NWEN 241 Exercise 4

System calls and Introduction to C++

Release Date: **7 May 2024**

Submission Deadline: **15 May 2024, 23:59**

Objective:

The objective of this exercise is to write and debug C programs that involve process management & socket programming, and provide an introduction to C++ .

At the end of this exercise, you should submit the required files to the Assessment System (`https://apps.ecs.vuw.ac.nz/submit/NWEN241/Exercise_4`) on or before the submission deadline. You may submit as many times as you like in order to improve your mark before the final deadline. Submissions beyond the deadline will not be marked and will receive 0 marks.

Exercise Requirements

For NWEN 241, it is highly recommended that you undertake all development using the computers in CO246. The computers in this lab use the Linux operating system. *This guide is written with the assumption that you are in CO246 lab.*

If you are not able to go the lab, you can remotely access similar computers via secure shell (ssh). Consult one of the remote study guides (see `https://ecs.wgtn.ac.nz/Courses/NWEN241_2023T1/RemoteStudyGuides`) and follow one that suits you the most.

Exercises

Activity 1: Process Management [30 Marks]

Copy and paste the following C program to your favourite text editor:

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<stdlib.h>

int main()
{
 int pid, ret_exec, status;
 pid=fork();

 switch(pid){

 case -1:
   /* code for case -1 */

 case 0:
    /* code for case 0 */

 default:
    /* code for case default */
 }
}
```

Save the file as `activity1.c`. Study the source file and complete the program by adding suitable code segments in the three switch case blocks as defined below:

1. `case -1`: Display error message - `Error` and exit from the program
2. `case 0`: Use `execl` system call to execute the `ps -A` command. If the `exec` system call is unsuccessful, display an error message - `Error executing exec`
3. `default`: Wait for termination of the child process, then if WIFEXITED(status) is set display the following values (in the same order):
   - process ID of the parent process
   - process ID of the child process and
   - the termination status of the child process.

Do not modify the order, otherwise, the automated marking might not work correctly. Compile and run the program. If you are happy with the program, submit it to the Assessment System for marking. Click on **Run checks** to initiate auto-marking.

Activity 2: Socket Programming 2 [30 Marks]

In this activity, you will write a C server program that runs on the local machine at port number 23456. The server should be programmed to receive a string from a client and return the reversed string back to the client. For example, suppose you simulate a `Netcat` client using the command `nc localhost 23456` and then send a string `"Hello"` to the server; the server would return `"olleH"` back to the client as shown below:

```
nc localhost 23456
Hello

olleH
```

Note: Remember to close the client socket after sending the reversed string back to the client.

Save the program as `activity2.c`. If you are happy with the program, submit it to the Assessment System for marking. Click on **Run checks** to initiate auto-marking.

Activity 3: Introduction to C++ [40 Marks]

In this activity, you will write a C++ program to represent complex numbers[1] of the form `a + bi`.

Define a class `complex` in a `namespace Complex`. The class should contain the following members:

- Private integer members `a` and `b`.
- A `constructor` with zero arguments and default values for `a` and `b` set to `1`;
- A `constructor` with two arguments which will be used to initialize `a` and `b`, respectively.
- `int geta( )` and `int getb()` member functions that return the values of `a` and `b`, respectively. These functions should be public.

Write a `main()` function that does the following:

[1] See `https://mathworld.wolfram.com/ComplexNumber.html` for more information about complex numbers.

- Declares a complex number using the default constructor (name this complex number as `c1`)
- Declares a complex number using the parameterized constructor (name this complex number as `c2`, and use 5 and 10 as the parameters)
- Displays the values of both the complex numbers. The display should look exactly like this:

  `Complex␣number␣1:␣`**`a1`**`␣+␣`**`b1`**`i\n`

  `Complex␣number␣2:␣`**`a2`**`␣+␣`**`b2`**`i\n`

  where `a1` and `b1`, and `a2` and `b2` are the values of the members a and b of the complex numbers. ␣denotes a single space character.

Use two `cout` statements in order (the first `cout` displays the `complex number 1` while the second displays the `complex number 2`). Note that your program should not use `printf()` when displaying the values of the complex numbers.

Save the program as `activity3.cpp`. If you are happy with the program, submit it to the Assessment System for marking. Click on **Run checks** to initiate auto-marking.