Kendrick Kwok / CSC 415 / 912351666 / 10/13

CSC415 OPERATING SYSTEM PRINCIPLES
Homework 1 Due: 4:00PM, Thursday, 10/13/2016

1. What is the difference between symmetric multiprocessing and asymmetric multiprocessing? (10 Points)

Asymmetric multiprocessing means that only one processor accesses the system data structure, then makes scheduling decision, and performs system activities, allowing other processors to run user code. Symmetric multiprocessing means that each processor is self-scheduling, and each processor may share the same ready queue or each has it's own private queue of ready processes.

2. Why do user programs have to make system calls rather than just executing the code for the system calls themselves? (10 Points)

There are two modes for systems, which is kernel mode and user mode. Kernel modules exist in kernel space which requires system calls in order to be accessed from user space. This basically means that users will not have the ability to have access the system, and need system calls to execute code for the system.

3. Assuming there are no errors, how many new processes will the following code create? Why? (10 Points)

Fork creates another child process. It duplicates another process. If there are no errors and assuming that a process has been created, depending on fork being n, there would be 2^n processes. Below is a sample code of the output from the program.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;
printf("this is a test\n");
        if (!fork()){
        printf("Hello1");
        fork();
        }
        else { fork();
                printf("Hello2\n");
                fork();
                printf("Hello3\n");
         }

return 0;}
```

```
Output :
Hello2
Hello2
Hello3
Hello3
Hello3
Hello3
Hello1
Hello1

^
^
```

There are three forks.
Fork is called eight times including the if statement fork.

4. What are the differences between short-term scheduler, medium-term scheduler, and long-term scheduler? (10 Points)

A short-term scheudler is also called CPU scheduler. Short-term scheduler is the change of ready state to the running state of the process. CPU scheduler selects just one process among processes that are ready to execute, and gives a CPU to one of them. A **medium-term scheduler** is part of swapping. It removes processes form memory and makes space for other processes. It is moved to secondary storage. **A Long-term** scheduler is called a job scheduler. Long-term scheduler determines which programs are being sent to the system, and which process needs to be executed.

5. Describe the actions taken by a kernel to context-switch between processes? (10 Points)
Context switch is the switching of the CPU to another process require a saved state of a current process and a state restored of a different process. When a context switch occurs, the kernel saves the context of the old process in a Process Control Block, then loads the saved context of a new process to run.

6. What are the two models for Inter-Process Communication? What are their differences? (10 Points)

The two models of inter-process communication is shared memory and message passing. Message passing is the communication process exchanging messages with one another to exchange information. Message passing useful for exchanging small data, no conflicts to avoid. Shared memory is the processes that use shared memory that creates and share memory system calls to gain memory owned by other processes. Two or more processes can exchange information. The difference is that shared memory allows maximum speed and very convenient in communication. Message passing is easy to implement that shared memory, because it handles and only can handle small amounts of data.

7. Can a multi-threaded solution using multiple user-level threads and many-to-one model achieve better performance on a multi-processor system than on a single processor system? (10 Points)

Many-to-One model are several user-level threads that are mapped to one kernel thread. Multi-thread solution using multiple user-level threads provides a mechanism for more efficient use of multiple computing cores. Multi-thread solution creates parallelism, which means more than one task can be performed simultaneously. This means that to achieve a maximum performance, a multi-processor system can be used to achieve this result better than a single processor system.

8. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds: (30 Points)

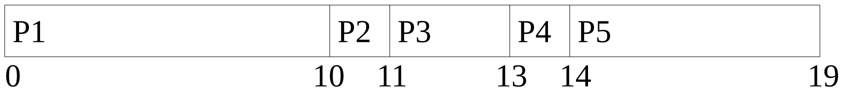
Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0. a. Draw four

Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling. b. What is the turnaround time of each process for each of the scheduling algorithms in Part a? c. What is the average waiting time for each of the scheduling algorithms in Part a?

FCFS

a. Gantt Chart



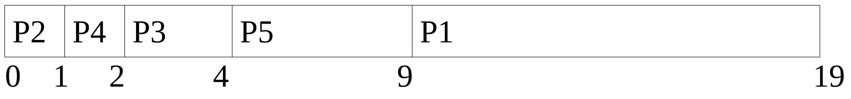
b. Turnaround time
P1 = 10, P2 = 11, P3 = 13, P4 = 14, P5 = 19

c. Average waiting time
P1 = 0, P2 = 10, P3 = 11, P4 = 13, P5 = 14

0 + 10 + 11 + 13 + 14 / 5
(21 + 27) / 5
= 9.6

SJF

a. Gantt Chart



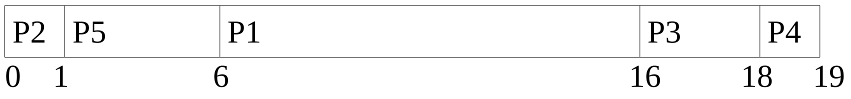
b. Turnaround time
P2 = 1, P4 = 2, P3 = 4, P5 = 9, P1 = 19

c. Average Waiting time
P1 = 9, P2 = 0, P3 = 2, P4 = 1, P5 = 4

0 + 1 + 2 + 4 + 9 / 5
(3 + 13) /5
= 3.2

Non-preemptive priority

a. Gantt Chart



b. Turnaround time
P2 = 1 , P5 = 6, P1 = 16, P3 = 18, P4 = 19

c. Average Waiting time
P1 = 6, P2 = 0, P3 = 16, P4 = 18, P5 = 1

0 + 1 + 6 + 16 + 18 / 5
(7 + 34) /5
= 8.2

RR (quantum 1)
a. Gantt Chart

P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1	P5	P1	P5	P1	P1	P1	P1	P1

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

b. Turnaround time
P1 = 19, P2 = 2, P3 = 7, P4 = 4, P5 = 14

c. Average Waiting time
(Average waiting time is calculated by when the process is finished my burst time)
P1 = 19 – 10 = 9
P2 = 2 – 1 = 1
P3 = 7 – 2 = 5
P4 = 4 – 1 = 3
P5 = 14 – 5 = 9

9 + 1 + 5 + 3 + 9 / 5 = 5.4