Craig Celestin
Linux for Noobs Challenges
Due 07/17/2021 at midnight

NOTES IN PROGRESS - Homework at Bottom
Intro to Linux
Linux = operating sys like Windows, MacOS - began in the 50s when OS ran batch processes, batch program w/o user interaction, read data from file/punch card then output to other file/printer. In 60s, interactive OS became popular allowing multiple users from different terminal to operate host at the same time (time-sharing OS), which challenged batch OS (dev of Unix)
Unix = free original release, then implemented TCP/IP protocol stack (early workstation OS).
90s -Unix = mainstream OS in server market, became commercialized. Linux became a competitor OS, simple, acquired attractive features. Linux = kernel of OS, basis of other running programs, implement multi-tasking, hardware management. Free apps w/ installation running on top of Kernel, required: CLI (user interaction, write shell scripts)
C compiler derived from GNU project of Free Software Foundation (began in 1984 with aim of developing free UNIX-like OS) Linux = GNU/Linux sys, 92-03 (TCP/IP Network, GUI sys attracted industry)

Linux Desktop Envi - no graphical interface (was xFree86), now there is a GUI in current distribution (Xorg). Desktop envi achieved via X Window System (X) = toolkit + architecture agreement, Xorg = framework of X implementation. Xorg = server providing GUI services like Apache. Achieving a complete desktop envi requires X client

Linux Terminal - Operate Linux sys via Shell. Linux sys provides a terminal emulator (the Terminal), Terminal and console are different (terminal - /dev/tty device on Linux where multi-user function is done. Linux provides 6 virtual consoles (CLI ‘terminals’ allowing user login)
In GUI, differences in user experience are caused by shell differences, UNIX/Linux kernel. Shell is a ‘provided user interface’ software, accepts user input commands. Shell - hides details of underlying OS (GNOME/KDE - 'virtual shell’/’graphical shell’), scripting language controls system. Shell = script interpreter that controls system boot, X11 boot, other utilities
Popular shells = Bash (sh), zsh, ksh, csh/Default shell = Bash, default desktop envi = GNOME/Unity

Input, Out, Err - Commands are most important (three streams)
touch - create file, cd /etc/ change to etc directory, pwd - print current directory
Tab - auto completes a command, $tail - next input has no reaction, $find / - continuous output
Ctrl+C (quits running command)
Shortcuts: ctrl+d (keyboard input ends/exits terminal), ctrl+s (pause current program, press key to resume), ctrl+z (put current program into back), ctrl+a (move cursor to beginning of input line, home equivalent), ctrl+e (move cursor to end of input line, end equivalent), ctrl+k (remove content from place cursor is to end of line), alt+backspace (delete word forward), shift+pgup(scroll up terminal), shift + pgdn (scroll down terminal)

Wildcards - special expression (* and ? matches strings), search folder w/ find command, use it instead / Wildcard input is handled in terminal by shell - search for possible matches as path/file name (match exists, replaced w/ path extension) - shell implementation of path extension function
$ls *.txt - list files ending in .txt
$touch love_{1..10}_linuxt.xt (create multiple files at once)
Common shell wildcards -

`*`	Match any zero or more characters
`?`	Match any single character
`[list]`	Match any single character of the list
`[!list]`	Match characters other than any single character of list
`[c1-c2]`	Match any single character in the range c1 to c2 such as [0-9], [a-z]
`{string1,string2,...}`	Match string1 or string2 (or more) with a string

`{c1..c2}`	Match all characters or numbers in the range c1 to c2 such as {1..10}

Section	Description
1	General order
2	System call
3	Library functions, covering the C standard library
4	Special files (usually devices in /dev) and drivers
5	File format and convention
6	Games and screensavers
7	Others
8	System management commands and daemons

man command sections: frmt (man 1 ls)
Format - name, synopsis, description, examples,
Specific parameters of a function: ls --help
Lab 0 - banner (graphic character output) - sudo apt-get update, sudo ap-get install sysvbanner
(2) banner labex or printerbanner -w 50 A
(-w *specifies print width*)

Linux User/Group + File Permissions

(1) Due to Linux user management, authority management, different users can't easily modify the files of others

$ who am i/$ who mom likes - indicates user name of user opening pseudo terminal
pts=pseudo terminal (relative to /dev/tty devices)
pts/#-corresponds to terminal number

Parameters of `who`:

Parameter	Description
`-a`	Print all information
`-d`	Print dead process
`-m`	Same as `am i`, `mom likes`
`-q`	Get number of users logged in and their user names
`-u`	Get the list of users logged in
`-r`	Get the current run level

Root account has supreme right of system like create/add users - root permission = system privilege like SYSTEM privilege, higher than admin privileges
Root =super admin user account, operate all system subjects / add, delete, modify, search for any file
Sudo can get root privileges, sudo command needs 2 reqs: password of user logged in, user must be in sudo user group
Su, su-, sudo
su<a> - switch to user a, command execution requires user password, run cm command at privilege level w/ sudo <cmd> require current user to belong to sudo group, enter current user password
Su - <user> (switch users) - $sudo adduser jack, enter custom pass
ls /home - shows home directory created for new user, log into new account - (su -l jack)
Each user has a home (user group) - collection of users who share resources, permission, private resources - similar to family structure
How to view user group name - $groups labex
You can view the /etc/sudoers.d/labex file. We have created this file in the /etc/sudoers.d directory to give sudo permissions to user label
$ cat /etc/group | sort (cat - read contents of specified file and print out, | sort - text sorted and output by dict sort (dictionary sort) (view contents of /etc/group file)
$ cat /etc/group | grep -E "labex" - filter out results from initial search for just labex
Each user group has a record - group_name:password:GID:user_list
Invite new user to join sudo user group for root privileges: su -l jack -> sudo ls
(after warning) user usermod to add user to user group /use the root user to add other users to user group
exit -> sudo groups jack -> sudo usermod -G sudo jack -> sudo groups jack
Delete a user: $ sudo deluser jack --remove-home

File Permissions
ls -l (list files in long format)
file type - everything in Linxus is a file, file permissions: read permission (r) : cat <file name> , read contents of a file
Write permission (w): read file contents, execute permission (x): refers to binary program file/script file (exe file on Windows), a directory w/ read and execute permissions can open and view contained files - only dir w/ write permissions can create other files in them
# of links - number of files linked to inode, file size - node size to rep file size (ls, -lh parameter view file sizes)
ls -A (display all files except . (current dir), .. (parent dir)/ ls -Al (use -A and -l parameters..list all files in long format)
view full dir properties: ls -dl <directory>/ ls -AsSh (show all file sizes) / ‘s’ (used for file size display), ‘S’ (sort file by file size)

Login as jack: sudo adduser jack -> su -l jack -> touch iphone2 -> cd /home/jack -> ls iphone2 -> exist -> sudo groups jack -> sudo usermod -G sudo jack -> sudo groups jack -> su -l jack (password) -> sudo chown labex iphone 2 (change file owner of ‘labex’) -> ll iphone 2 (file owner modified for labex)

Modify File Permissions
Document you don’t want other users to read, write, execute - 2 methods
Method 1: Binary # representation (file has 3 groups of permissions - owner, user group, others) corresponding to ‘rwx’ triplet. File ‘iphonex’ permissions changed to ‘only for owner to use’
echo “echo \”hello labex”\”” > iphone2 -> chmod 700 iphone2 (allow other users to read iphone2 file)
same results: chmod go-rw iphone2 (u,g,o represent user (file owner), group, others)
+, - (add/remove corresponding permissions)

Addition
useradd v adduser (useradd - only creates user, need to use passwd to set up password for newuser and then command adduser - creates user, directory, password)

```
labex:~/ $ sudo adduser labextest                                    [2:52:15]
Adding user `labextest' ...
Adding new group `labextest' (1000) ...
Adding new user `labextest' (1000) with group `labextest' ...
Creating home directory `/home/labextest' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for labextest
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]
labex:~/ $ su -l labextest                                           [2:52:54]
Password:
labextest@91fabe4d1ec1:~$ touch /opt/forlabex
touch: cannot touch '/opt/forlabex': Permission denied
labextest@91fabe4d1ec1:~$ exit
logout
labex:~/ $ sudo groups labextest                                     [2:54:06]
labextest : labextest
labex:~/ $ sudo usermod -G sudo labextest                            [2:54:14]
labex:~/ $ sudo groups labextest                                     [2:54:27]
labextest : labextest sudo
labex:~/ $ su -l jack                                                [2:54:36]
No passwd entry for user 'jack'
labex:~/ $ su -l labextest                                           [2:54:41]
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

labextest@91fabe4d1ec1:~$ sudo touch /opt/forlabex
[sudo] password for labextest:
labextest@91fabe4d1ec1:~$ sudo chown labex /opt/forlabex
labextest@91fabe4d1ec1:~$ ll /opt/forlabex
-rw-r--r-- 1 labex root 0 Jul 17 02:55 /opt/forlabex
labextest@91fabe4d1ec1:~$
```

File and Directory (Linux basic concepts - directory structure, abs path, relative path: cut, copy, rename, edit, file operations)
Directory Structure - Linux
Different implementation mechanism between Windows and Linux
Windows use disk symbol (C drive, D drive) for file management, another files can also be stored in any directory, after some time disk files are messy
UNIX - directory bases, Linux as well - tree like directory structure to build entire system
All stored on disk - Linux disk is mounted on a directory, every directory can use a file system on the network. Use NFS (Network File System) server to load specific directory

FHS Std - Normal operation of sys is based on directory structure, FHS defines use of every system area, minimal required files set, directories, provides exception handling and contradiction handling
2 Specifications - 1.Various directories below / contain specific files like setting files (/etc), executable files (/bin,/sbin) 2. Explicit definitions of subdirectories of 2 directories /usr, /var like /var/log to place system log files and /usr/share for placing shared data
$tree - view entire directory structure, command not found: sudo apt-get update -> sudo apt-get install tree

Directory Path
Path = route of where you go, use cd (switch directories), . (current directory), .. (upper directory/parent directory), ls-a (view hidden files), - (last used directory), ~ (current user home directory), pwd (get current path)
Absolute path - contains root directory, subdirectories (file/directory is contained)
/usr/local/bin ('bin' directory in local directory in the usr directory of root (/) directory)
Relative path - path relative to present working directory (pwd)
If in /var/log, desire change to /var/log/kernel - use relative path concept to change directory to kernel
Change directory to /usr/local/bin by using relative path concept: cd ~, cd ../../usr/local/bin
Using abs. path / relative path go to another directory (tab - auto fills path, avoid input errors, tab twice to show all results)

Basic Linux File Operations
Create new file - touch, change timestamp of existing file, w/o parameters, file name -> can create an empty file
Used (cd ~) to switch back to user's home/labex directory
~ cd -> touch test
Create new directory: mkdir mydir (create empty directory titled 'mydir')
Using mkdir w/ -p parameter (create parent directory if it does not exist)
Creating multi-level directories (mkdir -p father/son/grandson)

Copy
Copy Files - use cp to replicate file to specified directory/ copy prev file: cp test father/son/grandson
Copy Directories - cp w/ -r, -R parameter, (recursive copy) cp -r father family - avoids error that occurs when directly using cp to copy a directory
(file deletion) rm test
(Force deletion of files w/ read-only permissions) rm -f test
Delete Directories - rm -r family
Move Files - mv (move file to another place) (move 'file1' to directory named 'Documents') - mkdir Documents -> mv file1 Documents
-Rename Files- mv file1 myfile
Rename Multiple Files: touch file{1..5}.txt -> rename 's/\.txt/\.c/' *.txt -> rename 'y/a-z/A-Z/'*.c

3.5 - View Files
Commands used to print file content to std output (terminal), cat - prints contents in forward order, tac - prints contents in reverse order
Execute shell command, system open tree std files: std input file (stdin), std output file (stdout), standard error output file (stderr). Process gains input from std input file, output data to std output file. Any error - process sends error msg to std error file
View file passwd copied from /etc: cd /etc/ -> cat passwd (shows contents of passwd) and cat -n passwd (shows line numbers and contents in passwd file) (nl - add line number, print)

More and less - more (simpler vs less), rolls in one direction. Less - similar to more but allows backward movement in file + forward movement, does not have to read entire input file prior to starting. Less uses termcap, runs on a variety of terminals + limited support for hardcopy terminals. Head and tail allows for viewing of file - head (displays first count lines/bytes of each specified files), omitted count = 10, tail similar to head
Tail displays last count of files - new info at end of file (tail /etc/passwd)
Tail w/ -n parameter allows for viewing of contents of particular line in file
$tail -n 1 /etc/passwd (view contents of specific line in file) (in this example view last line (1st field of tail))

3.6 - View File Types
$ file /bin/ls - view file type
3.7 - File Edit
CLI Editors - allow for file editing (emacs, vim, nano), $file /bin/ls
$ vimtutor
CLI Editors - give the user an ability to edit files from CL. Most common is nano, vim req training (clone of Vi), extensible)primary editor, simple editor for changing files when SSH'd into server - Should be able to make edits on server w/ Vim, edits to config files, use Vim to manage git merges, conflicts
Mac: VIM - (initial homebrew install) `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
Brew install mercurial -> sudo mkdir -p /opt/local/bin -> cd ~ -> hg clone https://code.google.com/p/vim/ -> cd vim
./configure --prefix=/opt/local-> make -> sudo make install -> echo 'PATH=/opt/local/bin:$PATH' >> ~/.bash_profile
source ~/.bash_profile
Linux: run sudo apt-get install vim -> vim -v (exit screen -quit and discard changes (:q!))
(new file) vim mynewfile.txt -> create new file, editable in vim
Vim: command mode (execute commands) , insert mode (then exit via esc)
Entering command -> press : then wq (save edits) / quit Vim w/o saving changes (:q!)
vimtutor (vim's own tutorial) / move cursor using (h-left, k-up, l-right, j-down)
Enter Command Mode (ESC) -> (:q) (quite vim)
Text Edit - Deletion (click editor -> esc (command) -> x (navigate to desired letter to be deleted and then press)
Text editing requires entering Insert Mode (i-inserting (puts cursor before current position))

Appending - a (put cursor after current position), o (put cursor below line), 0 (put cursor above line)
Operators, Motions, Counts, Combine - Commands are decomposed into (operator, numbers, motions) [operator][number][motion]
1.Operators - actions, like verbs in a sentence (common ops: d-delete, c-change, y-yank, p-insert last deleted text after cursor (put command), r-replace)
2.Motions - w (until start of next word, exclude first character), e (end of current word, include last character), $ - end of line (include last character)
W-word forward, b-word back,)-next sen begin, (- begin of current sent, }-begin of next paragraph, {-begin of current paragraph,]-beginning of next sect, [-begin of current sect, H-screen top line, L-screen last line
Counts - optional, put a multiplier to command, motion
Word deletion - navigate to begin of word, enter (dw), in middle stope where word ends
Deletion to line end - delete everything til line end - d$, run command twice - d2$, delete four words - d4w, Undo cmd - u, Quick Page Nav - :G (file bottom), :gg (file start), :ctrl+g (view current page line), :123+G (jump to specific line)
Searching - (can use ctrl-f: jump around page), :/cats (search a page after cursor position), :?dogs (search page prior to cursor), n (navigate to next search match), N (nav to previous search match)
Matching search - function hippopotamus() {
//code lines
}
Press { -> % (jump to matching counterpart) / allows for quick jump around functions: (,),[,],{,}
Search+Replace
Desire: changes all words of ‘cat’ to ‘dog’ (:s/cat/dog), find+replace all instances (:s/cat/dog/g)
*Can do regular expression find + replaces, replace certain lines, sections

Execute External Command
-execute external commands from CLI in editor, start command w/ ‘!’
:!ls -al (list all files)
Configure Vim
*syntax highlight - not enabled by default - enabling on file req ‘:syntax on’
Configuring vim - file in home dir = .vimrc, force nable :syntax on as default
(1) vim ~/.vimrc -> syntax on -> :wq (save file w/ syntax on by default)
Set vim as default command line editor - sudo update-alternatives --config editor
Set vim as git default editor - git config --global core.editor “vim”
Summon pair of eyes for supervision - xeyes -> nohup exes ##nah
Create homework directory, put 1-10 file.txt, delete file range 1-5 file.txt
Mkdir homework -> touch homework/file{1..10}.txt -> rm -r file{1..5}.txt

-ENVI VAR AND FIND FILES-
Var: symbol used in computer to record value (character or string), used in different operations
Var and value: exclusive 1-1 relationship (‘declare’ command creates var named tmp: declare tmp)

Use = to assign variable tmp to labex
Echo, $ read variable value ($ - refers to value of variable): ‘echo $tmp’
Environment Variables: scope of environment variable (of shell can exert effect on shell, child processes) > custom variable
When process is created, inherits envi settings of parent process
Shell programs - run as processes on top of OS, commands in shell run as shell child processes
Three related commands - set, env, export (similar, used to print envi var info)
set - display variables of current shell, include built in envi vars, user vars, exported envi vars
env - display envi vars assoc w/ current user, allow cmd to run in specified envi
export - display vars exported from shell as envi vars, export custom vars as envi vars

```
$ temp=labex
$ export temp_env=labex
$ envsortenv.txt
$ exportsortexport.txt
$ setsortset.txt
```

-output of command is sorted using command sort, use vimdiff: compare differences between text
Correction: env|sort>env.txt -> export|sort>export.txt -> set|sort>set.txt
Use export to check validity of sub-shell variables
(1)set var temp=labex in shell (2)create subshell to see value of temp var
Make Envi Vars Permanent
-Permanent Vars: modify a config file then vars will be effectively permanent
-temporary vars: use export cmd at shell prompt, vars get invalid w/ shell closing

Linux (important files: /etc/bashrc, /etc/profile) - store shell vars, envi vars
Hidden file in every user directory .profile, use ls -a to view file:
cd homelabex -> ls -a
.profile (valid for current user), envi vars written in /etc/profile (valid for all users), want to add valid envi var, open /etc/profile, + envi var at file end

Entering a command into shell, shell knows where to find command via PATH var
PATH - saves search paths of commands
View PATH envi var contents: $echo $PATH
Creating simplest executable shell script: vi filename -> esc -> i (editing)
Enter:

```
#!/bin/bash
echo "Hello, World!"
echo "Knowledge is power."
```

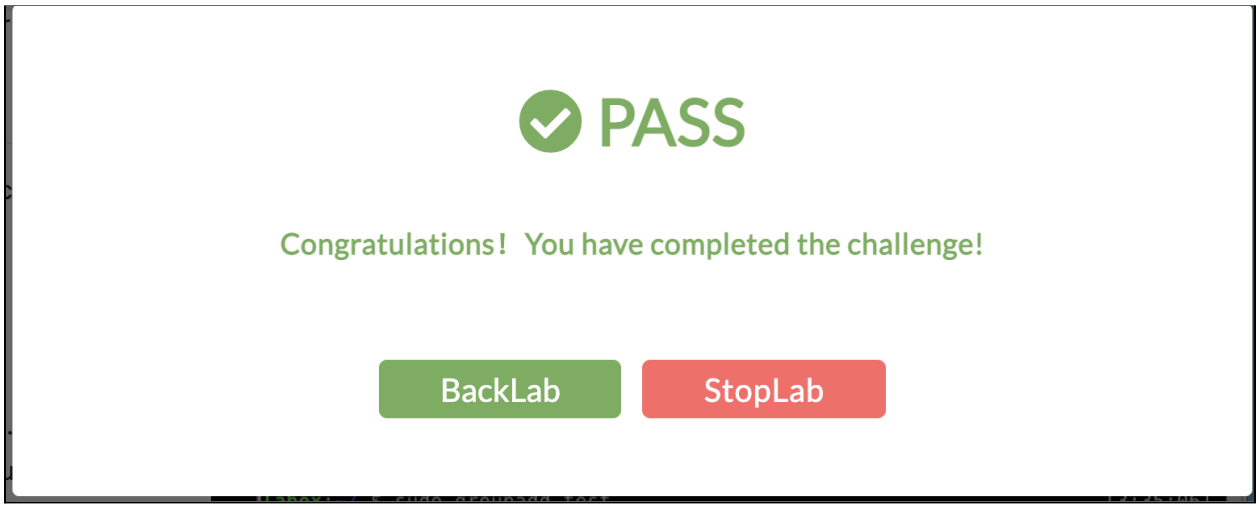
Then esc ->:w ‘file’ -> esc -> :wq -> (running script) ./hello.sh (try to run script, but receive an error) -> then chmod +x hello.sh -> ./hello.sh

Create Shell Script File (gedit hello_shell.sh)
Add following script (#!/bin/bash -> for ((i=0; i<10; i++)); do -> echo “hello shell”
Done
exit 0)
(add executable permission) - chmod 755 hello_shell.sh -> ./hello_shell.sh (prints hello shell 10 times)
Create ‘Hello World’ program: gedit hello_world.c

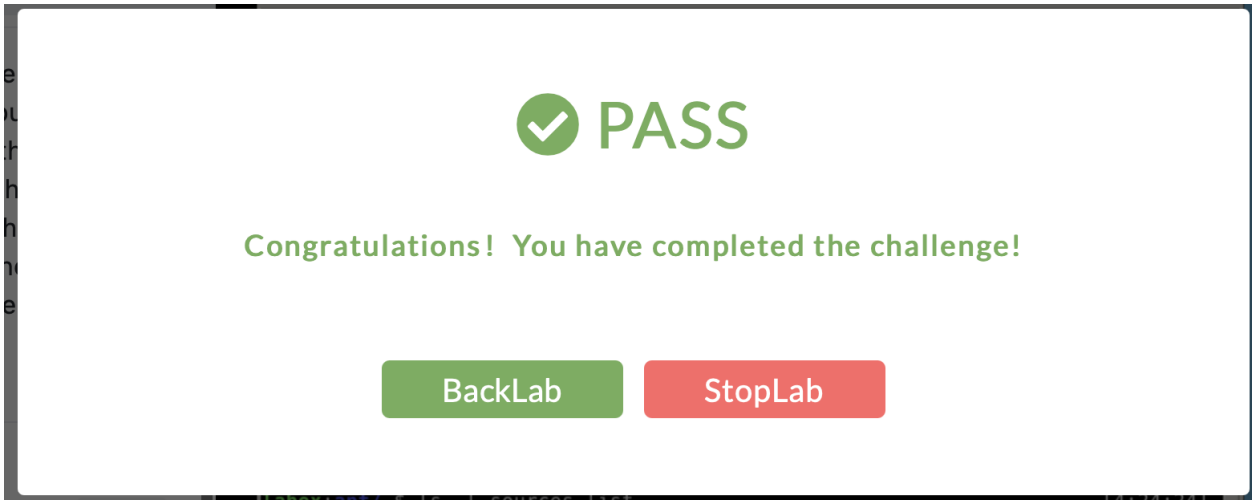
```
#include <stdio.h>
int main(void)
{
	printf(“hello world!\n”);
```

```
      return 0;
}, save + use gcc to gen exec file: gcc hello_world.c -o hello_world
(create dir) mkdir mybin, (move both previous files) mv hello_shell.sh hello_world
(change dir) cd mybin -> ./hello_shell.sh + ./hello_world
Executing programs outside of directories using sys commands to execute own prgms?
Add paths to directories via PATH envi var -
: (PATH delimiter), $PATH=$PATH:/home/'name'/mybin
Every home directory, config script run auto to initialize envi including other envi
vars - config files (.zsh - .zshrc, bash - .bashrc), more global config files under
/etc, usually config file mods in user directory
Add content command to .zshrc (echo PATH=$PATH:/home/labex/mybin >> .zshrc)
>> (std output redirected to file)
*modify variables - ${name#match string} (front to back delete shortest string
matching string), name##match - front to back, delete longest, name%match - back to
front + delete shortest, name%%match - back to front + delete shortest string,
${nameoldstringnewstring} - replace first tstring matching old string with nea
string, ${nameoldstringnewstring} - replace first str matching old string w/ new
Add PATH - PATH=$PATH:/home/shiyanlou/mybin
(add content to .zshrc) echo PATH=$PATH:/home/labex/mybin >> .zshrc
Modify path - path=$PATH -> echo $path -> path=${path:/home/shiyanlou/mybin}
-> path=${path:/mybin}
Delete envi var - $ unset temp
Modify config script *reopen terminal/restart host* use source command to modify
config script - source /zschrc (. .zshrc #.=source alias)
Whereis who - use cmd to find three paths: 2 are executable files, third = path of
man online help file - queries from database whereis: binary files (-b), man help
files (-m), source code files (-s)
Comprehensive search results - locate cmd (manual update via updatedb cmd, locate
used to find various file types: $locate etchsh - find files starting w/ 'sh')
Find all jpg files under usrshare: locate usrshare.jpg (find jpg files under
usrshare)
Add -c (count number of files), add -i (search ignoring cases)...
Find: search for files based on file attributes, command - search for file/directory
named interfaces in etc directory
sudo find etc -name interfaces
```

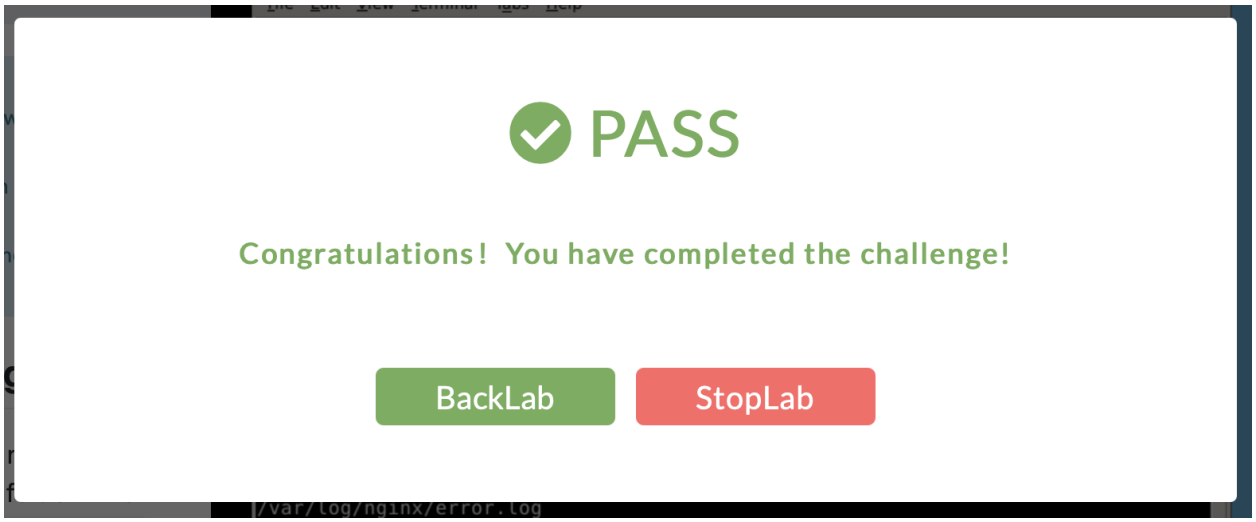
Add New User and Group Challenge
labex:~/ $ sudo adduser jack
labex:~/ $ usermod -d home/jack jack
labex:~/ $ usermod --shell /bin/zsh jack
labex:~/ $ sudo usermod -G labex jack
labex:~/ $ sudo groupadd dev
labex:~/ $ sudo usermod -aG dev jack
labex:~/ $ sudo adduser bob
labex:~/ $ usermod -d home/bob bob
labex:~/ $ sudo usermod --shell/bin/bash bob
labex:~/ $ sudo usermod -G labex bob
labex:~/ $ sudo groupadd test
labex:~/ $ sudo usermod -aG test bob



Find a File Challenge
cd /etc/
sudo find . -name sources.list
Output: ./apt/sources.list
cd ./apt -> sudo chown labex sources.list
Sudo chmod 600 sources.list -> ls -l sources.list (-rw----)



Backup System Log Challenge
mkdir /home/labex/tmp
sudo tar cvzf /home/labex/tmp/$(date +%F).tgz /var/log



Analyze Historical Commands Challenge

```
labex:~/ $ cat data1 | tr -s ' ' | cut -d ' ' -f3 | sort | uniq -dc | sort | tai
l -n 3 | cut -c 5- > result
labex:~/ $ cat result                                              [4:46:27]
  8 shutdown
 91 vim
  9 rabbitmqctl
```

Data Extraction Challenge

```
wget: unable to resolve host address 'labfile.oss-cn-hangzhou.aliyuncs.com'
labex:~/ $ wget http://labfile.oss-cn-hangzhou.aliyuncs.com/courses/1/data2
--2021-07-21 04:55:13--  http://labfile.oss-cn-hangzhou.aliyuncs.com/courses/1/d
ata2
Resolving labfile.oss-cn-hangzhou.aliyuncs.com (labfile.oss-cn-hangzhou.aliyuncs
.com)... 118.31.232.210
Connecting to labfile.oss-cn-hangzhou.aliyuncs.com (labfile.oss-cn-hangzhou.aliy
uncs.com)|118.31.232.210|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 246 [application/octet-stream]
Saving to: 'data2'

data2               100%[===================>]     246  --.-KB/s    in 0s

2021-07-21 04:55:14 (38.2 MB/s) - 'data2' saved [246/246]

labex:~/ $ ls                                                          [4:55:14]
Code  data2  Desktop
labex:~/ $ cd /home/labex                                              [4:55:15]
labex:~/ $ ls                                                          [4:55:20]
Code  data2  Desktop
```