Dustin Badillo
004715242
CSE 460
Kay Zemoudeh

Home-Work 4

4.8

Which of the following components of program state are shared across threads in a multithreaded process?

a. Register values
b. Heap memory
c. Global variables
d. Stack memory

4.14

A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between startup and termination, the program is entirely CPU bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread).

-How many threads will you create to perform the input and output? Explain.

Use as many threads as there are blocking system calls since they will be spent blocking. One for input and another for output.

-How many threads will you create for the CPU-intensive portion of the application? Explain

4 since there should be the same amount of threads as there are processing cores any less and you will waste cpu utilization.

4.16

As described in Section 4.7.2, Linux does not distinguish between processes and threads. Instead, Linux treats both in the same way, allowing a task to be more akin to a process or a thread depending on the set of flags passed to the clone() system call. However, other operating systems, such as Windows, treat processes and threads differently. Typically, such systems use a notation in which the data structure for a process contains pointers to the separate threads belonging to the process. Contrast these two approaches for modeling processes and threads within the kernel.

Since Linux does not distinguish between processes and threads when the process is taking place it can determine how much is to be shared between the child and parent tasks. Since some of the code is being shared a lot of space and time can be saved using this approach. In the windows approach, it treats the thread and process differently. Each application runs as a separate process and each process can contain one or more thread.

6.16

Consider the following set of processes,with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

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

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority(a larger priority number implies a higher priority), and RR (quantum = 2).

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

FCFS: 10.2
SJF: 8.6
Priority: 15.4
RR: 11.2

c. What is the waiting time of each process for each of these scheduling algorithms?

FCFS: 6.2
SJF: 4.8
Priority: 11.4
RR: 8

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Smallest job first gave the minimum average waiting time with 4.8 and First come first serve gave the second smallest with 6.2 average.

Extra Problem

Give a solution to the semaphore problem for 3 processes where P1 and P2 cannot be in their critical sections at the same time, and P2 and P3 cannot be in their critical sections at the same time, but P1 and P3 are free to be in their critical sections at the same time.