The Shell

EOAS Software Carpentry Workshop

September 20th, 2016

Getting Started

You need to download some files to follow this lesson. These files are found on the shell lesson website (see etherpad)

1. Make a new folder in your Desktop called shell-novice.
2. Download shell-novice-data.zip and move the file to this folder.
3. If it's not unzipped yet, double-click on it to unzip it. You should end up with a new folder called workshop.

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Why use the shell?

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Why use the shell?

- Connecting to supercomputers

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Why use the shell?

- Connecting to supercomputers
- Automate repetitive tasks

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Why use the shell?

- Connecting to supercomputers
- Automate repetitive tasks
- Reproducibility

Files and Directories

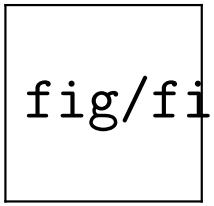
Learning Goals

1. Explain the similarities and differences between a file and a directory.
2. Translate an absolute path into a relative path and vice versa.
3. Construct absolute and relative paths that identify specific files and directories.
4. Explain the steps in the shell's read-run-print cycle.
5. Identify actual command, flags, and filenames in command-line call.
6. Demonstrate the use of tab completion, and explain its advantages.

Sample Code

- whoami
- pwd
- /
- ls
- ls -F data
- ls -F /data
- cd data
- cd ..
- ls north-pacific-gyre/2012-07-03
- ls no tab

Exercise

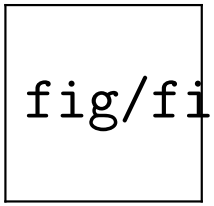


If `pwd` displays `/users/backup`, and `-r` tells `ls` to display things in reverse order, what command will display:

`pnas_sub/ pnas_final/ original/`

1. ls pwd
2. ls -r -F
3. ls -r -F /users/backup
4. Either #2 or #3 above, but not #1.

Exercise



If `pwd` displays `/users/backup`, and `-r` tells `ls` to display things in reverse order, what command will display:

`pnas_sub/ pnas_final/ original/`

1. ls pwd
2. ls -r -F
3. ls -r -F /users/backup
4. Either #2 or #3 above, but not #1.

Creating Things

Learning Goals

1. Create a directory hierarchy that matches a given diagram.
2. Create files in that hierarchy using an editor or by copying and renaming existing files.
3. Display the contents of a directory using the command line.
4. Delete specified files and/or directories.

Sample Code

- mkdir thesis
- cd thesis
- nano draft.txt
- rm draft.txt
- rm thesis
- rmdir thesis
- rm -r thesis
- mv thesis/draft.txt thesis/quotes.txt
- mv thesis/quotes.txt .
- cp quotes.txt thesis/quotations.txt

Exercise

Jamie is working on a project and she sees that her files arent very well organized:

```
$ ls -F
analyzed/  fructose.dat    raw/   sucrose.dat
```

The fructose.dat and sucrose.dat files contain output from her data analysis. What command(s) could you run so that the commands below will produce the output shown?

```
$ ls
analyzed   raw
$ ls analyzed
fructose.dat   sucrose.dat
```

Exercise

Jamie is working on a project and she sees that her files arent very well organized:

```
$ ls -F
analyzed/  fructose.dat    raw/   sucrose.dat
```

The fructose.dat and sucrose.dat files contain output from her data analysis. What command(s) could you run so that the commands below will produce the output shown?

```
$ ls
analyzed   raw
$ ls analyzed
fructose.dat   sucrose.dat
```

Solution

```
$ mv fructose.dat analyzed/fructose.dat
$ mv sucrose.dat analyzed/sucrose.dat
```

Pipes and Filters

Learning Goals

1. Redirect a command's output to a file.
2. Process a file instead of keyboard input using redirection.
3. Construct command pipelines with two or more stages.
4. Explain what usually happens if a program or pipeline isn't given any input to process.
5. Explain Unix's "small pieces, loosely joined" philosophy.

- cd molecules
- wc *.pdb
- wc -l
- wc -l *.pdb > lengths
- cat lengths
- sort lengths
- sort lengths > sorted-lengths
- head -1 sorted-lengths
- sort lengths | head -1
- wc -l *.txt
- wc -l *.txt | sort | head -5
- ls *Z.txt

Exercise

In our current directory, we want to find the 3 files which have the least number of lines. Which command listed below would work?

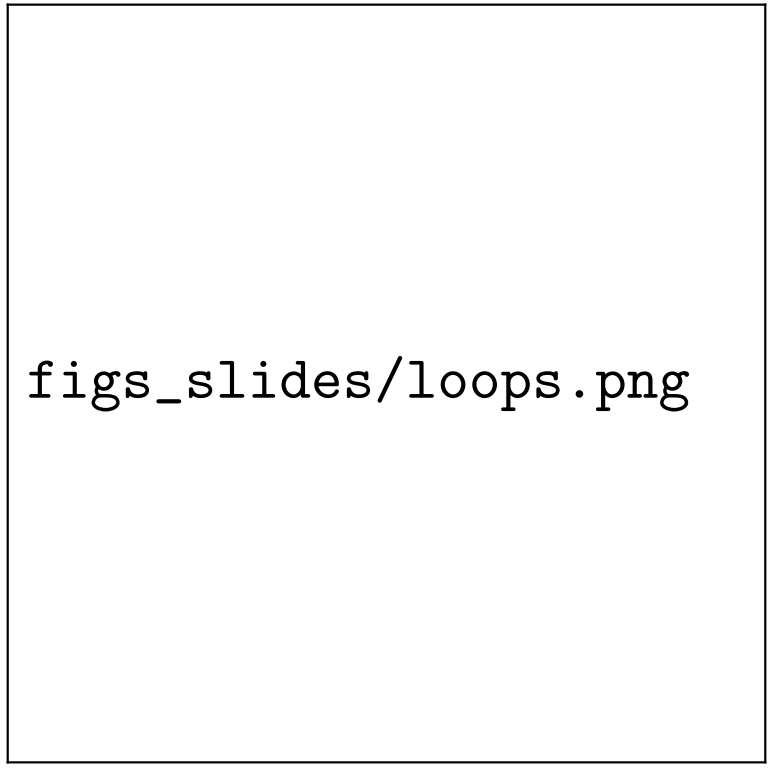
1. wc -l * > sort -n > head -3
2. wc -l * | sort -n | head 1-3
3. wc -l * | head -3 | sort -n
4. wc -l * | sort -n | head -3

Exercise

In our current directory, we want to find the 3 files which have the least number of lines. Which command listed below would work?

1. wc -l * > sort -n > head -3
2. wc -l * | sort -n | head 1-3
3. wc -l * | head -3 | sort -n
4. wc -l * | sort -n | head -3

Loops



https://xkcd.com/1411/

Loops

- Write a loop that applies one or more commands separately to each file in a set of files.
- Trace the values taken on by a loop variable during execution of the loop.
- Explain the difference between a variables name and its value.
- Explain why spaces and some punctuation characters shouldnt be used in file names.
- Demonstrate how to see what commands have recently been executed.
- Re-run recently executed commands without retyping them.

Variables in loops

Suppose that `ls` initially displays:

```
fructose.dat glucose.dat sucrose.dat
```

What is the output of:

```
for datafile in *.dat
do
 ls *.dat
done
```

Variables in loops

Suppose that `ls` initially displays:

```
fructose.dat glucose.dat sucrose.dat
```

What is the output of:

```
for datafile in *.dat
do
 ls *.dat
done
```

ANSWER:

fructose.dat glucose.dat sucrose.dat
fructose.dat glucose.dat sucrose.dat
fructose.dat glucose.dat sucrose.dat

Saving to a file in a loop

In the same directory, what is the effect of this loop?

```
for sugar in *.dat
do
 echo $sugar
 cat $sugar > xylose.dat
done
```

1. Prints `fructose.dat`, `glucose.dat`, and `sucrose.dat`, and the text from `sucrose.dat` will be saved to a file called `xylose.dat`.
2. Prints `fructose.dat`, `glucose.dat`, and `sucrose.dat`, and the text from all three files would be concatenated and saved to a file called `xylose.dat`.
3. Prints `fructose.dat`, `glucose.dat`, `sucrose.dat`, and `xylose.dat`, and the text from `sucrose.dat` will be saved to a file called `xylose.dat`.
4. None of the above

Saving to a file in a loop

In the same directory, what is the effect of this loop?

```
for sugar in *.dat
do
 echo $sugar
 cat $sugar > xylose.dat
done
```

1. Prints `fructose.dat`, `glucose.dat`, and `sucrose.dat`, and the text from `sucrose.dat` will be saved to a file called `xylose.dat`.
2. Prints `fructose.dat`, `glucose.dat`, and `sucrose.dat`, and the text from all three files would be concatenated and saved to a file called `xylose.dat`.
3. Prints `fructose.dat`, `glucose.dat`, `sucrose.dat`, and `xylose.dat`, and the text from `sucrose.dat` will be saved to a file called `xylose.dat`.
4. None of the above

Scripts

1. Write a shell script that runs a command or series of commands for a fixed set of files.
2. Run a shell script from the command line.
3. Write a shell script that operates on a set of files defined by the user on the command line.
4. Create pipelines that include user-written shell scripts.

Find the longest file with a given extension

Write a shell script called longest.sh that takes the name of a directory and a filename extension as its parameters, and prints out the name of the file with the most lines in that directory with that extension. For example:

```
$ bash longest.sh /tmp/data pdb
```

would print the name of the .pdb file in `/tmp/data` that has the most lines.

Why record commands in the history before running them?

If you run the command:

```
$ history | tail -5 > recent.sh
```

he last command in the file is the `history` command itself, i.e., the shell has added `history` to the command log before actually running it. In fact, the shell always adds commands to the log before running them. Why do you think it does this?

Script reading comprehension

Joel's data directory contains three files: `fructose.dat`, `glucose.dat`, and `sucrose.dat`. Explain what a script called `example.sh` would do when run as bash `example.sh *.dat` if it contained the following lines:

```
# Script 1
echo *.*
```

```
# Script 2
for filename in $1 $2 $3
do
cat $filename
done
```

```
# Script 3
echo $*.dat
```

Script reading comprehension

Joel's data directory contains three files: `fructose.dat`, `glucose.dat`, and `sucrose.dat`. Explain what a script called `example.sh` would do when run as bash `example.sh *.dat` if it contained the following lines:

```
# Script 1
echo *.*
```

ANSWER:

Prints

`example.sh fructose.dat glucose.dat sucrose.dat`

Script reading comprehension

Joel's data directory contains three files: `fructose.dat`, `glucose.dat`, and `sucrose.dat`. Explain what a script called `example.sh` would do when run as bash `example.sh *.dat` if it contained the following lines:

```
# Script 2
for filename in $1 $2 $3
do
cat $filename
done
```

ANSWER:

Shows contents of fructose.dat, glucose.dat, and sucrose.dat

Script reading comprehension

Joel's data directory contains three files: `fructose.dat`, `glucose.dat`, and `sucrose.dat`. Explain what a script called `example.sh` would do when run as bash `example.sh *.dat` if it contained the following lines:

```
# Script 3
echo $*.dat
```

ANSWER:

Prints

`fructose.dat glucose.dat sucrose.dat.dat`

Finding things

1. Use `grep` to select lines from text files that match simple patterns.
2. Use `find` to find files whose names match simple patterns.
3. Use the output of one command as the command-line parameters to another command.
4. Explain what is meant by ‘text’ and ‘binary’ files, and why many common tools don’t handle the latter well.

`find` pipeline reading comprehension

Write a short explanatory comment for the following shell script:

```
find .  -name *.dat' -print | wc -l | sort -n
```

Matching `ose.dat` but not `temp`

The `-v` flag to `grep` inverts pattern matching, so that only lines which do not match the pattern are printed. Given that, which of the following commands will find all files in `/data` whose names end in `ose.dat` (e.g., `sucrose.dat` or `maltose.dat`), but do not contain the word `temp`?

1. `find /data -name '*.dat' -print | grep ose | grep -v temp`
2. `find /data -name ose.dat -print | grep -v temp`
3. `grep -v "temp" $(find /data -name '*ose.dat' -print)`
4. None of the above.

Matching `ose.dat` but not `temp`

The `-v` flag to `grep` inverts pattern matching, so that only lines which do not match the pattern are printed. Given that, which of the following commands will find all files in `/data` whose names end in `ose.dat` (e.g., `sucrose.dat` or `maltose.dat`), but do not contain the word `temp`?

1. `find /data -name '*.dat' -print | grep ose | grep -v temp`
2. `find /data -name ose.dat -print | grep -v temp`
3. `grep -v "temp" $(find /data -name '*ose.dat' -print)`
4. None of the above.