FIT2070 Assignment 2

Josh Nelsson-Smith

16/09/2016

Installing the program

Start by downloading and unzipping the file named `25954113_A2.zip`, once unzipped you should see the folder contains the following files:

```
25954113_A2
├── README.md
├── README.pdf
├── TimingFiles
│   ├── 1string10long.txt
│   ├── 1string8long.txt
│   ├── 2strings8long.txt
│   ├── 3strings8long.txt
│   ├── 4strings8long.txt
│   └── processTiming.pdf
├── assignment2-coversheet-jxns.pdf
├── compScript.sh
├── permutations
├── src
│   ├── permutations.c
│   └── task2.c
└── task2

2 directories, 14 files
```

The project contains the `src` folder that houses the c code for the assignment. Within it are the `permutations.c` and `task2.c` files. Inside the `Timing Files` directory are a few sample time output files for various combinations of inputs and string lengths as well as `processTiming.pdf` that lists the averages for these files. The project also comes with a shell script that can be used to compile the code called `compScript.sh`. It is recommended you re-compile the code before running to ensure the executables work correctly on your architecture.

a. Compile using included script `compScript.sh`

```
$ ./compScript.sh
Compiling c program...
C programs compiled!
```

Note - you may need to alter permissions to allow the script to execute via something like:

```
$ chmod 777 compScript.sh
```

b. Compile the program manually via:

```
$ gcc -o permutations src/permutations.c
$ gcc -o task2 src/task2.c
```

Using the Programs

As said above, there are 2 programs included, `permutations` and `task2`. Permutations takes one argument, the string you want to show all the permutations of. For example running:

```
$ ./permutations abc
```

Will print out all the permutations of the string "abc". In addition to this, the permutations will be printed to a file called `output.txt` should you want to store them or if your terminal has a print limit. Another file `times.txt` will also be generated and will list how long it took to run permutations on your string. Also note that if you wish to run some tests on the permutations program, you can run them via:

```
$ ./permutations -t
```

The second executable `task2` is a little more complex. It runs the permutations executable from earlier, but by creating a new process for each of them. It's usage is `permutations n <string 1> <string 2> .. <string n>`. For example:

```
$ ./task2 3 abc def ghi
```

Would print the permutations of the three strings "abc", "def" and "ghi". Also note that because this runs the `permutations` program from earlier, it will also generate the associated `times.txt` and `output.txt` files.