EOPSY task3

QingqingYan 309598

Task:

> Create a configuration file in which all processes run an average of 2000 milliseconds with a standard deviation of zero, and which are blocked for input or output every 500 milliseconds. Run the simulation for 10000 milliseconds with 2 processes. Examine the two output files. Try again for 5 processes. Try again for 10 processes. Explain what's happening.

First Come First Served (FCFS) is a Non-Preemptive scheduling algorithm. FIFO (First In First Out) strategy assigns priority to process in the order in which they request the processor. The process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. As the CPU finishes each task, it removes it from the head of the queue and heads on to the next task.
The FCFS scheduling algorithm is nonpreemptive.
Nonpreemptive – once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

Process state:
As a process executes, it changes state.
The state of a process is defined in part by the current activity of that process.
Each process may be in one of the following states:
NEW: The process is being created.
RUNNING: Instruction are being executed.
WAITING: The process is waiting for some event to occur(Such as an I/O completion or reception of a signal)
READY: The process is waiting to be assigned to a processor.
Terminated: The processor has finished execution.

1. **Run the simulation for 10000 milliseconds with 2 processes.**

Configuration:

```
// # of Process
    numprocess 2
// mean deivation
    meandev 2000
// standard deviation
     standdev 0
```

```
// process       # I/O blocking
        process 500
        process 500
// duration of the simulation in milliseconds
      runtime 10000
```

Summary result:

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 4000
Mean: 2000
Standard Deviation: 0
Process # CPU Time IO Blocking    CPU Completed    CPU Blocked
0         2000 (ms)     500 (ms)  2000 (ms)        3 times
1         2000 (ms)     500 (ms)  2000 (ms)        3 times
```

Summary processes:

```
                            CPU Time     IO Blocking accumulated-time
Process: 0 registered... (2000              500                    0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
```

Analysing:

The processes were scheduled based on Non-Preemptive scheduling and FCFS(First Come First Served) algorithm. For FCFS algorithm ,the processor schedules in the order in which the task arrives. If the process is blocked or interrupted during execution, the next in the queue gets the CPU.As we observed that there are two processes, and process 0 arrived first. Because process 0 was the first process that required CPU, it was placed onto the head of the ready queue FIFO. At this moment, the CPU was free, so process 0, which was at the head of the ready queue, was allocated to the CPU. At this moment, process 0 was running state. Then when process 1 entered into the ready queue, process 1 was placed onto the tail of the ready queue FIFO. Because the processes are scheduled based on Non-Preemptive scheduling, the CPU would not be allocated to process one until process 0 completed the execution or gave up CPU actively or be interrupted. This moment, process 1 was ready state, which was waiting to get the CPU time for its execution. After 500 milliseconds, process 0 was interrupted and gave up occupying the CPU. It entered the ready state and waited for process 1 to release the CPU. Then at this moment, the CPU would be allocated to process 1, which was the next in the queue. Thus, there was a switch in every 500ms between running state and ready state, which means if the currently executing process was running ,the previous process would be ready state. When process 0 completed execution, it would be removed from the main memory, and its PCB was deleted from the FIFO queue. At this moment, the state of the process 0 was Terminated.

2. Run the simulation for 10000 milliseconds with 5 processes.

Configuration:

```
// # of Process
   numprocess 5
// mean deivation
   meandev 2000

// standard deviation
   standdev 0

// process        # I/O blocking
   process 500
   process 500
   process 500
   process 500
   process 500

// duration of the simulation in milliseconds
   runtime 10000
```

Summary result:

```
   Scheduling Type: Batch (Nonpreemptive)
   Scheduling Name: First-Come First-Served
   Simulation Run Time: 10000
   Mean: 2000
   Standard Deviation: 0
   Process #  CPU Time     IO Blocking   CPU Completed   CPU Blocked
```

0	2000 (ms)	500 (ms)	2000 (ms)	3 times
1	2000 (ms)	500 (ms)	2000 (ms)	3 times
2	2000 (ms)	500 (ms)	2000 (ms)	3 times
3	2000 (ms)	500 (ms)	2000 (ms)	3 times
4	2000 (ms)	500 (ms)	2000 (ms)	3 times

Summary processes:

```
                    CPU Time   IO Blocking accumulated-time
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
```

```
Process: 4 registered... (2000 500 1000 1000)
Process: 4 I/O blocked... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
```

Analysing:

As can be seen from the above Summary Result and Summary Processes that firstly, the simulation was run for 10000ms, because each process, in total, was running for 2000ms. Therefore 5 processes was 10000ms which did not exceed our total running time 10000ms.

The processes are based on non-preemptive scheduling and First-come First- serve algorithm. All processes are placed into the FIFO queue in the order. It means that the second process in FIFO queue is being proceeded only after the previous process in head of FIFO queue is interrupted or give up the CPU actively. When the second process is I/O blocked(interrupted) ,we go back to execute the previous process in the head of FIFO. This change happens every 500ms.From Summary result file, we can see all processes' CPU Completed time were 2000ms,but in Summary processes file, the process 4 did not show us "completed", I guess this is because the 10000ms run time was not enough for showing this message after the process 4 completed execution. Because the total CPU execution time was only 10000ms.

So we take a conclusion:

```
Process: 0 running state... (2000 500 0 0)
Process: 0 I/O ready state... (2000 500 500 500)
Process: 1 running state... (2000 500 0 0)
Process: 1 ready state(after it was interrupted)... (2000 500 500 500)
Process: 0 running state... (2000 500 500 500)
Process: 0 ready state (2000 500 1000 1000)
……..
```

Before the process 0 and process 1 completed the execution, other processes were ready state in the FIFO queue in the order to wait for the CPU. After the process 0 completed execution, it would be removed from the FIFO and the second process would become the head of the FIFO queue and so on until the number of processes in the FIFO queue is 0.

3. Run the simulation for 10000 milliseconds with 10 processes.

Configuration:

```
// # of Process
   numprocess 10

// mean deivation
   meandev 2000

// standard deviation
```

```
  standdev 0

// process         # I/O blocking
   process 500
   process 500
   process 500
   process 500
   process 500
   process 500
   process 500
   process 500
   process 500
   process 500

// duration of the simulation in milliseconds
  runtime 10000
```

Summary result:

```
  Scheduling Type: Batch (Nonpreemptive)
  Scheduling Name: First-Come First-Served
  Simulation Run Time: 10000
  Mean: 2000
  Standard Deviation: 0
  Process #    CPU Time     IO Blocking   CPU Completed   CPU Blocked
  0        2000 (ms)    500 (ms)      2000 (ms)     3 times
  1        2000 (ms)    500 (ms)      2000 (ms)     3 times
  2        2000 (ms)    500 (ms)      2000 (ms)     3 times
  3        2000 (ms)    500 (ms)      2000 (ms)     3 times
  4        2000 (ms)    500 (ms)      1000 (ms)     2 times
  5        2000 (ms)    500 (ms)      1000 (ms)     1 times
  6        2000 (ms)    500 (ms)      0 (ms)        0 times
  7        2000 (ms)    500 (ms)      0 (ms)        0 times
  8        2000 (ms)    500 (ms)      0 (ms)        0 times
  9        2000 (ms)    500 (ms)      0 (ms)        0 times
```

Summary processes:

```
                           CPU Time    IO Blocking accumulated-time
  Process: 0 registered... (2000 500 0 0)
  Process: 0 I/O blocked... (2000 500 500 500)
  Process: 1 registered... (2000 500 0 0)
  Process: 1 I/O blocked... (2000 500 500 500)
  Process: 0 registered... (2000 500 500 500)
  Process: 0 I/O blocked... (2000 500 1000 1000)
  Process: 1 registered... (2000 500 500 500)
```

```
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 5 registered... (2000 500 0 0)
Process: 5 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 5 registered... (2000 500 500 500)
```

Analysing:

In this case, the simulation time was not enough to simulate all 10 processes, , because we need at least 20000ms(2000ms * 10) to run all 10 processes. We can also observe that the process 4 did not complete the execution in this case, but it completed execution in the previous case which simulated 5 processes. This is because the 4th process was the last process and there was no process after it in FIFO. It could enter onto running state soon from the ready state after it was interrupted.

We can see only first 4 processes (from 0 to 3) were completed. Actually we can see that 6 processes ,four processes(No.0 to 3) show completed and 2 processes(process 4 and process 5) are being proceed. Because actual simulation time was out of the range we set, the rest processes didn’t start.