Linux

intro

Location

- pwd

  `print name of current/working directory`
- cd

  `change the working directory`
- ls

  `list directory contents`
- find

  `search for files in a directory hierarchy`
- mkdir

  `make directories`

Actions on files

- cp

  `copy files and directories`
- mv

  `move (rename) files`
- rm

  `remove files or directories`
- rsync

  `a fast, versatile, remote (and local) file-copying tool`

View of text files

- cat
  `concatenate files and print on the standard output`
- head
  `output the first part of files`
- tail
  `output the last part of files`
- more / less
  `file perusal filter for crt viewing`
- wc
  `print newline, word, and byte counts for each file`

Other need to know commands

- chmod

  `change file mode bits`
- sh

  `command interpreter (shell)`
- ssh

  `OpenSSH SSH client (remote login program)`
- man

  `an interface to the on-line reference manuals`

Unknown command?

http://explainshell.com/

write down a command-line to see the help text that matches each argument

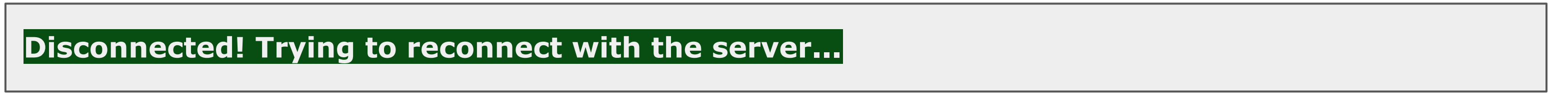
Exercise

A free online tutorial shell is available at:

https://www.tutorialspoint.com/unix_terminal_online.php

When linux commands are shown, the line of the command start with a $.
Lines without are often the output that the command returns.

If you are disconnected, just reload the page. (but you will lose all your current data)



```
Disconnected! Trying to reconnect with the server...
```

Exercise

1. Open the terminal (just go to the link). This is what you get:

```
sh-4.3$ /bin/bash
bash-4.3$
```

2. check your current directory

```
$ pwd
/home/cg/root
```

3. check which files and directory are in your current directory

```
$ ls
README.txt
```

4. check the permissions and size of the files in the current directory

```
$ ls -lh
total 4.0K
-rw-r--r-- 1 21664 21664 978 Nov 16 08:39 README.txt
```

Note:

1. the first part are the permissions. The first - stands for file (d for directory). There are 3 values describing the permissions for each owner, group and everyone (read (r), write (w) and execute (x)).
2. 21664 is the user. The next 21664 is the group.
3. 978 is the size of the file.
4. The creation date (Nov 16 08:39)
5. README.txt is the file name

Exercise

5. check the first lines of the README.txt file

```
$ head README.txt
We provide you an easy interface to Linux...
```

6. check the last lines of the README.txt file

```
$ tail README.txt
ln  -- Create...
```

7. view the complete README.txt file

```
$ cat README.txt
We provide you an easy interface to Linux...
$ more README.txt
We provide you an easy interface to Linux...
$ less README.txt
```

Note: less opens the file, and fills the complete terminal. You can use the arrows to scroll the file. Exit the file by pressing the q button.

8. Check the number of lines in the README.txt file

```
$ wc -l README.txt
23 README.txt
$ wc README.txt
 23 169 978 README.txt
```

Note: the -l option asks only for the number of lines. Without options, the output is: number of lines, number of words, number of characters and the file name

Exercise

9. print a line to the output

```
$ echo test
test
$ echo "echo test"
echo test
$ echo "echo \"test\""
echo "test"
```

Note: echo prints directly to the output. However it is possible to use echo in combination with complete sentences, it is safer to use quotes (‘ or “) at the start and end of your sentence. If you want to use quotes within the sentence, you need to escape them, using backslash (\)

10. print a line to a text file

```
$ echo "echo \"test\"" > script.sh
```

Note: any output can be directed to a file. > can be used to write to a file (the file can be non existing). If the file exists, it be overwritten. To append to a file use >>.
Here we add a sh extension, which indicates that this is a bash script.

11. check the text file, and its permissions

```
$ cat script.sh
echo "test"
$ ls -lh script.sh
-rw-r--r-- 1 21956 21956 12 Nov 16 08:58 script.sh
```

Note: if you want to reuse a previous command (like ls -lh), you can use the up and down arrows to browse the history. Shell is also smart, and uses tab-completion. When you start typing, and press tab, it will try to complete the filename or command. If not, try double tab. If multiple options are available continuing from your line, all options will be shown.

Exercise

12. before we execute the new script, we will put it in our mydir directory. But first this directory has to be created.

```
$ mkdir mydir
sh-4.3$ mkdir mydir
mkdir: cannot create directory 'mydir': File exists
sh-4.3$ mkdir -p mydir
```

Note: mkdir creates a directory, but give an error if it does not exists. The -p option does not give an error, when the directory already exists. Moreover, it creates the parent directories if they do not exists either.

13. Example of long path directory creation:

```
$ mkdir -p mydir/somedir/another
$ ls mydir/somedir
another
```

14. Go to the mydir directory, and show its content

```
$ cd mydir/
$ ls
somedir
```

Exercise

15. We do not need the somedir, so we will remove it.

```
$ rm somedir/
rm: cannot remove 'somedir/': Is a directory
$ rm -r somedir/
$ ls
```

Note: rm is used to delete files. For directories, the -r option has to be defined (-r is recursive). Beware that also the content of this directory will be deleted.

16. Now we copy the script to this directory. But first we check the name of the script in the parent directory

```
$ ls ..
README.txt  mydir  script.sh
```

Note: You can go to the parent by using .. For the parent of the parent use ../.. And this can go on until you are at the root (which is the / directory).

17. Copy the script to the current directory

```
$ ls
$ cp ../script.sh .
$ ls
script.sh
```

Note: For copying directory structures, you have to use the -r option (recursive).
The first argument (after the options which all start with -), is the file or directory to copy. The last argument is the location (for the current location, use .)

Exercise

18. Copy large files, remote files and monitor the progress (first remove the file again):

```
$ rm script.sh
$ rsync --progress -ahrL ../script.sh .
sending incremental file list
script.sh
            13 100%    0.00kB/s    0:00:00 (xfr#1,
to-chk=0/1)
$ ls
script.sh
```

Note: rsync can be used for remote file copy (like from your computer to the cluster). But rsync has to be installed at both sides.
--progress shows the progress of the copy.
-a says, copy everything.
-h is human readable (use bytes, Mb, Gb instead of bits)
-r is recursive (also copy the content of directories)
-L is copy links (a link will be changed to the actual file)

Exercise

19. Now we have the script in the proper place, let's execute it

```
$ sh script.sh
test
$ bash script.sh
test
$ ./script.sh
bash: ./script.sh: Permission denied
```

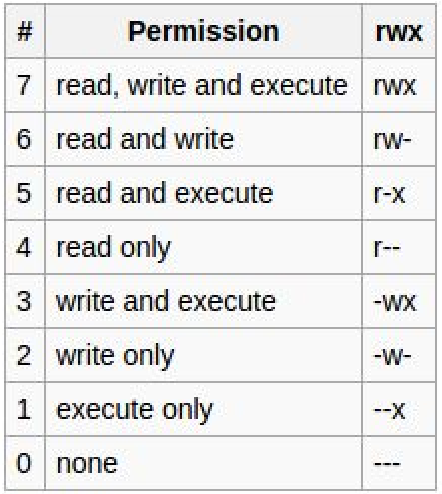
Note: sh and bash can be used to execute sh scripts.
Another way to execute scripts is by using ./script.sh
However this method requires the correct permissions (the user who executes the script needs execute permissions).

20. Changing permissions

```
$ ls -lh script.sh
-rw-r--r-- 1 23337 23337 12 Nov 16 09:51 script.sh
bash-4.3$ chmod u+x script.sh
bash-4.3$ ls -lh script.sh
-rwxr--r-- 1 23337 23337 12 Nov 16 09:51 script.sh
bash-4.3$ chmod 775 script.sh
bash-4.3$ ls -lh script.sh
-rwxrwxr-x 1 23337 23337 12 Nov 16 09:51 script.sh
```

Note: chmod is used to change all permissions. There are 2 ways to use it: with numbers (bit representation), or with letters.
The numbers version needs 3 numbers (0-7): owner, group and others. Each number tells the permissions.
The letter version needs the owner (u), group (g) or others (o), the thing that needs to happen: add (+) or remove (-), and the permission read (r), write (w) or execute (x).
-R option can be added for directories, doing the same permission operation on all files and directories in this directory.

Overview of the permissions in number format (bit format):



#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

Other command you can try:

- mv

```
$ mv script.sh ..
```

- man

```
$ man rsync
```

Remember: when you are getting stuck, and want to return to the command line, use Ctrl-c (control c) to terminate a running command or program.