Bash

FITSUM ALEBACHEW

Kernels

A kernel is the heart and core of an operating system:

- has control over everything in the system
- responsible for facilitating interactions between hardware and software components
- also responsible for disk, memory and task management

A common kernel used by many operating systems is the Linux kernel.

Shells

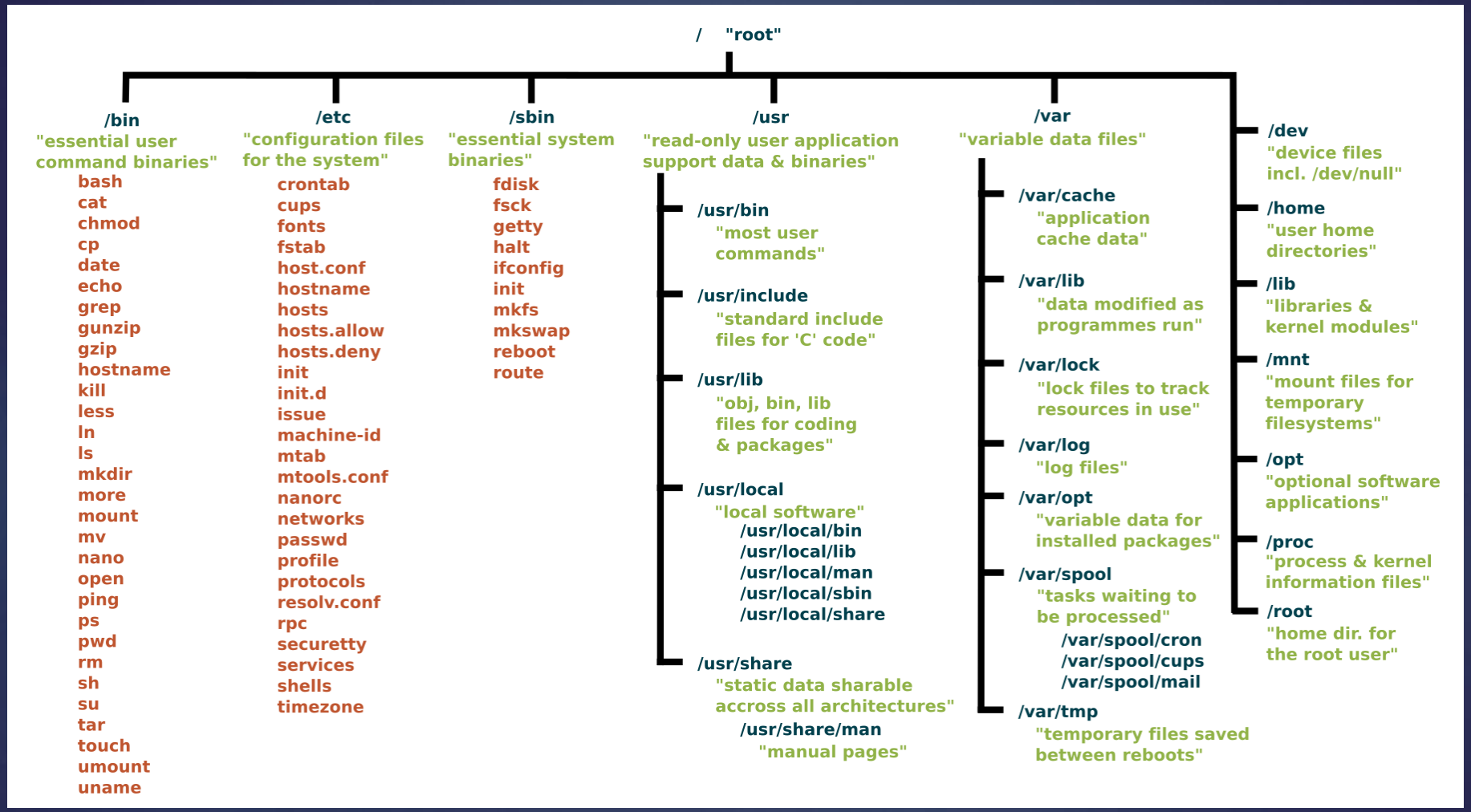
A shell is the program that interacts with the kernel:

- allows users and user applications to communicate with the OS
- uses either a GUI or a CLI (or both!)

There are many Unix/Linux shells:

- Bourne shell (bsh)
- C shell (csh)
- Korn shell (ksh)
- Bourne Again shell (bash)

Linux Filesystem



https://linuxfoundation.org/blog/classic-sysadmin-the-linux-filesystem-explained/

Navigating the Filesystem

In Linux, everything is a file. Therefore, it is essential to know how to navigate the filesystem:

- pwd – print working directory
- cd [path] – change directory
  - **.** : current directory
  - **..** : previous directory
  - **~** : home directory (usually /home/username)
- ls – view files in current directory (or supply path to directory)
  - **-l** : include file details
  - **-a** : show hidden files

Files and Folders

- mkdir [path] – create a new directory
- touch [path] – create a new file
- find -name [query] – find a file
- cp [source] [destination] – copy a file (-r for folders)
- mv [source] [destination] – move a file (-r for folders)
  - used to rename files/folders
- rm [path] – delete file permanently
  - **-rf**: recursively delete without warnings! (for folders)
    - **rm -rf /*** : one line killer (means delete EVERYTHING)

Viewing Files Contents

- cat – prints the contents of a file
- head/tail – print beginning/end of file respectively
  - -n [number]: print specific number of lines
- less – scrollable view of file
- wc – prints the length of a file in lines, words and characters
  - -w, -l, -c: only print word/line/byte count respectively

More options and functionality can be found in manual pages!

Important Commands

- grep – finds and prints lines of a file matching a pattern
  - grep [pattern] [path]
  - -v option to negate
  - -i for case insensitive
  - uses regex as pattern syntax
    - . : matches any character once
    - * : matches the previous character 0-any number of times
    - ^: start of line
    - $: end of line

Important Commands(cont.)

- cut [path] – finds and prints selective parts of each line of a file
  - -d “delimiter” -f [range]: print specific fields
- alias [new-command]=[definition] – create your own shortcut commands
  - no space around ‘=‘
  - Ex: alias la=‘ls –al’
- sort [path] – sorts the contents of a file
  - -n : numeric
  - -r : reverse
  - -k [n]: sort by field
- du [path] – show disk usage
- top – show CPU/memory utilization

Filename Expansion

Pathnames containing *, ? or [] are called wildcard patterns:

- ***** : matches any string (including empty string)
  - good for files starting/ending with a pattern
- **?** : matches any single character
- **[]** : matches characters in brackets (start with ^ to negate)
  - [abce], [a-ce] would match a, b, c, or e
  - [^abce], [^a-ce] would match any character except a, b, c, or e

Output Redirection and Piping

Redirect the output of a command to a file with:

- [command] > [path]
- use >> to append
- use 2> for error redirection (&> for both standard and error)
    - Ex: java test > output.log 2> error.log

Pipe the output of a command to another command with:

- [cmd1] | [cmd2]
    - Ex: du -s * | sort -nr | head -n 5 (prints the 5 biggest files/folders in directory)

Output Redirection and Piping


```
[picotte001] ~$ echo Hello World > file.txt
[picotte001] ~$ cat file.txt
Hello World
[picotte001] ~$ echo Second Line >> file.txt
[picotte001] ~$ cat file.txt
Hello World
Second Line
[picotte001] ~$
```

```
Print 5 Largest Files (NOT Folders)
[picotte001] demos$ ls -lR | grep ^- | sort -nrk5 | head -n5
-rw-r--r-- 1 fa496 vtune 166492663 Apr  5 12:29 Barnsley-Fern-Out.png
-rwxr-xr-x 1 fa496 vtune 83791872 Apr 25 12:49 lolcow_latest.sif
-rw-r--r-- 1 fa496 vtune 47885452 Apr 26 10:27 variables.data-00000-of-00001
-rw-r--r-- 1 fa496 vtune 2952882 Jun 13 09:51 libhpl.a
-rwxr-xr-x 1 fa496 vtune 1104640 Jun 13 09:51 xhpl
[picotte001] demos$
```

Variables

Variables can be defined in the shell as [name]=[value]

- no space around ‘=‘
- can be a number, character or string
- Ex: a=10, first_name=Alex

When using the value of variables, prepend with ‘$’

```
~$ a=10
~$ echo a
a
~$ echo $a
10
~$ echo "Max $a feet"
Max 10 feet
~$ echo 'Max $a feet'
Max $a feet
~$
```

Double quotes preserve the special character ‘$’. Use single quotes to treat as regular character

Conditional Statements

- && and ||
  - [cmd1] && [cmd2] : cmd2 is executed only if cmd1 succeeds
  - [cmd1] || [cmd2] : cmd2 is executed only if cmd1 fails
- if statements
  - if [test]; then [cmd1] ; elif [test]; then [cmd2] ; else [cmd3] ; fi
    - red parts required
    - test can be:
      - a command/series of commands
      - ‘’[[*expr*]]”, spaces required

*Bash conditional expressions: https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html#Bash-Conditional-Expressions

Conditional Statements


```
[picotte001] ~$ cat file.txt
Hello World
[picotte001] ~$ grep Hi file.txt && echo yes || echo no
no
[picotte001] ~$ grep Hello file.txt && echo yes || echo no
Hello World
yes
```

```
[picotte001] ~$ if ls |grep file.txt > /dev/null; then echo 'File found!!'; else echo 'File NOT found!!'; fi
File NOT found!!
[picotte001] ~$ touch file.txt
[picotte001] ~$ if ls |grep file.txt > /dev/null; then echo 'File found!!'; else echo 'File NOT found!!'; fi
File found!!
[picotte001] ~$
```

Loops

- for loops
  - for [var] in [list]; do [cmd]; done
- while loops
  - while [test]; do [cmd]; done

```
[picotte001] ~$ for a in {0..10..2}; do echo "Number is $a"; done
Number is 0
Number is 2
Number is 4
Number is 6
Number is 8
Number is 10
```

```
[picotte001] ~$ a=10
[picotte001] ~$ while [[ $a -ge 0 ]]; do echo "Number is $a"; let a=a-2; done
Number is 10
Number is 8
Number is 6
Number is 4
Number is 2
Number is 0
[picotte001] ~$
```

Bash Scripts

They are files with a list of bash commands

- should be an executable - (chmod +x [path])
- first line should be '#!/bin/bash'
- variables defined inside scripts are local

```
1   #!/bin/bash
2
3   echo Hello World
```

Questions?

- Feel free to attend my office hours every weekday 2 - 3 pm (any changes will be reflected on the URCF wiki main page): https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Main_Page#Talks_and_Workshops