Unix Terminal commands

Terminal commands

alias

- This allows you to create a shortcut for a command
- Syntax:

alias [name[='command']

Example:

alias quota='/usr/bin/quota –v'

So instead of typing a long command /usr/bin/quota –v

You just type quota on the terminal (the command)

Terminal commands

alias continued

- To see all the aliases created just type alias

alias entered

OpenSSH SSH client

```
charalg@compute0:~$ alias
alias quota='/usr/bin/quota -v'
alias rm='rm -i'
alias which='type -path'
charalg@compute0:~$
```

List of aliases shown

Terminal commands

- **alias continued**
- **To remove an alias command shortcut, use unalias [command]**

```
charalg@compute0:~/bscrt$ alias
alias ls1='ls -l'
alias quota='/usr/bin/quota -v'
alias rm='rm -i'
alias which='type -path'
charalg@compute0:~/bscrt$ unalias ls1
charalg@compute0:~/bscrt$ alias
alias quota='/usr/bin/quota -v'
alias rm='rm -i'
alias which='type -path'
```

Run alias command to list aliases

Lists the aliases set

Remove ls1 alias using unalias

Lists the aliases set, ls1 has been removed

Terminal commands

- at
- allows you to set a time to run a program/command
- Syntax:
    - **at time [date] commands**

- Example:
- ~/bscrt$ at 10.44 -f test1.sh > out1
- Here a job is set to run at 10.44 and it will run a script called test1.sh

the script will write to a file called out1 listing the date and the list the files in the local directory

Terminal commands

```
charalg@compute0:~/bscrt$ at 10.50 -f test1.sh
warning: commands will be executed using /bin/sh
job 13 at Tue Jul 27 10:50:00 2021
charalg@compute0:~/bscrt$ cat out1
hello charalg
Tue 27 Jul 10:50:00 BST 2021
total 128
-rw-r--r-- 1 charalg UnixStaff  0 Jul 27 10:28 a.out
-rw-r--r-- 1 charalg UnixStaff 43 Jul 27 10:50 out1
-rwxr-xr-x 1 charalg UnixStaff 86 Jul 27 10:43 test1.sh
```

Run script at 10:50

output from the at command

Display content of file out1

Content of out1 after script executed

Terminal commands

- at continued
- To list jobs set using the at command use –l option
- E.g at –l
- To remove at job use at –r [job_no]

```
 charalg@compute0:~/bscrt$ at -l
14      Tue Jul 27 11:50:00 2021 a charalg
charalg@compute0:~/bscrt$ at -r 14
charalg@compute0:~/bscrt$ at -l
charalg@compute0:~/bscrt$
```

Run at –l command lists jobs to run
Listing job number 14

Delete job 14
Then list jobs to run
Lists no jobs

Terminal commands

- Example of at with date

- ~/bscrt$ at 11.50 12/07/2021 -f test1.sh
- A job is created to run on the 7th of December at 11:50am
- charalg@compute0:~/bscrt$ at -l
- 15 Tue Dec 7 11:50:00 2021 a charalg

Listing jobs allocated

Terminal commands

- cal
- Displays the calendar month
- Example
- charalg@compute0:~/bscrt$ cal
- July 2021
- Su Mo Tu We Th Fr Sa
- 1 2 3
- 4 5 6 7 8 9 10
- 11 12 13 14 15 16 17
- 18 19 20 21 22 23 24
- 25 26 27 28 29 30 31

Terminal commands

- cal continued

```
charalg@compute0:~/bscrt$ cal 10 2025
   October 2025
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Run calendar command cal to display the month 10 i.e. October of the year 2025

Output from cal

Terminal commands

- cat
- Concatenates files and outputs to standard output
- Syntax:
    - **cat [option] [file]**
- **Options**
- **-n shows line numbers**
- **-A shows ALL; will include a dollar to mark the end of line**

Terminal commands

- Example using cat

cat -n 1 2 3

1 hello 1

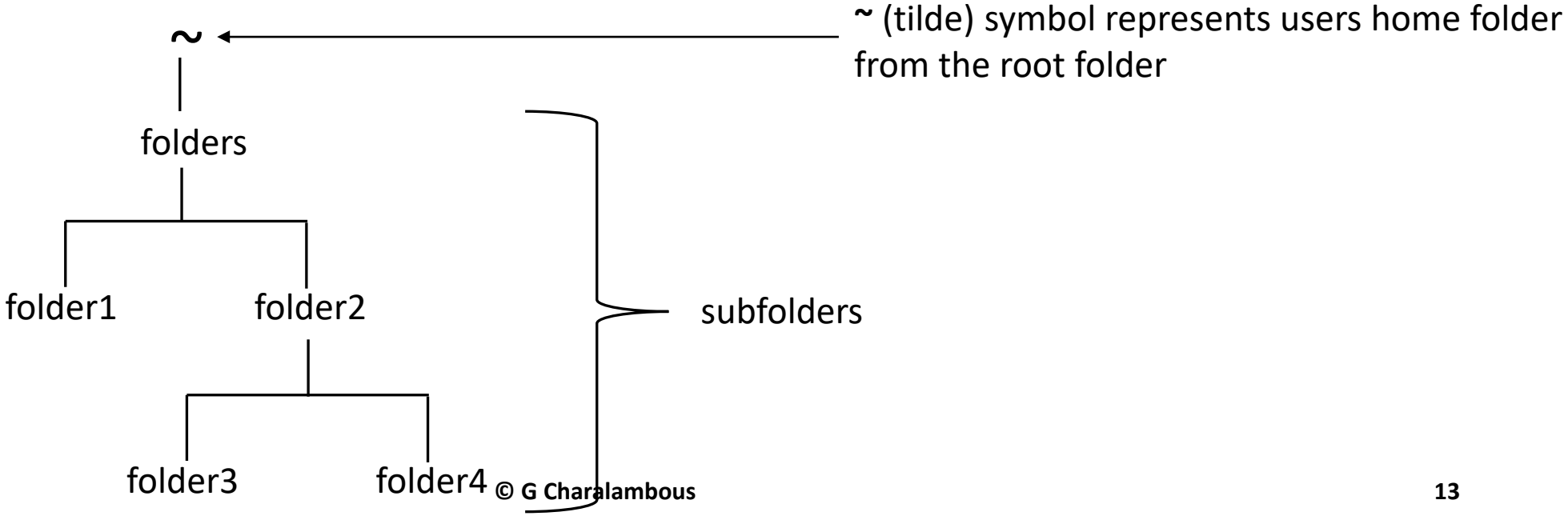
2 hello 2

3 hello 3

Concatenates and displays files 1, 2 and 3 with the option –n to show line numbers

Terminal commands

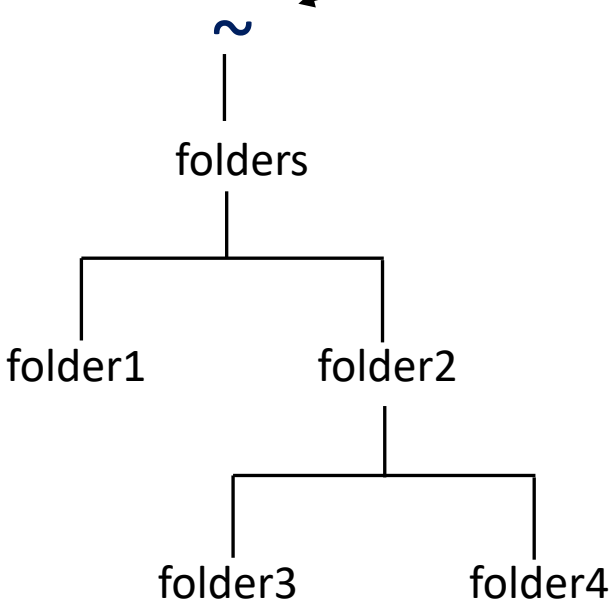
- cd
- Change directory
- Syntax ch [path/new_directory]
- Consider the following folder structure



Terminal commands

- cd

Assume we start at the users home directory



To move into folder1 we type

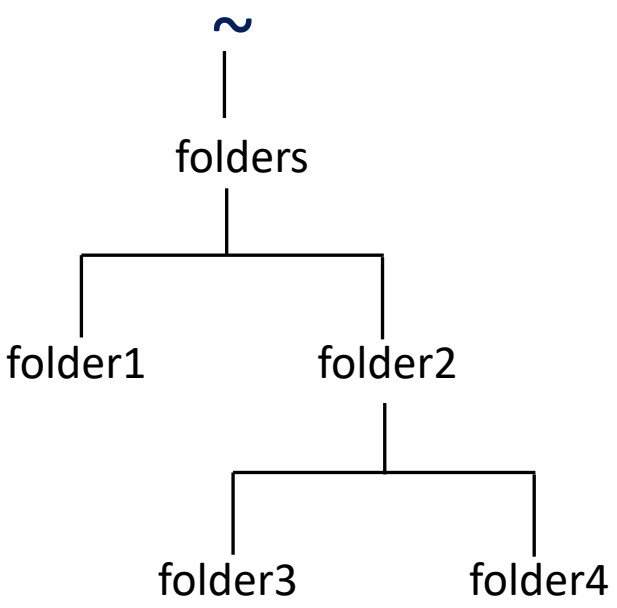
cd folders/folder1

To move into folder4 we type

cd folders/folder2/folder4

Terminal commands

- cd



Assume we are at folder4

To move into folder2 we type

cd .. (.. 2 dots back up one directory)

To move into folder3 we type

cd ../folder3

To move into folder1 we type

cd ../../folder1
OR
cd ~/folders/folder1

Terminal commands

- chmod
- Change file permissions for a file

- Each file has 3 sets of file permissions for access from
  - 1 user
  - 2 group
  - 3 others

For each set the file permission is represented by a 3 bit binary number

Most Significant bit == read
Middle bit == write
Least significant bit == executable

When the bit is set to 1 – this signifies will have the corresponding file person, otherwise its set to 0

Terminal commands

- chmod/file permissions continued
- Examples
- Binary values of file permission
  - 100 represents read permission only
  - 110 represents read and write permission only
  - 111 represents read, write and executable
- We don’t normally use binary, but octal (base 8)

Converting Binary to Octal

- Use place values

  for a 3 bit binary number we have the place values

  4 2 1

To convert to octal multiply the binary place value with the corresponding binary digit and sum the result

Example 110 (read-write permission)

$$\begin{array}{ccc} 4 & 2 & 1 \\ \times & \times & \times \\ 1 & 1 & 0 \\ \hline \end{array}$$

4 + 2 + 0 -> 06

We add a preceding 0 to define we are using base 8/ octal

Terminal commands

- chmod
- Syntax to change file permission
- chmod octal_number filename
- Examples
- chmod 0711 test1.sh
- We have the octal number

	7	1	1
PERMISSIONS FOR	User	group	others
Binary values	111	001	001
	Read, write, executable	executable	executable

So test1.sh
Can be read, write to and executed by the user

Any members of his/her group will be able to execute the file

Any others will be able to execute the file

Terminal commands

- chmod using symbolic modes
- class of user

  u user; g group; o others, a all groups
- operation

  + add permission; - remove permission; = set permission
- Access permission

  r read; w write; x execute;

Terminal commands

- chmod example
- chmod a-rwx,a+x,u+rw test1.sh

This will make test1.sh file have
read, write and executable permission for user
executable for group
executable for other

a-rwx : remove for all read, write and executable
a+x :add executable for all
u+rw :add read and write permission for user

Alternative way: **chmod 0711 test1.sh**

Terminal commands

- examples

The type of user whose permissions you want to change:

u: user
g: group
o: others
a: u g and o

Default permissions as produced by emacs. Only the user can write or read no one else can touch it.

```
-rw-------   1 aroollyt 800        22 Dec  2 18:16 coursewk1.c
```

Add read permissions for my group. Anyone in the same group can now copy file.

```
chmod g+r coursewk1.c
-rw-r-----   1 aroollyt 800        22 Dec  2 18:16 coursewk1.c
```

Add read and write permissions for all users. Anyone who can log on to the system can now copy, alter or even delete file.

```
chmod o+rw coursewk1.c
-rw-r--rw-   1 aroollyt 800        22 Dec  2 18:16 coursewk1.c
```

Remove read and write permissions for everyone except me.

```
chmod go-rw coursewk1.c
-rw-------   1 aroollyt 800        22 Dec  2 18:16 coursewk1.c
```

Remove read and write permission for user; user has no longer access to file!!

```
chmod u-rw coursewk1.c
----------   1 aroollyt 800        22 Dec  2 18:16 coursewk1.c
```

Terminal commands

- chown
- Change file owner
- Syntax chown [option] owner filename
- Options
  - -v (verbose) print out messages of change
  - -f silent, suppress error messages

Example:
>> chown -v charalg 1
>> ownership of '1' retained as charalg

Terminal commands

- cksum
- Carry out cyclic redundancy checksum and count the number of bytes in file
- Syntax : cksum filename
- Used to ensure file has not been changed
- Example

>> cksum test1.sh

3039837459 86 test1.sh

Output: CRC no_bytes filename

Terminal commands

- clear
- Clears terminal screen
- Syntax clear

Terminal commands

- cmp
- Compares two files byte by byte
- Syntax cmp [option] file1 file2
- Options
- -b print differing bytes
- -l output byte numbers and differing byte values
- -s suppress output; (used in scripts) returns

  0 files identical

  1 files different

  2 missing file(s)

Terminal commands

- cmp continued
- Example

>> cmp file1 file2

file1 file2 differ: byte 7, line 1

>> cmp -l file1 file2

7 61 62

>> cmp -b file1 file2

file1 file2 differ: byte 7, line 1 is 61 1 62 2

file1:

hello 1
hello a

file2:

hello 2
hello a

61 is the octal number for the character 1
62 is the octal number for the character 2

Terminal commands

- cp
- copies files
- Syntax
- cp [options] source destination
- cp [options] source directory
- Options
- -f overwrites without asking for permission
- -i prompts for permission to copy if file exists
- -p keeps file permissions, other will allocate permissions of user
- -r used to copy folders and subfolders

Terminal commands

- cp continued
- Example
- To copy a folder (folder1) and subdirectory to a directory new

  cp –r folder1 new

If new does not exist will create folder and copy files/subdirs from folder1

If new does exist wit will add a create subfolder called folder1 and copy all files/dir contained

Terminal commands

- date
- Displays date and time
- Syntax date [option] [+ format]
- Option
- -u print universal time
- -r [file_name]
- Example
- >> date

Wed 28 Jul 11:39:45 BST 2021

- >> date –u
- Wed 28 Jul 10:39:48 UTC 2021

Terminal commands

- date continued
- >> date –r file1

Tue 27 Jul 11:53:40 BST 2021 outputs last time/date file modified

Terminal commands

- date format options **+%[option]**
- %D – date format mm/dd/yy
- %d – day of month
- %m – month as number
- %B - month
- %H – hour
- %M – minutes
- %S - seconds

Terminal commands

- df
- Returns the amount of free space
- Syntax df [options] [drive_name]
- Options
- -i -give i-node data
- -l returns data on local file
- -h gives output in human readable form

Terminal commands

- df example

>> **bscrt$ df /dev/sda1**

```
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       46028800 22807700  20859924  53% /
```

>> **df -h /dev/sda1**

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        44G   22G   20G  53% /
```

>> **df -ih /dev/sda1**

```
Filesystem     Inodes IUsed IFree IUse% Mounted on
/dev/sda1       2.8M  767K  2.1M   27% /
```

Terminal commands

- diff
- Compares files displays differences
- Syntax diff [option] files
- Option
- -q only report when files differ
- -s only report if the files are identical
- -l paginate output
- -c output in context mode
- -u output in unified mode

Terminal commands

- diff continued
- Example consider 2 files

>> diff F1 F2

2,3c2

< hello 2

< hello a

> hello a this is a test

F1:

hello 1
hello 2
hello a

F2:

hello 1
hello a this is a test

2,3c2 reads from File F1 line 2 to 3 change to match file F2 line 2

Terminal commands

- diff continued
- Example consider 2 files

>> diff F1 F2

2,3c2

< hello 2

< hello a

> hello a this is a test

F1:

hello 1
hello 2
hello a

F2:

hello 1
hello a this is a test

2,3c2 reads from File F1 line 2 to 3 change to match file F2 line 2

Terminal commands

- diff continued
- Example consider 2 files

```
>> diff –c F1 F2
*** F1  2021-07-28   15:34:30.113383000 +0100
---   F2  2021-07-28   15:35:03.941677000 +0100
***************
*** 1,3 ****
  hello 1
! hello 2
! hello a
--- 1,2 ----
  hello 1
! hello a this is a test
```

F1:

hello 1
hello 2
hello a

F2:

hello 1
hello a this is a test

1st 2 lines
*** represents F1 and --- represents F2
showing name, modified date, modified time for each

Lists lines 1 to 3 for F1
Line 1 no ! Sign so matches with Line 1 in F2
Next two lines do not match

Lists lines 1 to 2 for F2
Line 1 no ! Sign so matches with Line 1 in F1
Next line does not match

Terminal commands

- diff continued
- Example consider 2 files

```
>> diff –u F1 F2
---     F1  2021-07-28   15:34:30.113383000 +0100
+++   F2  2021-07-28   15:35:03.941677000 +0100
@@ -1,3 +1,2 @@
 hello 1
-hello 2
-hello a
+hello a this is a test
```

F1:	**F2:**
hello 1	hello 1
hello 2	hello a this is a test
hello a	

1st 2 lines
--- represents F1 and +++ represents F2
showing name, modified date, modified time for each

@@ -1,3 +1,2 @@ shows the range compared wrt F1 and F2
-(file F1 from line 1 to 3) +(file F2 from line 1 to 2)

Lists changes required of file F1 to mark it identical to file F2
No sign for 1st line leave
- symbol: delete line 2
- symbol: delete line 3
+ symbol add the following line to F1

Terminal commands

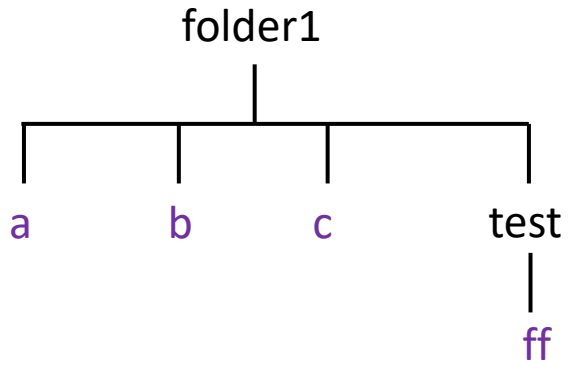
- du
- estimates file space usage
- Syntax du [options] [directories]
- Options
- -a includes totals for all files
- -c generate a grand total
- -h human readable sizes

Terminal commands

- du
- Example

```
>> du -hac folder1
32K     folder1/c
32K     folder1/a
32K     folder1/test/ff
72K     folder1/test
32K     folder1/b
208K    folder1
208K    total
```

Consider the following folder with its sub-folders (black) and files (purple)



Options:
-hac
h Human readable sizes
a generate totals for all files
c generate a final total

Terminal commands

- env
- Displays/changes environment variables

Terminal commands

- find
- Search for a file
- Syntax find pathname expression
- Example
- >> find ~ -name a –print

Search from home folder defined ~

Named file (-name) a

Display on screen (-print)

Terminal commands

- find
- Search for a file
- Syntax find pathname expression
- Example
- >> find . -name a –print

Search from home folder defined . Represents current folder

Named file (-name) a

Display on screen (-print)

Terminal commands

- find
- Search for a file
- Syntax find pathname expression
- Example
- >> find ./progs -name \[a-c]*.c –print

Search from current folder subfolder progs the named files beginning with a,b or c with the extension .c and display

Expression \[a-c]*.c
Here we are pattern matching look for 1st letter a,to c then * (any number of letters, followed by .c

Terminal commands

- hexdump
- Display file in ASCII,decimal,hexadecimal or octal
- Syntax: hexdump [option] filename
- Option
- -b one byte octal display : Display the input offset in hexadecimal
- -c one byte character : Display the input offset in hexadecimal
- -C Canonical hex+ASCII display : Display the input offset in hexadecimal
- -d Two-byte decimal display: Display the input offset in hexadecimal

Terminal commands

- Hexdump
- Example

>> hexdump -b file1

0000000 061 012

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- Hexdump
- Example

```
>> hexdump -C file1
00000000  31 0a                                       |1.|
00000002
```

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- Hexdump
- Example

```
>> hexdump -c file1
0000000   1  \n
0000002
```

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- Hexdump
- Example

>> hexdump -d file1

0000000 02609

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

We are representing our characters in 2 byte decimal number

00001010 (newline character) followed by 00110001 (‘1’ character)
00001010 00110001 in binary is 02609 in decimal (architecture – little endian)

Terminal commands

- history
- Lists commands typed on the terminal
- Usage

- history [n] lists last n commands typed
- Example : history 10 will list the last 10 commands typed

- To run rerun an instruction type !n where n represents the instruction number example !1081
- To run the previous command type !!

Terminal commands

- kill
- Used to terminate a process
- Syntax kill [option] IDs
- Option
- -l lists signal names with there values

Terminal commands

- kill

Example if we had a process with an id 1234

To terminate the process type kill -9 1234

-9 signal – represents stop process immediately

Terminal commands

- ls
- List files in the local directory
- Syntax ls [option] [filename]
- Options
- -l lists file details one per line
- -a list files including hidden files
- -g lists file details found in i-node
- -t lists files ordered by modification time

Terminal commands

- ls
- Example

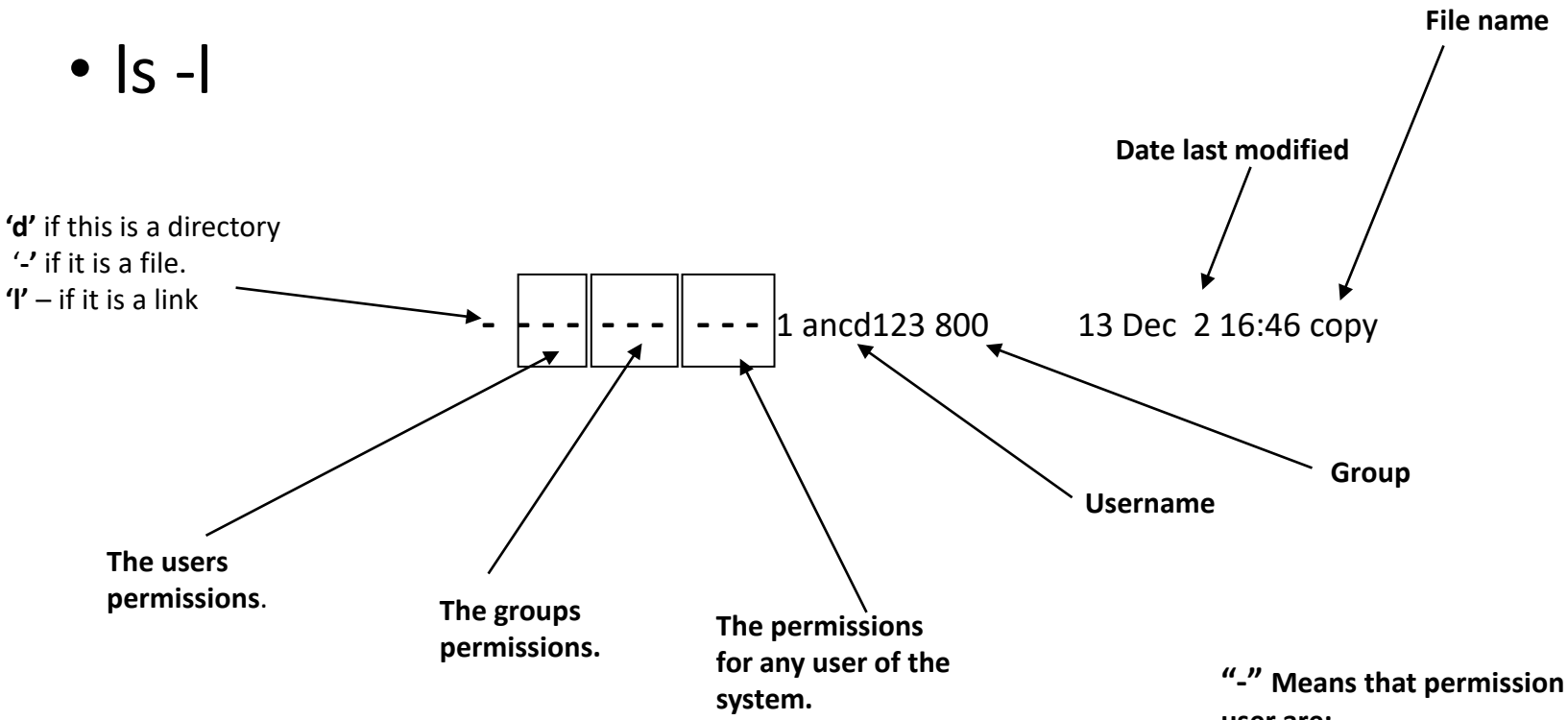
```
>> ls –lt
-rwxr-xr-x 1 charalg UnixStaff 8608 Jul 28 16:00 a.out
-rw-r--r-- 1 charalg UnixStaff   59 Jul 28 16:00 test.c
-rw-r--r-- 1 charalg UnixStaff   31 Jul 28 15:35 F2
-rw-r--r-- 1 charalg UnixStaff   24 Jul 28 15:34 F1
-rw-rw-rw- 1 charalg UnixStaff   47 Jul 28 15:30 1
-rw-r--r-- 1 charalg UnixStaff   24 Jul 28 15:06 1b
-rw-r--r-- 1 charalg UnixStaff   16 Jul 28 11:22 1bb
drwxr-xr-x 3 charalg UnixStaff   82 Jul 28 11:16 new
drwxr-xr-x 3 charalg UnixStaff   79 Jul 28 11:15 folder1
```

Terminal commands

- **ls –l output:**
- **drwxr-xr-x 3 fred123 UnixStaff 82 Jul 28 11:16 new**
- **d**:The d defines that new is a directory a – represents a regular file
- **rwxr-xr-x**: These are the permissions
  - The 1st 3 permissions rwx for the user (read, write and executable)
  - The next 3 permissions r-x for the group(read and executable)
  - The last 3 permissions r-x for the others(read and executable)
- **3**: is the link and directories inside this directory.
- **fred123**: is the user and owner
- **UnixStaff**: is the group fred123 belongs to
- **82**:size in bytes
- **Jul 28 11:16** :is the date/time modified
- **new** :is the name of the file/folder

Terminal commands

- ls -l



"-" Means that permission is not granted. Permissions for each class of user are:

'r' the file is readable by that class of user
'w' the file is readable by that class of user
'x' the file is readable by that class of user

Terminal commands

- man
- Manual pages – this is the help for the commands on the linux system
- Syntax man [command]
- Example man ls
- Note use your keyboard up and down arrows to move
- Or spacebar to page down and u to page up
- Press q to exit and go back to the terminal

Terminal commands

- mkdir
- Allows you to create directories/folders
- Syntax mkdir [folder_name] or mkdir –p [path_folder_name]
- Example
- mkdir folder1
- This will create a subfolder called folder1 in the current folder you are in

Terminal commands

- mkdir
- To create a folder path

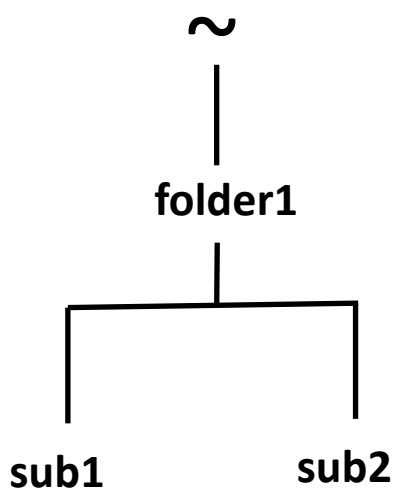
>> mkdir –p folder1/sub1

>> mkdir –p folder1/sub2

If a folder or subfolder exists on path defined

System will traverse to the point where in creates

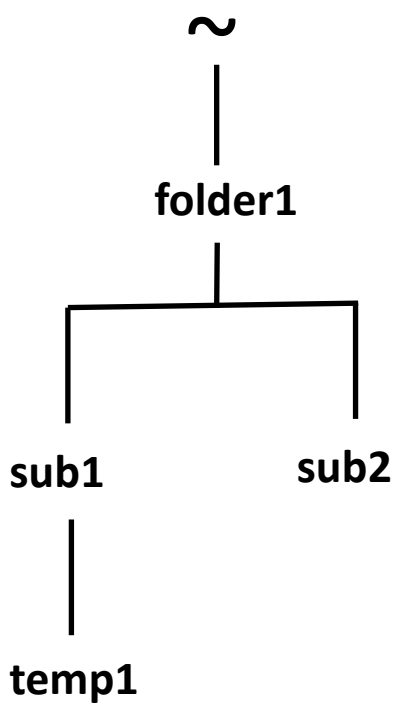
The remaining folders along path



Terminal commands

- mkdir
- To create a folder path

>> mkdir –p folder1/sub1/temp1



Terminal commands

- mv
- Move a file/directory or change name of file/directory
- Syntax mv [option] source_file target_file
- Options
- -f force move/rename
- -i ask for confirmation if an existing filename/foldername exists

Terminal commands

- mv
- Examples

>> mv test.c testf.c

Renames test.c to testf.c in the current folder

>> mv folder1 ~/folder1

Moves folder1 from current location to user home

>> mv test.c ~/

Moves file test.c to user home folder

If a folder does not exist in destination path mv will create it

Terminal commands

- od
- Octal/hexadecimal/ASCII dump
- Syntax od [option][filename]
- Option
- -a ASCII text
- -b Octal byte
- -c ASCII byte
- -d unsigned decimal
- -h short hexadecimal

Terminal commands

- od
- Example

>> od -a file1

0000000 1 nl

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- od
- Example

>> od -b file1

0000000 061 012

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- od
- Example

```
>> od -c file1
0000000   1  \n
0000002
```

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- od
- Example

>> od -d file1

0000000 2609

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- od
- Example

>> od -h file1

0000000 0a31

0000002

file1:

1

Char ‘1’ = 49 in decimal; 61 in octal; 31 in hexadecimal
Newline ‘\n’ = 10 in decimal, 12 in octal; A in hexadecimal

Terminal commands

- ps
- Displays process running for user
- Syntax ps [option]
- Option
- -e to see all processes
- -f to see user processes - detail
- -l to see user processes – long/

Terminal commands

- ps
- Example

>> ps –l

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	69080	29794	29793	0	80	0	-	4820	wait	pts/1	00:00:00	bash
0	R	69080	29952	29794	0	80	0	-	7230	-	pts/1	00:00:00	ps

Terminal commands

- PROCESS FLAGS

F

- The sum of these values is displayed in the "F" column, which is provided by the flags output specifier:

- 1 forked but didn't exec
- 4 used super-user privileges

Terminal commands

- PROCESS FLAGS

S

- The state of process values can be:
- D uninterruptible sleep (usually IO)
- R running or runnable (on run queue)
- S interruptible sleep (waiting for an event to complete)
- T stopped by job control signal
- t stopped by debugger during the tracin
- Z defunct ("zombie") process, terminated but not reaped by its parent

Terminal commands

- PROCESS FLAGS

UID – user ID;

PID – process ID,

PPID parent process ID,

C – CPU utilisation,

PRI – Priority (lower the value the higher the priority),

NI – nice value (lower the value the higher the priority,

ADDR – Memory address of the process

SZ – Virtual memory usage

WCHAN – waiting channel, ‘-’ process running; ‘wait’ - sleeping

TTY – terminal type,

T - Total CPU usage

CMD – the process name

Terminal commands

- pwd
- Displays pathway to your current directory

Terminal commands

- rm
- Remove/delete files
- Syntax: rm [option] filename
- Option
- -f – force the removal
- -i – prompt before removal
- -r – remove directories recursively

Terminal commands

- rmdir
- Remove empty directories
- Syntax rmdir [option] directory
- Option
- -p deletes specified directories defined by the path
- Example

>> rmdir folder1

Deletes folder1 if empty

>> rmdir –p folder2/folder3

Deletes subfolder folder3 if empty then deletes folder2

Terminal commands

- sort
- Sorts out the content of a text file
- Syntax sort [option] [filename]
- Option
- -b ignore leading blanks in a line
- -d ignore special characters, punctuation characters
- -f ignore case
- -i ignore non-printed characters
- -n carry out a numerical sort (considers -, decimal point)
- -r reverse oder of sort

Terminal commands

file1:

one theo

hello

a

f

g

two

- sort
- Example 1:

>> sort –fi file1

a

f

g

hello

one theo

two

- Example 2:

>> sort –fir file1

two

one theo

hello

g

f

a

Terminal commands

- tee
- read from terminal and write to standard output and files
- Syntax tee [option] [file]
- Option
- -a append output to a file

  if omitted will overwrite file

Terminal commands

- tee
- Example

>> echo "hello world" | tee out1

Here echo will print a string (display using standard output)

We use a special symbol | this is called a pipe

It will take the output from one command and pass it to a second command (process) as its input(standard input)

The tee command will display the string “hello world” (write it to the buffer standard output) as well as write it to the file out1

Terminal commands

- tee
- Example

```
>>  echo "hello world" | tee out1
hello world
>> cat out1
hello world
```