***Pledge:*** *I pledge my honor that I have abided by the Stevens Honor System.* - **Eric Altenburg**

**1: Does the busy waiting solution using the turn variable (Fig 2-23 in the book) solves the mutual exclusion problem when two processes are running on a shared-memory multiprocessor, that is, two CPUs sharing a common memory?**

Yes, the busy waiting solution using the turn variable solves the mutual exclusion problem when two processors are running on a shared-memory multiprocessor. From the figure, the "turn" variable is used to control the critical region between the two processes, and in this case, since it is on a multiprocessor, the two processes are running on different CPUs; however, the "turn" variable should now be in the shared-memory.

**2: Five batch jobs. A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.**

- **a Round robin.**
- **b Priority scheduling.**
- **c First-come, first-served (run in order 10, 6, 2, 4, 8).**
- **d Shortest job first.**

**For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d), assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.**

All batch jobs have been assigned a letter (a-e) in the order they were stated.

**(a)**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | A | B | C | D | E | A | B | D | E | A |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B | D | E | A | B | E | A | B | E | A | E | A | E | A | A |

The turnaround times in minutes for each job are as follows:

A 30
B 23
C 8
D 17
E 28

Mean Process Turnaround Time = $\frac{30+23+8+17+28}{5}$ = 21.2 minutes

**(b)**

| 1-6 | 7-14 | 15-24 | 25-26 | 27-30 |
|---|---|---|---|---|
| B | E | A | C | D |

The turnaround times in minutes for each job are as follows:

A 24
B 6
C 26
D 30
E 14

Mean Process Turnaround Time = $\frac{24+6+26+30+14}{5}$ = 20 minutes

**(c)**

| 1-10 | 11-16 | 17-18 | 19-22 | 23-30 |
|---|---|---|---|---|
| A | B | C | D | E |

The turnaround times in minutes for each job are as follows:

A 10
B 16
C 18
D 22
E 30

Mean Process Turnaround Time = $\frac{10+16+18+22+30}{5}$ = 19.2 minutes

**(d)**

| 1-2 | 3-6 | 7-12 | 13-20 | 21-30 |
|---|---|---|---|---|
| C | D | B | E | A |

The turnaround times in minutes for each job are as follows:

A 30
B 12
C 2
D 6
E 20

Mean Process Turnaround Time = $\frac{30+12+2+6+20}{5}$ = 14 minutes

**3: A soft real-time system has four periodic events with periods of 50, 100, 200, and 250 msec each. Suppose that the four events require 35, 20, 10, and x msec of CPU time, respectively. What is the largest value of x for which the system is schedulable?**

$$\begin{aligned}\frac{35}{50}+\frac{20}{100}+\frac{10}{200}+\frac{x}{250} &\leq 1\\ 0.95+\frac{x}{250} &\leq 1\\ \frac{x}{250} &\leq 0.05\\ x &\leq 12.5\\ x &= 12.5 \text{ msec}\end{aligned}$$

The largest value x can be is 12.5 msec.

**4: Consider the following state of a system with four processes, P1, P2, P3, and P4, and five types of resources, RS1, RS2, RS3, RS4, and RS5:**

$$C = \begin{array}{|c|c|c|c|c|}\hline \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{2}\\ \hline \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\\ \hline \mathbf{2} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \hline\end{array} \quad R = \begin{array}{|c|c|c|c|c|}\hline \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{2} & \mathbf{1}\\ \hline \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{2} & \mathbf{1}\\ \hline \mathbf{0} & \mathbf{2} & \mathbf{0} & \mathbf{3} & \mathbf{1}\\ \hline \mathbf{0} & \mathbf{2} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \hline\end{array} \quad \begin{array}{l} E = (24144)\\ A = (01021)\end{array}$$

**Using the deadlock detection algorithm described in Section 6.4.2, show that there is a deadlock in the system. Identify the processes that are deadlocked.**

1. Looking through the rows of R, it is clear that the 2nd row (01021) equals $A = (01021)$
2. Add the 2nd row of C (010101) to A, now $A = (02031)$
3. Looking through the rows of R now that $\text{P}_2$ is marked, the 3rd row (02031) equals $A = (02031)$
4. Add the 3rd row of C (00001) to A, now $A = (02032)$
5. Looking through the rows of R now that $\text{P}_2$ and $\text{P}_3$ are marked, none of the other rows in R are less than or equal to $A = (02032)$, thus the algorithm terminates.
6. All unmarked processes are in deadlock. Therefore, $\text{P}_1$ and $\text{P}_4$ are deadlocked.

**5: A system has four processes and five allocable resources. The current and maximum needs are as follows:**

| | **Allocated** | **Maximum** | **Available** |
|---|---|---|---|
| **Process A** | **1 0 2 1 1** | **1 1 2 1 2** | **0 0 x 1 1** |
| **Process B** | **2 0 1 1 0** | **2 2 2 1 0** | |
| **Process C** | **1 1 0 1 0** | **2 1 3 1 0** | |
| **Process D** | **1 1 1 1 0** | **1 1 2 2 1** | |

**What is the smallest value of x for which this is a safe state? Show your calculations using the Banker's Algorithm.**

Need Matrix = $Max - Allocation$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 0 | 2 | 1 | 0 | 0 |
| 1 | 0 | 3 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |

1. If $x = 0$, then there will be a deadlock, so 0 is not a valid value.
2. If $x = 1$, then $P_D$ can run to completion as $P_D$ = (00111) in the Need matrix is equal to Available = (00111). We then add $P_D$ from the Allocated matrix to Available, now Available = (11221).
3. $P_A$ can run to completion now as $P_A$ = (01001) in the Need matrix is less than Available = (11221). We then add $P_A$ from the Allocated matrix to Available, now Available = (21432).
4. $P_C$ can run to completion now as $P_C$ = (10300) in the Need matrix is less than Available = (21432). We then add $P_C$ from the Allocated matrix to Available, now Available = (32442).
5. $P_B$ can run to completion now as $P_B$ = (02100) in the Need matrix is less than Available = (32442).
6. The smallest safe value of x is 1, and the safe sequence is D, A, C, B.