Department of Computer Science
The University of Hong Kong
COMP3230B: Principles of Operating Systems
Assignment 1
Due Date: 23:59, Oct. 13, 2016

Objectives

- Practice to write a multiprocess parallel program with system calls *fork*, *exec, getpid*
- Grasp the sense of process memory space and inter-process communication

Specifications

You need to write ***tmem.c*** based on the given template ***tmem_template.c***. Follow the **ToDo** items in the template strictly.

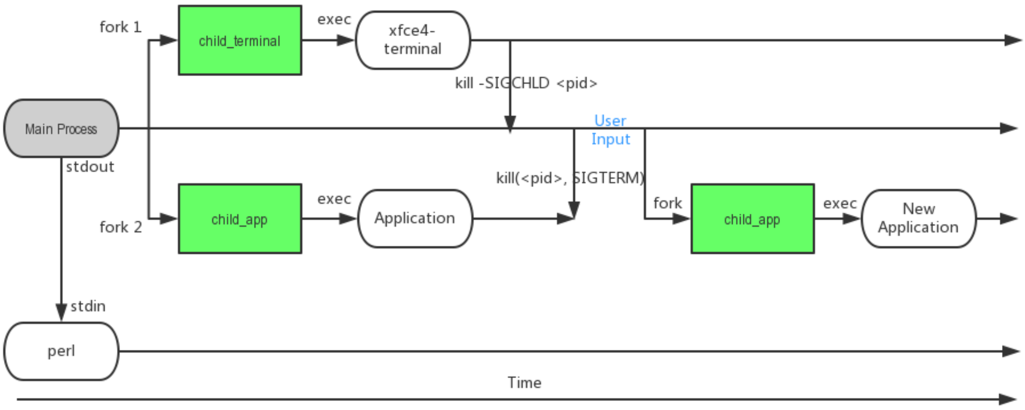
This assignment extends the function of *tmem* [1], a program that monitors the memory usage of a **target application**. The resulting program creates one more child process that becomes an **xfce4-terminal** (the terminal on X2Go) to ease the control, and allows users to change to another target application upon receiving the **SIGCHLD** signal not from children.

If you are not programming on X2Go, you can try gnome-terminal or look for what the terminal is on your machine. However, make sure your final submission uses xfce4-terminal.

The stdout of *tmem* will be passed to the stdin of ***driveGnuPlots.pl*** ([2], a perl script that plots the memory readings) with a pipe (i.e. |). If you are not running the program on X2Go, you should install gnuplot and gnuplot-x11 as demonstrated in Workshop 2. The program will be called as shown below:

```
./tmem <app_path> | perl driveGnuPlots.pl 4 400 400 400 400 vmsize vmdata vmstk vmRSS
```

Here is a graph that shows the interaction between the processes.



The program first creates 2 child processes, one becoming a terminal and one becoming the target application. The main process will then scan the directory */proc/pid/status* of the application process continuously, and output them to stdout. In this assignment, *tmem* is configured to read 4 numbers, VmSize, VmData, VmStk, and VmRSS.

VmSize:	Current virtual memory usage
VmData:	Size of "data" segment (data+bss+heap)
VmStk:	Size of stack
VmRSS	Resident set size

Users are expected to use the new terminal to send SIGCHLD to the main process:

```
$ kill -SIGCHLD <pid>
```

Upon receiving SIGCHLD not from children, the main process will prompt to let user input the path to the new target application. Then it will create a new child process to become the new application (by *execlp*), while terminating the original application. The **monitoring shall proceed** normally.

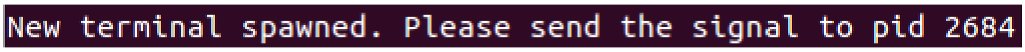
Tasks to Be Done

Please download the folder called *tmem* from Moodle. You can test *tmem* with the 2 given programs, ***malloc.c*** and ***sum.c***. Ubuntu built-in applications such as Firefox can also be monitored. There is also a pseudo-app embedded **null()**, which can be tested by inputting app_path as "null".

Task 1 New Terminal (30%)

Create a child process and then use the **execlp** system call. The path to the xfce4-terminal should be ***/usr/bin/xfce4-terminal***. You can verify this with the *whereis* command.

The parent process should then display the following message:



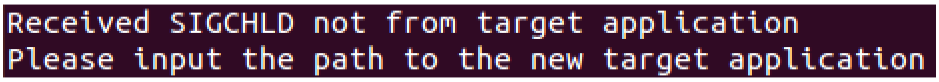
```
New terminal spawned. Please send the signal to pid 2684
```

The displayed pid belongs to the **main process**. **Beware** that the stdout has been passed to the perl script, so we need to use **stderr** to display on the console (**throughout the program**):

```
fprintf(stderr, <format>, var1, var2, ...)
```

Task 2 Replace Application (40%)

In the SIGCHLD handler **sig_chld()**, we can distinguish the source of the signal. When the signal is not from the children, display the following message:



```
Received SIGCHLD not from target application
Please input the path to the new target application
```

The program should then read the inputs from the console (the **original** one). You can assume that the new application does not take command line arguments. Make sure your program can switch among *./malloc*, *null* and *firefox*.

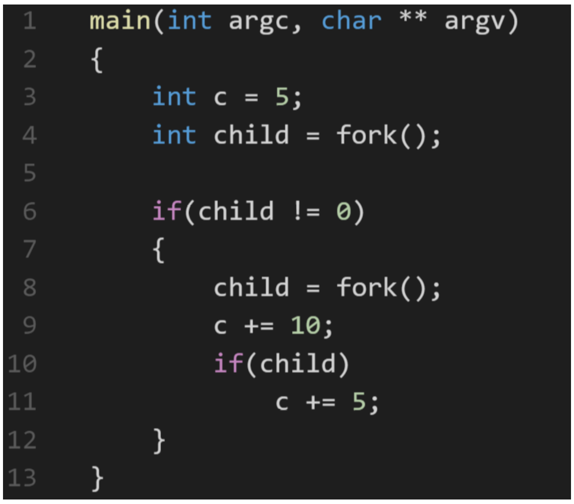
Hint: The first 2 arguments of **execlp()** can be the same string.

The program should then create a new child process that becomes the new application, while terminating the old application by *kill(<pid>, SIGTERM)*.

Task 3 Analysis (30%)

Analyze and answer the following questions in a **Word** file named with your UID. You may include any screenshot that can help you convey your ideas.

a) Given the following piece of code:



```
main(int argc, char ** argv)
{
    int c = 5;
    int child = fork();

    if(child != 0)
    {
        child = fork();
        c += 10;
        if(child)
            c += 5;
    }
}
```

How many copies of variable *c* are there at the end of the program? What are their values? (**10%**)

b) Show the curves of the given program ***malloc.c***. Discuss and explain the patterns of the curves. (**10%**)

c) The pseudo-app *null* makes modifications to the filename buffer after the sleeping for one second. Why does the VmRSS reading increase while the other 3 readings remain the same? (**10%**)

Submission

Please upload *tmem.c* and the Word file to Moodle on or before Oct. 13, 2016.

Reference

[1] tmem: http://locklessinc.com/articles/memory_usage/

[2] driveGnuPlots.pl: http://users.softlab.ntua.gr/~ttsiod/gnuplotStreaming.html