February 13, 2020

LAB 1 - CONTEXT SWAPPING

CS 4375 - OS

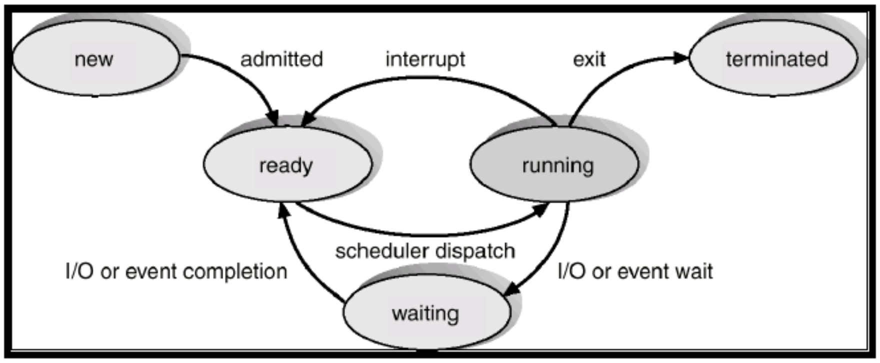
Dilan Ramirez
The University of Texas at El Paso

Table of Contents

Introduction

The main purpose of Lab 1 – Context Swapping is to measure the cost, in this case, the time it takes for the operative system to swap processes. According to tutorialspoint.com, context swap process means, “It involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier.” The steps to do it are the next:

1. Save the context of the process which is running on the CPU.
2. Change the process state to blocked and store it in the queue.
3. Select a new process from the ready list to execute.
4. Update the process state to running.
5. Update memory as required.
6. Restore the context of the process which was previously running after it is loaded again on the CPU.
7. Repeat.



A process runs on the CPU until it is swapped to start another process. Swapping can occur for many reasons. Among these reasons are the process has used its timeslice, the process needs to use some other resource not available, the process state has changed, etc.

Methods

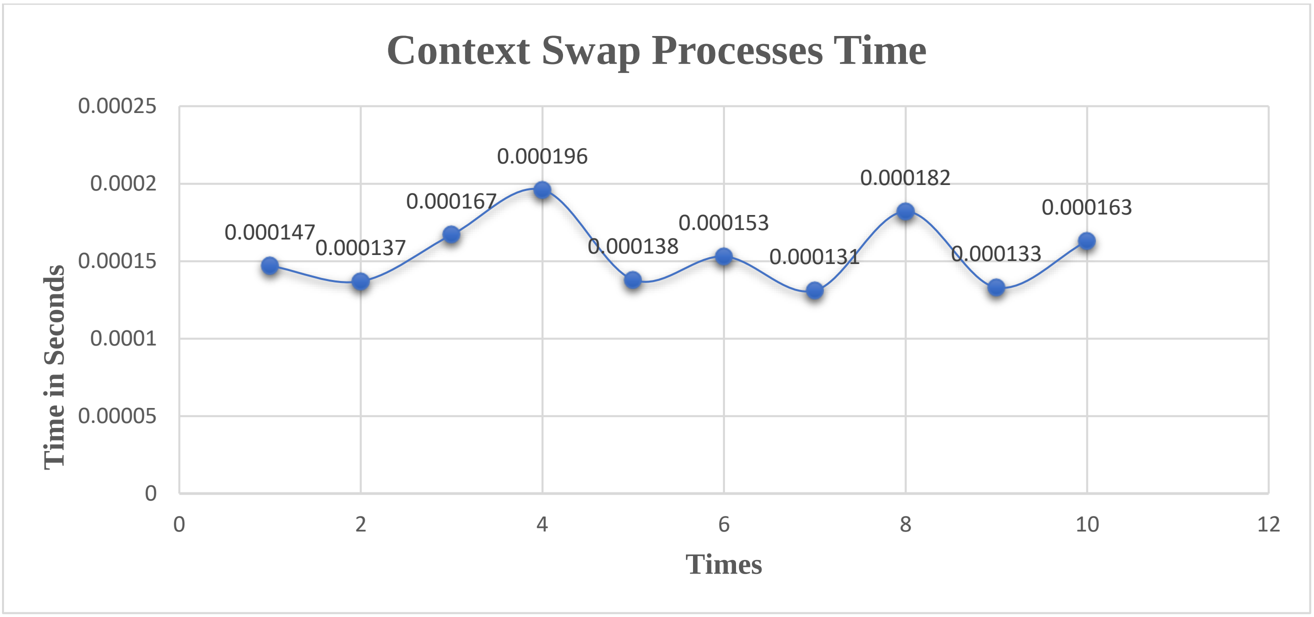
I will describe all the functions that my code uses to measure the context swap time between both processes. To start, I coded a function called startTime() using the library time.h which starts a timer. Then, I coded a function called ednTime() which takes as parameter the start time to then calculate the total time as a result. This function returns the total time. In the context of swapping, there are some problems. One of these is that a computer can have more than one CPU and the CPU can have more that one core. As a result, one of the processes will never get blocked because it can run in another core or CPU. To avoid that, I used my two functions called printCPUs() and setOneCPU(). printCPUs() uses the method sched_getaffinity() and sysconf () to go through the system and recognize the number of cores my computer has. I used the function setCPUs() to set all the processes to run in only one core, avoiding the problem mentioned before. I use CPU_SET(0, &mask) to set all the processes to run in the core number 0.

Once the CPU is configured to avoid all these problems, I coded my function contextSwapping() to show how two processes communicate with each other through a pipe. First, I used the fork method which split one process into two. It creates a copy of itself. Both processes are known as parent process and child process. Then, I used the pipe method to concatenate a string into another through the pipes. I checked if both pipes did not return -1 which means that pipes failed. After that, I check that fork is not less than 0. If fork returns a value less than 0, it means that the fork method has failed. After everything is working correctly, I start the two processes, parent and child. If the fork returns a value greater than 0, it means that the process is the parent process. I have two processes, two pipes. Process A reads from pipe 0 and writes to pipe1. Process B reads form pipe 1 and writes to pipe 0. This is the moment when the context swap occurs. After the process parent writes, it gets blocked and waits for process child to read pipe 0. I start the timer just before the process parent gets blocked waiting for the process child, and before the child process starts to read. After the process child starts to read and then write. I closed both pipes 0 to then write the concatenated string. It returns the total time it took the context swap.

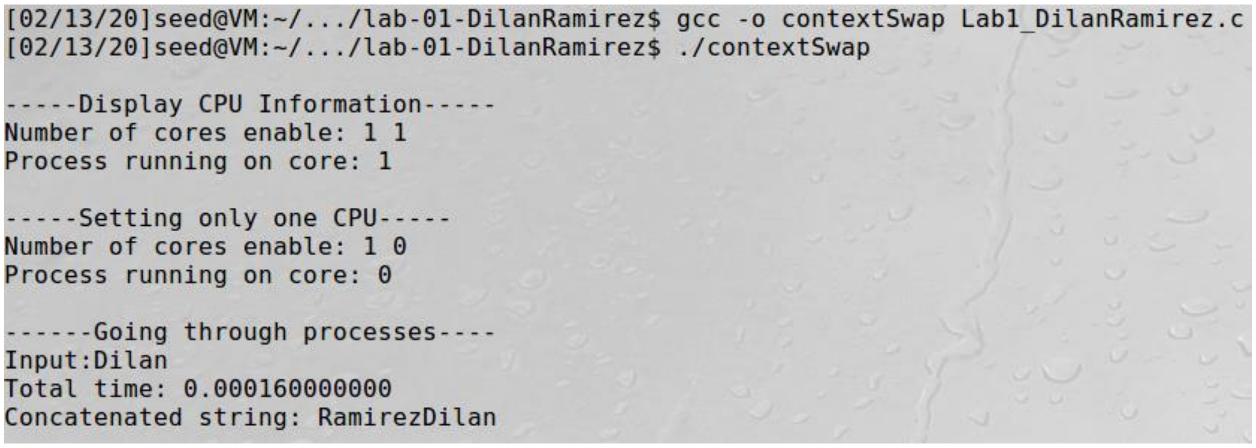
Results

After ran my code 10 times, I got the same result shown in the below graph1 with the next equation y = 7E-08x + 0.0002. The next picture 1 is a screenshot of the output of my code. I only used one core of my CPU. Therefore, the time I got was the context swap time when one process gets blocked by the CPU and starts the other process.

Times	1	2	3	4	5	6	7	8	9	10	Avg
Time sec	0.000147	0.000137	0.000167	0.000196	0.000138	0.000153	0.000131	0.000182	0.000133	0.000163	0.000157



Graph 1



```
[02/13/20]seed@VM:~/.../lab-01-DilanRamirez$ gcc -o contextSwap Lab1_DilanRamirez.c
[02/13/20]seed@VM:~/.../lab-01-DilanRamirez$ ./contextSwap

-----Display CPU Information-----
Number of cores enable: 1 1
Process running on core: 1

-----Setting only one CPU-----
Number of cores enable: 1 0
Process running on core: 0

------Going through processes----
Input:Dilan
Total time: 0.000160000000
Concatenated string: RamirezDilan
```

Picture 1

$\mathbf{X}$	$\bar{\mathbf{x}}$	$\mathbf{X} - \bar{\mathbf{x}}$	$(\mathbf{x} - \bar{\mathbf{x}})^2$
0.000147	0.000157	-0.00001	0.0000000001
0.000137	0.000157	-0.00002	0.0000000004
0.000167	0.000157	0.00001	0.0000000001
0.000196	0.000157	0.000039	0.000000001521
0.000138	0.000157	-0.000039	0.000000001521
0.000153	0.000157	-0.000004	0. 0000000016
0.000131	0.000157	-0.000026	0.000000000676
0.000182	0.000157	0.000025	0.000000000625
0.000133	0.000157	-0.000024	0.000000000576
0.000163	0.000157	0.000006	0.000000000036

$$s^2 = \frac{\sum(\mathbf{X} - \bar{\mathbf{x}})^2}{n - 1} \qquad s^2 = \frac{0.000000007155}{9} \qquad s = \mathbf{2.819574435974337}\boldsymbol{e} - \mathbf{5}$$

S = 0.02819574435 milliseconds.

Discussion

After examining and analyzing the previous results, I found out that the time it takes to the OS to swap between processes after one process gets blocked is very low. The time I tracked might not be accurate because there could be other factors in between. However, it could show that the time is very short and this time exists. The variance factor calculates in “s” shows how much these outputs differ from the expected value. This values shows how far this set of time is spread out from the mean. The time is very short because I can possibly say that the CPU is optimized to handle the context switching of a process. The OS finds out that for some reason, a process got blocked, so it takes another process in the ready queue and executes it.

In conclusion, I can say that I learned how the context swap of processes is done behind the scene. I understood what the steps are to swap a process. Besides, I understood the diagram flow of the context swap of a process. It was a very useful lab. It was a challenge to understand many new definitions that were involved not only with context swapping but also with other things too, but it worked.

Citation

“What Is Context Switching in Operating System?” *What Is Context Switching in Operating System?*, Tutorialspoint.com, www.tutorialspoint.com/what-is-context-switching-in-operating-system.