UNIX Programming:

A brief introduction to UNIX history and most useful commands.

Toni Hermoso Pulido
Bioinformatics Core Facilty

UNIX History (I)

UNIX is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.

Un*x is still is a trademark.

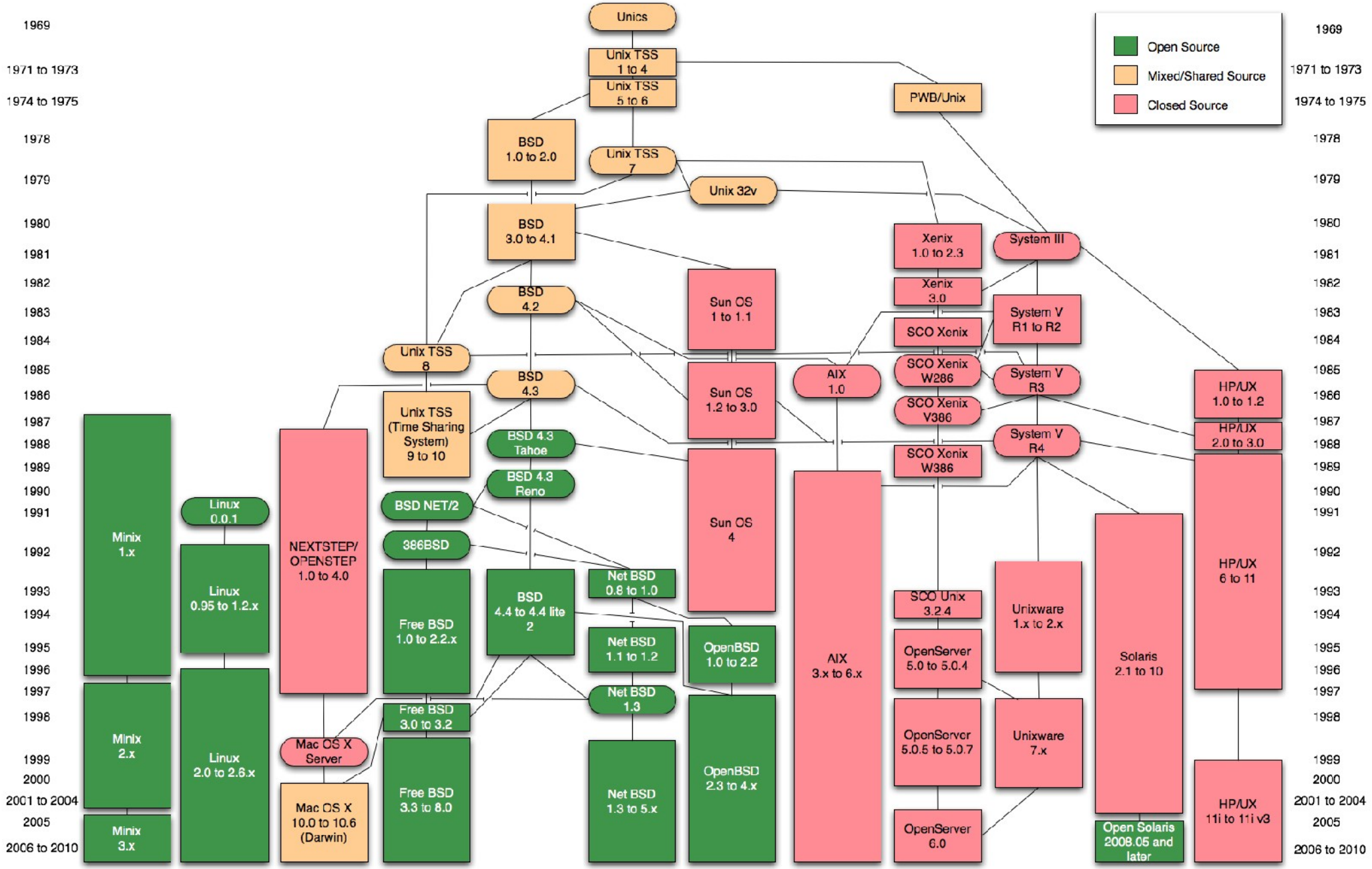
Intimately related to C programming language.

It was intended to be ported to several kinds of machines.

UNIX-like derivatives spread:

- BSD
- AIX
- HP-UX
- etc.

UNIX History (II)



http://en.wikipedia.org/wiki/File:Unix_history-simple.en.svg

UNIX Philosophy (I)

- Portable

  Same code should work the same in different machines

- Multi-tasking

  Different processes can run simultaneously

  Every process has a unique identifier (PID)

- Multi-user

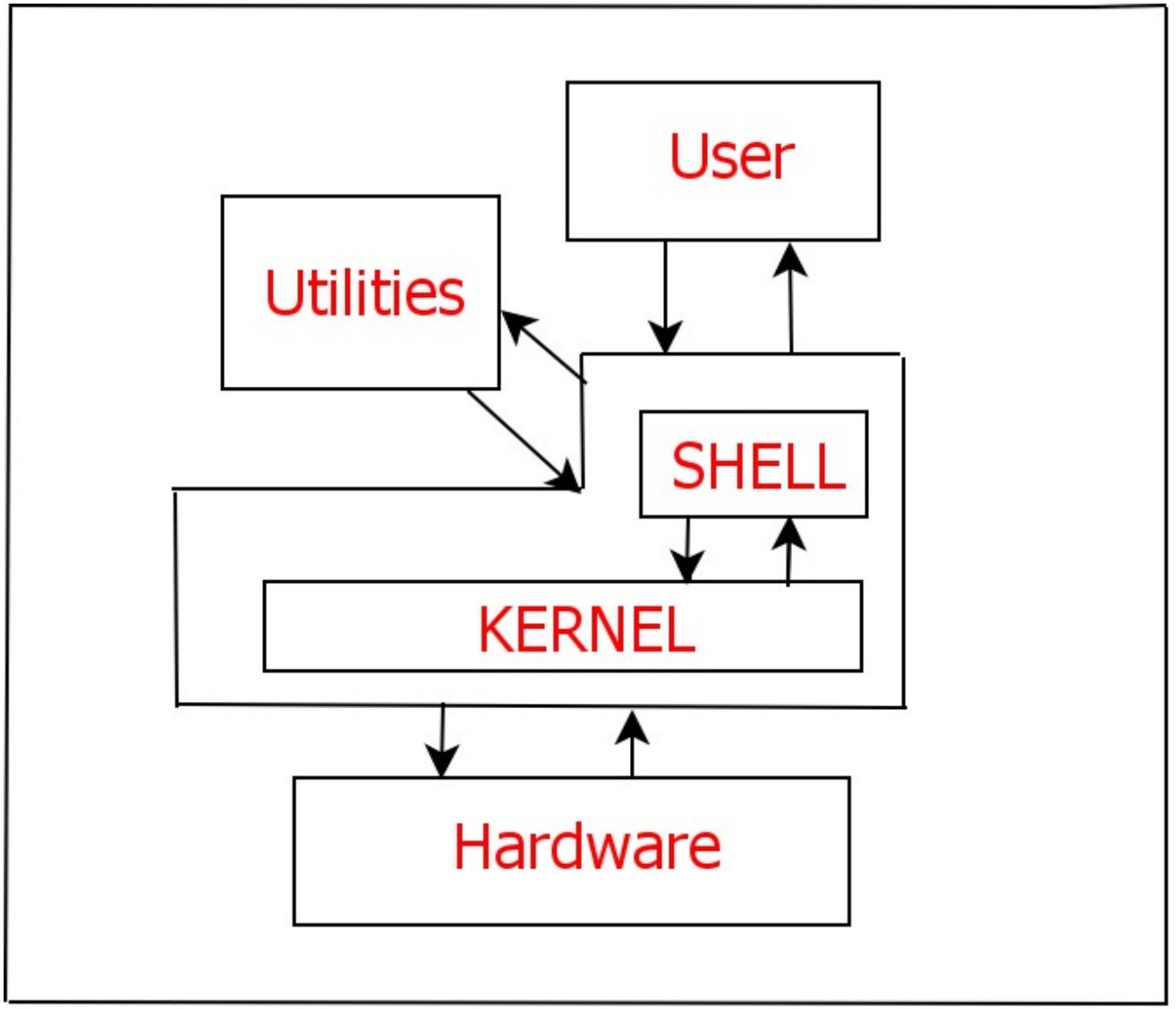
  Many people can use the same machine at the same time

  Users can share resources and processes

UNIX Philosophy (II)

- Use of plain-text for storing data
  also configuration files
- Hierarchical file system
- Almost everything is a file.
  That includes devices and some information of processes
- Use of small programs all together to retrieve an output instead of an only multifunctional one.

UNIX layers



http://eglug.org/node/456

GNU (GNU is not UNIX)

Principle

UNIX implementation based uniquely on free software

After Richard M Stallman dramatic experience with a printer in MIT.

Revival of academic principles of code sharing and reviewing.

Outcome

- Most basic tools are from GNU project.
- Most widespread license (GPL) as well.
- However, a kernel (core of the OS) was missing

Linux

Kernel & OS

Most popular UNIX-like operating system nowadays.
Based on Linux kernel -> after Linus Torvalds
Inspired on MINIX, an educational demo of UNIX

Deployed in many systems: computers, laptops,mobiles, video game consoles, supercomputers, etc.

Many distributions or tastes:

- Desktop / workstation: Ubuntu, Fedora
- Server: Debian, RedHat.
- Handset: Android, MeeGo, etc.

Mac OS X

Kernel & OS

Derivation from Mach kernel + BSD, NeXTSTE
BSD License significance

Open-source based OS behind: Darwin

Getting common UNIX software

- Xcode: Development tools.
http://developer.apple.com/technologies/tools/xcode.html
- X: Common Linux Window Graphical System
http://xquartz.macosforge.org

<u>Ports</u>

- MacPorts: http://www.macports.org/
- Fink (Debian-like): http://finkproject.org/

Cygwin

UNIX-like environment and user interface for Microsoft Windows.

Possible to run X (graphical environment).

Download many common programming languages and libraries from an installer.

http://www.cygwin.com/

Terminals and I/O

Terminals

CLI (Command-Line Interfaces) are the traditional way to work with UNIX machines.

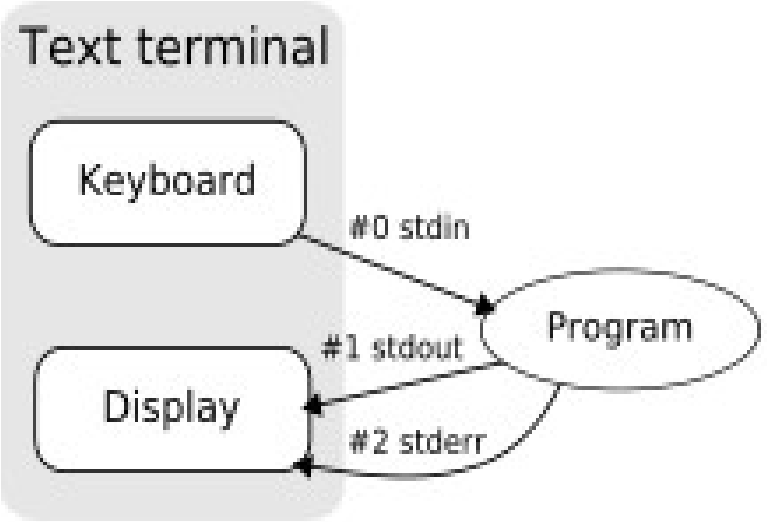
Terminal (Ctrl+Alt F1, F2, F3, ...)

Terminal emulator (xterm, etc.)

Xterminal (terminal in X environm

Input/Output (I/O)

- STDIN
- STDOUT
- STDERR



Shell

Definition

A Unix shell is a command-line interpreter or shell that provides a traditional user interface for the Unix operating system and for Unix-like systems.

Types of shell

Bourne-again shell (bash, sh)

- Derivation from original one.
- More scripting and OS control

C shell (csh)

- Modeled after C
- More interactive usage

Important to check which one is in your system. Nowadays it's more popular bash. Older Bioinformatics applications might use C shell for some installation steps

Terminal emulator in Mac OS X

Exercise

Open a terminal (emulator) in your computer.

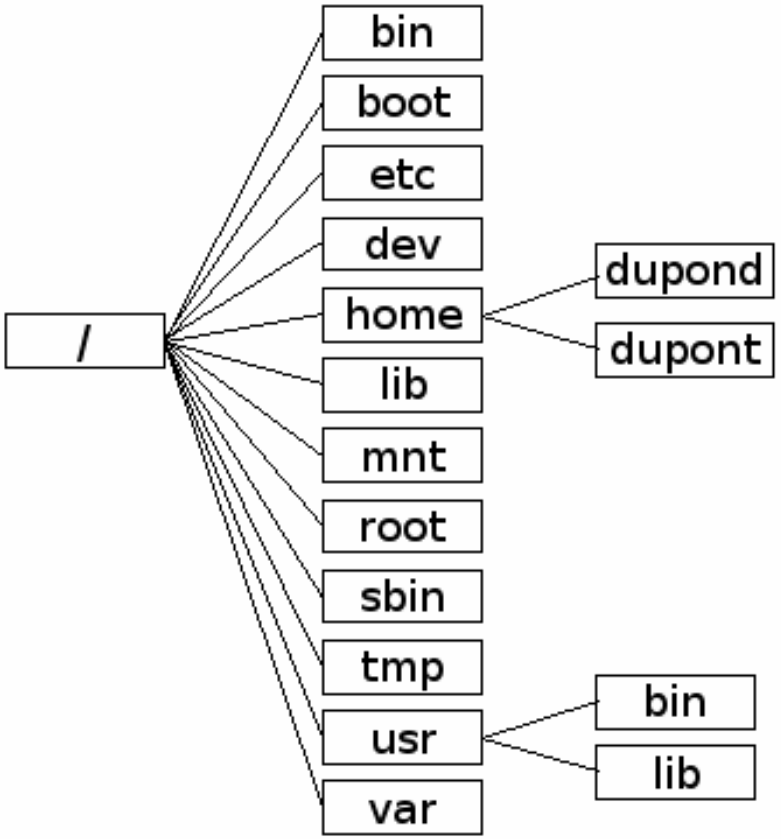
In Mac OS X:

Open Applications >> Utilities subfolder >> Terminal

Recommended terminal for Mac OS X:
http://iterm.sourceforge.net/

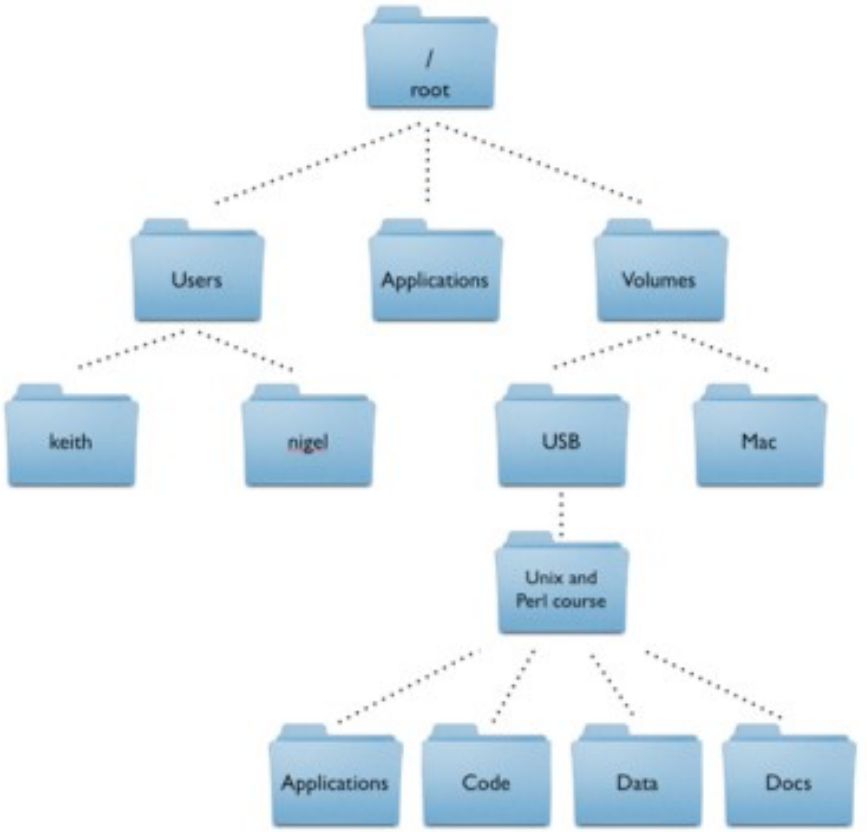
UNIX Directory tree

GNU/Linux



http://labor-liber.org/en/gnu-linux/introduction/all

http://korflab.ucdavis.edu/Unix_and_Perl/



Mac OS X

Simple commands (I)

ls

cd

pwd

touch

mv

rm

Exercise:

```
ls
ls -a
ls -la
ls -Fa
```

```
cd Desktop
cd ..
pwd  --> Absolute path
cd /Users/username/Desktop
```

```
cd; cd Desktop
touch example.txt
ls
mv example.txt ex.txt
rm ex.txt
```

Relative vs absolute paths

Relative paths

```
cd -> go to home
cd ../ -> go one level back
cd Desktop -> go to Desktop
cd ../../ go two levels back
cd ../Desktop/.. -> don't move
```

Absolute paths

```
cd / -> go to root
cd /users/username/Desktop
cd ~ -> go to home
```

Simple commands (II)

cp

mkdir

rmdir

cp -rf; rm -rf

cat

more or less (q for exiting)

Exercise:

```
cd Desktop
mkdir prova
ls
cp -rf prova prova2
rmdir prova
rmdir prova2
```

```
cd; cd Desktop
mkdir prova
cd prova
touch file
cp file file2
cd ..
rmdir prova
rm -rf prova
```

```
cat /etc/services
more /etc/services
less /etc/services
```

Command help

UNIX commands usually provide some hints and help about their parameters and usage

```
ls -h
```

```
nano --help
```

Depending on the program, one of the two former ways might not work.

```
man nano
man ls
```

Manual pages

Exercise:

```
Try man and –help with the different commands we learnt.
Hint1: exit man with q
Hint2: Stop programs executing in shell with CTRL+C
```

Text console editor

vi(m)

emacs

nano (former pico)

Exercise:

Try nano...
Get used to the commands: save, exit, etc.

File permissions (I)

UNIX FILE Permissions

- user (u), group (g), others (o)
- read (r), write (w), execute (x)

- `ls -l`

```
drwxr-xr-x    4 toniher  staff      136 Sep 27  2009 mpeg
-rw-r--r--    1 toniher  staff     1053 Dec 13 12:57 nanorc.nanorc
```

Changing Permissions

- `chmod g+rwx file.txt`
- `chmod a+rx mydirectory/`
- `chmod -R go-w mydirectory/`

File permissions (II)

Octal notation

0 --- no permission
1 --x execute
2 -w- write
3 -wx write and execute
4 r-- read
5 r-x read and execute
6 rw- read and write
7 rwx read, write and execute

```
"-rwxr-xr-x" -> 755
"-rw-rw-r--" -> 664
"-r-x------" -> 500
```

Changing permissions

`chmod 740 file.txt` (all owner, read group)
`chmod -R 755 mydirectory/` (all owner, read the rest)

Environment variables

Information about the system and the terminal

- Command: **env**
- Provide shortcuts
- Helpful to guess why something may not be working
- Example output:

```
MANPATH=/sw/share/man:/opt/local/share/man:
TERM_PROGRAM=iTerm.app
SHELL=/bin/bash
TERM=xterm-color
TMPDIR=/var/folders/Y6/Y6ZCmJ-vGk89i9u8YvRC8++++TI/-Tmp-/
Apple_PubSub_Socket_Render=/tmp/launch-vdCkbl/Render
OLDPWD=/sw/etc
USER=toniher
COMMAND_MODE=legacy
PATH=/sw/bin:/opt/local/bin:/opt/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/
usr/X11/bin
PWD=/Users/toniher
EDITOR=/usr/local/bin/mvim
PS1=\[\033[0;31m\][\t]\[\033[0m\]\[\033[0;32m\][\u@\h\[\033[0m\]\[\033[0;36m\]
| \w]$ \[\033[0m\]
HOME=/Users/toniher
SHLVL=2
LOGNAME=toniher
DISPLAY=/tmp/launch-6RaVBP/org.x:0
_=/usr/bin/env
```

.bashrc (I)

Storage of custom variables between sessions

No need to set environment variables one time and another.

Files

Depending on the system.
And different conventions what to put in each one.

`.bashrc`
`.profile`
`.bash_profile`
`.bash_login`

System wide: `/etc/profile /etc/bashrc`

.bashrc (II)

Example

```
export PATH=/sw/bin:$PATH
export MANPATH=/sw/share/man:$MANPATH
export EDITOR=/usr/local/bin/mvim

export PS1="\[\033[0;31m\][\t]\[\033[0m\]\[\033[0;32m\][\u@\h\\[\033[0m\]\
[\033[0;36m\] | \w]\$ \[\033[0m\]"

alias casa='ssh thermoso@casa.crg.es'
```

Exercises

- Add custom alias of some commands used
- Add new directories the PATH

Networking from the terminal

Download a file from an URL

```
wget http://nin.crg.es/bioinfo/test.tar.gz

curl http://nin.crg.es/bioinfo/test.tar.gz  > test.tar.gz
```

Compression and archiving

Archiving and compression

- zip --> zip -r archive.zip files

- gzip

- tar z:
- Compression -> tar zcf archive.tar.gz files
- Extraction -> tar zxf archive.tar.gz

- bzip2
- tar j:
- Compression -> tar jcf archive.tar.bz2 files
- Extraction -> tar jxf archive.tar.bz2

Exercise

- Uncompress `test.tar.gz`
- Compress back in `tar.bz2 format`

Piping (I)

STDIN

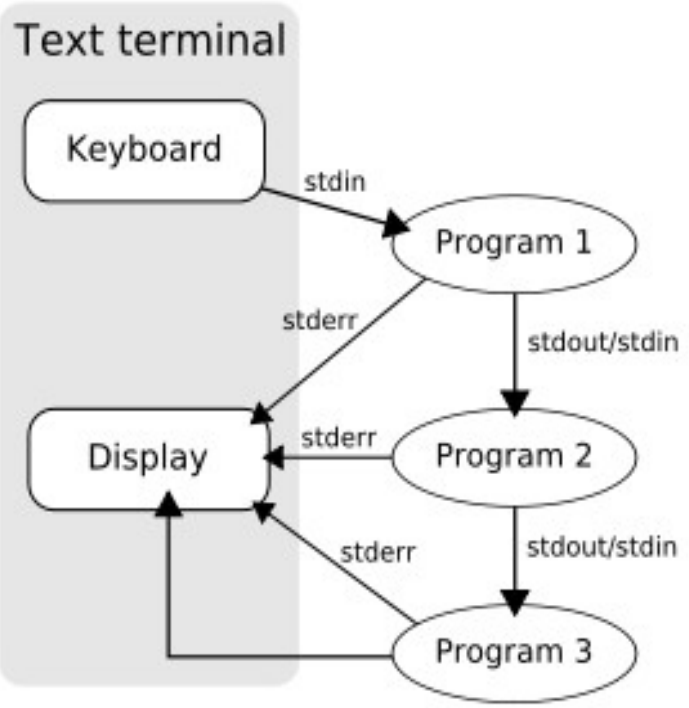
< Input, instead of interactive, from a file.

```
perl program.pl < input.txt
```

STDOUT

(>) Overwrite
(>>) Append

```
perl program.pl < input.txt >out.log
perl program.pl < input.txt >>out.log
```



Piping (II)

STDERR 2> 2>>

(2>) Overwrite
(2>>) Append

```
perl program.pl < input.txt 2>out.log
perl program.pl < input.txt 2>>out.log
```

STDIN, STDOUT, STDERR

All together

```
perl program.pl < input.txt &>out.log
```

PIPING THROUGH PROGRAMS

```
cat out.log | less
```

Exercise:

Create input.txt file with random content
Download http://biocore.crg.cat/toniher/program.pl
Follow the examples above
Create file.txt file with random content
Repeat the process

Executables and shebang

Exercise (continued)

First, Make program.pl executable
Repeat the previous procedure omitting perl before program.pl and adding ./ or absolute equivalent path.

That is:

./program.pl < input.txt 2>>out.log (relative)
or
/users/toniher/program.pl < input.txt 2>>out.log (absolute)

Why?

open program.pl file

#!/usr/bin/perl

#! -> shebang

Process management

Background

- Instance of running program and associated information: ID, parent process, owner, priority in the system, etc.
- If a process is a finished but non responsive and resident, it is named *zombie*.

Useful commands.
Exercise, try them:

- `top` (quit pressing q)
- `ps` (list process)
- `ps -ef` (GNU/Linux) || `ps -Au` (all system processes)
- `kill` -> stop program by process ID. `kill 666`
- `killall` -> stop program by command name. `killall antichrist`
- `kill -9` -> if process is zombie
- `killall -9` -> if process is zombie