Week 10 Laboratory Sample Solutions

Objectives

- Developing scripting skills relevant to systems

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

Getting Started

Create a new directory for this lab called `lab10`, change to this directory, and fetch the provided code for this week by running these commands:

```
$ mkdir lab10
$ cd lab10
$ 2041 fetch lab10
```

Or, if you're not working on CSE, you can download the provided code as a zip file or a tar file.

EXERCISE:

What is Behind That URL?

Write a Shell program, **which_webserver.sh** which given the URL of 1 or more webservers as command-line arguments prints what software the webserver is running, as indicated by the server field returned in a request to the web server.

Match the output and behaviour in the example below exactly:

```
$ ./which_webserver.sh https://www.unsw.edu.au
https://www.unsw.edu.au Apache/2.4.34 (Red Hat) OpenSSL/1.0.1e-fips PHP/5.6.25
$ ./which_webserver.sh https://www.google.com
https://www.google.com gws
$ ./which_webserver.sh https://www.netflix.com
https://www.netflix.com nq_website_nonmember-prod-release b2b0bc7b-bffd-4024-b8b8-d060e658f2ac
$ ./which_webserver.sh https://www.abc.net.au https://www.sydney.edu.au http://www.bom.gov.au
https://www.abc.net.au Apache/2.4.43 (Unix)
https://www.sydney.edu.au Apache
http://www.bom.gov.au Apache
```

Hint

```
man curl
```

Assumptions

Your answer must be Shell. You can not use other languages such as Perl, Python or C.

No error handling is necessary.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest which_webserver
```

When you are finished working on this exercise, you must submit your work by running `give`:

```
$ give cs2041 lab10_which_webserver which_webserver.sh
```

before **Tuesday 11 August 21:00** to obtain the marks for this lab exercise.

Sample solution for `which_webserver.sh`

```
#!/bin/sh

for webserver in "$@"
do
    server_string=$(
        curl --head --silent "$webserver"|
        egrep -i '^server: '|
        head -1|
        sed 's/^[^:]*: //'
    )
    echo "$webserver $server_string"
done
```

EXERCISE:

Find Those Makefiles

Write a Shell program, **make_tree.sh** which is given the pathname of a directory tree.

It should search this directory tree for directories containing a makefile.

For each of these directories it shouldprint the directory pathname and then run **`make`** in the directory.

Match the output and behaviour in the examples below exactly:

```
$ unzip /web/cs2041/20T2/activities/make_tree/examples.zip
$ find simple
simple
simple/b.c
simple/Makefile
$ ./make_tree.sh simple/
Running make in simple
clang -Wall   -c -o b.o b.c
clang -Wall -o p b.o
$ find medium
medium
medium/a
medium/a/x.c
medium/a/h.c
medium/a/Makefile
medium/a/l.c
medium/b
medium/b/b.c
medium/b/j.c
medium/b/k.c
medium/c
medium/c/p.c
medium/c/Makefile
$ ./make_tree.sh medium
Running make in medium/a
clang -Wall   -c -o h.o h.c
clang -Wall   -c -o l.o l.c
clang -Wall   -c -o x.o x.c
clang -Wall -o c h.o l.o x.o
Running make in medium/c
clang -Wall   -c -o p.o p.c
clang -Wall -o d p.o
$ find hard
hard
hard/a
hard/a/p.c
hard/a/r.c
hard/a/Makefile
hard/a/q.c
hard/b
hard/b/b
hard/b/b/b
hard/b/b/b/b
hard/b/b/b/b/b
hard/b/b/b/b/r.c
hard/b/b/b/b/Makefile
hard/c
hard/c/d
hard/c/d/o.c
hard/c/d/Makefile
hard/c/e
hard/c/e/a.c
hard/c/e/b.c
hard/c/e/f
hard/c/e/f/a.c
hard/c/e/f/Makefile
$ ./make_tree.sh hard
Running make in hard/a
clang -Wall   -c -o p.o p.c
clang -Wall   -c -o r.o r.c
clang -Wall   -c -o q.o q.c
clang -Wall -o b p.o r.o q.o
Running make in hard/b/b/b/b
clang -Wall   -c -o r.o r.c
clang -Wall -o r r.o
Running make in hard/c/d
clang -Wall   -c -o o.o o.c
clang -Wall -o m o.o
Running make in hard/c/e/f
clang -Wall   -c -o a.o a.c
clang -Wall -o i a.o
```

The test directories are also available as a zip file

If **make_tree.sh** is give extra command line arguments they should be passed to each make command, for example:

```
$ cat simple/Makefile
CC=clang
CFLAGS=-Wall

p: b.o
$ (CC) $(CFLAGS) -o $@ $^

.PHONY: clean

clean:
    rm -f b.o

clobber: clean
    rm -f p
$ ./make_tree.sh medium clobber
Running make in medium/a
rm -f h.o l.o x.o
rm -f c
Running make in medium/c
rm -f p.o
rm -f d
$ ./make_tree.sh hard clean
Running make in hard/a
rm -f p.o r.o q.o
Running make in hard/b/b/b/b
rm -f r.o
Running make in hard/c/d
rm -f o.o
Running make in hard/c/e/f
rm -f a.o
```

Hint

Use **`find`** to find all Makefiles in the directory tree.

Assumptions

You can assume makefiles are named **`Makefile`**

Your answer must be Shell. You can not use other languages such as Perl, Python or C.

No error handling is necessary.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest make_tree
```

When you are finished working on this exercise, you must submit your work by running `give`:

```
$ give cs2041 lab10_make_tree make_tree.sh
```

before **Tuesday 11 August 21:00** to obtain the marks for this lab exercise.

Sample solution for `make_tree.sh`

```
#!/bin/sh

if test -z "$1"
then
    echo "Usage: $0 <directory>"
    exit 1
fi

directory="$1"
shift

for makefile in $(find "$directory" -name Makefile)
do
    directory=$(dirname "$makefile")
    echo "Running make in $directory"
    # could also do this with cd/pwd
    make --no-print-directory -C $directory "$@"
done
```

EXERCISE:

Should I Compress That File

Write a Shell program, **compress_if_needed.sh** which takes the pathnames of 0 or more files as command line arguments.

It should compresses each file with **xz** if and only if, this results in a smaller file.

Match the output and behaviour in the example below exactly:

```
$ unzip /web/cs2041/20T2/activities/compress_if_needed/examples.zip
$ ls -l file*
-rw-r--r-- 1 andrewt andrewt 4096 Aug  3 17:10 file1.bin
-rw-r--r-- 1 andrewt andrewt 4096 Aug  3 17:10 file2.bin
$ ./compress_if_needed.sh file1.bin file2.bin
file1.bin 4096 bytes, compresses to 4156 bytes, left uncompressed
file2.bin 4096 bytes, compresses to 2160 bytes, compressed
$ ls -l file*
-rw-r--r-- 1 andrewt andrewt 4096 Aug  3 17:10 file1.bin
-rw-r--r-- 1 andrewt andrewt 2160 Aug  3 17:10 file2.bin.xz
```

The two test files are also available as a zip file

Assumptions

Your answer must be Shell. You can not use other languages such as Perl, Python or C.

You should use xz's default compression. xz has many options to change compression behaviour. Do not use these.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest compress_if_needed
```

When you are finished working on this exercise, you must submit your work by running `give`:

```
$ give cs2041 lab10_compress_if_needed compress_if_needed.sh
```

before **Tuesday 11 August 21:00** to obtain the marks for this lab exercise.

Sample solution for `compress_if_needed.sh`

```
#!/bin/sh

for file in "$@"
do
    xz --keep $file || continue

    n_bytes_before_compression=$(stat -c %s "$file")
    n_bytes_after_compression=$(stat -c %s "$file.xz")

    echo -n "$file "$n_bytes_before_compression" bytes, "
    echo -n "compresses to $n_bytes_after_compression bytes"

    if test "$n_bytes_before_compression" -lt "$n_bytes_after_compression"
    then
        echo ", left uncompressed"
        rm "$file.xz"
    else
        echo ", compressed"
        rm  "$file"
    fi
done
```

Submission

When you are finished each exercises make sure you submit your work by running `give`.

You can run `give` multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Tuesday 11 August 21:00** to submit your work.

You cannot obtain marks by e-mailing your code to tutors or lecturers.

You check the files you have submitted here.

Automarking will be run by the lecturer several days after the submission deadline, using test cases different to those `autotest` runs for you. (Hint: do your own testing as well as running `autotest`.)

After automarking is run by the lecturer you can view your results here. The resulting mark will also be available via give's web interface.

Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```
