Griffith UNIVERSITY
Queensland, Australia

Assignment 1- Milestone 1

2803ICT System and Distributed Computing
School of ICT, Griffith University
Trimester 2, 2022

Assignment 1, Milestone 1 (7%)
Milestone 1 is due on 11:59pm Sunday 21st August 2022
With peer-review and demonstrations in week 5.

Instructions:

- **Due: Milestone 1** - 11:59pm, 21st August
- **Marks: 7% of your overall grade**
- **Late submission:** Late submission is allowed but a penalty applies. The penalty is defined as the reduction of the mark allocated to the assessment item by 7% of the total weighted mark for the assessment item, for each day that the item is late.
- **Extensions:** You can request for an extension of time on one of two grounds:
  - Medical
  - other (e.g. special family or personal circumstances, unavoidable commitments).

  All requests must be made through the myGriffith portal.
- **Individual Work:** You must complete this assignment individually. You are encouraged to discuss the assignment with your fellow students, but eventually this should be your own work. Anyone caught plagiarising will be penalised and reported to the university.
- **Presentation:** You **must** present/demonstrate your work on **Week 5** to two other students, and provide a feedback to the other students' presentations. You may also be asked to demonstrate your code to a teaching team member. Your work will not be marked if you do not present it.
- **Submission:** For this milestone you should submit to L@G:
  - a zip file that includes your source code and makefile.
  - Feedback page to be submitted on week 5 - further details will be provided by week 4.
- **Marking**: For marking details please refer to the marking rubric.
- **Objectives:**

  ✓System programming using C

  ✓Write programs that interact with file systems

2803ICT – System and Distributed Computing – T2 2021
Assignment 1 – Milestone 1

Requirement:

1. Write a C code for 'shell' program so that it operates as a command shell. When you run it, it waits for you to type in the command name, then executes it and prints the results to *stdout*. It then waits for and executes the next command in an endless loop, unless the entered command is '**quit**'.
2.
3. Your code should work on linux using Cygwin.

4. The following 7 commands are to be supported by your shell program.
   - **calc** ***expr.*** - prints out the result of the mathematical prefix expression that comes after the command.
   - **time** - prints out the current local time and date
   - **path** - prints out the current working directory
   - **sys** - prints the name and version of the OS and CPU type
   - **put** ***dirname filename(s) [-f]*** – put files *filenames* in the directory *dirname*
   - **get** ***filename*** - prints the content of the file *filename* to the screen
   - **quit** - ends the program

Notes:

- If the command is not one of the above, you should print an error message and wait for a new command.
- You can assume the expression after the **calc** command is a valid prefix expression containing only '+' and '–' signs. You can also assume a space character separates any two numbers/signs. e.g. "+ + 2 3 - 4 5".
- Note that if you want to use the prefixadd() function from week 2 workshop you will need to store the expression as a ragged array of strings (see week 2 exercise 3).
- **time** – you can use the functions defined in **<time.h>** (link). Hint: look at **time()**, **localtime()** and **asctime()** functions.
- **path** - you can use the linux system function **getcwd()** (link).
- **put** - The **put** command will create a new directory called ***dirname*** and **copy** the file (or files) listed in the command, in this directory. If the directory exists you should only print an error message, unless **-f** has been specified, in which case the directory will be completely overwritten (old content is deleted). If a file(s) doesn't exist, you will need to print a 'file not found' message for that file.
- **get** - The **get** command will dump the file contents to the screen 40 lines at a time and pause, waiting for a key to be pressed before displaying the next 40 lines etc.

3. Write a **Unix makefile** that creates an executable program called '**shell**' for your program.