Navigating the Linux file system

Oxford Biomedical Data Science Training Programme
University of Oxford
2020-04-27

Overview

- The Linux shell
  - Linux commands
  - Navigating the Linux file system
- Environment variables and shell configuration
- Working with files
  - Finding, comparing
  - Compressing, transferring, checking
- File properties & permissions
- Managing processes
- System resources

The Linux shell

- Command line interface (CLI) - allows user to interact with the computer by typing in commands
- The shell is the software that handles the command line interface - takes commands from the keyboard and gives them to the operating system to perform (command interpreter)
- Most widely used is BASH (Bourne Again Shell)
- Terminal emulator - program that lets you interact with the shell
  - Mac - Terminal, iTerm2
  - Windows - PuTTy

Linux commands

- Bash is a command interpreter
  - **A shell builtin:** command built into the shell itself e.g. `cd` (change directory)
  - **An alias:** User-defined command built from other commands
  - **An executable program:** Any external software in the system path
- Commands have options that modify their function

```
type <command>          # shows how a command is interpreted
type cd          cd is a shell builtin
type less          less is /usr/bin/less

which <command>          # displays which executable program will be executed
which less          /usr/bin/less
```

Getting help with commands

Find out about a command's function and options:

```
man <command>
```
display a command's manual page

```
info <command>
```
display a command's info entry

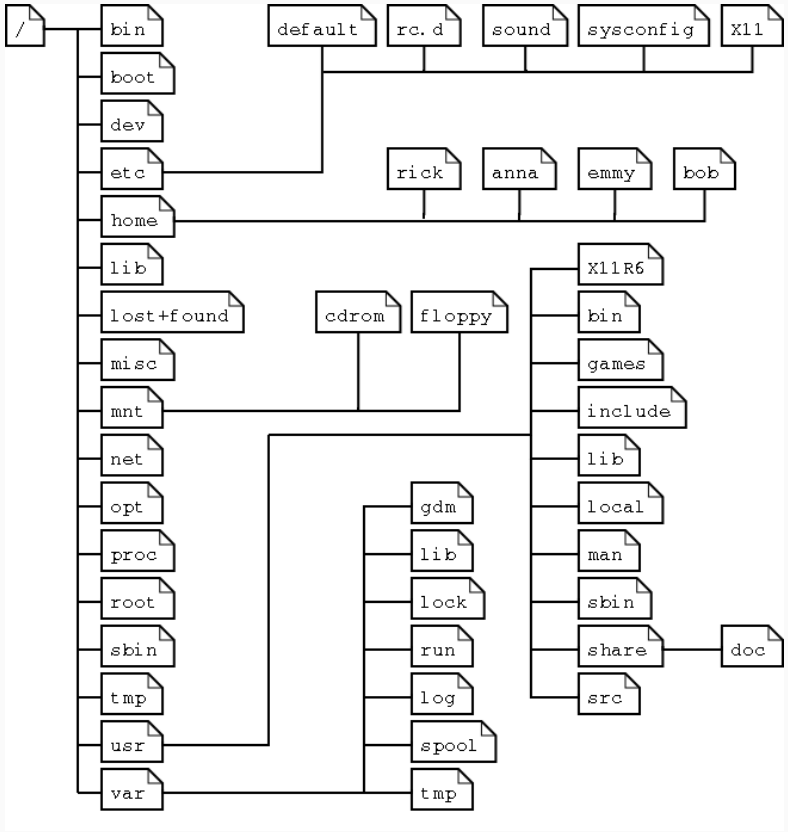
Or there is always...

Bash shortcuts

- CTRL-c # abort current command
- CTRL-z # pause current foreground process
- CTRL-l # clear the screen
- CTRL-a # go to start of line
- CTRL-e # go to end of line
- CTRL-u # cut from start of line
- CTRL-k # cut to end of line
- CTRL-r # search history
- CTRL-d # can also type `logout` or `exit`
- Up arrow # access previous commands
- Tab # autocomplete (will prompt if ambiguous)

The Linux file system

- In Linux, everything is a file (or a process)
- A directory is just a file containing names of other files
- Tree structure
  - Finder (macOS)
  - File Explorer (Windows)



Red Hat file system layout

Remote connection to a Linux server

- Secure shell - SSH
- Encrypted network protocol
- Remote command line login to a Linux system
- Uses a username and password to authenticate a remote user
- Supports X11 forwarding - graphical interface (slow)

e.g.

```
ssh -X <user>@cgatui.imm.ox.ac.uk          # CGAT system, -X turns on X11 forwarding

ssh -X <user>@deva.molbiol.ox.ac.uk        # CBRG system
```

Navigating the file structure

```
pwd          # print working directory
```

/ifs/obds-training/apr20/shared

```
ls           # list directory contents
```

week1

week2

week3

etc.

```
cd week1          # change to week1 directory
```

```
pwd
```

/ifs/obds-training/apr20/shared/week1

```
ls
```

bash, bedtools, conda etc.

Changing how you view files

Options for ls

Option	Long Option	Description
`-a`	`--all`	List all files, even those with names that begin with a period, which are normally not listed (i.e., hidden).
`-d`	`--directory`	Ordinarily, if a directory is specified, `ls` will list the contents of the directory, not the directory itself. Use this option in conjunction with the `-l` option to see details about the directory rather than its contents.
`-F`	`--classify`	This option will append an indicator character to the end of each listed name. For example, a "/" if the name is a directory.
`-h`	`--human-readable`	In long format listings, display file sizes in human readable format rather than in bytes.
`-l`		Display results in long format.
`-r`	`--reverse`	Display the results in reverse order. Normally, `ls` displays its results in ascending alphabetical order.
`-S`		Sort results by file size.
`-t`		Sort by modification time.

`man ls` # manual page for ls command

`ls -l` # long format

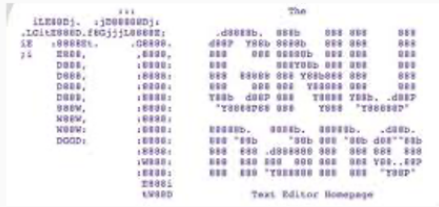
`ls -lhF` # long format, human readable, classify

Viewing text files on the command line

```
cat <filename>          # print the file contents to standard out

head -n 20 <filename>          # print the first n lines to standard out

tail -n 20 <filename>          # print the last n lines to standard out

tail -f <filename>          # follow changes to the end of the file on standard out

more <filename>          # display output in the terminal one page at a time

less <filename>          # like more but allows backward movement
```

Terminal text editors

- No graphical interface/mouse required
- Can be heavily customised
  - Language-specific syntax highlighting
  - Code autocompletion



- Vim, Emacs
  - Advantage - been around for > 25 years, very widely used, large number of add-ons
  - Disadvantage - need to learn a set of keyboard shortcuts to use
- Nano
  - Advantage - very simple and easy to use
  - Disadvantage - not as powerful as Vim/Emacs

Creating files and directories

```
touch <filename>          # create new empty file (can create several at a time)
vim <filename>            # open file/empty file in vim terminal text editor
emacs <filename>          # open file/empty file in emacs terminal text editor
nano <filename>           # open file/empty file in nano terminal text editor

mkdir <dir1>              # create new directory
mkdir <dir1> <dir2>       # create multiple new directories
```

Copying, moving and renaming files

Options for cp

Option	Meaning
`-a, --archive`	Copy the files and directories and all of their attributes, including ownerships and permissions. Normally, copies take on the default attributes of the user performing the copy.
`-i, --interactive`	Before overwriting an existing file, prompt the user for confirmation. **If this option is not specified, `cp` will silently overwrite files.**
`-r, --recursive`	Recursively copy directories and their contents. This option (or the `-a` option) is required when copying directories.
`-u, --update`	When copying files from one directory to another, only copy files that either don't exist, or are newer than the existing corresponding files, in the destination directory.
`-v, --verbose`	Display informative messages as the copy is performed.

```
cp file1 file2      # copy file to new file

cp file1 dir1/      # copy file to specified directory

mv file1 file2      # rename file

mv file1 dir1/      # move file to specified directory
```

Deleting files and directories

Options for rm

Option	Meaning
`-i, --interactive`	Before deleting an existing file, prompt the user for confirmation. **If this option is not specified, `rm` will silently delete files.**
`-r, --recursive`	Recursively delete directories. This means that if a directory being deleted has subdirectories, delete them too. To delete a directory, this option must be specified.
`-f, --force`	Ignore nonexistent files and do not prompt. This overrides the `--interactive` option.
`-v, --verbose`	Display informative messages as the deletion is performed.

```
rm file1      # rm file1

rm *.perl     # remove all files that end in .perl

rm -r dir1    # remove directory and contents

rmdir dir1    # remove empty directory
```

Hard and symbolic links

- A hard link is an additional name for an existing file
- A symbolic link is a file that contains a text pointer to a target file or directory
- Symbolic links overcome the two disadvantages of hard links:
  - Hard links cannot span physical devices (disks)
  - Hard links cannot reference directories but only files
- Symbolic links become unusable if target file is moved or deleted

```
ln file1 link1          # create hard link
```

```
ln -s file1 link1       # create symbolic link
```

Connecting to cgath1 via ssh

1. ssh username@cgatui.imm.ox.ac.uk
    - e.g. ssh lucy@cgatui.imm.ox.ac.uk
2. Type yes to authenticate host
3. Type password
4. ssh cgath1 or ssh cgath2 - work should be performed on one of these two hosts
5. Type password again

Setting up passwordless connection to cgath1 or cgath2

1. In a new local terminal, move to your home directory and then to the .ssh directory

   ```
   cd ~
   ```

   ```
   cd .ssh
   ```

2. Create an SSH key pair

   ```
   ssh-keygen -t rsa
   ```

3. Choose file name (id_rsa is the default) and enter a passphrase (or can leave blank). You should now have two files e.g. id_rsa and id_rsa.pub.

Setting up passwordless connection to cgath1 or cgath2

4. Add your SSH key to the ssh-agent

- `eval $(ssh-agent -s)"` # start the ssh-agent in the background
- If you are using macOS Sierra 10.12.2 or later, modify your ~/.ssh/config file to automatically load keys into the ssh-agent and store passphrases in your keychain.

```
Host *
        AddKeysToAgent yes
        UseKeychain yes
        IdentityFile ~/.ssh/id_rsa
```

- `ssh-add -K ~/.ssh/id_rsa` # add your SSH private key to the ssh-agent and store your passphrase in the keychain

Setting up passwordless connection to cgath1 or cgath2

5. Copy text from id_rsa.pub to ~/.ssh/authorized_keys file on the server - if there is no authorized_keys file, make one

6. Add the following to the .ssh/config file on your local computer - replace lucy with your cgat username

```
nano config          # edit config file in nano
```

```
Host    h1
 hostname     cgath1
 ForwardAgent yes
 ForwardX11   yes
 User         lucy
 ServerAliveInterval 300
 ServerAliveCountMax 2
 ProxyCommand ssh lucy@cgatui.imm.ox.ac.uk nc %h %p
 IdentityFile ~/.ssh/id_rsa
```

Setting up passwordless connection to cgath1 or cgath2

7. You should now be able to type ssh h1 and enter directly into cgath1

```
ssh h1
```

Exercise 1 - commands and files

1. Open a terminal emulator and connect to cgath1 via ssh.
2. Change to your personal /ifs/obds-training/apr20/username directory
3. Create a new directory called obds
4. Copy all the files and directories from /ifs/obds-training/apr20/shared/week1 to obds (hint: use -r option)
5. View the file week1/bash/coding_gene_region.bed on the command line - try cat, more, less, head and tail
6. Make a new directory called linux in your home directory
7. Make a copy of coding_gene_region.bed in your linux directory
8. Open the copy of coding_gene_region.bed in a text editor and remove the first 10 lines
9. Rename the file to edit.bed
10. Make a new empty file called transcripts.bed
11. Create a link to edit.bed and a symbolic link to transcripts.bed
12. Remove both links, the files transcripts.bed and edit.bed, then remove the linux directory

Shell configuration

- .bashrc is a shell script that is run every time a user opens a new shell
- Can be called .profile or .bash_profile
- Located in user's home directory - hidden file so use `ls -a` to list
- Set environment variables
- Set up virtual environments e.g. load Conda environment
- Set aliases - shortcuts

```
#!/usr/bin/env bash

# Source global definitions
if [ -f /etc/bashrc ]; then
 . /etc/bashrc
fi

# Set umask
umask 002

# Use Oracle Java
export JAVA_HOME=/usr/java/latest

if [[ $PS1 ]]; then
  # redefine the module function to work properly on
the cluster
  unset -f module
  module()  { eval `/usr/bin/modulecmd bash $*`; }

fi # if PS1
```

Environment variables

- Variables in the shell
- Can be accessed by any program run in the shell
- Can be user defined and modified
- Accessed using the $ notation
- PATH is a key environment variable - list of places to look for executable programs

```
echo $PATH      # see what is currently in your path
```

Aliases

- Shortcut for command
- Reduce keystrokes and improve efficiency
- Defined in .bashrc

```
alias rm="rm -i"
```
prompt before every removal

```
alias obds="cd /ifs/obds-training && pwd && ls -lh"
```
change to obds-training directory, print the working directory and list the contents

The .inputrc file

- Customise how keystrokes are interpreted in the terminal emulator
- Found in your home directory (hidden file)
- Enable reverse history search with up arrow
  - `"\eOA": history-search-backward`
  - `"\e[A": history-search-backward`
  - `"\eOB": history-search-forward`
  - `"\e[B": history-search-forward`
- Enable moving cursor word by word - Ctrl + right/left arrow
  - `"\e[1;5D": backward-word`
  - `"\eOd": backward-word`
  - `"\e[1;5C": forward-word`
  - `"\eOc": forward-word`

Exercise 2 - shell configuration

1. Look at the contents of the $PATH variable - where is the system looking for executable files?
2. Find the .bashrc and .inputrc files in your copy of the week1/bash folder
3. Move the .bashrc and .inputrc files to your home directory
    - N.B. If you have existing .bashrc or .inputrc files, back up first
4. View the contents of the .inputrc file in the terminal
5. Open .bashrc in a terminal text editor and add the `alias ll="ls -lhF"`
6. Add another alias to provide a shortcut to your personal obds-training directory
7. Create a symbolic link to the .bashrc file in your home directory called .profile
8. Create a symbolic link to the .bashrc file called .bash_profile

Working with files

- Counting lines, words and characters within files
- Searching for files
- Comparing files
- File compression
- Checking file integrity - checksums
- Transferring files across systems

Counting characters, lines or words

Options for wc

-c, --bytes	print the byte counts.
-m, --chars	print the character counts.
-l, --lines	print the newline counts.
-L, --max-line-length	print the length of the longest line.
-w, --words	print the word counts.
--help	display a help message, and exit.
--version	output version information, and exit.

`wc -m <filename>` # count characters in file

`wc -l <filename>` # count lines in file

`wc -w <filename>` # count words in file

Searching for files

Find command - very powerful e.g. search for recently modified files, recently accessed files, empty files etc.

```
find ~ -type f -name "*.JPG"
```

where to search (~ is home directory, . is current directory, / is root directory)

what to search for (files with .JPG extension)

```
find . -name "*.tsv" -exec wc -l {} \;
```

where to search

what to search for (files with .tsv extension)

execute command on each file found, brackets signify where to put current file name in command

Comparing files

- diff is a tool to compare documents
- Supports many output formats (see -c and -u options)
- Often used by programmers to examine changes between versions of source code

```
diff file1.txt file2.txt          # find differences between file1 and file2
```

File compression

Good practice to compress all non-trivial text files to save hard disk space

- `gzip <file1>` # compress file1 in place
- `gunzip <file1>` # decompress file1 in place
- `gunzip -c <file1>` # decompress file1 to stdout
- `zcat <file1>` # print compressed file to stdout
- `zless <file1>` # less compressed file
- `tar cf file.tar files` # create an archive from multiple files
- `tar xf file.tar` # extract files from an archive
- `bzip2 <file1>` # stronger compression, less widely used

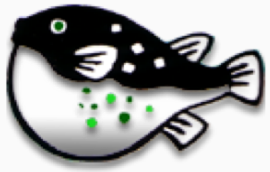
File transfers

- Based on SSH:
  - Secure File Transfer Protocol (SFTP)
  - Secure Copy Protocol (SCP)
  - Rsync - file synchronisation
- Curl - library for transferring data with URLs
- wget:
  - Supports HTTP/HTTPS/FTP/SFTP
  - Downloading files from the internet
  - Used to mirror websites

FTP/SFTP GUIs

FileZilla

Cyberduck



Fugu

Checksums

- Quickly compare files to see if they are identical
- md5sum command
- Prints a 32-character (128-bit) checksum ("hash") of a given file
- Hash is based on the file contents - will change if file is changed

```
md5sum file1.txt file2.txt > hashes.txt
```
calculate the hash values for file1.txt and file2.txt and store in the hashes.txt file

```
bbbf0c67c08ce89e6b70e334b13b4f49  file1.txt
7dda572cd5dd8e39e4a251f77046cb51  file2.txt
```

```
md5sum --check hashes.txt
```
check hashes of file1.txt and file2.txt after transferring to a different server

```
file1.txt: OK
file2.txt: OK
```

Exercise 3 - working with files

1. Find all the files containing "fMuscle" in the week1/bedtools folder
2. Count the number of lines in each of these files using find command
3. Copy week1/bedtools/gwas.bed to your home directory
4. Edit the file to remove some lines
5. Compare the original and modified files (diff)
6. Count the number of lines and characters in the original and modified files
7. Compress the modified file in place
8. View the compressed file
9. Generate a checksum for your compressed file
10. Download the compressed file to your laptop
11. Check the file downloaded correctly

File properties and permissions

```
[davids@cgath2 shared]$ ls -l
total 43286
-rwxrwxr-- 1 davids     obds 38578443 Jan 21  2019 nomachine_5.3.12_10.dmg
drwxrwxr-x 6 davids     obds       90 Sep 15 21:17 week1
drwxrwxr-- 2 davids     obds       41 Sep 24 11:03 week2
drwxrwxr-- 4 davids     obds       50 Nov 14 17:18 week3
drwxrwxr-x 7 davids     obds      161 Oct 10 13:28 week4
drwxrwxr-- 5 davids     obds      157 Oct 15 12:51 week5
drwxr-xr-x 2 davids     obds        0 Sep 14 20:25 week6
```

file permissions | # hard links | owner | group | size (bytes) | Modification date | file name

File permissions - owner

- Linux is a multi-user operating system
- Access to read, write and execute each file/directory is controlled
- Each file is owned by exactly one user

File permissions - group

- A group is a collection of one or more users
- Each user can be a member of multiple groups
- A file can be owned by exactly one group
- To see the groups that your user currently belongs to type: `groups`

File permissions - other

- The last category that you can assign permissions for is the "other" category
- "other" - any user that is not the file owner and is not a member of the group that owns the file
- This allows you to set permissions that will apply to anyone outside of the two control groups

Types of permissions

- Each permissions category (owner, group, other) can be assigned permissions that restrict their ability to read, write or execute a file
- For a regular file:
  - read permissions are required to read the contents of a file
  - write permissions are necessary to modify it
  - execute permissions are needed to run the file as a script or an application
- For directories:
  - read permissions are necessary to list the contents of a directory
  - write permissions are required to modify the contents of a directory
  - execute permissions allow a user to change directories into a directory

Alphabetic notation

- Linux represents these types of permissions using two separate symbolic notations:
  - Alphabetic
  - Octal

Alphabetic:

```
ls -l          # list the contents of a directory in long format
```

Octal notation

- File permissions can be displayed in octal format

```
stat -c "%a: %n" *          # %a - access rights in octal, %n - file name
```

755 jupyter_notebooks
644 mm10.blacklist.bed.gz

- Each permissions category (owner, group, other) is represented by a number between 0 and 7
- Each type of permission is assigned a numerical value:
  - 4 = read permission
  - 2 = write permission
  - 1 = execute permission
- We sum the numbers for each category to give a single number

Octal notation

+ 4 = read permission
+ 2 = write permission
+ 1 = execute permission

```
421401401 = 755
rwxr-xr-x
```

```
420400000 = 640
rw-r-----
```

Changing file permissions

- We can change file permissions with the `chmod` command using alphabetic notation
- u = user, g = group, o = other/r = read, w = write, x = execute/- = deny access, + = give access

-rw-rw-r-- 1 lucy obds 19 Apr 18 10:22 file1.txt

```
chmod go-rwx file1.txt
```

```
ls -l
```

-rw------- 1 lucy obds 19 Apr 18 10:22 file1.txt

Changing file permissions

- We can also change file permissions using the octal notation (4 = read, 2 = write, 1 = execute)

```
chmod 751 file1.txt
```

-rwxr-x--x 1 lucy obds 19 Apr 18 10:22 file1.txt

- We can also change the owner and group

```
chown <user> file1.txt          # however only root user has permission to do this
```

```
chgrp <newgroup> file1.txt
```

-rwxr-x--x 1 lucy usersfgu 19 Apr 18 10:22 file1.txt

Exercise 4 - file permissions

1. List the contents of your home folder in long form
2. Change the permissions of your .bashrc file so that only you and your group can read, write and execute the file
    - Use alphabetic notation
3. Change the permission of your .inputrc file so that only you and your group can read, write and execute the file
    - Use octal notation

Linux processes

- Any Linux command creates a process
- Every process has a process ID (PID)
- Processes can run in the foreground or background
- Processes can be stopped (killed)

```
ctrl-z          # pause process in foreground
bg              # run a paused process in the background
command &       # & at the end of a command starts it in the background
fg              # bring a background process to the foreground
ps              # give the status of processes running for a user
ps <PID>        # give the status of a particular process
kill <PID>      # kill a particular process
killall <name>  # kill all processes for a particular user
```

Exiting the terminal

- `logout` or `exit`
- Sends hangup signal (HUP) to all dependent processes
- Nohup
  - Ignore the hangup signal
  - Send output to nohup.out instead of the terminal
  - Avoids terminating job when a remote session drops

```
nohup command          # run a command immune to hangups
```

Exercise 5 - Linux processes

1. Start a new process using the command `sleep 3000` # delay for 3000 seconds
2. Pause the sleep process and send it to the background
3. Check the status of running processes
4. Start a new sleep process in the background
5. Kill the first process
6. Bring the second process to the foreground and terminate it

Managing system resources

`top` # details all active processes

Upper section (summary area) - statistics on processes and resource usage

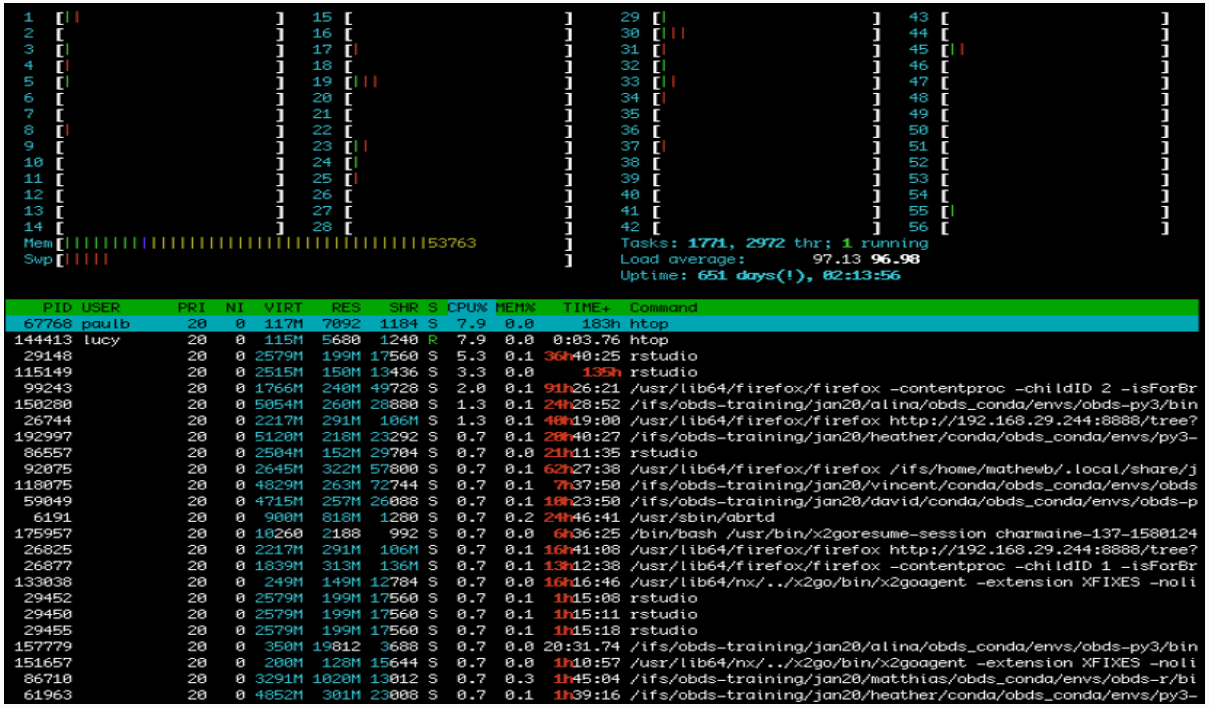
Lower section - list of currently running processes

```
Tasks: 287 total,   1 running, 286 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.2%us,  0.1%sy,  0.0%ni, 99.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16418440k total,  4039976k used, 12378464k free,   215652k buffers
Swap:  1048572k total,        0k used,  1048572k free,  2036108k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3451 root      20   0  344m 188m  32m S  1.7  1.2  84:21.12 Xorg
 3945 davids    20   0  214m  15m 9688 S  0.7  0.1   0:13.53 gnome-terminal
20164 davids    20   0 19304 1440  976 R  0.3  0.0   0:00.35 top
29511 davids    20   0 2404m 307m  73m S  0.3  1.9  21:29.15 firefox
    1 root      20   0 23488 1580 1260 S  0.0  0.0   0:01.56 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.18 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:31.01 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   1:36.70 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
```

Process id	User	Priority	Memory usage (Virtual Reserved Shared)	Status (s = sleeping r = running)	Run time	Name of command

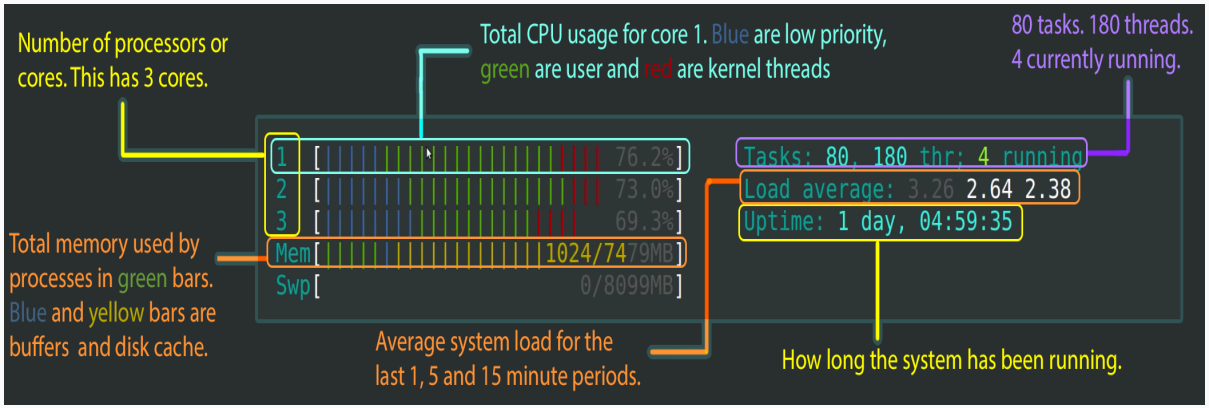
Managing system resources

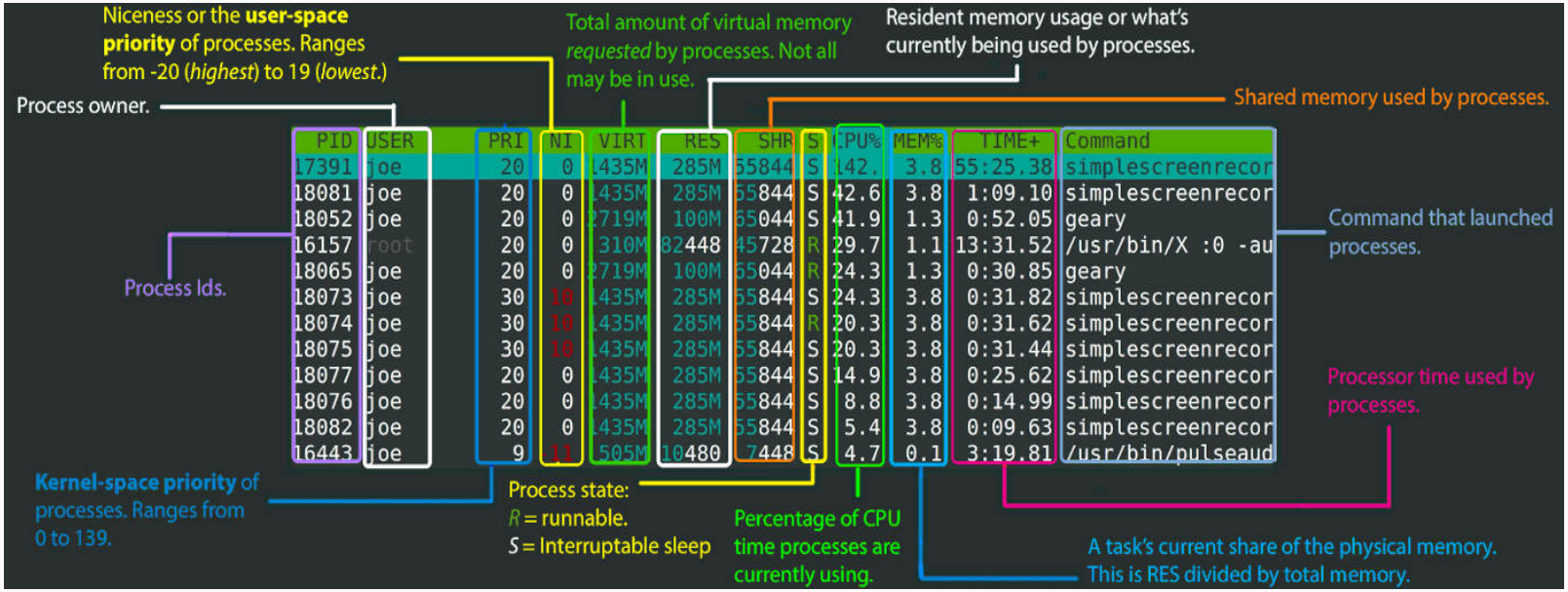
```
htop        # available on CGAT server but not installed by default on CBRG server
```



Press `u`, select or type your name, and press Enter to see only your processes

Managing system resources





Memory and disk monitoring

```
df          # free hard disk space on your system

du          # show directory space usage

free          # gives free RAM on your system
```

Exercise 6 - system resources

1. Look at the man page for top to see how you can change the display
2. Look for all the processes for your username
3. Sort the output by memory usage
4. Check the total processor load on the server
5. Check the total disk usage in your home folder
6. Check the total free RAM on the server