Linux part III

Working with commands;
shell expansion and quoting;
permissions

How to capitalize the first letter of each word in the sentence

```
$ echo how to capitalize each word | sed -e "s/\b./\u\0/g“
```

\b – word boundary

. – match any character

\u – turn the next character to uppercase

\0 – everything that was matched, in this case everything that was matched is only ‘.’, as the word anchor is not included into the match

Working with commands

type displays what kind of command we are dealing with

```
$ type cp
$ type whoami
$ type ls
$ type type
```

which - where is the executable for a command located?

```
$ which sed
$ which which
$ which bash
$ which cd
```

whatis – what is this command?

```
$ whatis which
$ whatis cp
```

Working with commands

Get help for the command:
help <command> - works only for built-ins, like cd

```
$ help cd
```

<command> --help – print help for the command

```
$ grep -help
$ ls -help
```

man <command> - view the manual for the command in ***less***

```
$ man sed
```

apropos <topic> – display appropriate commands that deal with certain topic

```
$  apropos schedule task
```

alias - create aliases for command

```
$ type c # check if command exists
$  alias c='clear’
```

Shell expansion and quoting

Shell expansion

```
$ echo *

$ echo *.txt

$ echo /usr/*/share

$ echo [[:upper:]]
```

Hidden files start with **.**, for example ***.bashrc***

```
$ ls -l ~/.* # expand all hidden files and directories in the home folder
```

Arithmetic expansion

```
$ echo $((2 + 2))
$ echo $((10 * 4))
echo $(( (4 * 2) / 4  ))
```

Shell expansion and quoting

Arithmetic expansion (continued)

```
$ echo $((5/2))
$ echo $(( 0.5 * 2  )) # bash can handle only integers, use bc for non-integer calculation
```

Brace expansion (create text of a pattern between braces)

```
$ echo sample_{1..10}
$ echo sample_{A,B,C}
$ echo {1,2,3}_{_,@}
$ echo {A{1,2},B{1,2}}
$ mkdir {2022..2023}_{01..12}
```

Command substitution (use an output of the command in brace expansion)

```
$ ls -l $(which cp)
$ ls -l `which cp`
```

Shell expansion and quoting

Quoting

```
$ echo a    b     c
$ echo “a    b     c”
```

Command expansion works in double quotes

```
$ echo $(date)
$ echo “$(date)”
```

Use single quotes to suppress expansions and treat the argument as simple text

```
$ echo '$(date)’
```

Escape special characters with back slashes

```
$ echo \$\(date\)
```

Permissions

id – display user identity

chmod – change mode of the file

umask – sets default permissions

su – run a shell as another user

sudo – run a command as another user

chown – change file’s owner

chgrp – change file’s group ownership

passwd – change users password

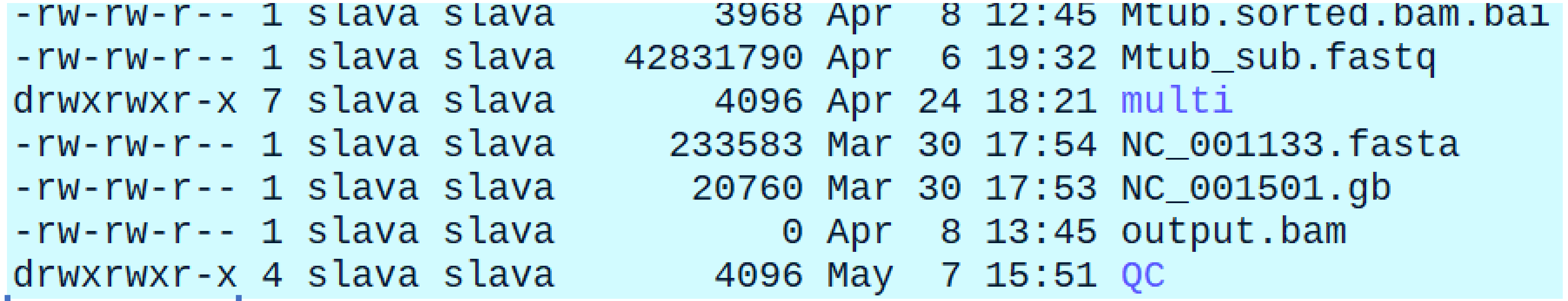
Permissions

Show user identity

$ id

View file permissions

$ ls -l



```
-rw-rw-r-- 1 slava slava       3968 Apr  8 12:45 Mtub.sorted.bam.bai
-rw-rw-r-- 1 slava slava   42831790 Apr  6 19:32 Mtub_sub.fastq
drwxrwxr-x 7 slava slava       4096 Apr 24 18:21 multi
-rw-rw-r-- 1 slava slava     233583 Mar 30 17:54 NC_001133.fasta
-rw-rw-r-- 1 slava slava      20760 Mar 30 17:53 NC_001501.gb
-rw-rw-r-- 1 slava slava          0 Apr  8 13:45 output.bam
drwxrwxr-x 4 slava slava       4096 May  7 15:51 QC
```

Permissions

The first character in permission line can be: - - file; d – directory; l – symbolic link; c – terminal or dev/null; b – block special file, like hard drive device, DVD, etc

Permissions

Permission example: -rw-rw-r--

r – the file is open for reading

w – the file can be written to

x – the file can be executed

(1)rwx (2)rwx (3)rwx

(1)Owner (2)Group (3)World

Permissions

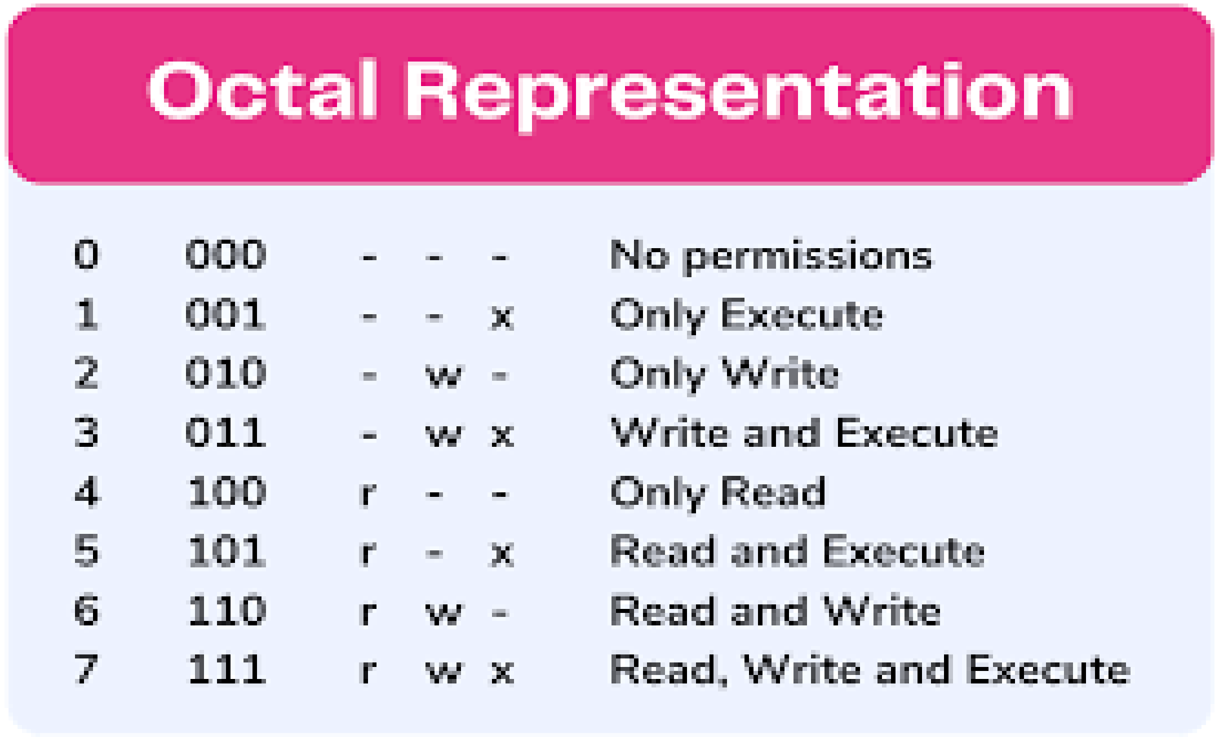
-rwx------ - the file is readable, writable, executable by the owner, but not by anyone else

-rw-r--r-- - the file can red and written to by the owner and read by everyone else

drwxrwx – directory that can be opened and written to by everyone

chmod – change mode of the file

Modes in octal notation



Octal Representation

0	000	-	-	-	No permissions
1	001	-	-	x	Only Execute
2	010	-	w	-	Only Write
3	011	-	w	x	Write and Execute
4	100	r	-	-	Only Read
5	101	r	-	x	Read and Execute
6	110	r	w	-	Read and Write
7	111	r	w	x	Read, Write and Execute

Permissions

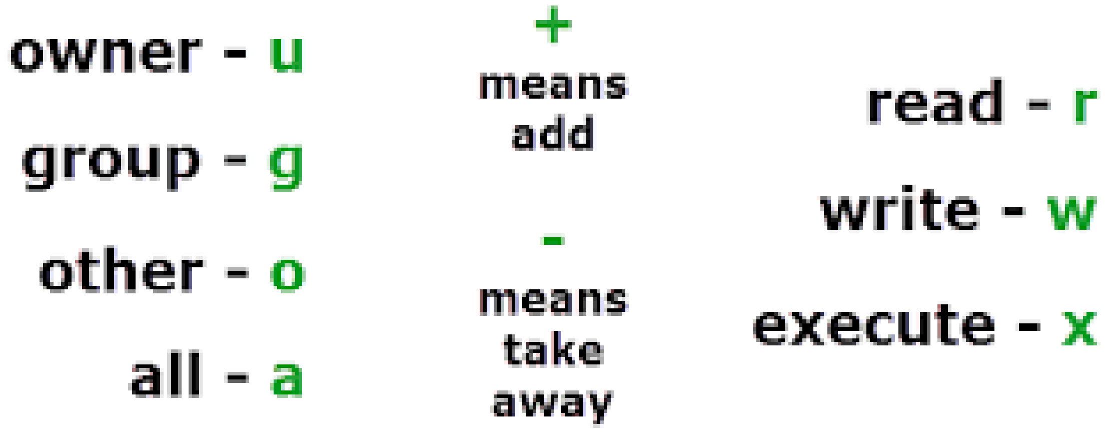
```
chmod 600 <some_file>
```

- `6 – Owner – read and write`
- `0 – Group – no permissions`
- `0 – World – no permissions`

```
Result: -rw-------
```

Permissions

Symbolic notation



owner - u
group - g
other - o
all - a

+ means add

- means take away

read - r
write - w
execute - x

Give an owner permission to execute the file

```
chmod u+x <some_file>
```

Take away permission to read, write and execute from the user group

```
chmod g-rwx <some_file>
```

Permissions

Create a test directory

```
$ mkdir test_dir
$ chmod u-rwx test_dir # take away all permissions from the user
$ ls test_dir
```

Give back the permissions

```
$ chmod u+rwx test_dir
$ ls -l test_dir
```

Set default permissions with umask

```
$ umask # show default permissions
$ umask 0000 # set default permissions
```

su – run shell as a different use

```
$ su -l steve # run shell as user steve
$ su - # run shell as root
```

Permissions

sudo – run command as a different use

```
$ sudo apt get install bowtie
```

chown – change file owner

```
$ touch test_file
$ sudo chown steve test_file # overturn file ownership to steve
$ sudo chown steve:workshop test_file # make steve and user-group “workshop” owners of the file
$ sudo chown :workshop test_file # give the file ownership to the “workshop” user group
```

passwd – change password

```
$ passwd # change your password
$ sudo passwd steve # change a password for steve
```