COMP 304- Operating Systems: Assignment 1

Due: 2 March 2018, 6.00 pm

Notes: This is an individual assignment. Any forms of cheating will be harshly punished! No late assignment will be accepted. Submit your answers through blackboard. This assignment is worth 4% of your total grade.
Corresponding TA: Najeeb Ahmad (nahmad16ku.edu.tr)

Problem 1

(15 points) Provide concise answers (2-3 sentences). Use your own words. We will deduct credits for unnecessarily long answers.
a) Explain the purpose of system calls. Why would a programmer prefer using an API rather than directly invoking system calls?
b) What are the advantages of using loadable kernel modules?
c) In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.
- What are the advantages of a multiprogramming and time-sharing environment?
- What kind of mechanisms are used to overcome security problems?

Problem 2

(10+10 points)
a) Write a C program on Linux that forks a child process that immediately becomes a zombie process. This zombie process must remain in the system for at least 5 seconds. A zombie process is created when a process terminates but its parent does not invoke wait() immediately. By not invoking wait(), the child maintains its PID and an entry in the process entry table.
b) Write a C program on Linux that a process becomes orphan. This orphan process must remain in the system for at least 5 seconds. A process becomes an orphan when its parent terminates before the process terminates.
For these two programs;

- Use the sleep() system call API for the time requirement.
- Run these programs in the background (using the & parameter) and then run the command **ps -l** to determine whether the child is a zombie or orphan process.
- The process states are shown below the S column; processes with a state of Z are zombies in the ps command output.

- The process identifier (PID) of the child process is listed in the PID column, and that of the parent is listed in the PPID column.
- Remember that orphan processes are adopted by the *init* process.
- Verify that you can locate the zombie and orphan processes with the ps command.
- You don't want to create too many zombies. If you need, you can kill the parent process with **kill -9 PID**.
- Provide the source code (only .c) and the screen shots of the ps commands.

Problem 3

(30 points) In this program you will use both pipes and shared memory objects for interprocess communication.
Write a C program on Linux that takes two parameters; input filename and output filename. The program should then perform the following steps:

- The program starts as a single process which reads the content of the input file.
- Parent forks a child process A and sends the content of the input file to this child using an ordinary pipe.
- The child process A receives the content of the file, creates a shared memory object and writes the content of the file into this object. Then it sends 'I am done' message as an acknowledgment to parent process using an ordinary pipe.
- On receiving the acknowledgment message, the parent creates another child B.
- The process B reads the data from the shared memory object and writes it into the output file. Then this second child sends 'I am done' message as an acknowledgment to the parent again using an ordinary pipe.
- On receiving the acknowledgment, parent process terminates.
- After the execution of the program, the contents of input and output files should be the same.

You are required to submit your .c source code file and a snapshot of a sample run on your terminal. You may want to refer to the ordinary pipe example and shared memory object example in the textbook.
How to compile and execute the program:

```
1 bash-3.2$ gcc -lrt problem3.c -o problem3  //need to compile with -lrt
2 bash-3.2$ ./problem3 inputfile.txt outputfile.txt
```

Problem 4

(10+25 points) In this question, you will learn how to create a kernel module and load it into the Linux kernel. Note that you need to be a superuser on the computer for this assignment.

a) Read Linux Kernel Modules under the Programming Projects from the book in Chapter 2 (Page 120 also provided in Blackboard). Perform Part I and Part II of the assignment. You can arbitrarily set the day, month, year variables. Use the makefile and template program provided as a starting point.

b) Design a separate kernel module that outputs the following characteristics of the processes:
- PID (process ID)
- its parent
- executable name,
- its sibling list (their process ids and executable names),

You need to read through the *task_struct* structure in $< linux/sched.h >$ to obtain necessary information about a process. The kernel module should take an PID as an argument. If no process ID is provided or the process ID is invalid, print an error message to kernel log.

Some Useful References:
- Info about task link list (scroll down to Process Family Tree):
http://www.informit.com/articles/article.aspx?p=368650
- Linux Cross Reference:
http://lxr.free-electrons.com/source/include/linux/sched.h
- You can use the **pstree** command to check if the sibling list is correct. Use -p to list the processes with their PIDs.
- Refer this website on how to pass arguments to a kernel module:
http://www.tldp.org/LDP/lkmpg/2.6/html/x323.html

******** You are required to submit your .c source code files and a snapshot of a sample run on your terminal for all programming assignments. We will deduct points for those who submit executables or object files. *********