



Review Test Submission: Quiz 2

User	Sim Jia Ren .
Course	2110 ISTD - 50.005 : Computer System Engineering
Test	Quiz 2
Started	2/18/21 9:00 AM
Submitted	2/18/21 9:14 AM
Due Date	2/18/21 9:16 AM
Status	Completed
Attempt Score	10 out of 15 points
Time Elapsed	13 minutes
Results Displayed	All Answers, Submitted Answers, Correct Answers

Question 1

2 out of 2 points

Consider a class called Lock that implements a function called `getAndSet(boolean new_value)` thats guaranteed to be **atomic** by hardware although it consists of multiple instructions.

Then a thread A utilizes lock (an instance of Lock that's shared by all threads) as follows:

```
while (true){  
    while (lock.getAndSet(true)) Thread.yield();  
    //.. CS here  
    lock.set(false);  
    // ... remainder section  
}
```

When thread calls **yield**, it remains runnable but it yields control to **CPU scheduler** to pick the next thread to run.

Which of the following correctness properties are guaranteed by the above solution? Select ALL that applies. Selecting an incorrect answer will result in penalty. the minimum score for this question is 0.

← OK

Selected Answers: ☒ Mutual Exclusion
☒ Progress
Answers: ☒ Mutual Exclusion
☐ Bounded Waiting
☒ Progress

Question 2

0 out of 1 points

_____ involves saving the PCB of a process and restoring the saved PCB of another process. It is a key mechanism that allows the kernel to switch the use of the CPU among different processes.

Selected Answer: ☒ Mode switch
Answers: ☒ Context switch
☐ Interrupt Hardware
☐ Process Control Block
☐ Mode switch

Question 3

1 out of 1 points

A process' address space is private and not accessible from other processes by default.

Selected Answer: ☒ True
Answers: ☒ True
☐ False

Question 4

0 out of 1 points

The time taken to perform context switch is dependent on hardware support.

Selected Answer: ☒ False
Answers: ☒ True
☐ False

Question 5

1 out of 1 points

A buggy thread in a process can corrupt / affect the stack of another thread in the same process.

Selected Answer: ☒ True

Answers: ☒ True
☐ False

Question 6

1 out of 1 points

Threads allow support for concurrency without protection (protection in the sense like how processes are isolated from one another).

Selected Answer: ☒ True

Answers: ☒ True
☐ False

Question 7

1 out of 1 points

For a uniprocessor system, concurrency is an illusion.

Selected Answer: ☒ True

Answers: ☒ True
☐ False

Question 8

1 out of 1 points

Assume that a system call to:

- Create and attach a shared memory segment of size 2048 bytes takes 200 CPU clock cycles
- Send 128 bytes of data via message passing / socket takes 10 CPU clock cycles
- Receive 128 bytes of data via message passing / socket takes 10 CPU clock cycles

You can ignore the time taken to synchronize between communicating processes, and other socket system calls to create, listen, accept, and connect. Loading or storing 2048 bytes of data to the shared memory takes 5 CPU clock cycles.

Suppose process i want to send 2048 bytes of data on its **stack** to another process j. How many CPU clock cycles will it take to transfer the data to process j's **stack** if it were to use shared memory method instead? Provide your answer in terms of numbers only, e.g: 10.

Selected Answer: ☒ 210

Correct Answer: ☒ 210

Answer range +/- 0 (210 - 210)

Question 9

1 out of 1 points

If a process performs a *signal/release* operation on a counting semaphore whose value is **zero**, and there are one or more other processes waiting on the semaphore, exactly **one** of the other processes will be unblocked.

Selected Answer: ☒ True

Answers: ☒ True
☐ False

Question 10

2 out of 2 points

Consider the code snippet below:

```
int main ()
{
    pid_t pid;
    /* fork another process */
    pid = fork () ; /* both parent & child return from fork() */
    if (pid < 0 )
    { /* error occurred */
        fprintf (stderr, "Fork Failed" ) ;
        exit ( - 1 ) ;
    }
    else if (pid > 0 )
    {
        execlp ( "/bin/ls" , "ls" , NULL ) ; // runs ls program instead of shown code
        printf ( "ls has returned." )
    }
    else
    {
        wait ( NULL ) ;
        printf ( "Exiting..." ) ;
        exit ( 0 ) ;
    }
}
```

What can be the **possible** outcome(s) from running the code snippet above? Assume that it compiles successfully.

Select **all correct answers**. Selecting an incorrect answer will result in penalty. The minimum score for this question is 0.

Selected Answers: ☒ 'ls' might be successfully executed if the binary exists in the given path.

☒ wait(NULL) will not block the calling process at all.

Answers: ☒ 'ls' might be successfully executed if the binary exists in the given path.

☐ The program will crash during runtime due the line wait(NULL)

☐ The message "child complete" might be printed on the terminal.

☒ wait(NULL) will not block the calling process at all.

Question 11

0 out of 1 points

Can a bug (malicious instructions) in a user Process A corrupt the operations of another user Process B?

Selected
Answer:



No. Because Process A and Process B might run on different cores.

Answers:



No. Both processes run on different VM and are isolated from one another.

Yes. Process A can turn into Kernel mode and corrupt Process B.

No. Because Process A and Process B might run on different cores.

Yes with the help of shared memory or message passing mechanism.

Question 12

0 out of 2 points

Consider the following C-code:

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(){
    int x = 5;

    pid_t pid = fork();


    if (pid < 0){
        perror("Fork fails");
        exit(1);
    }


    x ++;

    if (pid == 0){
        x--;
        printf("The value of x is : %d\n", x);
    }

    return 0;
}
```

Both parent and child process increases the value of x. The value of x that will be printed out from the above code is _____. (Provide numerical answer only, e.g: 1).

Selected Answer:  6

Correct Answer:  5

Answer range +/- 0 (5 - 5)

Sunday, February 28, 2021 6:02:45 PM SGT