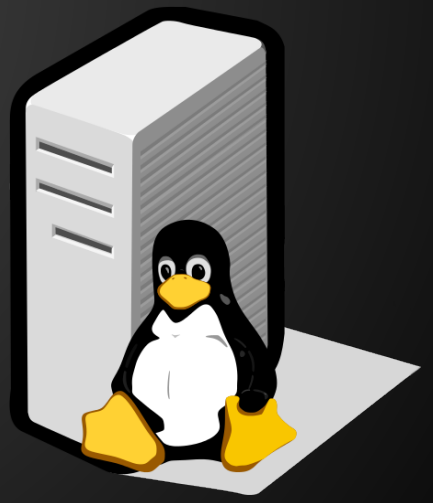
More Linux

Alan Orth



Recap - rationale

- Bioinformatics is the application of information technology and computer science to the field of molecular biology

- Data sets are getting bigger, we need more processing power power!

- … computers with that kind of power use Linux :)
- extremely efficient and stable
- excellent tools for text processing

Recap - commands

A few commands to help jog your memory.

cd – *change directory*
mkdir – *make directory*
mv – *move* a file or directory
cp – *copy* a file or folder
whoami – print the name of the current user
who – print a list of other users who are logged in
date – print the current date and time on the server
cal – print a calendar for the current month
echo – print a text string to the screen

Command structure

Linux commands come in various forms. Some are simple, and can be used by themselves:

```
whoami
cal
ls
date
```

Command structure

… other times you can add “arguments” to change the behavior of the command.

Arguments are separated by one or more spaces:

```
cal 2013
```

Some commands require arguments (they don’t make sense to run by themselves, like `mkdir`).

Commands and their arguments

A few common ones...

`ls -lh`	(“long” list of files)
`ls -la`	(“long” list of hidden files)
`ls -lh file`	(“long” list of file)
`mv file1 file2`	(rename file1 to file2)
`cp file filecopy`	(copy file to filecopy)
`rm file`	(delete file)
`rm -i file`	(delete file, but ask first)

Navigating the file system

Files and folders are organized in a hierarchical fashion. The top of the hierarchy is called the “root.”

The “root” of your home directory, for example:

```
/home/user1
```

Navigating the file system

Create some directories and get the hang of moving around them:

```
mkdir one
mkdir two
mkdir two/three
cd one
```

How do we get to `two`?

Navigating the file system

If we want to move to the directory “two” we have to first move back up in the directory hierarchy. Once we move back to “user1” we will be able to move into “two.”

```
cd ..
cd two
```

In Linux “..” means “parent directory”.

Your first shell script

A shell script is a text file with a list of commands inside. Shell scripts are good for automating tasks you use often, or running “batch” jobs.

Enter the following in a new file, call it `script.sh`:

```
echo “Date and time is:”
date
echo “Your current directory is:”
pwd
```

Your first shell script

Run the script like this:

```
sh script.sh
```

It should output something like this:

```
“Date and time is:”
Tue Oct  8 10:15:00 EAT 2052
“Your current directory is:”
/home/aorth
```

Your second shell script

Create a new script, `script2.sh`:

```
DATE=$(date)
PWD=$(pwd)

echo “Date and time is: $DATE”
echo “Your current directory is: $PWD”
```

More shell scripts

A more advanced shell script utilizing a loop:

```
for num in 1 2 3
do
   echo “We are on $num…”
done
```

What do you think it does? Can you try to run it? What is a good use case for this?

I/O Redirection

By default, command line programs print to *stdout* (“standard out”). I/O redirection manipulates the input/output of Linux programs, allowing you to capture it or send it somewhere else.

Two main kinds of redirection:

>	to a file
|	to another program

I/O Redirection

Redirect the output of your script to a file:

```
sh script.sh > script.out
```

… and to another program, ie `less`:

```
sh script.sh | less
```

Voila!

Anything else?

Questions? I promise not to laugh.