MIIC1225 Operating Systems

MII MAHANAKORN Institute of Innovation

LAB Assignment Week #6

Topic: Race condition and critical-section problem

Name/ID of paticipant(s)

Instructions:

- You will be working in a group of two. Choosing your own team.
- You are allowed to discuss freely within your group. Avoiding seeking solution from other groups.
- Two computers are provided per group.
- Turn in the result by the end of class period.

Activity 1: Understanding race condition

- Run lab5_1.cpp (Linux) and lab5_2.cpp (Windows) and explain the results of both examples.
- Compare the situation against lab5_0.cpp and explain the problem in general.

Activity 2: Visiting Peterson's solution for critical-section problem

- Run lab5_3.cpp (Linux) and Lab5_4.cpp (Windows)
- Compare the result against the result from activity #1.

Activity 3: Using hardware interlocking mechanism for critical-section problem

- Run lab5_7.cpp (Linux) and Lab5_8.cpp (Windows) and observe the result
- Run lab5_9.cpp (Linux) and Lab5_9.cpp (Windows) and learn how we implement hardware interlocking mechanism in both POSIX and Windows environment
- Discuss any abnomality you might find from the result of both examples (lab5_9.cpp and lab5_10.cpp)
- Run lab5_11.cpp (Linux) and lab5_12.cpp (Windows) and learn how we implement an algorithm to correct the problem

Activity 4: Using interlocking mechanism provided by operating systems for critical-section problem

- Run lab5_13.cpp (Linux) and Lab5_14.cpp/lab5_15.cpp (Windows) and observe the result
- Run lab5_9.cpp (Linux) and Lab5_9.cpp (Windows) and learn how we implement hardware interlocking mechanism in both POSIX and Windows environment

Activity 5: Learning the basic concept of implementation of critical-section in your programs

- Use the program you created for activity #2 from week#4 laboratory.
- Change your program a little bit so that it would not affect the outcome of the program. (The modified one should run and produce the result the same as the original)

Original code	Modified code

while(1){
.....
scanf("%d",&x);
.....
if(x<0) break;
}
	int xx; //define a new local variable
	while(1){
	.....
	scanf("%d",&xx);
	x=xx;
	.....
	if(xx<0) break;
	}

```
If you defined two variables, the code should be something like this:
int xx,yy;
while(1){
   ......
   scanf("%d",&xx);
   scanf("%d",&yy);
   x = xx;
   y = yy;
   ......
   if(xx<0){break;}
}
```

- Then, apply entry section and exit section to your program using mutex locks (or CRITICALSECTION in windows). Considering there might be some rearrange the code or change something to suit the modification. Test the program to ensure the result and its behaviour are still the same.

Note: your program might not require the use of critical section. What we implement above is only to demonstrate how we implement critical section in a program.