Problem Set 3

Matthew Kaiser

1. To implement deadlock avoidance, the Banker's Algorithm is applied to the following system, where the total number of resources available for each of four resource types is R0=6, R1=4, R2=4, and R3=2. Is the system in a safe state? If not, then explain why not, or if so, find a safe sequence. Show your work.

Max Claims:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R0 | R1 | R2 | R3 |
| P0 | 3 | 2 | 1 | 1 |
| P1 | 1 | 2 | 0 | 2 |
| P2 | 1 | 1 | 2 | 0 |
| P3 | 3 | 2 | 1 | 0 |
| P4 | 2 | 1 | 0 | 1 |

Current Allocation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R0 | R1 | R2 | R3 |
| P0 | 2 | 0 | 1 | 0 |
| P1 | 1 | 1 | 0 | 0 |
| P2 | 1 | 1 | 0 | 0 |
| P3 | 1 | 0 | 1 | 0 |
| P4 | 0 | 1 | 0 | 1 |

Yes, the system is in a safe state, the sequence is as follows:

* Give resources to P2 and update:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 2 | 2 | 2 | 1 |

* Give resources to P3 and update:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 3 | 2 | 3 | 1 |

* Give resources to P4 and update:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 3 | 3 | 3 | 2 |

* Give resources to P1 and update:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 4 | 4 | 3 | 2 |

* Give resources to P0 and update:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 6 | 4 | 4 | 2 |

* Job is complete

2. Suppose we have 5 dining philosophers and 5 shared chopsticks, and that the philosophers are circularly deadlocked. Assume also that there are two serving spoons, one allocated to philosopher 1, and the other available. Assume that philosopher 4 wants a serving spoon. Show how this situation can be modeled according to the various data structures, e.g. Allocation and Request matrices, used in the Deadlock detection algorithm. Run the Deadlock detection algorithm on your model to prove that the dining philosophers are in deadlock.

* R0 is the Chopstick
* R1 is the Spoon
* P0-P4 are the five philosophers
* Max Claims:

|  |  |  |
| --- | --- | --- |
|  | R0 | R1 |
| P0 | 2 | 1 |
| P1 | 2 | 1 |
| P2 | 2 | 1 |
| P3 | 2 | 1 |
| P4 | 2 | 1 |

* Current Allocation:

|  |  |  |
| --- | --- | --- |
|  | R0 | R1 |
| P0 | 1 | 1 |
| P1 | 1 | 0 |
| P2 | 1 | 0 |
| P3 | 1 | 0 |
| P4 | 1 | 0 |

* What is needed:

|  |  |  |
| --- | --- | --- |
|  | R0 | R1 |
| P0 | 1 | 0 |
| P1 | 1 | 0 |
| P2 | 1 | 0 |
| P3 | 1 | 1 |
| P4 | 1 | 0 |

* P3 needs a spoon and its available
* P0-P4 need another chopstick, however none are available
* No philosopher can get enough resources due to the lack of chopsticks, this results in deadlock

3. Consider the following set of processes, with the length of the CPU execution time given in seconds:

|  |  |  |
| --- | --- | --- |
| ­­Process | Execution Time | Priority |
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

Draw four Gantt charts that illustrate the timeline of execution of the processes when scheduled according to FCFS, SJF, nonpreemptive priority (priority 1 is most important), and round robin (time slice=1). Which one has the lowest average wait time? Which one has the lowest average turnaround time?

FCFS:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | //P1-> | //P2-> | //P3-> | //P4-> | //P5-> | // |
| Time | 0 | 10 | 11 | 13 | 14 | 19 |

The // notation represents a change in process, the beginning and/or end

The -> represents that the process is running

Averages:

* Wait Time: (10+11+13+14)/5 = 9.6 seconds
* Turnaround Time: (10+11+13+14+19)/5 = 13.4 seconds

SJF: Shortest times yielded from this scheduler

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | //P2-> | //P4-> | //P3-> | //P5-> | //P1-> | // |
| Time | 0 | 1 | 2 | 4 | 9 | 19 |

The // notation represents a change in process, the beginning and/or end

The -> represents that the process is running

Averages:

* Wait time: (1+2+4+9)/5 = 3.2 seconds
* Turnaround Time: (1+2+4+9+19) = 7 seconds

Non-Preemptive Priority:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | //P2-> | //P5-> | //P1-> | //P3-> | //P4-> | // |
| Time | 0 | 1 | 6 | 16 | 18 | 19 |

The // notation represents a change in process, the beginning and/or end

The -> represents that the process is running

Averages:

* Wait Time: (1+6+16+18)/5 = 8.2 seconds
* Turnaround Time: (1+6+16+18+19)/5 = 12 seconds

Round Robin:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | //P1-> | //P2-> | //P3-> | //P4-> | //P5-> | //P1-> | //P3-> | //P5-> | //P1-> |
| Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Continuing with Round Robin…

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| //P5-> | //P1-> | //P5-> | //P1-> | //P5-> | //P1-> | //P1-> | //P1-> | //P1-> | //P1-> | // |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

The // notation represents a change in process, the beginning and/or end

The -> represents that the process is running

Averages:

* Wait Time: 5.4 seconds
* Turnaround Time: 9.2 seconds

**Shortest Job First yields the shortest average wait and turnaround times in this problem**

4.Suppose the Completely Fair Scheduler (CFS) algorithm is applied to the following scenario.  There exist two processes P1 and P2 that need to be scheduled.  P1 has two threads T1 and T2 that are implemented as kernel threads.  P2 has three threads T3, T4, and T5 that are implemented as user space threads.  We apply round robin scheduling to all schedulable tasks, where each task schedulable by the kernel receives a time slice of size T seconds.

a. Draw the table of wait times (times owed on an ideal CPU) for one round robin of this scenario, i.e. what are the wait time balances as each time slice is allocated to a kernel-schedulable task, and what is the final wait time for each such task?  See for example slide 21 of the Chapter 6.3 lecture slides.

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | T2 | P2 | T1 |
| -> |  |  |  |

The -> represents that the process is running

|  |  |  |  |
| --- | --- | --- | --- |
| Give time(T) to task | T1 | T2 | P2 |
| T1 | -2T/3 | T/3 | T/3 |
| T2 | -T/3 | -T/3 | 2T/3 |
| P2 | 0 | 0 | 0 |

b. Is this system fair at the level of schedulable tasks?  Is this system fair at the level of processes?  Justify your answers.

* At the level of schedulable tasks the system is fair because all of the tasks are shared equally.
* At the level of processes the system is not fair because the processes run on multiple threads, sharing resources among them(processes).