CSCE-4600 Operating Systems Design

Homework #2

Due 2-14-2020 11:59pm

Submission per Canvas

1. Consider a system of 9 processes, **P** = {p1, …, p9}
   Associated with the system are 6 memory cells, **M** = {M1, .., M6}

   The domain and range for each process is given in the following table:

Process pi	Domain D(pi)	Range R(pi)
p1	M1, M2	M3
p2	M1	M5
p3	M3, M4	M1
p4	M3, M4	M5
p5	M3	M4
p6	M4	M4
p7	M5	M5
p8	M3, M4	M2
p9	M5, M6	M6

$$\begin{tikzcd}
 & & 1 \arrow[d] & & \\
 & & 2 \arrow[dll] \arrow[d] \arrow[drr] & & \\
3 \arrow[dr] & & 4 \arrow[dl] \arrow[dr] & & 5 \arrow[dl] \arrow[ddll, bend left] \\
 & 6 \arrow[dr] & & 7 \arrow[dd] & \\
 & & 8 \arrow[dr] & & \\
 & & & 9 &
\end{tikzcd}$$

   In addition, you are given the following precedence relation:
   ➔ = {(P1,P2), (P1,P6), (P2,P3), (P2,P4), (P2,P5),(P3,P6),(P3,P8),(P4,P6),
   (P4,P7), (P5,P7), (P5,P8), (P6,P8), (P6,P9), (P7,P9), (P8,P9)}

a. (20pts) Construct the Precedence Graph (not containing any redundant or transitive edges)

b. (20pts) Determine if the system above is always determinate. If it is not, add to ➔ necessary elements to make it determinate.

   Add:
   ➔ { (P7, P8) }

$$\begin{tikzcd}
1 \arrow[d] & \\
3 \arrow[d] & 2 \arrow[dl] \\
4 \arrow[d] & \\
5 &
\end{tikzcd}$$

2. (30 pts) In the first problem, there were 9 processes, many of which were listed as pairs under the precedence relation. Suppose we are dealing with a system of only 5 processes named P1 through P5. You are given a set of constraints that are expressed by the following precedence relation:
   ➔ = {(P1,P3), (P1, P5), (P2,P4), (P3, P4), (P4, P5)}

Provide pseudocode to show how you can use semaphores to enforce these constraints (i.e., the precedence relation ➔).

```
semaphore s1, s2, s3;

[Process 1]     [Process 2]     [Process 3]     [Process 4]     [Process5]
  begin:          begin:          begin:          begin:          begin:
  …               …               …               …               …
  V(s1, 1)        V(s2, 1)        P(s1)           P(s2)           P(s3)
  …               …               execute         P(s2)           execute
  end;            end;            V(s2, 1)        execute         …
                                  …               V(s3, 1)        end;
                                  end;            …
                                                  end;
```

3. (30pts) Implement a system of three processes which read and write sequence numbers to a file. Each of the three processes P1, P2, and P3 must obtain 200 integers from the file. The file only holds one integer at any given time. Given a file, F, containing a single integer, each process must perform the following steps:

    1. Open F
    2. Read the integer N from the file
    3. Close F
    4. Output N and the process' PID (either on screen or test file)
    5. Increment N by 1
    6. Open F
    7. Write N to F (overwriting the current value in F)
    8. Close F.

    Will the numbers that are read by the processes P1, P2, and P3 be always unique, or do you observe duplicates of numbers being obtained by different processes (i.e., does a particular integer **x** appear in the output of more than on process)?

    Yes

    Rewrite your implementation of these 3 processes to guarantee that no duplicate numbers are ever obtained by the processes. In other words, each time the file is read by any process, that process reads a distinct integer.

    Briefly describe why the code sequence above can lead to duplication of integer values and discuss how your solution will avoid these duplicates.

NOTE: Programs must compile and execute on the CSE machines. It is imperative **that the file F containing in the integer is located on the local disk. On Linux, the \tmp directory is located on the local file system.**

The code described above creates three processes that can access the critical section at the same time however the solution I provided mutual exclusion on the critical section through semaphores. I created a process-shared semaphore shared between the three processes, placed in a shared memory region. The semaphore provides a mutual exclusion over all resources of the critical section.