NWEN 241 Exercise 4

System calls and Introduction to C++

Release Date: **4 May 2021**

Submission Deadline: **14 May 2021, 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_2021T1/RemoteStudyGuides`) and follow one that suits you the most.

Exercises

You may download a copy of the base source files used in the activities from `https://ecs.wgtn.ac.nz/foswiki/pub/Courses/NWEN241_2021T1/Exercises/nwen241_exercise4_files.zip`.

Activity 1: Process Management [30 Marks]

Copy and paste the following C program[1] to your favorite text editor:

```

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

int main()
{
    int pid, ret, sta;
    // 1) Call fork() and assign its return value to pid

    switch (pid) {

        case -1:
        if (pid<0) {
            printf("Error\n");
            // 2) Call exit() with status value of 1
        }
        break;

        case 0:
        // 3) Use execl() to execute command ps -A, and assign its return value to ret
        printf ("Error executing exec\n");
        break;

        default:
        // 4) Call wait(), use sta variable to store status information, and assign return value to pid

        // 5) If WIFEXITED(sta) is set, print parent process id, child
```

[1] You can extract a copy of this file from `https://ecs.wgtn.ac.nz/foswiki/pub/Courses/NWEN241_2021T1/Exercises/nwen241_exercise4_files.zip`.

```
           process id and termination status of child process.
        //    Do this by replacing the comments below with the
           appropriate expressions.
        if (/* expression to check whether WIFEXITED(sta) is set */) {
            printf("%d %d %d\n", /* parent process id*/, /* child
               process id */, /* termination status of child process
               */);
        }
        break;
    }
}
```

Save the file as `activity1.c`. Study the source file and complete the program by adding suitable code segments in the appropriate places as indicated by the numbered comments:

1. Call `fork()` and assign its return value to the variable `pid`.
2. Call `exit()` with status value of 1.
3. Use `execl()` to execute the command `ps -A`, and assign its return value to the variable `ret`.
4. Call `wait()`, use `sta` variable to store status information, and assign its return value to the variable `pid`.
5. If `WIFEXITED(sta)` is set, print parent process id, child process id and termination status of child process.

Compile and run the program. If you are happy with the program, submit it to the Assessment System for 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. You can use netcat (the `nc` command in Linux) as the client.

To illustrate how the server works, run `nc localhost 23456` to connect to your server (make sure that you are running the server program in another terminal before you run `nc`). Suppose you send a string `"Hello"` to the server; the server should return `"olleH"` 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.

Remarks about the testing: The testing procedure for this activity is similar to the testing procedure in Assignment 3. You will need two terminals: one to run the server program, and the other one to run `nc`.

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

In this activity, you will write a C++ program to represent complex numbers[2] 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:

- 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.

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.

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