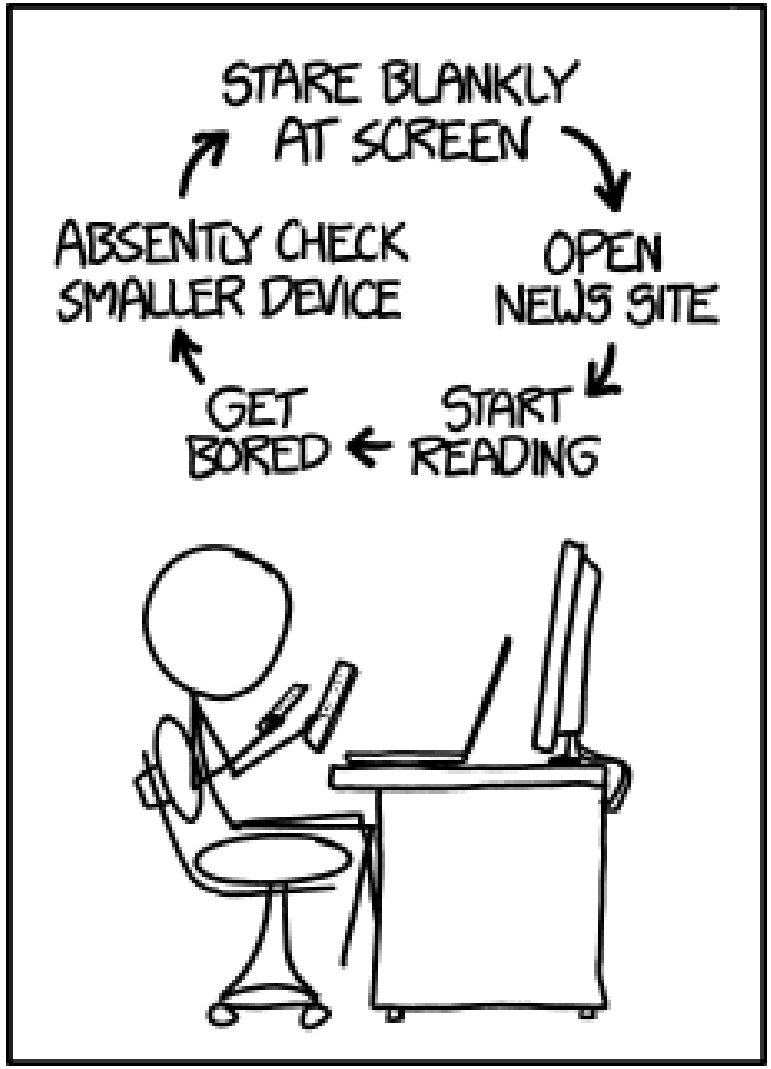
Loops, scripts and finding things in the shell

EOAS Software Carpentry Workshop

September 22nd, 2015

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 variableÕs name and its value.
- Explain why spaces and some punctuation characters shouldnÕt 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
```

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

Nested loops and command-line expressions

The `expr` does simple arithmetic using command-line parameters:

```
$ expr 3 + 5
8
$ expr 30 / 5 - 2
4
```

Given this, what is the output of:

```
for left in 2 3
do
 for right in $left
 do
 expr $left + $right
 done
done
```

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
```

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.