



# H3ABioNet

Pan African Bioinformatics Network for H3Africa

## 16SrRNA Intermediate Bioinformatics Online Course

# Introduction to the command line and R

## Introduction to the command line



**H3ABioNet**

Pan African Bioinformatics Network for H3Africa



16SrRNA Intermediate Bioinformatics Online Course:

Int\_BT\_2019

Trainer name

# Learning Objectives

- To give a background on Unix and the fundamentals of the OS
- To introduce the shell
- To give a breakdown of command usage and some of the most important commands
- To give an intro to command line editors
- To give an intro to a compute cluster and resources manager
- To give an intro to workflow tools
- To give an intro to software containerisation

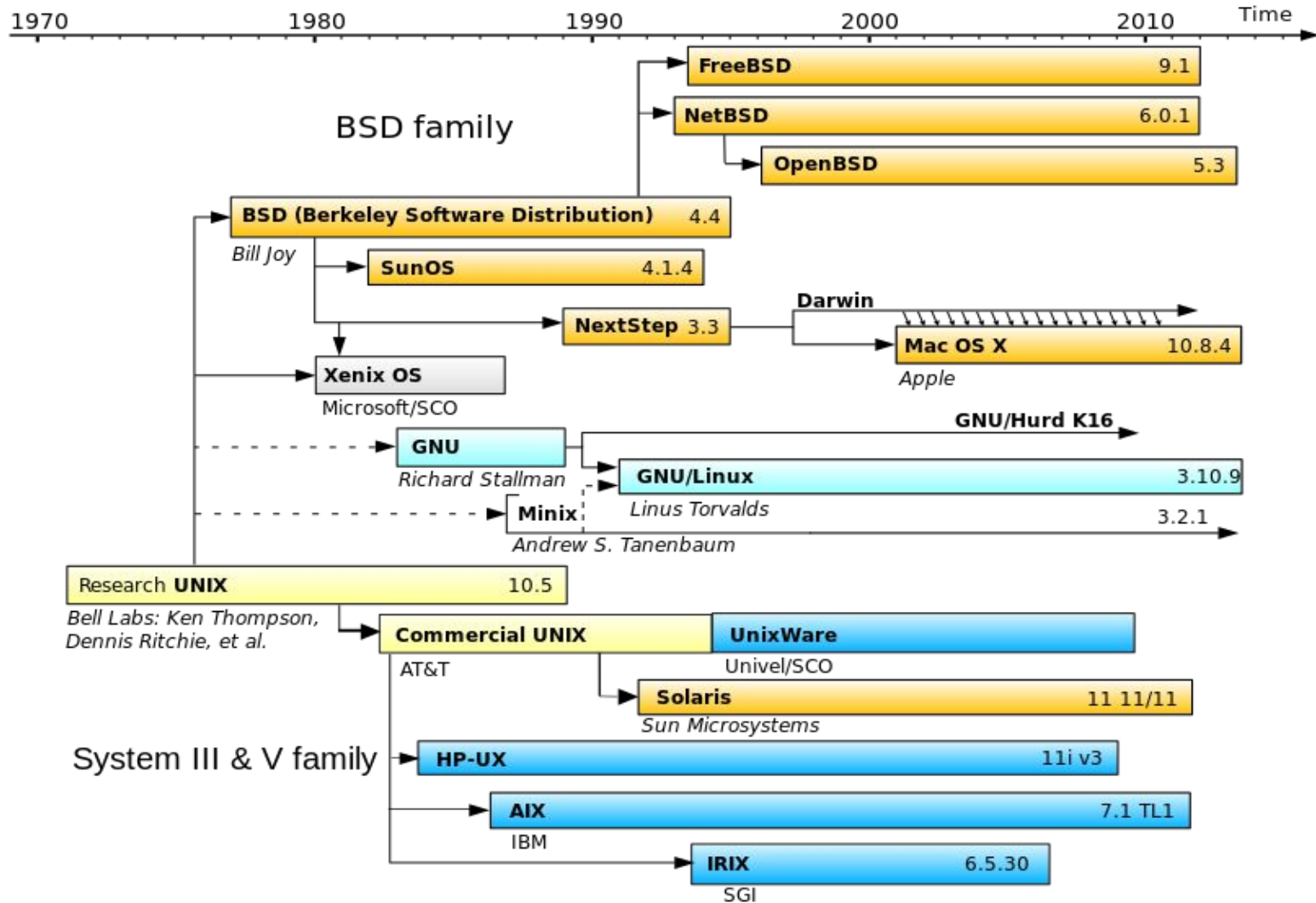
# Learning Outcomes

- To understand the background of the Unix environment
- To understand the need for using the command line for scientific computing
- To be comfortable with navigating the environment, doing file and directory manipulations, and working with pipes and filters
- To understand a cluster environment basics and be able to submit an interactive job
- To be familiar with workflow tools
- To be familiar with software containerisation

# What is Unix

- A stable, multi-user, multitasking operating system for servers, desktops and laptops that exists in many variants
- Unix systems are characterised by a modular design:
  - a set of simple tools that each perform a limited, well-defined function
  - with a unified filesystem as the main means of communication and
  - a shell scripting and command language to combine the tools to perform complex workflows.
- Unix flavours
  - Sun Solaris, Mac OS X, GNU/Linux, UnixWare, FreeBSD, OpenBSD, IBM IAX, HP UX
- GNU/Linux distributions
  - Difference is in package managers, directory structure, file naming, suitability (servers vs desktops)
  - Debian, Ubuntu, Fedora, openSUSE, SUSE Linux Enterprise, Scientific Linux, Redhat, CentOS

# Timeline



From: [http://commons.wikimedia.org/wiki/File:Unix\\_timeline.en.svg](http://commons.wikimedia.org/wiki/File:Unix_timeline.en.svg) , license: released in Public domain

# Fundamentals

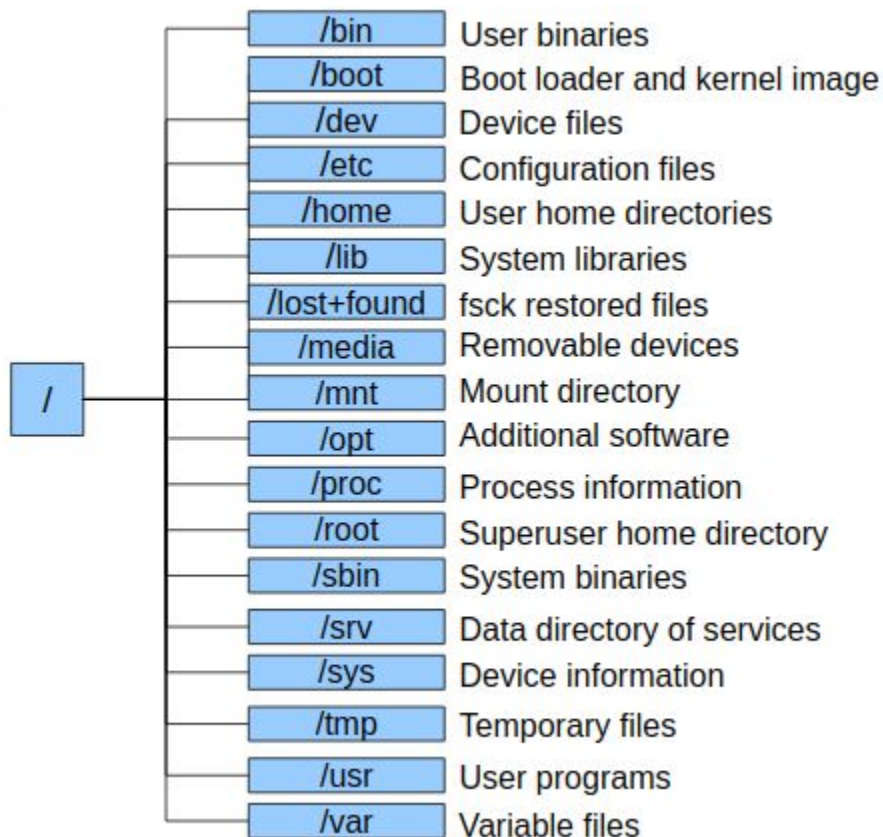
- Different Unix flavours but fundamentally the same
  - Kernel
    - Allocates time and memory to programs, handles storage and communication
  - Shell
    - Interface between user and kernel. Command line interpreter (CLI)
  - Terminal / Console
    - Interface to the shell
- Comply to POSIX standards
- Username, password, home directory, group, permissions, default shell
- Processes (PID)
- Directory structure and files

# Ubuntu files structure

```

trainee@mgwvm: ~
trainee@mgwvm:~$ ls -ld /*/ | sort
/bin/
/boot/
/cdrom/
/dev/
/etc/
/home/
/lib/
/lib32/
/lib64/
/lost+found/
/media/
/mnt/
/opt/
/proc/
/root/
/run/
/sbin/
/selinux/
/srv/
/sys/
/tmp/
/usr/
/var/

```



# Permissions

- In multi-user systems, access and access restrictions are key
- Typically, you are the owner of every file/directory you create or bring into a system
- What other files and directories you can read, write or execute will depend on how the system is set up

```
[gerrit@headnode001 filter-vcf] $ ls -la /global5/datasets/
total 204
drwxr-xr-x 15 ayton  root    245 Mar  7 16:48 .
drwxr-xr-x  7 root   root   4096 Nov 17  2017 ..
drwxr-x---  3 ayton ayton   25 Feb 18  2016 afmap
drwxr-x---  4 gerrit agvp    42 Jul  6  2015 agvp
drwxr-s---  4 ayton  root   20480 Aug 23  2016 ashk
drwxrwx--- 10 gerrit h3a    4096 Feb  8 14:34 baylor
drwxr-x---  3 root   root    27 Jan 14  2016 gdap
drwxr-xr-x  4 gerrit gerrit  70 Oct 19  2018 giab
drwxr-xr-x  3 ayton  root    52 Mar  7 16:51 hapmap3
drwxr-xr-x  8 root   root   4096 Feb 26  2016 kgp
drwxr-s---  6 root   root    88 Nov 10  2015 other_chipdesign
drwxr-x--- 27 gerrit saals   4096 Oct 23  2018 saals
drwxr-s---  6 gerrit sahgp   171 Feb  8 13:33 sahgp
drwxrws---  2 ayton  root  126976 Oct  6  2015 simons
drwxr-xr-x  6 root   root    101 Feb  8 14:05 trypanogen
```

↑  
permissions

↑  
user / group  
ownership

↑  
size

↑  
time of last  
change

↑  
file name



# Shell

- Command line interface to the operating system (browsing file system and issuing commands)
- All in all a programming language
- Shell scripts provide a way to automate repetitive tasks
- Different shells with different features

# Command anatomy

**command** -options/flags      **[input]**      **[output]**

- Depending on the command the **input** can be read from the *stdin* stream, the output from a previous command or from a file
- Depending on command the **output** can be send to the *stdout* stream, to the **input** of the next command or to a file
- Error messages are normally send to the *stderr* stream
- Do get more detail information about command flags and usage make use of the man pages. e.g. `man ls`

# List of important commands

File management	
ls	List files
cd	Change directory
pwd	Print working directory
mkdir	Make a new, empty directory
rmdir	Remove an empty directory
cp	Copy files
scp / ssh	Do a secure copy from or to a remote machine / login to remote machine
rm	Remove files
mv	Move files or directories (rename)
find	Find files or folders based on name, date, size, ownership or other parameters
tar	Build an archive of files
gzip / bzip2	Compression tools
Permissions	
chown	Change ownership of files/directories
chgrp	Change group ownership of files/directories
chmod	Change permissions (mode) of files/directories
groups	Report the group(s) you belong to
id	Report your username, userid, group(s) and group id(s)
newgrp	Set you default group for the current shell

# List of important commands

Resource monitoring	
top	See the top resource-hungry processes
df	See how much disk space is free
du	Report numbers and sizes of files on disk
free	Show available and cached memory
Job control	
which	Display full path of command
ps	View active processes
<cntrl>-c	Kill current job
<cntrl>-z	Suspend current job
bg	Put suspended job in background
fg	Bring a suspended job into the foreground
Filtering/Reporting	
grep	Search for substrings in a file or pipeline
awk / sed	Pattern scanning and text processing / stream editor for transforming text
sort	Sort lines alphabetically or numerically
wc	Count lines, words, characters
cat	Print file or files to standard output
more / less	Text viewing programs
head / tail	View start or end of a file or pipeline

# Redirection and piping

Piping and redirection summary	
> file	Output redirection, overwrite
>> file	Output redirection, append
< file	Input redirection
commandA   commandB	Pipe output from commandA to commandB

# Things to be aware of

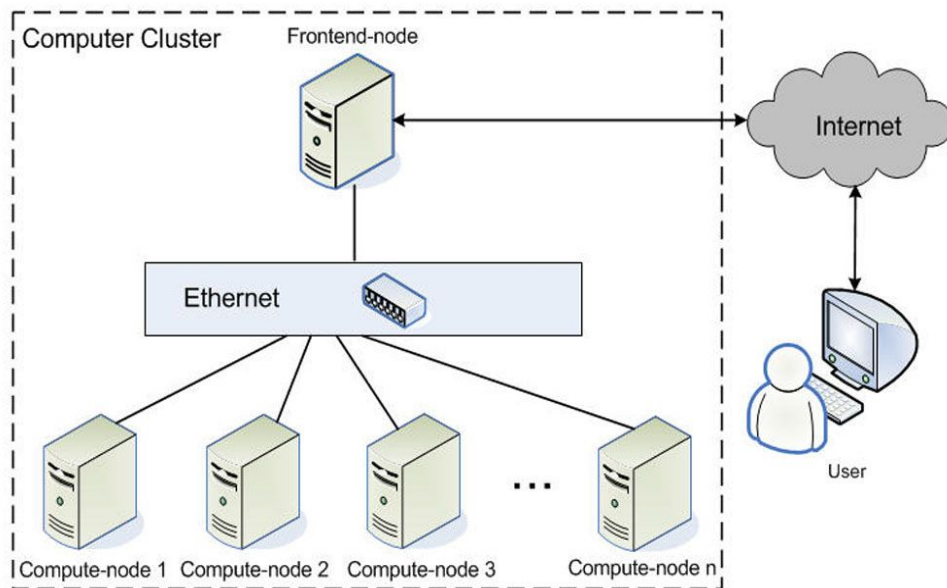
- File and directory naming is case sensitive.
- Keep away from using strange characters in file names otherwise it needs to be escaped.
- Bash shell features
  - Command and filename completion with **tab** key
  - Browse through history with up and **down** keys
  - Search for previous commands with **cntrl-r**
- Many ways to accomplish the same thing using various combinations of piping together commands.
- In the shell deleting files is forever.

# Editors

- Learn to use a shell editor and the shortcut keys
- Command line options for editors: `vi`, `emacs`, `nano`, `pico`
- `nano` is a easy editor to start with

# Compute cluster

- A compute cluster consists of single machines that work together so that in many respects they can be viewed as a single system.
- A resource and scheduler monitors job resources and assigns jobs to nodes.
- A job scheduler operates efficiently if resources are fairly distributed across users of the system.
- A job scheduler uses certain criteria to decide if jobs should run e.g.
  - Job priority
  - Compute resource availability (memory/cpu)
  - Execution time allocated to user
  - Number of simultaneous jobs allowed for a user
  - Estimated execution time
  - Elapsed execution time
  - Job dependency





- PBS and SLURM are popular job scheduling systems
- Both have different ways in defining submissions scripts. See <https://github.com/grbot/batch-system-comp>

## PBS

```

1  #!/bin/bash
2  #PBS -l nodes=1:ppn=1
3  #PBS -l walltime=01:00:00
4  #PBS -N ech_count
5  #PBS -d .
6  #PBS -M gerrit.botha@uct.ac.za
7  #PBS -m abe
8  #PBS -q UCTlong
9
10 ##### Running commands
11 echo "Date is ";date
12 echo "hostname is ";hostname
13 exe="echo_count.sh count.txt 100"
14 $exe

```

## SLURM

```

1  #!/bin/bash
2  #SBATCH -N 1
3  #SBATCH --ntasks-per-node=1
4  #SBATCH -t 01:00:00
5  #SBATCH -J echo_count
6  #SBATCH -o echo_count.o%j
7  #SBATCH -e echo_count.e%j
8  #SBATCH --mail-user=gerrit.botha@uct.ac.za
9  #SBATCH --mail-type=begin
10 #SBATCH --mail-type=end
11 ##SBATCH -p Main
12
13 ##### Running commands
14 echo "Date is ";date
15 echo "hostname is ";hostname
16 exe="./echo_count.sh count.txt 100"
17 $exe

```

# PBS and SLURM interactive sessions

- PBS doing an interactive submission (getting onto a compute node, requesting 1 node and 1 core and reserve for 2 hours)

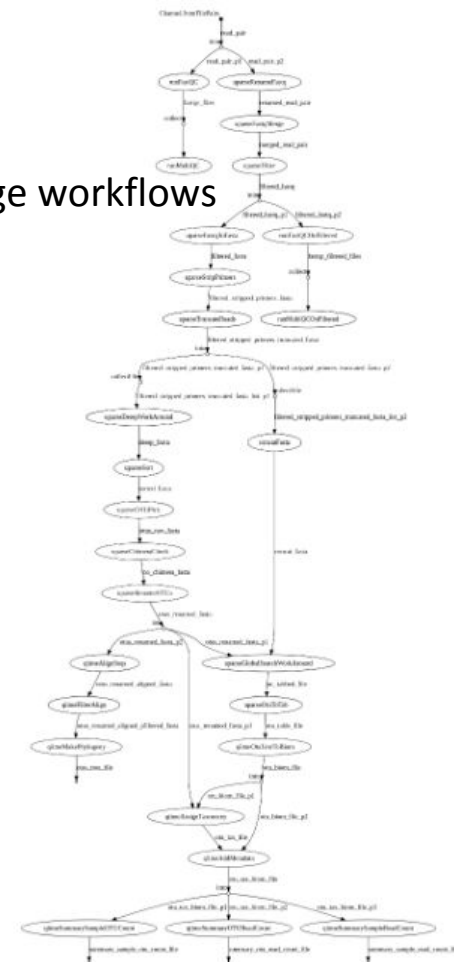
```
- $ qsub -I -l nodes=1:ppn=1 -l walltime=02:00:00
```

- SLURM doing an interactive submission (getting onto a compute node, requesting 1 node and 1 core and reserve for 2 hours)

```
- $ srun --nodes=1 --ntasks=1 --time=120 --pty bash
```

# Workflow tools

- [Nextflow](#)
  - Provides a language and engine
  - Language are based on Groovy
  - Relative easy to learn and can easily be applied to simple and large workflows
  - Not modular at moment but will be in DSL 2.0
  - Nextflow example code on Ilifu are [here](#) with demo [here](#)
- [CWL](#) (Common Workflow Language)
  - Language only, tools such as cwltool an Toil provides the engine
  - Modular and very structured
  - Large community support similar to what is found for Nextflow.
- [WDL](#) (Workflow Descriptive Language)
  - Language only, Cromwell provides the engine
  - Used by Broad Institute and supported in GA4GH workflows.



# Software containers

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
- Use operating-system-level virtualization to deliver software in packages called containers
- Most popular technologies are Docker and Singularity
- Depending on what is available at you facility it is sometimes a less administrative hassle to package your software or software suite in a container.
- Before building a container have a look at <https://quay.io/organization/biocontainers> to see if an image of the tool does not exist already.
- Singularity images can be build from Docker images
- Due to Docker security issues it is most likely that only Singularity is supported on your cluster environment.