Tutorial 2 Answer Sheet

CS3103 Operating Systems

Student Name: LIU Hengche

Student No. : 5 7 8 5 4 3 2 9

Day: ☐Tuesday ☑Wednesday ☐Friday

Time: ☐ 10:00 - 10:50 ☐ 11:00 - 11:50 ☐ 16:00 - 16:50 ☑ 18:00 - 18:50

The Process

Submission:

• Deadline: Sunday, February 25, 2024, 23:59 pm HKT.

• Answers are allowed in text only. Any form of image/snapshot is not allowed.

• Submit this answer sheet via Canvas->Assignments->Tutorial 2.

Questions

Question 1: Run `process-run.py` with the following flags: `-l 5:100,5:100`. What should the CPU utilization be (e.g., the percent of time the CPU is in use?) Why do you know this? Use the `-c` and `-p` flags to see if you were right.

Answer:

The CPU utilization should be 100% because the flag generates 2 process with 5 instructions and the instructions are all CPU instructions. So the CPU is always busy, then the percent of time the CPU is in use is 100%. Below is the output with the -c and -p flags:

hengchliu2@ubt20a:/public/cs3103/tutorial2$ python2 ./process-run.py -l 5:100,5:100 -c -p

Time	PID: 0	PID: 1	CPU	IOs
1	RUN:cpu	READY	1	
2	RUN:cpu	READY	1	
3	RUN:cpu	READY	1	
4	RUN:cpu	READY	1	
5	RUN:cpu	READY	1	
6	DONE	RUN:cpu	1	
7	DONE	RUN:cpu	1	
8	DONE	RUN:cpu	1	
9	DONE	RUN:cpu	1	
10	DONE	RUN:cpu	1	

Stats: Total Time 10

Stats: CPU Busy 10 (100.00%)

Stats: IO Busy 0 (0.00%)

Question 2: Now run with these flags: `-l 4:100,1:0`. These flags specify one process with 4 instructions (all to use the CPU), and one that simply issues an I/O and waits for it to be done. How long does it take to complete both processes? Use `-c` and `-p` to find out if you were right.

Answer:

It takes 6 times to complete both processes. I am not correct because the default IO-length is 5 instead of 1.

It takes 10 times to complete both processes. Below is the output:

Time	PID: 0	PID: 1	CPU	IOs
1	RUN:cpu	READY	1	
2	RUN:cpu	READY	1	
3	RUN:cpu	READY	1	
4	RUN:cpu	READY	1	
5	DONE	RUN:io-start	1	
6	DONE	WAITING		1
7	DONE	WAITING		1
8	DONE	WAITING		1
9	DONE	WAITING		1
10*	DONE	DONE		

Stats: Total Time 10

Stats: CPU Busy 5 (50.00%)

Stats: IO Busy 4 (40.00%)

Question 3: Switch the order of the processes: `-l 1:0,4:100`. What happens now? Does switching the order matter? Why? (As always, use `-c` and `-p` to see if you were right)

Answer:

Now the total time becomes 4. The order switching matters. Because now the first process is IO operation, then when the IO is waiting, we could switch to the other process so that the CPU could be in use for the second process. Below is the output:

Time	PID: 0	PID: 1	CPU	IOs
1	RUN:io-start	READY	1	
2	WAITING	RUN:cpu	1	1
3	WAITING	RUN:cpu	1	1
4	WAITING	RUN:cpu	1	1
5	WAITING	RUN:cpu	1	1
6*	DONE	DONE		

Stats: Total Time 6

Stats: CPU Busy 5 (83.33%)

Stats: IO Busy 4 (66.67%)

Question 4: We'll now explore some of the other flags. One important flag is `-S`, which determines how the system reacts when a process issues an I/O. With the flag set to `SWITCH_ON_END`, the system will **NOT** switch to another process while one is doing I/O, instead waiting until the process is completely finished. What happens when you run the following two processes (`-l 1:0,4:100 -c -S SWITCH_ON_END`), one doing I/O and the other doing CPU work?

Answer:

Now the time is 9. Because the system will not switch to another process while one is doing I/O, then the CPU instructions could only be done if the first process(IO) is done. So now the total time is 9. Below is the output:

Time	PID: 0	PID: 1	CPU	IOs
1	RUN:io-start	READY	1	
2	WAITING	READY		1
3	WAITING	READY		1
4	WAITING	READY		1
5	WAITING	READY		1
6*	DONE	RUN:cpu	1	
7	DONE	RUN:cpu	1	
8	DONE	RUN:cpu	1	
9	DONE	RUN:cpu	1	

Stats: Total Time 9

Stats: CPU Busy 5 (55.56%)

Stats: IO Busy 4 (44.44%)

Question 5: Now, run the same processes, but with the switching behavior set to switch to another process whenever one is **WAITING** for I/O (`-l 1:0,4:100 -c -S SWITCH_ON_IO`). What happens now? Use `-c` and `-p` to confirm that you are right.

Answer:

Now the total time is 6 again. By default, it is -s SWITCH_ON_IO meaning that the system could change to another process while one is doing IO. So during the waiting time, the CPU instructions could be done. Below is the output:

Time	PID: 0	PID: 1	CPU	IOs
1	RUN:io-start	READY	1	
2	WAITING	RUN:cpu	1	1
3	WAITING	RUN:cpu	1	1
4	WAITING	RUN:cpu	1	1
5	WAITING	RUN:cpu	1	1
6*	DONE	DONE		

Stats: Total Time 6

Stats: CPU Busy 5 (83.33%)

Stats: IO Busy 4 (66.67%)

Question 6: Now run with some randomly generated processes: `-s 1 -l 3:50,3:50` or `-s 2 -l 3:50,3:50` or `-s 3 -l 3:50,3:50`. See if you can predict how the trace will turn out. What happens when you use the flag `-I IO_RUN_IMMEDIATE` vs. `-I IO_RUN_LATER`? What happens when you use `-S SWITCH_ON_IO` vs. `-S SWITCH_ON_END`?

Answer:

-s 1 -l 3:50,3:50:

Process 0

cpu

io-start

io-start

Process 1

cpu

cpu

cpu

-s 2 -l 3:50,3:50:

Process 0

io-start

io-start

cpu

Process 1

cpu

io-start

io-start

-s 3 -1 3:50,3:50:

Process 0

cpu

io-start

cpu

Process 1

io-start

io-start

Cpu

"-I IO_RUN_IMMEDIATE" indicates that when an IO operation ends, the process will immediately resume execution without any delay.

"-I IO_RUN_LATER" means that when an IO operation ends, the process will be rescheduled to run later, allowing other processes to execute in the meantime.

"-S SWITCH_ON_IO" implies that the scheduler will switch between processes when an IO operation occurs.

"-S SWITCH_ON_END" means that the switch between processes will happen when a process completes its execution.