Xi'an Jiaotong-Liverpool University
西交利物浦大學

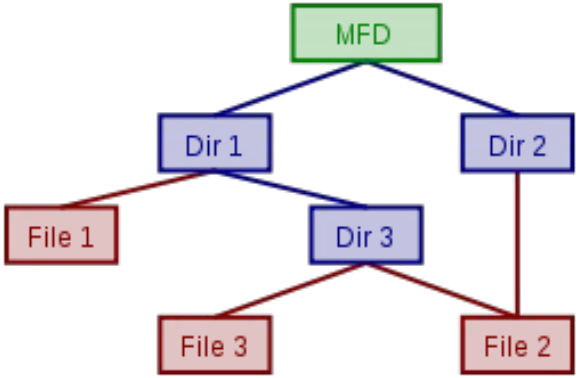
CPT104 - Operating Systems Concepts

Lab 6

Linux 2

Filesystem

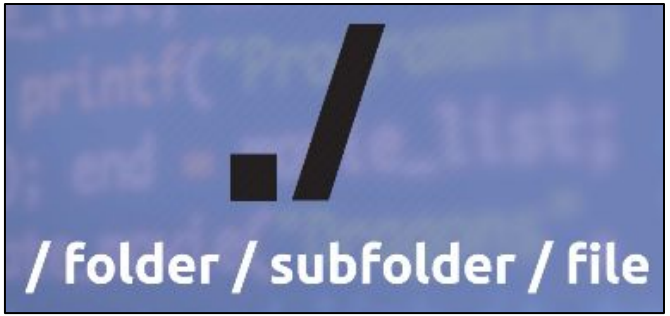
- Recall that one role of the operating system is to manage files using a file management system, or filesystem
- For the user, a filesystem is best visualized as *a tree*
  - Files are grouped into **directories**, or **folders**
  - These directories contain further files and/or other directories
  - There is a **root directory** (MFD — Master File Directory) and then a number of subdirectories



- Such an organization generates a hierarchy of directories and files organized into a tree

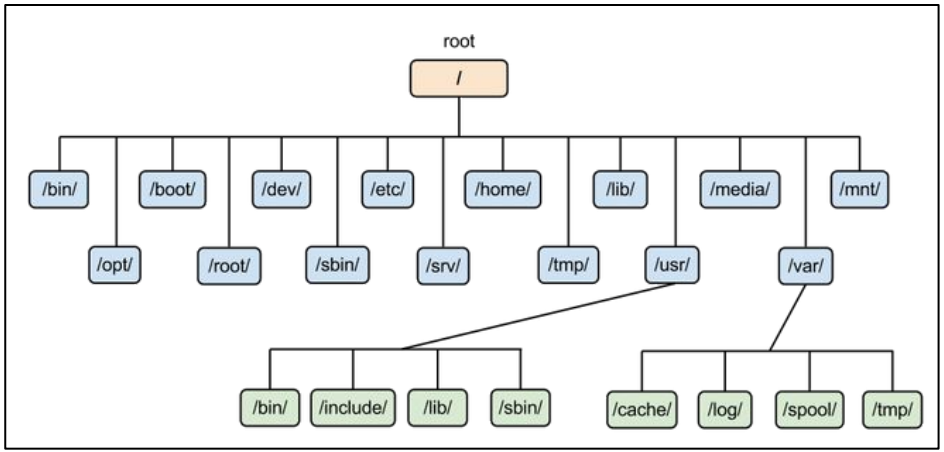
Root, Folder, Subfolder, File in Linux

- On Unix/Linux, the root of the file system is denoted **slash /**
  - The root folder would, therefore, be slash
- A *folder* and a *subfolder* will be denoted slash folder slash subfolder
- A *file* in that subfolder will be denoted slash folder slash subfolder slash file
- It is up to the users to decide how to organize the hierarchy of their own files and folders



Filesystem Standard

- The operating system itself has lots of files to keep track of such as utility programs, applications, methods to communicate with devices
- In August 1993, the process of standardizing the file system hierarchy began
- **FSSTND**, Filesystem Standard, is a standard specific to the filesystem hierarchy to GNU Linux
- The vast majority of GNU Linux distributions do not strictly follow this standardization format



- We're going to give you some directory names that can be found in the vast majority of Linux distributions

Directories in Linux (1)

- **/bin** is the directory which contains all of the basic commands needed to start and use a minimal system
  - the directory name is derived from the abbreviation of binary
  - note that the executable file of an application is called ***a binary***
- **/sbin** directory contains commands (executables, binaries) for the **super user**, also called **root user**
  - the root user is the administrator of the computer and has more powers than simple users (superpowers)
- **/home** contains *all of user directories*, for example, /home/erick
- **/root** is the home directory of the root user, the user that has the superpowers
- **/etc** directory contains configuration files
  - it is the abbreviation of Editable Text Configuration

Directories in Linux (2)

- **/lib** is where the system stores software libraries required for executables in /bin and /sbin
- **/tmp** directory is for temporary files.
  - sometimes, it is located in /var/tmp or /run/tmp
  - it is usually emptied every time you restart the computer, so be careful, when you use this folder and restart your computer, everything will be destroyed inside
- **/var** contains various files used by the operating system, such as databases, email boxes, and history
  - for example, a log of what happened on the system is in /var/logs

Directories in Linux (3)

- **/usr** is the acronym of Unix System Resources, and not user, as one might think
  - this folder contains some subfolders, similar to those present that the roots, but they are used to extend the system operations
  - for example, this folder includes **/usr/bin** that contains executable binaries that are not already present in /bin and, therefore, not essential to a minimal system
  - or, **/usr/lib** that contains those libraries that are not essential for a minimal system
- Finally, **/dev** contains files, each of which corresponds, directly or indirectly, to a physical device
  - dev is an abbreviation for device
  - this is one of the strengths of Linux: it considers ***everything as a file***, including devices, such as /dev/printer, /dev/audio, /dev/mem, /dev/networkcard

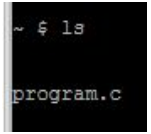
Navigating Linux Filesystem

- We are going to navigate through the file system using *Linux commands* in the Linux command line, using WebLinux
  - Please try everything in today's lab yourself, while reading this lab notes

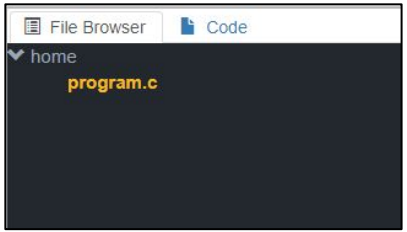
- The first command in today's lab: **pwd**
  - It means print working directory
  - it is the directory in which you are working just now
  - so if you press Enter, well, you see /home/user which is your **home directory**
- Remember last week, whoami, your username is user . I am user.
  - so when you type pwd when you first log into a Linux system, you will have /home/ and then you're log in, if it is erick, it would be /home/erick

ls and File Browser

- To list the files and folders, directories inside the working directory, use **ls** for list
  - inside, we have only *one file* that is *program.c*
  - this file is open in *the editor* in the left



- Equivalently, you can also click **File Browser** to see all the files and directories in your *home directory*
  - home here stands for /home/user
  - you see there only one file: program.c
  - if you double-click on it, it will open it in the editor



cd from root to home directory

- **cd** is used and stands for change directory
- To go to the root directory: **cd /**
  - recall that slash / is the root directory
- You can see in the command prompt that you have now is / $
  - if you pwd, now you are in /
  - if youl list what is in / , you will see folders bin, etc, lib, home, and so on
- To go to home: **cd /home**
  - try ls, you have the folder user
  - go to user: **cd user**
  - and now you will see the tilde ~ $
  - ~ stands for the home directory
  - you are back in your home directory !

bin and go back to home directory

- Go back to the root directory **cd /** then go to bin **cd /bin**
  - ls and you will see a lot of files
  - you see the binary of ls, that we are using just now
  - you also see the binary of pwd
  - indeed, the standard Linux commands in the /bin directory
- Go back to the root and go inside the sys directory, and list
  - you see other directories: block, class, devices
  - how do you go back to home directory?

bin and go back to home directory

- Go back to the root directory **cd /** then go to bin **cd /bin**
  - ls and you will see a lot of files
  - you see the binary of ls, that we are using just now
  - you also see the binary of pwd
  - indeed, the standard Linux commands in the /bin directory
- Go back to the root and go inside the sys directory, and list
  - you see other directories: block, class, devices
  - how do you go back to home directory?
  - to go back to the home directory is to type **cd**
    - alone like that, and hit enter
  - it will go anyway to the home directory and you see the tilde, and if you pwd it's indeed /home/user

ls -a, current directory and parent directory

- List with ls the files you have in your home directory
  - you only see program.c, since it is a simplified view
- To have a more advanced view and to see the hidden files, add an option to ls :
  - **ls -a** will list the files and also the hidden files
  - you see you actually have hidden files,
    where the names of the hidden files are starting with a dot .
  - you also have something named dot . and dot dot ..
    what are those?

ls -a, current directory and parent directory

- List with ls the files you have in your home directory
  - you only see program.c, since it is a simplified view
- To have a more advanced view and to see the hidden files, add an option to ls :
  - **ls -a** will list the files and also the hidden files
  - you see you actually have hidden files,
    where the names of the hidden files are starting with a dot .
  - you also have something named dot . and dot dot ..
- **dot** is the ***current directory*** and **dot dot** is the ***parent directory***
  - so if you type **cd ..** then it will go to the parent directory of current directory
  - now type cd .. then it will go to the parent directory of my home directory,
    which is /home
  - type cd .. again then we are back to the root directory,
    which is the parent of /home

Relative Path vs Absolute Path

- Go back to the home directory with cd
- Let us now try cd with what is called **a relative path**: **cd ../..**
  - it means the parent of the parent *relative* to this directory
  - it will go to the root directory, indeed
- We want to now go to sys using relative path
  - what is the command?

Relative Path vs Absolute Path

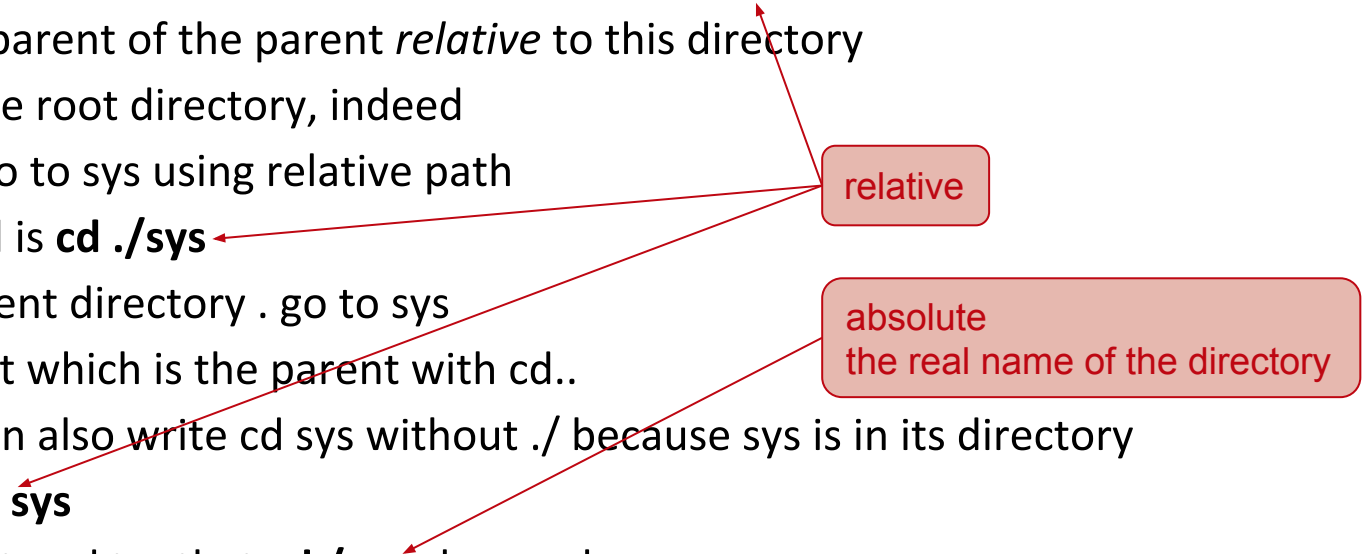
- Go back to the home directory with cd
- Let us now try cd with what is called **a relative path**: **cd ../..**
  - it means the parent of the parent *relative* to this directory
  - it will go to the root directory, indeed
- We want to now go to sys using relative path
  - the command is **cd ./sys**
  - from the current directory . go to sys

Relative Path vs Absolute Path

- Go back to the home directory with cd
- Let us now try cd with what is called **a relative path**: **cd ../..**
  - it means the parent of the parent *relative* to this directory
  - it will go to the root directory, indeed
- We want to now go to sys using relative path
  - the command is **cd ./sys**
  - from the current directory . go to sys
- Go back to the root which is the parent with cd..
- To go to sys, we can also write cd sys without ./ because sys is in its directory
  - so we type **cd sys**

Relative Path vs Absolute Path

- Go back to the home directory with cd
- Let us now try cd with what is called **a relative path**: **cd ../..**
  - it means the parent of the parent *relative* to this directory
  - it will go to the root directory, indeed
- We want to now go to sys using relative path
  - the command is **cd ./sys**
  - from the current directory . go to sys
- Go back to the root which is the parent with cd..
- To go to sys, we can also write cd sys without ./ because sys is in its directory
  - so we type **cd sys**
- Go back to the root and try that **cd /sys** also works
  - But this time, the last one above is **an absolute path**
  - Absolute path ***always start with the root /***



More on Relative and Absolute Path

- You are now in /sys and you want to go back to your home directory /home/user using relative path
  - What is the command?

More on Relative and Absolute Path

- You are now in /sys and you want to go back to your home directory /home/user using relative path
  - Use relative paths as follow: **cd ../home/user**
  - In other words, go to the parent which is root, go to home inside root, and go to user inside home

More on Relative and Absolute Path

- You are now in /sys and you want to go back to your home directory /home/user using relative path
  - Use relative paths as follow: **cd ../home/user**
  - In other words, go to the parent which is root, go to home inside root, and go to user inside home

- Go back to /sys, and this time, try to write it with the absolute path!
  - What is the command?

More on Relative and Absolute Path

- You are now in /sys and you want to go back to your home directory /home/user using relative path
  - Use relative paths as follow: **cd ../home/user**
  - In other words, go to the parent which is root, go to home inside root, and go to user inside home

- Go back to /sys, and this time, try to write it with the absolute path!
  - So it is **cd /home/user**
  - Because it is absolute, you have to start from the / followed by the path from the beginning and it is home/user
  - So from *anywhere* in the system, you could reach /home/user using the absolute path because it starts from the root

More on Relative and Absolute Path

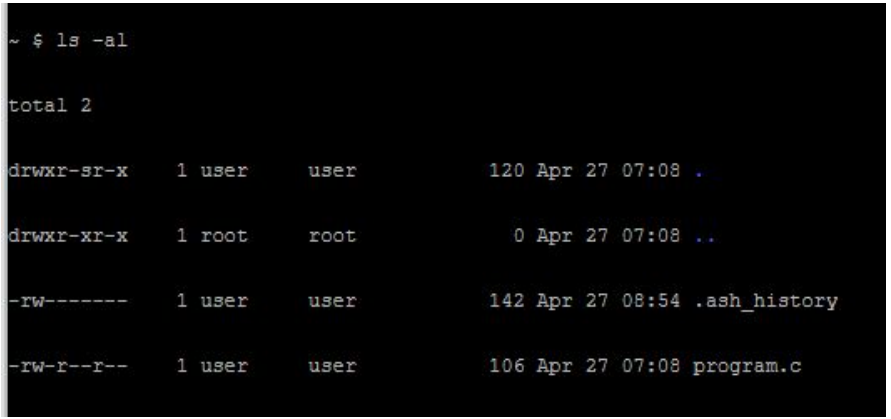
- You are now in /sys and you want to go back to your home directory /home/user using relative path
  - Use relative paths as follow: **cd ../home/user**
  - In other words, go to the parent which is root, go to home inside root, and go to user inside home

- Go back to /sys, and this time, try to write it with the absolute path!
  - So it is **cd /home/user**
  - Because it is absolute, you have to start from the / followed by the path from the beginning and it is home/user
  - So from *anywhere* in the system, you could reach /home/user using the absolute path because it starts from the root

explore WebLinux, go 3, 4 folders deep from root and go back to home using relative path

ls -al

- Adding another option, **ls -al** will list all files and folders with more detail



- In WebLinux, blue ones are folder, white ones are files
- You cannot cd a file

file

- In home directory, type command **file program.c**
  - it says that here is a C source
  - it's written in ASCII text, which is a way to encode characters, so it is encoded in ASCII
- use **file** command to get the type of a file

nano

- Let's create a new file using nano
  - recall that nano is a file editor, a very basic one
  - type **nano f** to create a file called f
  - it will open the editor, and we can edit it in a GNU nano
- Now, write something like: hello
  - Look the menu below, WriteOut means save, so save by Control + O
  - File Name to Write: f, Enter
  - Exit by Control + X

nano, File Browser, and file

- Let's create a new file using nano
  - recall that nano is a file editor, a very basic one
  - type **nano f** to create a file called f
  - it will open the editor, and we can edit it in a GNU nano
- Now, write something like: hello
  - Look the menu below, WriteOut means save, so save by Control + O
  - File Name to Write: f, Enter
  - Exit by Control + X
- If you ls, you will see that now you have f and program.c
  - also shown in the File Browser
  - you can double-click on f, it will open it in the code editor
- Find out the type of the file f
  - type **file f** and that type is just ASCII text, indeed

Path of a File

- type **realpath f**
  - it will show the absolute path of the file f

Another type

- Go to root directory with cd /
- Type **file bin**
  - bin is a directory, so you can cd into it, cd bin
- Now **file cat**

Another type

- Go to root directory with cd /
- Type **file bin**
  - bin is a directory, so you can cd into it, cd bin
- Now **file cat**
  - cat is a symbolic link to /bin/busybox
  - now that's one interesting cat, isn't it?
  - it's kind of a link to another file, which is this busybox

Another type

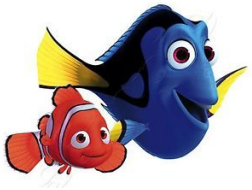
- Go to root directory with cd /
- Type **file bin**
  - bin is a directory, so you can cd into it, cd bin
- Now **file cat**
  - cat is a symbolic link to /bin/busybox
  - now that's one interesting cat, isn't it?
  - it's kind of a link to another file, which is this busybox
- So next **file busybox**
  - it's a binary that is executable
- What is the absolute path of cat ?

Another type

- Go to root directory with cd /
- Type **file bin**
  - bin is a directory, so you can cd into it, cd bin
- Now **file cat**
  - cat is a symbolic link to /bin/busybox
  - now that's one interesting cat, isn't it?
  - it's kind of a link to another file, which is this busybox
- So next **file busybox**
  - it's a binary that is executable
- What is the absolute path of cat ?
  - **realpath cat** gives you /bin/busybox
  - make sense since cat is just a link to another file

explore WebLinux,
try check the type and
the realpath of some files

Finding Path

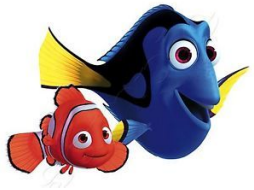


- We know a name of a program, and we want to get the path of binaries that are installed on the system
  - for example, the program realpath we just used
  - type **which realpath**
  - now we know realpath is installed in /usr/bin/realpath

  - where is the binary of cat located?

  - where is the binary of which located?

Finding Path



- We know a name of a program, and we want to get the path of binaries that are installed on the system
  - for example, the program realpath we just used
  - type **which realpath**
  - now we know realpath is installed in /usr/bin/realpath

  - where is the binary of cat located?
  - type **which cat**

  - where is the binary of which located?
  - type **which which**

Create a File 2

- Last time, we already know how to create a file using nano
  - we created f with nano f
- Another simpler way to do that is to use touch
  - type **touch f2** and Enter
  - it will just create an empty file named f2
  - type ls to see the files

Delete a File

- Last time, we already know how to create a file using nano
  - we created f with nano f
- Another simpler way to do that is to use touch
  - type **touch f2** and Enter
  - it will just create an empty file named f2
  - type ls to see the files
- Delete a file with command rm
  - type **rm f**
  - type **rm f2**
  - type ls to see that those files are no longer there

Create many files

- **touch my file**
- ls -l
  - you will see ***two*** (empty) ***files*** are created (in two separate lines)

- thus, you need to be careful when using space in the name of your file when you use touch!
- remove those two files

Create file with space in name

- **touch my\ file**
- ls -l

- this will create one file with space in name using touch
- remove that file
  - use backslash again

Create file with space in name 2

- **touch 'my file'**
- ls -l

- so that will also create a single file with space in its name
- remove that file
  - use a pair of single quote again

Stuck in Single Quote

- Try typing: **rm '** and then **Enter**
  - a greater than sign **>** appears
  - try typing some more: **ls Quit Enter**
  - it just continues waiting for more commands …
  - the Enter actually becomes part of the name of the file to be removed!

- How do you return to the command prompt?

Stuck in Single Quote

- Try typing: **rm '** and then **Enter**
  - a greater than sign **>** appears
  - try typing some more: **ls Quit Enter**
  - it just continues waiting for more commands …
  - the Enter actually becomes part of the name of the file to be removed!

- How do you return to the command prompt?
  - type **'** and **Enter**
  - you actually finish the rm with file name command

- Similar situation with double quote " " and braces/curly brackets { }
  - try them yourself!

Stuck in Single Quote 2

- Try typing: **rm '** and then **Enter**
  - a greater than sign **>** appears
  - try typing some more: **ls Quit Enter**
  - it just continues waiting for more commands …
  - the Enter actually becomes part of the name of the file to be removed!

- How do you return to the command prompt?
  - another way to get out of that situation is by typing **Ctrl + D**
  - you may need to type it ***twice***

Read a File

- Let us first create a file and write in it
  - nano file.txt
  - write in it "first line" Enter "second line" Enter
  - save it with Ctrl + O
  - exit with Ctrl + X
- ls to check that new file is there
- Now, to read a file in a fast manner, use the cat command
  - **cat file.txt**

Create a File 3

- Last time, we already know how to create a file using nano and touch

- Another simpler way to do that is by redirecting the output of a command into a file
  - recall that echo "Hello World" print the string on your screen
  - now we redirect that output into a file, for example:
    echo "Hello World" > file.txt
  - read the file using cat
    cat file.txt

Create a File 3.5

- Last time, we already know how to create a file using nano and touch

- Another simpler way to do that is by redirecting the output of a command into a file
  - recall that echo "Hello World" print the string on your screen
  - now we redirect that output into a file, for example:
    echo "Hello World" > file.txt
  - read the file using cat
    cat file.txt
  - now you can also redirect the output of that cat into another file!
    cat file.txt > file2.txt
  - check that file2 contains the same string, then delete both files

Create a File 4

- Yet another way to create a file by cat and redirection is as follows
  - with no filename before > you write:
    cat > file.txt
  - when you Enter, you enter a writing state, so write:
    first line Enter
    second line Enter
  - end by typing Ctrl + D (twice if you don't want newline after second line), and you will back to command prompt
  - read the file, delete the file

Reading a long file 1

- try **cat /etc/services**
  - it is a long file that is describing all the services of the operating system
  - you cannot see the beginning of the file because you are at the very end

- one way it to do this: **cat /etc/services | more**
  - you redirect the output of cat into the input of another command more, using the pipe |
  - hit *spacebar* to go to the next page
  - if you want to exit more, hit q or Ctrl + C

Reading a long file 2

- Another way to do that: **cat /etc/services | less**
  - you redirect the output of cat to less using pipe
  - less is more advanced than more
  - you can scroll through the document using arrows
  - use j to scroll one line at a time to down,
    and k one line up
  - use spacebar to scroll page by page
  - use uppercase G to go to the end of the document,
    and lowercase g to the beginning
  - you can look up other less commands yourself
  - quit the command using q

you can use command more or less followed by name file to read it as well

ls and less

- ls /usr/bin will give you a long list of files and directories
  - redirect to less with ls **/usr/bin | less**
  - now you can scroll to browse the files

Searching using less

- ls /usr/bin will give you a long list of files and directories
  - redirect to less with ls **/usr/bin | less**
  - now you can scroll to browse the files
- You can also do search as follows:
  - forward search by typing slash and then the pattern, for example **/joe** Enter
  - to go to the next one, type **n**, you will get
  - capital **N** to the previous match

Searching using less

- ls /usr/bin will give you a long list of files and directories
  - redirect to less with ls **/usr/bin | less**
  - now you can scroll to browse the files
- You can also do search as follows:
  - forward search by typing slash and then the pattern, for example **/joe** Enter
  - to go to the next one, type **n**, you will get
  - capital **N** to the previous match
  - backward search from where you are is similar, for example **?joe** Enter

- Add statistic by add some options: **ls /usr/bin | less -NM**

Create a Directory/Subdirectory

- In the home directory, let us make directory mkdir command
  - **mkdir folder**
  - you can see the folder in the File Browser
- Get inside folder, then create two files
  - **touch f1**
  - **touch f2**
- Then, create a subfolder in it
  - **mkdir subfolder**
- Check with ls that you have f1, f2 subfolder in the directory folder
- Create three files f3, f4, f5 in the subfolder with touch
  - in the File Browser, open subfolder
  - check that you see similar to this following picture :

Delete a Directory

- Go back to home, try rm folder, it won't work
- You have to delete a folder recursively, use **rm -r folder**
  - use with caution, you just deleted a whole bunch of files and a subdirectory

Delete a Directory

- Go back to home, try rm folder, it won't work
- You have to delete a folder recursively, use **rm -r folder**
  - use with caution, you just deleted a whole bunch of files and a subdirectory

- Create the directory and files again
- This time, delete with **rm -ri folder**
  - you will get asked for each time you want to delete
  - you can answer with y or n

Delete a Directory

- Go back to home, try rm folder, it won't work
- You have to delete a folder recursively, use **rm -r folder**
  - use with caution, you just deleted a whole bunch of files and a subdirectory

- Create the directory and files again
- This time, delete with **rm -ri folder**
  - you will get asked for each time you want to delete
  - you can answer with y or n

- You can automate answering yes by command yes
  - try **yes**, quit with Ctrl + C
  - so we can redirect its output to input of rm with pipe as follows

    yes | rm -ri folder

Create Multiple Directories

- You can create multiple directories by
  mkdir d1 d2 d3

- What if you want to create directories D3 inside D2 inside D1 ?

Create Multiple Directories

- You can create multiple directories by
  mkdir d1 d2 d3

- What if you want to create directories D3 inside D2 inside D1 ?
  - use the option -p as follows:
    mkdir -p D1/D2/D3

- Now delete them all

Move a File

- In home directory, create a file f1 and a directory d1
- While looking at File Browser, move the file f1 to the directory d1 by
  mv f1 d1
- Create another directory d2
- How do we move directory d1 into directory d2 ?

Move a Folder

- In home directory, create a file f1 and a directory d1
- While looking at File Browser, move the file f1 to the directory d1 by
  mv f1 d1
- Create another directory d2
- How do we move directory d1 into directory d2 ?
  mv d1 d2
- Go inside d1
- How do we move f1 to the home directory ? Using relative path ?

Move a File

- In home directory, create a file f1 and a directory d1
- While looking at File Browser, move the file f1 to the directory d1 by
  mv f1 d1
- Create another directory d2
- How do we move directory d1 into directory d2 ?
  mv d1 d2
- Go inside d1
- How do we move f1 to the home directory ? Using relative path ?
  mv f1 ../..
- Still inside d1, create a directory d3, then move f1 into d3 :

Move a File and Relative Path

- In home directory, create a file f1 and a directory d1
- While looking at File Browser, move the file f1 to the directory d1 by
  mv f1 d1
- Create another directory d2
- How do we move directory d1 into directory d2 ?
  mv d1 d2
- Go inside d1
- How do we move f1 to the home directory ? Using relative path ?
  mv f1 ../..
- Still inside d1, create a directory d3, then move f1 into d3 :
  mkdir ../../d3
  mv ../../f1 ../../d3

Continue to Move

- Can we move the current directory to another directory, say home directory?
  - try **mv . ../..**
  - we cannot move a folder we're currently in

Rename a Directory

- Go to home directory
- We rename the directory d2 to d4 also by move command

  mv d2 d4

Moving and Renaming a File

- In home directory, we want to move f1 inside d3 to home directory, and rename it to f2. How do we do that in one command ?

Moving and Renaming a File

- In home directory, we want to move f1 inside d3 to home directory, and rename it to f2. How do we do that in one command ?
  mv d3/f1 f2

- Now, move d1 that is inside d4 into d3 and rename it to d2 !

Moving and Renaming a Folder

- In home directory, we want to move f1 inside d3 to home directory, and rename it to f2. How do we do that in one command ?
  mv d3/f1 f2

- Now, move d1 that is inside d4 into d3 and rename it to d2 !
  mv d4/d1 d3/d2

Move a File to a File ?

- In home directory, create two files f3 and f4
  - cat > f3↵ this is f3↵ Ctrl + D
  - cat > f4↵ this is f4↵ Ctrl + D
- Now try to rename f3 to f4
  mv f3 f4
- The result is that the f4 is gone forever, and cat f4 will give this is f3
  - need to be careful with this!
- Type **mv --help**, you will see that option -i and -n could be useful in this situation
  - mv -i will prompt you before you erase/overwrite an existing file, when you only want to rename

Copy a File

- To copy a file f4 to a new file f5 use command

  cp f4 f5
- Verify with cat that they have the same content

Copy with Cat

- You can also copy with cat and redirection

  cat f4 > f5

  will produce the same result

Copy a Directory

- Create a directory d5 and move f5 into d5
- We want to copy d5 to d6

  cp d5 d6 won't work
- We have to copy recursively with option -r

  cp -r d5 d6
- Notice that there is an f5 inside d6 too

- Now what will happen if you repeat the same command once more?

Copy a Directory into Existing Directory

- Create a directory d5 and move f5 into d5
- We want to copy d5 to d6

  cp d5 d6 won't work
- We have to copy recursively with option -r

  cp -r d5 d6
- Notice that there is an f5 inside d6 too

- Now what will happen if you repeat the same command once more?
  - since now d6 exists, it will copy d5 into d6

- If you repeat it the third time, it will overwrite
  - similarly, you can add option -i to prompt before overwrite

Locate

- To search files and directories by name in a typical Linux environment operating system, use a command that is called locate
- Unfortunately it's not installed on WebLinux
  - because locate uses a big database that would slow the system very much
  - you can use it by locate followed with the filename
  - you will need to update the database as well by updatedb

- Instead, let us explore some other simpler ways

Search with Echo

- In home directory, type **echo p***
  - it will list all files in it that starts with p
- **echo /usr/bin/a*** will print all the files starting with a in the /usr/bin
- **echo /usr/bin/a??** will print the files that have three letters starting with a, and then followed by exactly two letters
- **echo /usr/bin/???** will print all the files in /usr/bin that are three letters long

Find

- type just **find** or **find .** will print all the files within a directory
- **find /** will list all the files within the the root of the system and recursively in all the directories
    - it will stop until we don't have any more memory to print all the results

Find

- type just **find** or **find .** will print all the files within a directory
- **find /** will list all the files within the the root of the system and recursively in all the directories
    - it will stop until we don't have any more memory to print all the results

- **find . -name "program.c"** will find in current directory file with name in double quotes
- **find . -name "prog*"** to find all files in . starting with prog

Find

- type just **find** or **find .** will print all the files within a directory
- **find /** will list all the files within the the root of the system and recursively in all the directories
    - it will stop until we don't have any more memory to print all the results

- **find . -name "program.c"** will find in current directory file with name in double quotes
- **find . -name "prog*"** to find all files in . starting with prog

- what if we try to search in the root instead? for example:
  find / -name "hello"
  you will find two search results, and lots of error message saying permission denied, because user doesn't have access to many different file systems.
  how do we get rid of these error messages?

Find File

- The command is: **find / -name "hello" 2> /dev/null**
  - we want to redirect the error messages
  - we use the greater than sign for redirection, but just before, put a 2
  - 2 means the second output, which is the error output
  - you could redirect it to a file, then delete it, but there is a better way!
  - redirect it to /dev/null, a file which acts like a black hole — anything you write inside it is lost forever

Find File

- The command is: **find / -name "hello" 2> /dev/null**
  - we want to redirect the error messages
  - we use the greater than sign for redirection, but just before, put a 2
  - 2 means the second output, which is the error output
  - you could redirect it to a file, then delete it, but there is a better way!
  - redirect it to /dev/null, a file which acts like a black hole — anything you write inside it is lost forever

- Now try **find / -iname "*joe*" 2> /dev/null**
  - the additional option i means ignoring the case
  - and we look for file with filename that contains substring joe

Find Directory

- To find not only the files but also the directories, we need to search within path
  find / -ipath "*joe*" 2> /dev/null

- You will see a lot of results, more than a page — what do we do in that case?

Find Directory

- To find not only the files but also the directories, we need to search within path
  find / -ipath "*joe*" 2> /dev/null

- You will see a lot of results, more than a page — what do we do in that case?
  - we pipe it to less
    (take the output of this command and put it as the input of less)
    find / -ipath "*joe*" 2> /dev/null | less

  - you can go through all the results using the same less keys like down and up arrows or the space bar
  - highlight joe by forward search **/joe**

Thank you for your attention !

- In this lab, you have learned:
  - Filesystem and Important Directories in Linux
  - Navigating the filesystem
    - Absolute vs Relative Path
  - Various ways of creating and reading files
  - Manipulating files and folders
  - Searching files and folders
  - Redirecting outputs