UNIX Tutorial 5 - C and Makefiles

Today, we are going to use the Clang C compiler and the Make utility. These are pre-installed on the Glasgow Linux systems. However if you are running your own Linux distro, you may need to install extra packages. Try:

```
$ sudo apt-get install build-essential
$ sudo apt-get install clang
```

on Ubuntu distros. It is important to become familiar with these utilities since you will be using them extensively in the Advanced Programming 3 course later on.

Writing a C Program

Let's write a simple hello world C program.

```
$ cd ~
$ vim hello.c
```

This should drop you into the vim screen, where you can start to edit the file. Press `i` to enter *insert mode* then type the C program, something like...

```c
#include <stdio.h>

int main(void) {
    printf("hello world\n");
    return 0;
}
```

Now save hello.c and quit vim (Escape `:` `w` `q` Enter).
Now let's compile this C program.

```
$ clang hello.c -o hello
```

which should generate an executable file called hello, if the compilation succeeds. You can check whether the executable file is there by listing the files in the directory.

```
$ ls -l
```

You should be able to see a file called 'hello' which has a recent timestamp and has the x permission set to indicate that it is an executable file.

Now let's run this program.

```
$ ./hello
```

and watch it print its string to the standard output.

Creating a Makefile

Now let's use the make utility to automate our build process. A Makefile is a script that specifies how to build target files from source files. In the same directory as your `hello.c` program, create a file called `Makefile`.

```
$ vim Makefile
```

The file should have the following format:

```
# my first makefile (this is a comment)
# my student id

hello: hello.c
    clang hello.c -o hello
```

Notice that lines beginning with the hash character are comments (like Python). The target is separated from the source file by a colon. The line(s) underneath the target specify the commands to create the target. Note that each of these command lines *must* start with a TAB character.

Save your file. To check it works, run the command `make` in this directory.

```
$ make
```

It will either tell you that 'hello is up to date' or it will rebuild the hello executable. If hello is up to date, then you can change the timestamp on the hello.c file to make it more recent than the executable file—using the touch command. Try this:

```
$ touch hello.c
$ make
```

This will force a recompilation. The other way to force the recompilation is to use the `-f` flag for make.

Note that the first target in the Makefile is the 'default'. To create a different target, you must specify its name when you run 'make' i.e.

```
$ make hello
```

Some Puzzles for You

0.1 Shorten it!

You can rewrite your Makefile to shorten it considerable. In a rule (specifying how to convert a source into a target), the target name can be written as `$@` and the source name can be written as `$<`. Rewrite your Makefile to use these meta-variables instead. Check `http://www.linux-pages.com/2013/02/gnu-makefile-special-variables-dollar-at/` for more details.

0.2 Do it for Java

Find (or quickly write) a Java program and write a Makefile to automate its compilation, using the `javac` command.

0.3 Lots of Dependencies

(**difficult**) In the work above, we looked at the `touch` command. Use `man touch` to find out more about it. In general, `touch` takes a single filename argument. If that file exists, then it updates its timestamp to 'now'. If that file does not exist, then `touch` creates an empty file with the current timestamp. Ok . . . imagine a process where you want to generate a file `degree`, which requires you to have three courses files `a, b, c`. Suppose each course (e.g. `a`) requires you to have `a.exam` and `a.coursework`. You have a nice set of dependencies now! Can you turn this into a Makefile, using the touch command to 'create' each file? I will start you off...

```
a: a.course a.exam
    touch $@

a.course:
    touch $@

a.exam:
    touch $@
```

3