SOLUTIONS TO THE SECOND 3360/6310 QUIZ

Jehan-François Pâris
Summer 2016

First question

- Consider the following System V Release 4 scheduler:

```
#ts_quantum  ts_tqexp  ts_slpret  ts_maxwait  ts_lwait  LVL
 1000        0         1          16000       1         # 0
  500        1         2          8000        2         # 1
  200        1         3          4000        3         # 2
  100        2         3          2000        3         # 3
```

  - Which events can ***increase*** the priority of a process at ***level 2***?
  - Which events can ***lower*** it?

First question

- Consider the following System V Release 4 scheduler:

```
#ts_quantum   ts_tqexp   ts_slpret   ts_maxwait   ts_lwait   LVL
 1000         0          1           16000        1          # 0
  500         1          2           8000         2          # 1
  200         1          3           4000         3          # 2
  100         2          3           2000         3          # 3
```

  - Which events can ***increase*** the priority of a process at ***level 2***?
    - ***Process returns from the BLOCKED state***
    - ***Process spends more than 4s in the READY state***

First question

- Consider the following System V Release 4 scheduler:

```
#ts_quantum  ts_tqexp  ts_slpret  ts_maxwait  ts_lwait  LVL
 1000        0         1          16000       1         # 0
  500        1         2          8000        2         # 1
  200        1         3          4000        3         # 2
  100        2         3          2000        3         # 3
```

  - Which events can ***lower*** the priority of a process at ***level 2***?
    - ***Process exceeded its 200ms processor time slice***

Second question

- Give an example of a data structure that is ***much costlier*** to pass to a ***remote procedure*** than to a ***regular procedure***.

Second question

- Give an example of a data structure that is ***much costlier*** to pass to a ***remote procedure*** than to a ***regular procedure***.
  - ***Any dynamic structure***
    - ***Especially large ones!***
  - ***Any big array***

Third question

- What is the main difference between the **signal** and **notify** primitives?

Third question

- What is the main difference between the **signal** and **notify** primitives?
  - ***A signal will take control of the monitor away from the calling procedure if any other procedure was waiting on that signal***
  - ***A notify will have no effect until the procedure releases the monitor***

Third question

- Which one is better and why?

Third question

- Which one is better and why?
  - ***Notify because it***
    - ***Causes fewer context switches***
    - ***Can be put anywhere in monitor procedures.***

Fourth question

- What is the major disadvantage of ***busy waits****?*

Fourth question

- What is the major disadvantage of ***busy waits****?*
  - ***Busy waits waste processor cycles and cause additional context switches***
    - *Can also cause* ***priority inversions***

Fourth question

- What is the major advantage of ***streams*** over ***datagrams***?

Fourth question

- What is the major advantage of ***streams*** over ***datagrams***?

  - ***Stream are better than datagrams for transmitting large amounts of data because they guarantee that all data will arrive***
    - ***Without errors***
    - ***In the right order***

Fourth question

- How can we prevent deadlocks by ***denying*** the ***circular wait condition***?

Fourth question

- How can we prevent deadlocks by ***denying*** the ***circular wait condition***?
  - ***We should force all processes to acquire all resources in the same linear order***

Fourth question

- Why is ***lottery scheduling*** said to be ***non-deterministic***?

Fourth question

- Why is ***lottery scheduling*** said to be ***non-deterministic***?

  - ***Because lottery scheduling runs a <u>lottery</u> to decide which process in the ready queue should get the CPU***

Fourth question

- What is the sole possible initial value for a ***mutex***?

Fourth question

- What is the sole possible initial value for a ***mutex***?
  - ***ONE***
    - *Recall that Pthread mutexes are* ***automatically initialized to one***
      - *User has no other option*

Fourth question

- What is the difference between ***blocking sends*** and ***non-blocking sends***?

Fourth question

- What is the difference between ***blocking sends*** and ***non-blocking sends***?
  - ***Blocking sends wait until the message they send have been received by their destination***
  - ***Non-blocking sends return as soon as their messages have been accepted for delivery***

Fifth question

- A bar room has a maximum occupancy of **40** guests.
- Complete the following code fragments to ensure that
  - The room will never exceed its maximum occupancy,
  - All parties will always enter the room ***together in the order they arrived***.

Fifth question

- ```
  semaphore room   = _____;
  semaphore access = _____;
  ```
  ```

Fifth question

```
party_arrives (int nguests) {
  int i;

  ____________________________________

  for (i = 0; i < nguests ; i++)

      ____________________________

  ____________________________________

  get_all_inside();
} // party_arrives
```

Fifth question

```
party_leaves (int nguests) {
  int i;
  for (i = 0; i < nguests ; i++)
      ____________________________________
} // party_leaves
```

Fifth question

```
semaphore room   = 40;
semaphore access = 1 ; // mutex?
```

- Semaphore room will represent the room capacity
  - Initially set at 40
  - Before entering the room, each guest must do a P(&room)
  - When he or she leaves the room, each guest must do a V(&room)

Fifth question

```
party_arrives (int nguests) {
  int i;

  ____________________________________

  for (i = 0; i < nguests ; i++)
       P(&room);

  ____________________________________

  get_all_inside();
} // party_arrives
```

Still has problems

Fifth question

```
party_leaves (int nguests) {
  int i;
  for (i = 0; i < nguests ; i++)
       V(&room);
} // party_leaves
```

A remaining problem

- What if a small party arrives while a larger party is in the process of accessing the **`room`** semaphore?
  - Could get ahead
  - Must add a ***mutex***

Fifth question

```
party_arrives (int nguests) {
  int i;
  P(&access);
  for (i = 0; i < nguests ; i++)
       P(&room);
  V(&access);
  get_all_inside();
} // party_arrives
```

The correct solution

Sixth question

- What are the major advantage and the major disadvantage of ***atomic transactions***?

Sixth question

- What are the major advantage and the major disadvantage of ***atomic transactions***?
  - ***Atomic transactions ensure that all remote procedure calls will either execute correctly of have no effect***
    - *Eliminate the risks of partial of duplicate executions*
  - ***They are costly***

Seventh question

- What are the two possible values for the register **%eax** after the instruction is executed and their meanings for the process that executed the instruction?
  - **If %eax equals _____ then the process can enter the critical section.**
  - **If %eax equals _____ then the process must wait.**

Seventh question

- What are the two possible values for the register **%eax** after the instruction is executed and their meanings for the process that executed the instruction?
  - **If %eax equals 0 then the process can enter the critical section.**
  - **If %eax equals 1 then the process must wait.**