H3ABioNet

Pan African Bioinformatics Network for H3Africa

16SrRNA Intermediate Bioinformatics Online Course

Introduction to the command line and R

Introduction to the command line

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

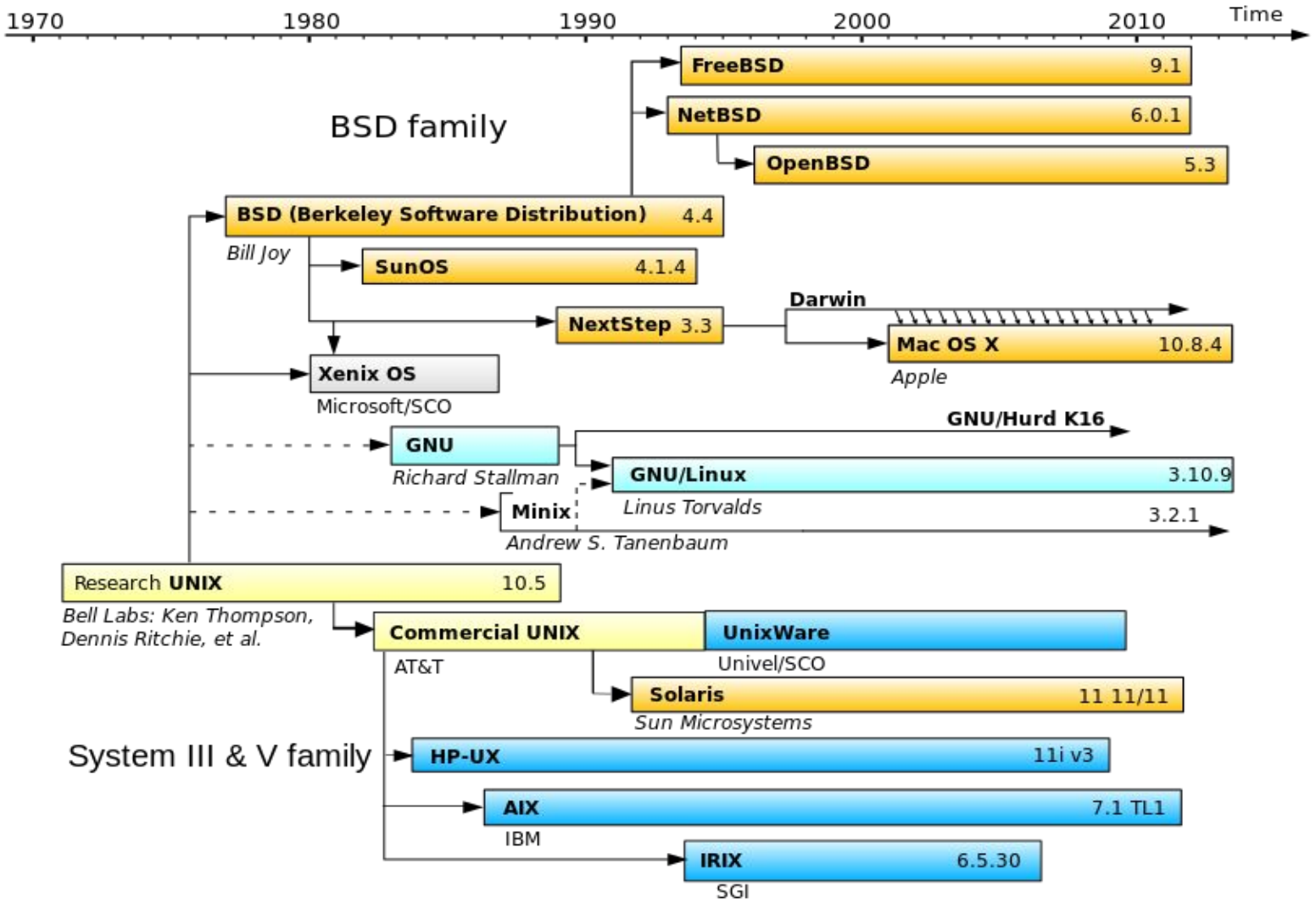
- A stable, multi-user, multitasking operating system for servers, desktops and laptops that exists in many variants

- 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



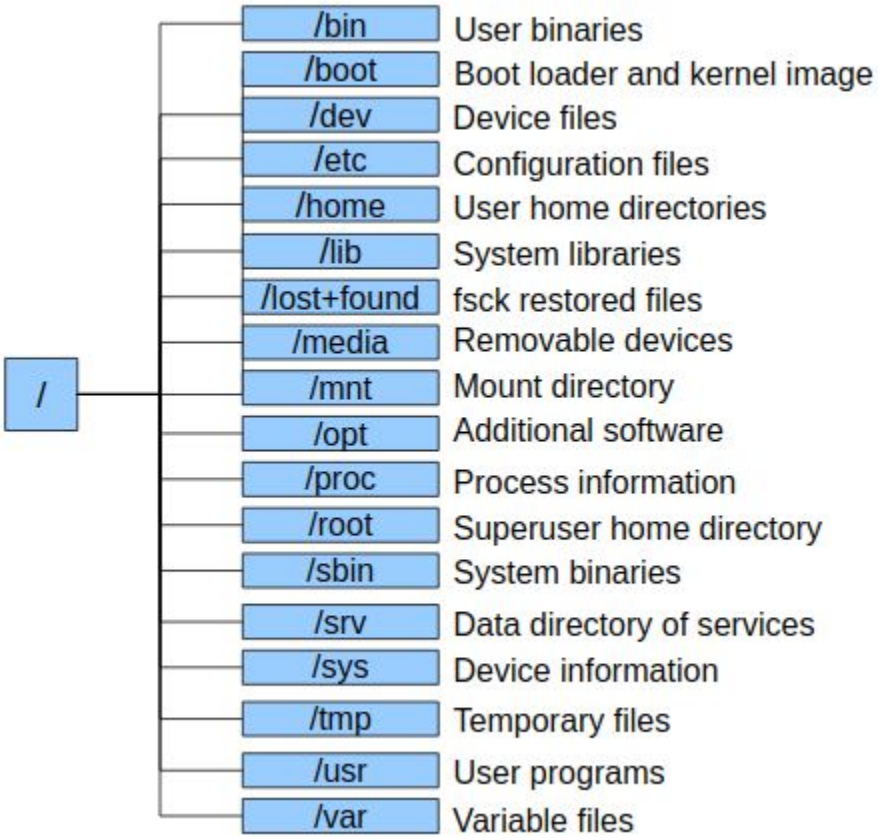
From: From: http://commons.wikimedia.org/wiki/File:Unix_timeline.en.svg , license: released in Public domain

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

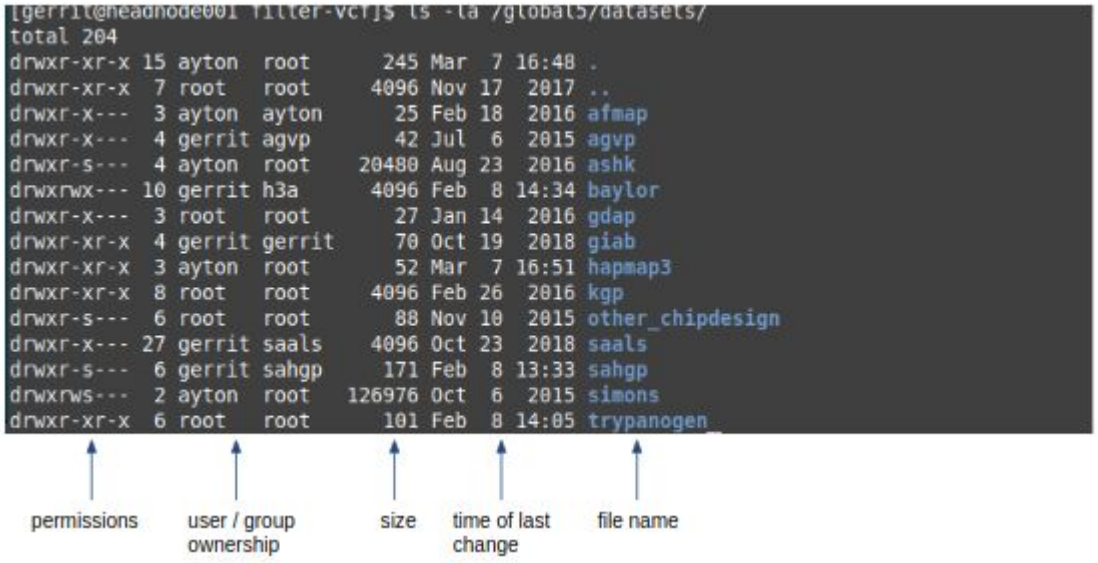
Ubuntu files structure

```
trainee@mgwvm: ~
trainee@mgwvm:~$ ls -1d /*/ | 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



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

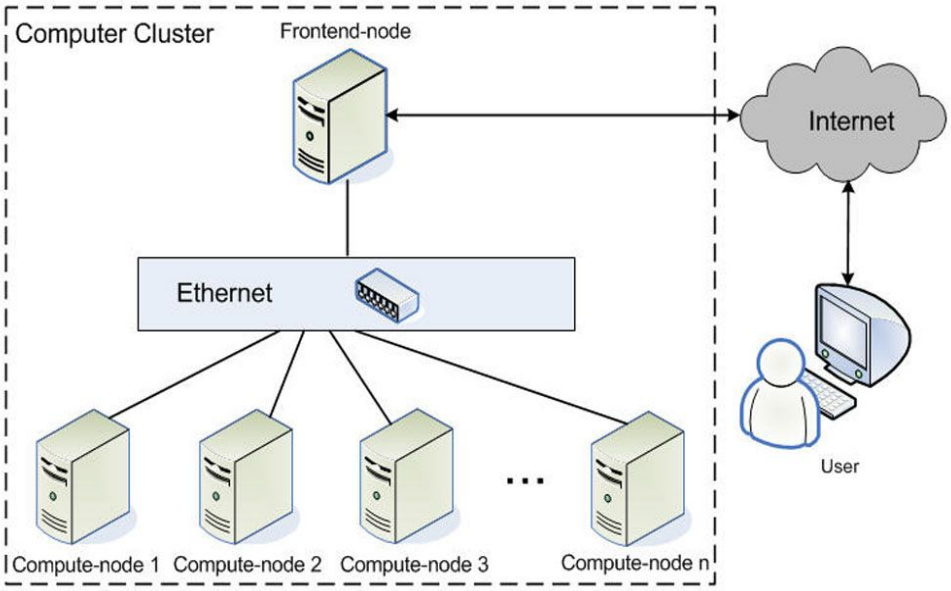
- File and directory naming is case sensitive.
- Keep away from using strange characters in file names otherwise it needs to be escaped.
- Bash shell features
  - Command and filename completion with **tab** key
  - Browse through history with up and **down** keys
  - Search for previous commands with **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 consist 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

- 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

```
#!/bin/bash
#PBS -l nodes=1:ppn=1
#PBS -l walltime=01:00:00
#PBS -N ech_count
#PBS -d .
#PBS -M gerrit.botha@uct.ac.za
#PBS -m abe
#PBS -q UCTlong

##### Running commands
echo "Date is ";date
echo "hostname is ";hostname
exe="echo_count.sh count.txt 100"
$exe
```

SLURM

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -t 01:00:00
#SBATCH -J echo_count
#SBATCH -o echo_count.o%j
#SBATCH -e echo_count.e%j
#SBATCH --mail-user=gerrit.botha@uct.ac.za
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
##SBATCH -p Main

##### Running commands
echo "Date is ";date
echo "hostname is ";hostname
exe="./echo_count.sh count.txt 100"
$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)

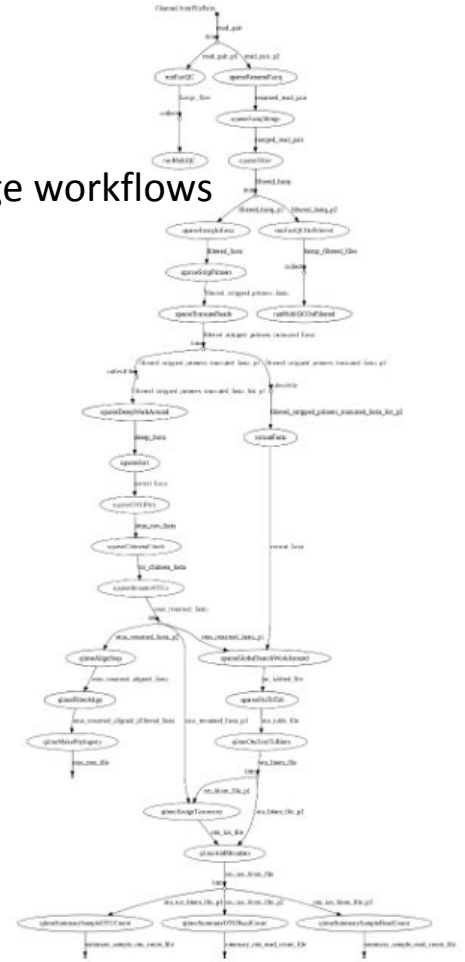
  - `$ srun --nodes=1 --ntasks=1 --time=120 --pty bash`

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.

H3ABioNet
Pan African Bioinformatics Network for H3Africa

16SrRNA Intermediate Bioinformatics Online Course: Int_BT_2019

Introduction to the command line and R

Introduction to the command line

Practical

Intro

For this practical we are going to go through the Software Carpentry's Introduction to Unix shell.

Please ask your teaching assistant what is the appropriate why for you to get to a shell / command line. Whether it is working on your own machine, a remote server or working on a compute node by launching an interactive job.

The breakdown the specific sections can be found here

We will focus on

1. Introducing the shell
2. Navigating files and directories
3. Working with files and directories
4. Pipes and filters

You have 20 minutes to spend on each section. After that we will quickly run through the section, look at the solutions and key points and then move on to the next.

You would have received an assignment with questions for each section that needs to be filled in and returned.

Intro

First we need to setup the data you would be working with. Follow the instructions here.

- `wget http://swcarpentry.github.io/shell-novice/data/data-shell.zip`
- `unzip data-shell.zip`

The `data-shell` directory contains the data that you will be working with.

Now you are setup and ready and can move onto section 1, then 2, then 3 and finally 4.

Note: In this practical we will not be doing the section Loops, Shell Scripts, Finding Things and Shell Extras. Feel free to work through those section in your own time because it contains tools and methods that will improve your command line efficiency.