SOLUTIONS FOR THE SECOND 4330/6310 QUIZ

Jehan-François Pâris

Spring 2015

First Question

- Consider the following solution to the mutual exclusion problem and explain when it fails (5 points) and what happens then. (5 points)
- **shared int locked[2] = {0, 0}; // global variable**
- **void enter_region(int pid) { // always 0 or 1**
  while (locked [1 - pid]); // busy wait
  locked[pid] = 1; // reserve
  } // enter_region
- **void leave_region(int pid) {**
  locked[pid] = 0;
  } // leave_region

Answer

- When **two processes arrive in lockstep**

  then **both processes will enter the critical region.**

Alternate first question

- Consider the following solution to the mutual exclusion problem and explain when it fails (5 points) and what happens then. **shared int locked[2] = {0, 0}; // global variable**
- **void enter_region(int pid) { // always 0 or 1**
  locked[pid] = 1; // reserve
  while (locked [1 - pid]); // busy wait
  } // enter_region
- **void leave_region(int pid) {**
  locked[pid] = 0;
  } // leave_region

Answer

- When **two processes arrive in lockstep**

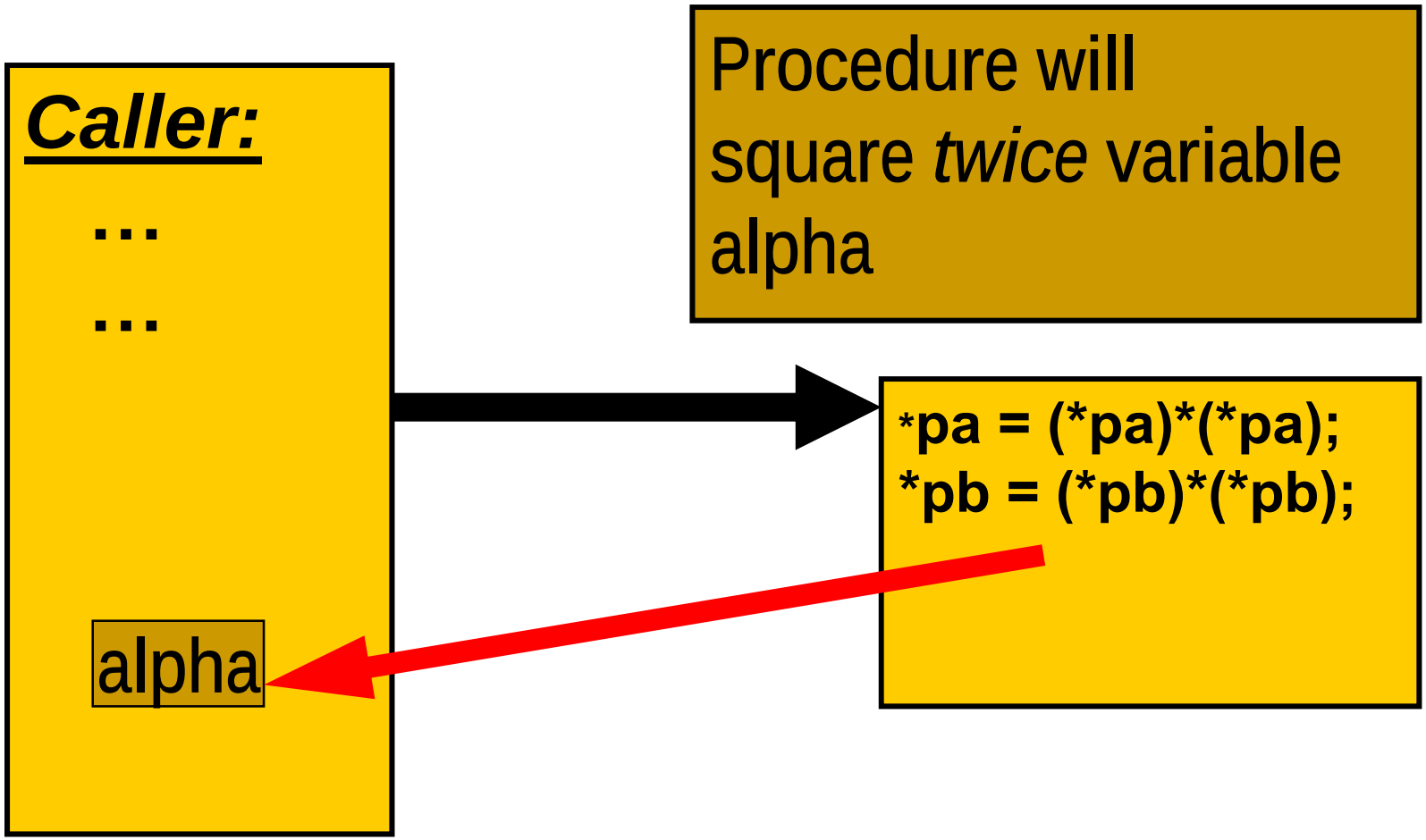
  then **we have a deadlock.**

Second question

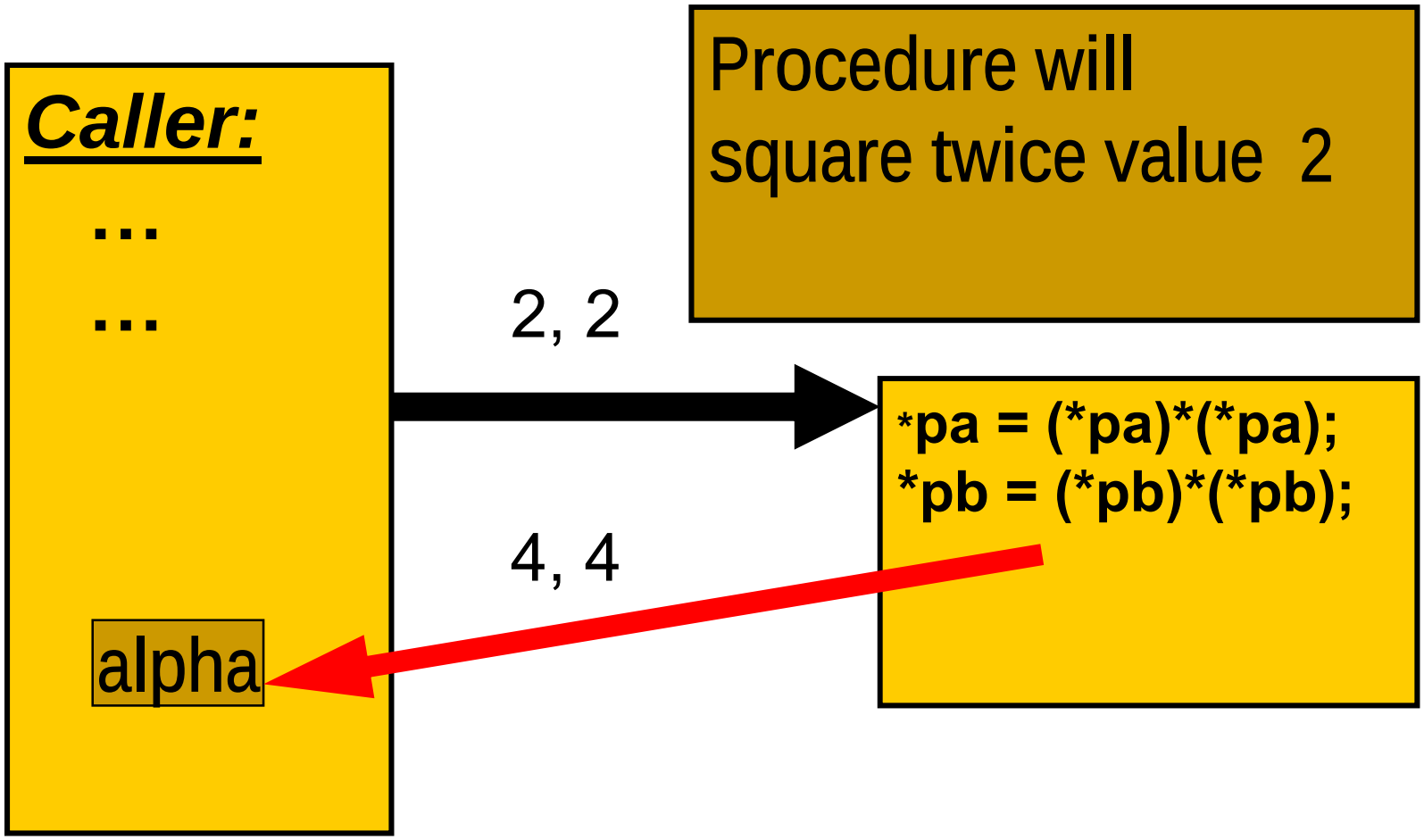
- Consider the function
  - **void squarethem(int *pa, int *pb) {**
    ***pa = (*pa)*(*pa);**
    ***pb = (*pb)*(*pb);**
    } // squarethem

  and assume the following calling sequence:
  - **int alpha = 2;**
    squarethem (&alpha, &alpha);

Passing by reference



Passing by value and result



Second question

- What will be the value of **alpha *after the call*** assuming that the call was:
  - A ***conventional procedure call***?
    - **alpha = 2×2×4 = 16**
  - A ***remote procedure call***?
    - **alpha = 2×2 = 4**

Alternate second question

- Assume now **alpha = 3**
- What will be the value of **alpha *after the call*** assuming that the call was:
  - A ***conventional procedure call***?
    - **alpha = 3×3×9 = 81**
  - A ***remote procedure call***?
    - **alpha = 3×3 = 9**

Third question

- Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- Most programmers like to put all their signal operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize binary semaphores.
- You cannot combine non-blocking sends and blocking receives.

Third question

- ***TRUE:*** Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- Most programmers like to put all their signal operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize binary semaphores.
- You cannot combine non-blocking sends and blocking receives.

Third question

- ***TRUE:*** Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- ***TRUE:*** Most programmers like to put all their signal operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize binary semaphores.
- You cannot combine non-blocking sends and blocking receives.

Third question

- ***TRUE:*** Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- ***TRUE:*** Most programmers like to put all their signal operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize binary semaphores.
- You cannot combine non-blocking sends and blocking receives.

Third question

- ***TRUE:*** Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- ***TRUE:*** Most programmers like to put all their signal operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- ***FALSE:*** One cannot initialize binary semaphores.
- You cannot combine non-blocking sends and blocking receives

Third question

- ***TRUE:*** Most scheduling policies decrease the priority of processes that have exhausted their slice of CPU time.
- ***TRUE:*** Most programmers like to put all their signal operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- ***FALSE:*** One cannot initialize binary semaphores.
- ***FALSE:***You cannot combine non-blocking sends and blocking receives.

Alternate third question

- Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- Most programmers like to put all their notify operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize monitor conditions.
- You cannot combine non-blocking sends and blocking receives.

Alternate third question

- ***FALSE:*** Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- Most programmers like to put all their notify operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize monitor conditions.
- You cannot combine non-blocking sends and blocking receives.

Alternate third question

- ***FALSE:*** Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- ***FALSE:*** Most programmers like to put all their notify operations at the end of their monitor procedures.
- Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize monitor conditions.
- You cannot combine non-blocking sends and blocking receives.

Alternate third question

- ***FALSE:*** Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- ***FALSE:*** Most programmers like to put all their notify operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- One cannot initialize monitor conditions.
- You cannot combine non-blocking sends and blocking receives.

Alternate third question

- ***FALSE:*** Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- ***FALSE:*** Most programmers like to put all their notify operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- ***TRUE:*** One cannot initialize monitor conditions.
- You cannot combine non-blocking sends and blocking receives,

Alternate third question

- ***FALSE:*** Most scheduling policies increase the priority of processes that have exhausted their slice of CPU time.
- ***FALSE:*** Most programmers like to put all their notify operations at the end of their monitor procedures.
- ***TRUE:*** Peterson's algorithm assumes the existence of shared variables.
- ***TRUE:*** One cannot initialize monitor conditions.
- ***FALSE:*** You cannot combine non-blocking sends and blocking receives,

Fourth question

- A cruising boat can carry up to 80 passengers. These passengers can embark or debark through a narrow gangway that can accommodate one person at a time
- Complete the two following monitor procedures to ensure that neither the boat nor its gangway will ever be overloaded.
- **class Boat {**
  private int npassengers ;
  private condition notfull;

Answer

```
public synchronized void embark(){
        if (npassengers == 80)
            notfull.wait;
                npassengers++;
        walk();
    } //embark
public synchronized void debark(){
        walk();
        npassengers--;
        notfull.signal;
} //debark
```

Fifth question

- What are the sole correct values of X, Y and Z in the following System V.4 scheduler:
- ```
  #ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait  LVL
     1000        X         1         50000       1     #  0
      500        0         2         20000       2     #  1
      200        1         3         10000       3     #  2
      100        2         Y         10000       Z     #  3
  ```
  ```

Fifth question

- What are the sole correct values of X, Y and Z in the following System V.4 scheduler:
- ```
  #ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait  LVL
     1000        X         1         50000       1     #  0
      500        0         2         20000       2     #  1
      200        1         3         10000       3     #  2
      100        2         Y         10000       Z     #  3
  ```
- **X = 0**
- **Y = 3**
- **Z = 3**
  ```

Alternate fifth question

- What are the sole correct values of X, Y and Z in the following System V.4 scheduler:
- ```
  #ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LVL
     1000        X         1         50000       1     #  0
      500        0         2         20000       2     #  1
      200        1         3         10000       3     #  2
      100        2         4         10000       4     #  3
      100        3         Y         10000       Z     #  4
  ```
  ```

Alternate fifth question

- What are the sole correct values of X, Y and Z in the following System V.4 scheduler:
- ```
  #ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LVL
     1000        X         1         50000       1     #  0
      500        0         2         20000       2     #  1
      200        1         3         10000       3     #  2
      100        2         4         10000       4     #  3
      100        3         Y         10000       Z     #  4
  ```
- **X = 0**
- **Y = 4**
- **Z = 4**
  ```

Sixth question

- What is the ***main disadvantage*** of the ***round-robin*** CPU scheduling policy?
  - **It causes too may context switches when the system is heavily loaded.**

Sixth question

- Why does the ***web http protocol*** use streams instead of datagrams?
  - **Because replies from an http server will not always fit in a single packet and we want these packets to arrive to the client in order without lost packets, damaged packets or duplicates.**

Sixth question

- What is the ***main disadvantage*** of ***non preemptive*** CPU scheduling policies?
  - **They let CPU-bound processes monopolize the CPUs.**

Sixth question

- What is the main disadvantage of ***spin locks***?
  - **They waste CPU cycles while waiting for the lock (and generate context switches).**

Sixth question

- What is the difference between ***virtual circuits*** and ***streams***?
  - **Virtual circuits preserve message boundaries while streams do not.**

Sixth question

- What us the difference between a ***blocking receive*** and a ***non-blocking receive***?
  - **A blocking receive waits until the process receives a message while a non-blocking receive does not.**

Sixth question

- What us the difference between a ***blocking send*** and a ***non-blocking send***?
  - **A blocking send does not return until the message has been delivered to its recipient.**

Sixth question

- How can you implement the ***at most once semantics*** in a remote procedure call package?
  - **We should attach a sequence number to each message sent by a specific client and instruct the server to reject requests with duplicate sequence numbers.**