Quo Vadis Workshop 2016

---

Software installation
Intro to Computing on Linux
Welcome!
Quo Vadis Workshop 2016

Pete Meyer
Ilyas Hamdi
Mick Timony
Andrew Morin
Justin O’Connor
Michelle Ottaviano
Carol Herre
Saythyda Corrado

Piotr Sliz
Rob DiCamillo
SBGrid: Structural Biology Research Computing

---

Who we are:

Non-profit Consortium based in BCMP @HMS focused on Structural Biology computing

Structural Biologists, IT pros, software engineers, programmers, software policy advocates, postdocs, students
Quo Vadis Workshop 2016 - 22 May 2016

---

Software Installation for QV2016

Intro to Linux Computing
Quo Vadis Workshop 2016

---

CCP4 : (COOT, iMosflm, ...)

DIALS :

XDS : (XDSGUI, XDSSTAT, XDSViewer)
Software for QV2016

---

https://sbgrid.org/wiki/sbgrid-qv2016-installation

1) Download the script:

```bash
```

2) Make executable and execute:

```bash
chmod +x sbgrid-qv2016
./sbgrid-qv2016
```

3) Load the environment:

```bash
source /programs/sbgrid.shrc  (sbgrid.cshrc in tcsh)
```
Software for QV2016

---

Installing updates:

./$HOME/programs/share/bin/sbgrid-qv2016
Datasets for QV2016

Datasets can be downloaded via rsync

```
rsync -av rsync://data.sbgrid.org/10.15785/SBGRID/$id
```

(where $id is replaced by the dataset ID).
Intro to Linux Computing

- Introduction to Linux
  Why Linux?

- The Linux interface
  (Understanding the Shell)

- Scientific Computing on Linux -
  Sysadmin’s tips and tools for computing and research
Linux

Linux is an open-source operating systems modeled on UNIX developed by Linux Torvalds in 1991.

Comp.os.minix

Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. ...
Linux

The GNU (Gnu’s Not Unix) was an effort to develop free and open source OS and applications.

Torvalds developed ‘kernel’ and combined it with software from Richard Stallman @ MIT.
Linux

Multi-user, multi-tasking

Many users on the same machine at once, running many programs

Multi-platform

runs on many different processor types
Linux

Multi-user, multi-tasking

Many users on the same machine at once, running many programs

Multi-platform

runs on many different processor types

Linux is a Unix-like system free of proprietary software for which source code is available and freely distributed
Linux

---

Multi-user, multi-tasking

Many users on the same machine at once, running many programs

Multi-platform

runs on many different processor types

Linux is a Unix-like system free of proprietary software for which source code is available and freely distributed
Linux

But why did Linux succeed? 
( and not HURD, BSD, Minux, etc ...)

- Decentralized Development
- Pragmatic (Not an academic or ideological exercise)
- Technological Superiority
- Luck?
"Everything is a file"

Defining features of *nix

Resources (documents, directories, keyboards, printers, storage, network communications, etc)

Are simple streams of bytes exposed through the filesystem name space
The Shell

---

The ‘shell’ is the Command Line Interface for Linux

This is an program that interprets what you type, keeps track of programs on the system, etc.

Common Shells:

tcsh : exTended C SHEll
bash : Bourne Again SHEll
ksh : Korn SHEll
csh : C SHEll (early popular shell)
sh  : the original shell, often a synonym for bash now
The Shell : Commands

---

Navigation (cd, pwd, ls)

Manipulating Files (cp, rm, mv, mkdir)

Search (grep, find)

Permissions (chmod, chown, chgrp)

Job Control (jobs, ps, fg, bg, nohup)

http://linuxcommand.org/lc3_learning_the_shell.php
The Shell: The Environment

---

The shell environment is configured globally per user in files and startup scripts

- Settings for variables
- Function definitions
- Aliases

Except for the reserved Shell special parameters variable names can be set by the user

Quotes remove special meaning from one or multiple characters
The Shell: The Environment

---
The ‘printenv’ command
The Shell: The Environment

---

The ‘printenv’ command

Shell variables:

- **PATH**
  - Where executables can be found
- **HOME**
  - User’s home directory
- **USER**
  - User’s username
- **SHELL**
  - Default shell setting
The Shell : The Environment

---

The ‘printenv’ command

Shell variables:

    PS1
        Shell prompt settings

    LD_LIBRARY_PATH
        Primary search path for library directories

...
The Shell: The Environment

---

The ‘alias’ command

An alias is a shortcut or abbreviation.

Great for avoiding typing a long command sequences

Aliases do NOT get passed to scripts (sub-shells)
The Shell: The Environment

---

Functions: The ‘declare -f’ command

Functions are subroutines: a code block (list of commands) that implements a set of operations.
The Shell : stdout stdin stderr

---

stdout :
  Output from commands
  written to the screen

stdin :
  Input for commands
  usually come from the keyboard

stderr :
  Error messages from processes
  usually written to the screen
The Shell: stdout stdin stderr

---
Pipe (|):
    stdout of one command to stdin of another command

Output Redirection (>):
    stdout of a command to a file

Output Appending (>>):
    stdout of a command appending to a file

Input Redirection (<):
    stdin of a command from a file

Use "-" to read this from standard input
The Shell: stdout stdin stderr

---

Stderr redirection

For tcsh
   &> filename

For bash
   2>&1 filename
The Shell : stdout stdin stderr

---

Most Linux (*nix) commands can be strung together

Example:

How many image files do I have?

```
ls -l *img | wc
```

How many image files do I have that are not have ‘native’ in the name?

```
ls -l *img | grep -v “native” | wc
```
The Shell: stdout stdin stderr

Most Linux (*nix) commands can be strung together

Example:

A list of all my image files:

```bash
ls *img > my_images.txt
```

A list of all my images sorted in reverse numerical order?

```bash
ls -l *img | sort -rn -k 9 > sorted_files.txt
```
The Shell script

Shell scripts are text files of variables, functions, and commands

A ‘shebang’ (#!/bin/bash, ...) is required to indicate which interpreter the OS programs loader should use

Conditional expressions: if/else, case
Loops: for, while, until

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
Scientific Computing: tools, tips and tricks

- Getting there and moving data
- What resources does this computer have (and what is it doing)?
- Reproducibility and collaboration
Scientific Computing: tools, tips and tricks

SSH

provides a secure channel (encrypted) over an unsecured network in a client-server architecture

- remote command-line login
- remote command execution
- any network service can be secured with SSH.
Scientific Computing: SSH

SSH public-key authentication allows login and command execution without passwords based on a public/private key pair.

Setup: create keys, set a password

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

Public key goes on the remote server in your .ssh directory in the file $HOME/.ssh/authorized_keys

Private key stays in $HOME/.ssh

Ssh-agent manages keys – typically running by default on most Linuxes

Use ssh-add command to add key, No more passwords!
Scientific Computing: SSH

SSH public-key authentication allows login and command execution without passwords based on a public/private key pair

Use ssh -X to forward X11 for graphics access
Scientific Computing: Moving data with rsync

---

rsync

- Transfers only changes in a file tree
- Local and remote synchronization of data file and directories

```bash	rsync -rv /my/files/here/ /my/files/over_there/
```

- Can be run over ssh for secure transfer

```bash	rsync -rv /my/files/here/ remote.server.org:/my/files/over_there/
```

- Ideal for data backup

```bash
man rsync
```

For more info
Scientific Computing: Hardware

---

CPU, Storage, memory, USB and PCI

- `cpuinfo` or `cat /prog/cpu`
- `df` or `lsblk`
- `free`
- `lsusb`
- `lspci`
Scientific Computing: What is running

---

CPU and memory use, jobs, IO

- top, uptime
- ps
- sar
Scientific Computing: history

---

history

The shell records all commands. This record can be accessed with the ‘history’ command.

Some relevant variables:

- **HISTSIZE**
  Define number of commands

- **HISTFILE**
  Define file

- **HISTCONTROL=ignoredups**
  Ignore duplicates
Scientific Computing: Terminal multiplexer

---

Tmux (or screen)

A terminal multiplexer is terminal-based program that gives the user

- Ability to detach and reattach sessions from a terminal
  - Sessions persist on the remote machine
  - A terminal session can be accessed from multiple machines
  - Persist through network disconnection

- Multiple separate login sessions inside a single terminal window
Scientific Computing: Version Control

---

VCS

Version control systems are designed for software development, and are great for scientific computing projects.

Version control software keeps track of every modification to the code.

Earlier versions of code are retained and can be accessed.
Scientific Computing: Version Control

---

Git / hg /svn

Preserves the modification history of scripts and configuration files ‘stream of snapshots’

Allows branching for new projects, ideas and experimentation

Designed for collaboration (labs)

Github, bitbucket, etc.
Scientific Computing: Getting started with git