Operating System
10610 CS342302
Prof. Pai H. Chou
TA:

Hsu PO-CHUN
Student ID: s105062803
Submitted: Oct. 22, 2017
Due Date: Oct. 22, 2017

Assignment 5

1 Problem 1

[20 points] Prove that the algorithm satisfies all three requirements for the critical-section problem.

```
1.do {
2.   flag[i] = true;
3.   while (flag[j]) {
4.      if (turn == j) {
5.         flag[i] = false;
6.         while (turn == j) ; /* do nothing */
7.         flag[i] = true;
8.      }
9.    }
    /* critical section */
10.    turn = j;
11.    flag[i] = false;
    /* remainder section */
12.} while (true);
```

1.1 Answer

The three requirements fir the CS problem are mutual exclusion, progress and bounded waiting. For mutual exclusion, we note that each P_i enter CS only if flag[j]==false in Line3 or turn == i in Line6. And if both processes can enter CS at the same time, then the flag[0] is true and flag[1] is true as well in Line7. These imply that P_0 and P_1 cannot enter CS at the same time because turn can just be i or j not both. The while loop in Line3 repeat until flag[j] == false. The wile loop in Line6 does nothing until turn == i and set flag[i] = true in Line7. Thus, when one process is ready for enter the CS, the other process is not satisfied for the conditions to enter the CS which satisfy mutual exclusion.
And a process P_i is blocking to enter the CS with flag[j] == true or turn == j . Once P_i exits the CS , it will set turn == j in Line10 and set flg[i] = false in Line11, thus allowing P_j to enter its CS.(progress) after at most one entry by P_i(bounded waiting).#

2 Problem 2

[10 points] 6.4 Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user-level programs.

2.1 Answer

If a user-level program can disable the interrupts, it can disable the timer and the scheduler interrupts to prevent the context switches from being taken, it will break the synchronization mechanism.

[10 points] 6.23 How does the signal() operation associated with monitors differ from the corresponding operation defined for semaphores?

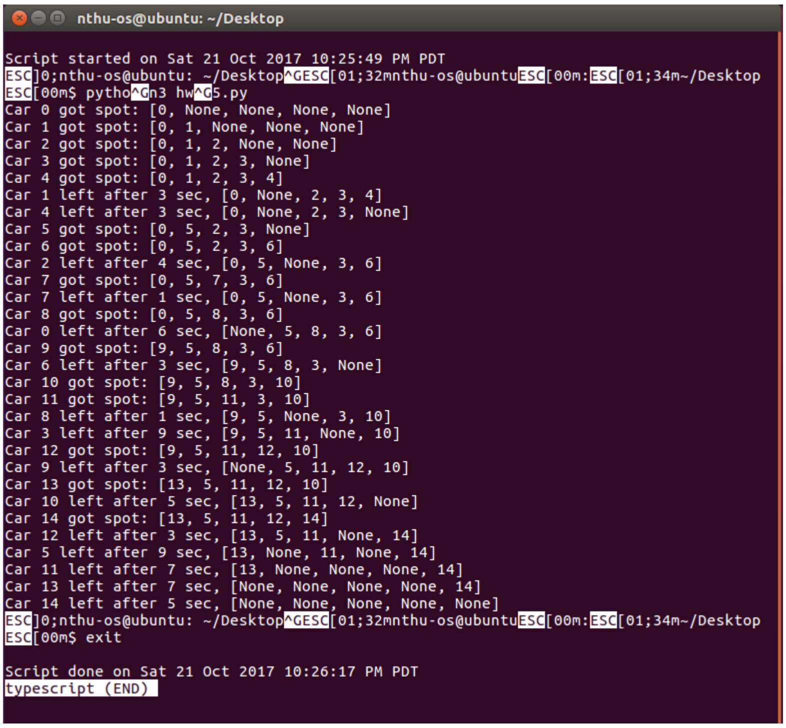
2.2 Answer

In monitor x.signal() operation resumes exactly one suspended process. If no process is suspended, the the x.signal has no effect. And the signal operation associated with semaphores always affects the state of the semaphore.

3 Program exercise 2.4

[5 points] Show your typescript. Run your code multiple times. Does it show the same or different output? Why?

3.1 Result



```
nthu-os@ubuntu: ~/Desktop

Script started on Sat 21 Oct 2017 10:25:49 PM PDT
ESC]0;nthu-os@ubuntu: ~/Desktop^GESC[01;32mnthu-os@ubuntuESC[00m:ESC[01;34m~/Desktop
ESC[00m$ pytho^Gn3 hw^G5.py
Car 0 got spot: [0, None, None, None, None]
Car 1 got spot: [0, 1, None, None, None]
Car 2 got spot: [0, 1, 2, None, None]
Car 3 got spot: [0, 1, 2, 3, None]
Car 4 got spot: [0, 1, 2, 3, 4]
Car 1 left after 3 sec, [0, None, 2, 3, 4]
Car 4 left after 3 sec, [0, None, 2, 3, None]
Car 5 got spot: [0, 5, 2, 3, None]
Car 6 got spot: [0, 5, 2, 3, 6]
Car 2 left after 4 sec, [0, 5, None, 3, 6]
Car 7 got spot: [0, 5, 7, 3, 6]
Car 7 left after 1 sec, [0, 5, None, 3, 6]
Car 8 got spot: [0, 5, 8, 3, 6]
Car 0 left after 6 sec, [None, 5, 8, 3, 6]
Car 9 got spot: [9, 5, 8, 3, 6]
Car 6 left after 3 sec, [9, 5, 8, 3, None]
Car 10 got spot: [9, 5, 8, 3, 10]
Car 11 got spot: [9, 5, 11, 3, 10]
Car 8 left after 1 sec, [9, 5, None, 3, 10]
Car 3 left after 9 sec, [9, 5, 11, None, 10]
Car 12 got spot: [9, 5, 11, 12, 10]
Car 9 left after 3 sec, [None, 5, 11, 12, 10]
Car 13 got spot: [13, 5, 11, 12, 10]
Car 10 left after 5 sec, [13, 5, 11, 12, None]
Car 14 got spot: [13, 5, 11, 12, 14]
Car 12 left after 3 sec, [13, 5, 11, None, 14]
Car 5 left after 9 sec, [13, None, 11, None, 14]
Car 11 left after 7 sec, [13, None, None, None, 14]
Car 13 left after 7 sec, [None, None, None, None, 14]
Car 14 left after 5 sec, [None, None, None, None, None]
ESC]0;nthu-os@ubuntu: ~/Desktop^GESC[01;32mnthu-os@ubuntuESC[00m:ESC[01;34m~/Desktop
ESC[00m$ exit

Script done on Sat 21 Oct 2017 10:26:17 PM PDT
typescript (END)
```

Figure 1: typescript

After running the code multiple times, it shows different outputs. Because the Car parking period is randomized by the program in the interval between 1 to 10 seconds. Every car parks within different times for each simulation, thus, it results in a different output.