CS 377: Operating Systems Lab

Assignment No. 1: 10 January 2014

(*Note:* A process is an execution of a program.)

1. Answer the following questions using the proc file system. Also mention the name of the file of proc file system which contains the answer.

   (a) What is the cache size, address sizes (physical and logical) of any processor.

   (b) What is the limit on the number of open files of a process? (Execute q3.c to find it.)

   (c) Write a 8-10 line paragraph **in your own words** explaining process-preemption. (Read linux literature to find relevent information)

2. A process executes in the *user mode* most of the time and in the *kernel mode* some of the time. A simple explanation of these modes is as follows: A *privileged instruction* is an instruction in the CPU which a user process is not allowed to execute. If a process contains some privileged instructions, the kernel executes them on behalf of the process. During this time, the process is said to be in *kernel mode*. The process also executes in the kernel mode when it makes a *system call*.

   Consider the given two programs q2_1.c and q2_2.c. Compile and execute the programs using following commands:

   ```
   gcc q2_1.c -o first
   gcc q2_2.c -lm -o second
   ./first > /dev/null
   ./second > /dev/null
   ```

   Use proc filesystem to find out the time spent by each of the above two processes in kernel mode and user mode by running both the processes for 30 seconds. Observe the values of the times spent in the kernel and user modes. Study the programs and give a logical explanation about the relative magnitudes of these times.

   P.S. End the process's by (Ctrl+c) after completion.

3. A *context switch* occurs when the OS switches the CPU from execution of one process to execution of another process. A *voluntary context switch* occurs when a process voluntarily yields the CPU (by explicitly calling sleep, or waiting for an event or some resource to become available); whereas an *involuntary context switch* occurs when the OS scheduler decides to execute some other process.

   (a) You have been given a program q3.c. Compile the same program twice to produce two executables as follows:

   ```
   gcc q3.c -o first
   gcc q3.c -o second
   ```

   Execute these programs with different scheduling priorities as follows:

   ```
   nice -n +5 ./first output_file1
   nice -n +11 ./second output_file2
   ```

   where output_file1 and output_file2 are command line arguments giving names of their respective output files.

      i. Count the no of a's in both the files after 30 seconds.
      ii. Observe the CPU percentage consumed by these processes.
      iii. Observe the values of voluntary and involuntary context switches. (You can get them from status file of the proc filesystem.)

   (b) Change the priority of process `first` as `renice +20<PID of the process>` (You can get the pid of the process by the `ps -A` command.) Answer questions (i)–(iii) of part (a) for the new priorities.

4. Consider the program q4.c, which reads from the standard input file and writes to the standard output file. It accepts the buffer size, i.e., number of characters to be read at a time, as a command line argument.

   (a) Create the executable of p2 and run it as
   `time ./p2< file_name> /dev/null number_of_bytes`
   where the file `file_name` is any file larger than 8 MB, and `number_of_bytes` is a command line argument indicating the buffer size. Vary the buffer size from 1 to as large as file size in at least 10 steps. Obseve the buffer size and the user, system, and real times

of the process. (*Hint:* Use the `time` command to get the user time, system time and real time spent in executing the process given as an argument.)

(b) In the command of part (a), instead of `/dev/null` redirect the output to another file as follows:

`time ./p2< file_name> output_file number_of_bytes`

where `output_file` is the name of the output file. Repeat the above observations for this case.

Compare the results with those of the first part and explain the reason for their difference.

5. The operating system uses the notion of a *state* to represent the current activity in a process and changes the state as the activity in the process changes.

   You are given a program q5.c. The process that represents its execution spawns three *child processes* by using the `fork` command during its execution. The process that spawns other processes is called the *parent process*. You will notice that the three child processes execute different functions in q5.c.

   (a) From Linux literature, find the meaning of the various states.

   (b) Now study the program q5.c and code the functions executed by the three child processes such that one of them will be in the zombie state, one will be in the sleep state and the third one will be in some other state.

   (c) Submit your program.

   (d) How can you get your process in traced or stopped (T) state?

6. (a) The parent-child relationships between processes can be represented in the form of a *process tree* with branches pointing from a parent to its children. The "pstree" command is useful for printing a tree of all running processes. There are various options to control the output of the command (see man page). For parts i and ii, mention the exact commands (with arguments) that you executed to get the answer.

      i. List all the processes started by root.

      ii. List all the processes started by a user (any one user is fine).

iii. Is the list from (ii) a subset of the list from (i)? If yes, explain why. (Hint: read up about the "init" process.)

(b) run the following commands (in q3 folder):

- `gcc q3.c -o q_6b`
- `./q_6b output_6b`

Use the man page of pstree to find out all the ancestors of this running process.
(Make sure you delete output_6b file after completing this part)

7. (a) Use the proc file system to find the current load average of the system.

   (b) Run the program q7.c.

   (c) Use the proc file system to again find the current load average of the system.

   (d) Read about load averages. Analyze and reason the difference between the load averages.

   (e) Modify program q7.c so that you obtain a significant difference in load averages. Explain how you achieved it.