What kind of shell are we running?

- Interactive vs. non-interactive bash shells
- Login s. non-login bash shell

How to identify the different types of shells

Type of Shell	How to identify it
Interactive	You can type commands
Non-Interactive	You cannot (You read commands from a file (script))
Login	You had to provide credentials when starting the shell
Non-Login	You did not

How to start the different types of shells

Type of Shell	Interactive	Non-Interactive
Login	• `CTRL + ALT + F3` to virtual console (then `CTRL + ALT + F2` to go back to the GUI) • `Bash --login` does not prompt credentials but follows all other steps • `ssh alessio@penguin.edu`	• `echo "ls –l ~/" | ssh alessio@penguin.edu` • `bash --login -c 'ls -l /'`
Non-Login	• Type bash in another shell • Open new terminal window • Open new tab in terminal window	• `bash myscript.sh` • `./myscript.sh`

FROM → https://askubuntu.com/questions/879364/differentiate-interactive-login-and-non-interactive-non-login-shell

How to tell if a shell is interactive?

- Check the contents of the $- variable.
  - For interactive shells, it will include i
- Interactive shell, just running a command in a terminal
  - `echo $- himBHs`
- Non-interactive shell
  - `bash -c 'echo $-' hBc`

FROM → https://askubuntu.com/questions/879364/differentiate-interactive-login-and-non-interactive-non-login-shell
More About $- at https://stackoverflow.com/questions/42757236/what-does-mean-in-bash

How to tell if the shell is a login shell?

- There is no portable way of checking this but, for bash, you can check if the `login_shell` option is set:
- Normal shell, just running a command in a terminal: interactive
  - `shopt login_shell`
    `login_shell off`
- Login shell;
  - `ssh localhost`
    or
  - `bash --login`
  - `shopt login_shell`
    `login_shell on`

FROM → https://askubuntu.com/questions/879364/differentiate-interactive-login-and-non-interactive-non-login-shell

Putting it all together….

Type of Shell	Interactive	Non-Interactive

Login	`bash -l`
	`echo $-`
	`himBHs`
	`shopt login_shell`
	`login_shell on`
		`echo 'echo $-;shopt login_shell'|ssh 127.0.0.1`
		(Pseudo-terminal not allocated since stdin is not a terminal.)
		`hBs`
		`login_shell on`

Non-Login	`echo $-`
	`himBHs`
	`shopt login_shell`
	`login_shell off`
		`bash -c 'echo $-; shopt login_shell'`
		`hBc`
		`login_shell off`

Back to looking up initialization files in the Bash manpage

- Focus on `INVOCATION` section
- Let's clarify it a bit what the manpage says:

Type of Shell	Interactive

Login	1. `/etc/profile`
	2. 1st of the following that exists:
	`~/.bash_profile`
	`~/.bash_login`
	`~/.profile`

Non-Login	1. `/etc/bash.bashrc` 2. `~/.bashrc`

A little experiment

```
sudo gedit /etc/profile /etc/bash.bashrc ~/.profile
~/.bashrc &
```

- Add `echo` commands to each of them, start with `/etc/profile`

- Enter bash to a bash prompt → not triggered
- `bash --login` → triggers it

…and so on so forth…

What the manpage says… vs our experiment

Type of Shell	Interactive
Login	1. /etc/profile 2. /etc/bash.bashrc 3. 1st of the following that exists: ~/.bash_profile ~/.bash_login ~/.profile 4. ~/.bashrc
Non-Login	1. /etc/bash.bashrc 2. ~/.bashrc

Why are the yellow highlights also loaded?

What the manpage says… vs our experiment

Type of Shell	Interactive
Login	1. /etc/profile 2. /etc/bash.bashrc 3. 1st of the following that exists: ~/.bash_profile ~/.bash_login ~/.profile 4. ~/.bashrc
Non-Login	1. /etc/bash.bashrc 2. ~/.bashrc

Why are the yellow highlights also loaded?

/etc/profile calls
. /etc/bash.bashrc

Hard to introduce configurations that affect only non-login shells, as long as the profile files can call the local ones too

Remember that users can do what they want in their ~/.profile e.g., loading ~/.bashrc or even /etc/bash.bashrc

Topic

IE1 Solutions

Q1 – Magical Touch

We want to create the following empty files in the current working directory. Please note that, before we create the files, the current working directory is completely empty.

- file_blue_cat
- file_blue_dog
- file_blue_panther
- file_blue_wolf
- file_green_cat
- file_green_dog
- file_green_panther
- file_green_wolf
- file_red_cat
- file_red_dog
- file_red_panther
- file_red_wolf

Provide a single shell line, that invokes the touch command one time only, with parameters that expand to all these filenames.

```
touch file_{blue,green,red}_{cat,dog,panther,wolf}
```

Q2 – Conditional ls

We want to use the ls command to list all the files from the current working directory which names match the following pattern:

- if the filename starts with a letter (lower or upper case)
  - it is followed by 3 characters: a digit, an uppercase letter, and a lowercase letter.
- Else
  - it is followed by 5 characters: a digit, any character, an uppercase letter, any character, and a lowercase letter.

Provide a single ls command that will list the above-described files in the current working directory.

```
ls @(   [a-zA-Z][0-9][A-Z][a-z]    |    [^a-zA-Z][0-9][0-9]?[A-Z]?[a-z]    )
```

Q3 – Invisible Touch

We want to use the touch command in order to set the time of last modification of all the files in our current working directory to the current time.

To this end, we issue the following command:

```
touch *
```

However, we realize that hidden files are not affected.

How do we change that without having to modify the "`touch *`" command that we are using?

shopt -s dotglob

Q4 – Unknown File formats Folder

- We have a folder named "/home/tux/pictures" that contains all our photos. Over the years, we have imported them from various sources and various formats. We want to move out of this folder all the photos who are not of a well-known format like gif jpg or jpeg, and put them into a subfolder name "/home/tux/pictures/unknown_formats/".
- Provide one line of shell that will move these files in the appropriate subfolder, regardless of what our current working directory is.
- Along with this one line of shell, provide any command to set specific options of the shell so that your solution works regardless of the default options being turned on right now.

```
shopt -s extglob
mv    /home/tux/pictures/!(*.gif,*.jpg,*.jpeg)
      /home/tux/pictures/unknown_formats/
```

Q5 – Folders, all the way down…

We want to setup the following hierarchy of folders in our home directory (/home/tux/):

This could take a lot of shell commands. Try to obtain the same results with as small of an effort as possible. Your grade will be commensurate to how much your solution uses the shell syntax to avoid repetitive work. You will also make sure that your command works when invoked from any current working directory.

```
mkdir -p /home/tux/EDU/
{CIS4930,CEN6084,COP{4610,3515,2512,2513}}
/{slides,videos,exercises}
```

EDU

- COP4610
  - slides
  - videos
  - exercises
- COP2512
  - slides
  - videos
  - exercises
- COP2513
  - slides
  - videos
  - exercises
- CIS4930
  - slides
  - videos
  - exercises
- CEN6084
  - slides
  - videos
  - exercises
- COP3515
  - slides
  - videos
  - exercises

Q6 – RTFM --ps

We want to list all the processes running on our system but, instead of displaying the default information on each, we want to display the following columns (in that specific order):

- Username of the user who started the process
- Niceness level of the process
- Process Identifier
- Start time of the process
- Name of the command that was issued to start it (along with its arguments)

Please note:

It is part of this exercise to have you search through the ps manpage for some of the necessary syntax if it has not all been introduced in the slides examples.

```
ps -e -o ruser,nice,pid,start,command
```

Q7 – Rogue Background Process

I have a process with job ID 1 running in the background. It is draining the CPU usage on the shared server on which I am logged in so I would like to pause its execution for now.

Provide all the different syntaxes and ways to do so that you can think of.

```
kill -19 %1
kill -STOP %1
kill -s STOP %1
fg %1 then ^Z
```

Q8 – Data Science CPU Usage

I just logged remotely to a Linux server on which I want to run a program named /home/tux/bin/sat_brute_force that will do some computations for a very long time. Since I share access to that Linux server with other users, I want to run the program as follows:

- It must run in the background
- It must use the lowest priority that I can set
- It must keep running in the background after I logout of the remote server
- Its output must be preserved in a file

Provide the command line that you would use to start this program.

```
nohup nice -n 19 /home/tux/bin/sat_brute_force
```

Q9 - mkcd

We want to create a folder name /home/tux/EDU/CIS4930/case_studies/ using a single command line and regardless of what our current working directory is.

If, for some reason, the creation of the folder did not work, we want to display a message saying “Sorry, something went kablemo.”. On the other hand, if the creation of the folder went well, we want to cd into it. All this as part of the one command line.

```
mkdir something/else && cd something/else || echo "FAILED"
```

Q10 – Two Commands, Two Failure Messages

Consider the 4 following commands:

- { echo "one" ; true;} && { echo "two" ; true; } || echo "one or two FAILED"
- { echo "one" ; true;} && { echo "two" ; false; } || echo "one or two FAILED"
- { echo "one" ; false;} && { echo "two" ; true; } || echo "one or two FAILED"
- { echo "one" ; false;} && { echo "two" ; false; } || echo "one or two FAILED"

Using the built-in true and false, as well as by grouping them with an echo, we are able to simulate the success or failure of a command displaying one word on the screen.

Right now, our solution is able to execute echo “one” and, if it is followed by true, execute echo “two”. If one of the two echo commands is followed by false, then we display the message “one or two FAILED”.

Improve the above so that we are able to display instead a specific error message indicating which of the two echo commands failed.

For example:

- One FAILED
- One worked but two FAILED

```
{ echo "one" ; false;} && { { echo "two" ; false; } || echo "two FAILED"; } || echo "one FAILED"
```

Topic

Interlude: PA2a

These Practice Exercises are meant to help you review for Intermediary Exam: IE2.

Bash Quoting Revisited

Provide the *echo* command you would type in your shell to display each of the following outputs;

- The dog & the cat; a tale of getting along just fine I remember now...
- "Bash is fun", they said. Liars!
- It's rather annoying (and even at times infuriating) to see quoting fail
- Why are we using \\ when we want just a \ to be displayed

You will provide 2 more solutions, each one using only one of the following;

- Single quotes ‘ …. ‘
- Double quotes “ …. “

What will the following display & why?

Command	Output
`echo "\ \\ \\\ \\\\"`	
`echo '\ \\ \\\ \\\\'`	
`echo "$HOMEr"`	
`echo "${HOME}r"`	
`echo '$HOMEr'`	
`echo '${HOME}r'`	

Command Substitution

The commands `whoami` and `date` are quite handy. We want to display with a single `echo` command the following message;

```
Hello, tux. Today is Tue May 8 17:20:40 EDT 2018
what do you want to do?
```

Obviously, we want "tux" to be replaced by the name of the user you are logged as, and the date by the current date of the system.

There are two syntaxes that will allow to specify what part of an *echo* statement you want the shell to actually execute and substitute by its output. Use both to achieve the above goal.

Dynamical PS1

- The above message would be a nice personalized prompt for a shell. How would we go about telling our current shell to use it as such for the rest of the session?
- Do you notice something strange about the so-called current date that is now being displayed in your prompt for each line?
- How would you fix this? This question is tricky. Do not simply try things at random and make sure you understand why things are working, or not.

It's meta.

- We want to assign to a variable called `META` the value `$RANDOM`. Please note, we don't want to assign a random number but the value `$RANDOM` as a string. How do you do that?
- We then want to display a message that states: Your luck number is `112233`. The value `112233` will be a random number, different every time we re-execute the line, and generated by using the variable `META`
- Why doesn't the following work?

```
ash-3.2$ echo  $( echo $var)
$RANDOM
bash-3.2$ echo $(echo $( echo $var))
$RANDOM
bash-3.2$
```

- Hint: check out the doc for the `eval` builtin