EE4013 Assignment 1

Yashwanth G - EE18BTECH11007

Download all Codes from

https://github.com/yashwanthguguloth24/EE4013/tree/master/Assignment1/codes

Download all latex-tikz codes from

https://github.com/yashwanthguguloth24/EE4013/tree/master/Assignment1/Assignment1.tex

1 Problem

Consider the following multi-threaded code segment (in a mix of C and psuedo-code), invoked by two processes P1 and P2, and each of the processes spawns two threads T1 and T2:

```
int x = 0; // global
lock L1; // global
main(){
 create a thread to execute foo(); // Thread T1
 create a thread to execute foo(); // Thread T2
 wait for the two threads to finish execution;
 print(x);}

foo() {
 int y = 0;
 Acquire L1;
 x = x + 1;
 y = y + 1;
 Release L1;
 print(y);}
```

Which of the following statement(s) is/are correct?
(A) Both P1 and P2 will print the value of x as 2.
(B) At least one of P1 and P2 will print the value of x as 4.
(C) At least one of the threads will print the value of y as 2.
(D) Both T1 and T2, in both the processes will print the value of y as 1.

2 Solution

Both Options (A) and (D) are Correct.
In the question it is given that the code segment is invoked by 2 processes(a program in execution).
For the Process P_1:
1.Two threads are created in the main let they be T_{11} and T_{12}.
2.Both the threads executes the foo() function and they don't wait for each other.Due to the locking mechanism in the foo() function, as soon as one of the threads enters the foo() it locks the function and doesn't allow other thread to execute simultaneously.
3.Let T_{11} be the first thread to enter foo(),then it locks the function and increments **x to 1**, y being the local variable it prints **y as 1** and releases the lock.
4.As soon as the T_{11} releases the lock, T_{12} executes foo() by locking, incrementing **x to 2** (since x is global variable), printing **y as 1**(local variable), unlocking.
5.Finally, as soon as both the threads finishes their execution, **x** will be printed as **2**.
For the Process P_2:
The execution for P_2 will be same as P_1 because both the processes have different stack space, heap, text, data and also there is no part of the code which has mechanisms like shared memory,files etc.. so,both the processes will run independently.
Options:
(A) **True**, because both the processes being independent have their own copy of variables so,they finally print x as 2.
(B) **False**, as both proccess print x as 2, none of them prints x as 4.
(C) **False**, since y is an local variable so both the threads have their own copy of variable y, which they would increment to 1 and print them as 1.
(D) **True**, both threads will print x as 1 as mentioned above in (C) option.

3 Implementation

As the code-segment was invoked by 2 processes, the following C code

https://github.com/yashwanthguguloth24/EE4013/tree/master/Assignment1/codes/invoke.c

invokes the following C code(multithreading.c) with two processes(creating using fork()) via executable file, which represents the given code-segment with multiple threads

https://github.com/yashwanthguguloth24/EE4013/tree/master/Assignment1/codes/multithreading.c

The code invoke.c(creating multiple processes) produces the following output:

```
**********************************
Executing Process P1....
value of y is 1
value of y is 1
value of x is 2
Finished executing process P1.
**********************************
Executing Process P2....
value of y is 1
value of y is 1
value of x is 2
Finished executing process P2.
**********************************
```

Finally, verfying the output with given options,gives **(A)** and **(D)** as correct answer.