Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming
Laboratory Session Week_10

Learning Objectives

- To produce examples of fork() usage
- To produce examples of signal handling
- To become familiar with threads

Exercises

You should attempt the exercises below by using only the C constructs that you learnt up to teaching week 11, and:

1. Write pseudo code to describe the required algorithm to solve the exercise (or draw up a flowchart), before writing and testing the actual code.
2. Add comments to your code.
3. Make your code neat, by using indentation and parenthesis (where appropriate).
4. Give meaningful names to functions and variables.

Exercise 1

Write a program that performs a `fork()` and then invokes the ***runChild*** function (given in **Figure 1**) in the child process, and five seconds later, the parent prints a message saying, **"I'm still here!"** and the **process id** of both parent and child processes. Use the ***sleep*** library function to put the process to sleep for five seconds.

```
void runChild (void)
{
   printf("\n I am the child! My pid is %d\n",getpid());

}
```

Figure 1

Exercise 2

Take the program you wrote in Exercise **1** and instead of sleeping for 5 seconds **make the parent process simply wait for the child to terminate and print the exit status of the child.**

Exercise 3

Reuse the code you wrote for Exercise 2, and change the ***runChild*** method so that the child process execute the **"ls -aF /"** command using `execlp()` function.

Exercise 4

The function given in **Figure 2** tests whether a word is a *Palindrome*. Examples of palindromes are: *did, mum, level,* etc. Write a program that tests the user capabilities to find palindromes. The program should ask the user to enter a palindrome of 3 letters first, if the user gives the correct answer in 20 seconds the program continues by asking the user to enter a palindrome of 4 letters in 30 seconds and so on (5 letters in 40 seconds, 6 letters in 50 seconds, etc.). If the user gives an incorrect answer or the waiting time expires (displaying "TIME IS UP!!!"), the program terminates and displays the final score (how many correct answers in total). If the user presses ***Ctrl C*** the program also terminates and displays the final score. The program must make use of signals. **Hint:** See the example given in class for signals using alarms.

```
#include<stdio.h>
#include<string.h>

void testPalindrome(int len)
{
      char s[len];
      int i,j;
      printf("Enter a word of length %d:\n",len);
      fflush(stdin);
     gets(s);
     for (i=0,j=len-1;i<j;i++,j--){
           if (s[i]!=s[j]){
            printf("\nNot a Palindrome\n");
           raise(SIGINT);
           break;
     }
     else
           printf("\nPalindrome\n");
      }
}
```

Figure 2

Exercise 5

Write a program including **15 threads** where **each thread increments** a *global* variable called ***count*** from 1 to 100 in units of 1 (the expected result would be 100*15). The main program then prints out the final value of the variable **count**. Verify if the result is what you expected. Now, change your program so that each thread now increments the variable **count** from 1 to 1000000 (the expected result would be 1000000*15). What is the actual output value of the program? Explain the reason for the obtained value in the output.

Exercise 6

Rewrite the program to overcome the issue in the Exercise 5 (Threads LN2).

ECS501U – END of LAB Week_10