

CITS4407 Open Source Tools and Scripting

Semester 1, 2021

Week 7 workshop – Assignment 1, conditionals

Before starting this workshop, make sure you’ve reviewed the recommended reading for weeks 1–6, and completed the lab sheets for weeks 2–5.

1. Assignment work

We will use the lab/workshop time this week and next week to work on Assignment 1.

If you have a question about the assignment, we ask that you post it on the MS Teams “Assignment 1” channel, so that (a) we can reduce duplicated effort if multiple people have the same question, and (b) teaching staff have some idea of how to allocate their time across the different questions.

2. Assignment tips

Using Docker to access the test environment

The wiki pages on the MS Teams “Assignment 1” channel include advice on how to get a Bash shell in the standard Ubuntu environment we’ll be using for testing your code.

But note that you don’t have to do all your work within the test environment.

The wiki pages suggest running a command like

```
$ docker run --rm -it -v $PWD:/work adstewart/cits4407-2021-env:0.1.3
```

to access the test environment.

This makes the directory you’re currently in (“\$PWD”) visible within the test environment. So you can use graphical editors and so on *outside* the test environment in a separate terminal, and just use the test environment for periodically running tests.

Using arrays to store lists of things

Although they are not essential to completing the assignment scripts, you may find it useful to read through the chapter of the textbook (Shotts, chapter 35) on *arrays* in Bash.

Arrays allow us to store and manipulate a “list” of strings, and access or modify individual elements in that list.

Try typing:

```
$ my_array=(aardvak bonus category delineate estimate)
```

This creates an array called “my_array” which holds 5 words. Try the following:

```
$ echo "${my_array[2]}"
```

and you should see Bash print the word “category”. Here we have accessed the element at position 2 in the array; positions start from 0, so “aardvark” is at position 0, “bonus” at position 1, and “category” from position 2.

Try typing:

```
$ my_array[2]="some other string"
```

Now repeat the `echo` command from above – what do you see? The assignment statement above should have modified the string held in position 2 of the array.

It’s possible to read lines from a file into an array using code like the following:

```
IFS=$'\n' myarray=($(cat myfile))
```

Here, IFS stands for the “Internal Field Separator” – the characters that Bash considers separate different “words” when it’s asked to read or parse a line. Bash by default splits strings up into *words* separated by whitespace, and we don’t want that – we want to preserve all spaces – so we temporarily alter the value of “IFS”. (Covered on p. 439 of the textbook.)

declare -p

The command “`declare -p some_variable`” can be useful when debugging a program.

The “-p” argument means that instead of creating a variable (which is what `declare` usually does), it *prints* the variable in human-readable format to the screen.

This can be very useful for showing the contents of arrays, in particular, since the “`echo`” command won’t necessarily do what you expect for arrays. (It will only print the first element.)