CS344: Operating Systems Lab

Lab # 02 (1 Questions, 50 Points)
Held on 29-Aug-2023
Lab Timings: 09:00 to 12:00 Hours Pages: 4
Submission: 12:00 Hrs, 29-Aug-2023
Instructor Dr. V. Vijaya saradhi
Head TAs Adithya Moorthy & Laxita Agrawal
Department of CSE, IIT Guwahati

Submission instructions:

Single file Question 1 (a) should be implemented in one single file. Implement using C language.

2.5 marks Write a shell program: `compile-1a-and-execute.sh` which is responsible for

a. Compiling question 1 (a)

b. At the time of compiling, name the executable by prefixing your roll number. Example, `210101999-q1a` corresponds to executable file of roll number 210101999 for question 1 (a).

c. Unset any previously declared environment variables.

d. Set appropriate environment variables as necessary.

e. Run the executable.

Single file Question 1 (b) should be implemented in one single file. Implement using C language.

2.5 marks Write a shell program: `compile-1b-and-execute.sh` which is responsible for

a. Compiling question 1 (b)

b. At the time of compiling, name the executable by prefixing your roll number. Example, `210101999-q1b` corresponds to executable file of roll number 210101999 for question 1 (b).

c. Unset any previously declared environment variables.

d. Set appropriate environment variables as necessary.

e. Run the executable.

Single file Question 1 (c) should be implemented in one single file. Implement using C language.

2.5 marks Write a shell program: `compile-1c-and-execute.sh` which is responsible for

a. Compiling sub-question 1 (c)

b. At the time of compiling, name the executable by prefixing your roll number. Example, `210101999-q1c` corresponds to executable file of roll number 210101999 for question 1 (c).

c. Unset any previously declared environment variables.

d. Set appropriate environment variables as necessary.

e. Run the executable.

Single file Question 1 (d) should be implemented in one single file. Implement using C language.

2.5 marks Write a shell program: `compile-1d-and-execute.sh` which is responsible for

a. Compiling question 1 (d)

b. At the time of compiling, name the executable by prefixing your roll number. Example, `210101999-q1d` corresponds to executable file of roll number 210101999 for question 1 (d).

c. Unset any previously declared environment variables.

d. Set appropriate environment variables as necessary.

e. Run the executable.

The TA will only execute the shell script file `compile-1a-and-execute.sh` `compile-1b-and-execute.sh` `compile-1c-and-execute.sh` OR `compile-1d-and-execute.sh`

Zip file Create a zip file containing four C program files corresponding to questions 1 (a), 1 (b), 1 (c), 1 (d) and the shell script files `compile-1a-and-execute.sh`, `compile-1b-and-execute.sh`, `compile-1c-and-execute.sh` and `compile-1d-and-execute.sh`.

Naming Name the zip file with your roll number.

Upload the zip file.

a. This assignment is based on chapter 3, `Process Management` in the book `Operating System Principles, Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne`.

b. In order to perform this assignment, understanding of system calls `fork(), wait(), exit(), getenv(), execle(), execlp(), execl()` are essential.

c. Carefully read the manual pages for the above system calls.

d. Example program in chapter 3 covers `fork, wait, waitpid, execve & exit`

Question 1: (50 points)
Program execution: Implement the following:

a. (5 marks) Write a 1^{st} C program that takes n as command line argument and generates sequence of numbers using the following equation (stop sequence generation when $n = 1$)

$$n = \begin{cases} \frac{n}{2} & \texttt{if n is even} \\ \\ 3 \times n + 1 & \texttt{if n is odd} \end{cases} \tag{1}$$

i. The input n will be given through command line argument. If no argument is presented, assume a default value of 100.

ii. If more than one argument is passed to this program, you should compute the above function for each value passed. Examples listed below

No arguments `./a.out`. In this case, compute the sequence for $n = 100$.

One argument `./a.out 35`. In this, compute the sequence for $n = 35$.

More than one argument `./a.out 35 47`. In this, compute the sequence for $n = 35$ and sequence for $n = 47$.

b. (5 marks) Write a 2^{nd} C program that reads the shell environment variable n and computes the sequence described in 1^{st} program. This program should NOT take any command line arguments.

i. To set an environment variable in shell use `export n="10"`

ii. Read the shell environment variable n. When variable n is not set, then the sequence should be computed using default value 100.

iii. When more than one number is assigned to environment variable, sequence should be computed for each number in the environment variable. That is `export n="10 20 30"` then sequence should be computed for $n = 10$, $n = 20$ and $n = 30$ respectively.

c. (15 marks) **Process creation and execution** Write a 3^{rd} C program in which the main function creates three child processes. Immediately after creating the children, their process image should be over-written with the executable of

i. 1^{st} program with no arguments.

ii. 1^{st} program with one argument.

iii. 1^{st} program with ten arguments.

Demonstrate the process memory over-writing using the following system calls:

i. (5 marks) `execle`

ii. (5 marks) `execl`

iii. (5 marks) `execve`

iv. Perform the above three system calls by creating three child processes one for each of the above system call and over-writing the child image with executable of 1^{st} C program.

d. (15 marks) **Process creation and execution** Write a 4^{rd} C program in which the main function creates three children. Immediately after creating the children, their process image should be over-written with the executable of

i. 2^{nd} program with environment variable not set
ii. 2^{nd} program with environment variable is assigned one value
iii. 2^{nd} program with environment variable is assigned 10 values
iv. Important note: While implementing this question, you should not pass any arguments to the executable. The environment variables should be used to pass the appropriate arguments.

Demonstrate the process memory over-writing using the following system calls:

i. (5 marks) `execle`
ii. (5 marks) `execl`
iii. (5 marks) `execve`
iv. Perform the above three system calls by creating three child processes one for each of the above system call.