- ○ Anything between 0 and 100

The function of the accumulator register in Intel X86 is as follows:

- ○ Performing ALU operations and storing the results.
- ○ Performing memory read/write operations.
- ○ Deciding if the outcomes of ALU operations for the sake of making appropriate jumps and / or returns.
- ◉ None of the options.

Suppose you call the write() system call to write "ABCD" to an empty file and then immediately a different program reads the same file. What is the expected behavior?

- ◉ (A) It will read "ABCD"
- ○ (B) It will find the file empty
- ○ (C) It will get an error
- ○ (D) Either (A) or (B)

Suppose a parent process quits without waiting for child process. Who inherits the child process?

- ◉ The init process
- ○ The closest sub-reaper process on the process tree
- ○ The kthreadd process
- ○ None of the options

What is the output of the following program?

```
void swap (int a, int b) {
                int c;
                c = b;
                b = a;
                a = c;
}

int main (void) {
                int x=3, y=4;
                swap (x, y);
                print ( "%d %d" , x, y);
}
```

- ○ 4 3
- ◉ 3 4
- ○ 3 3
- ○ 4 4

What is the time complexity of current Linux scheduler?

- ○ O(1)
- ◉ O(log n)
- ○ O(n)
- ○ O(n^2)

Suppose you create a thread using the pthread interface. Which of the following system calls is used by Linux to implement this action?

- ○ fork
- ◉ clone

- ○ exec
- ○ none of the options, as pthread is a library function

Suppose we write a program hello.c in C on our personal computer. We first (1) run the preprocessor to get hello_ex.c, (2) then compile it to assembly get hello_ex.S, (3) assemble it to get hello_ex.o and (4) finally link it to get hello.out as executable. Suppose you move all these files starting from hello.c to a Raspberry Pi, with ARM processor. From which step would you need to re-run to ensure that the executable runs on the Raspberry Pi?

- ○ Step (1)
- ○ Step (2)
- ◉ Step (3)
- ○ You cannot run the same program on Raspberry Pi

Suppose a child process does not want the parent to wait for its completion. Is this possible, and if so, how?

- ◉ The child process can send a signal to the parent process
- ○ The child process can call another fork and then quit
- ○ The child process can call wait
- ○ This is not possible using any technique

Which of the following assembly language instructions are incorrect?

- ○ ADD rax, [rbx]
- ○ ADD [rax], rbx
- ◉ ADD [rax], [rbx]
- ○ None of the options

Fork(), pthread_create() and vfork() differ in the following ways:

- ○ Vfork() does a lazy copy while fork() creates copies of all program memories – code, stack, data, BSS, heap and RODATA.
- ○ pthread_create() and vfork() do a lazy copy while fork() creates copies of all program memories – code, stack, data, BSS, heap and RODATA.
- ◉ Both vfork() and fork() do the same thing. Pthread_create() however is different, in that it is used for thread creation.
- ○ None of the options

Which of the following is true about the `init' process.

- ○ It is a kernel thread.
- ○ It is a background process (zombie) and no PCB is allocated for it.
- ○ It does not have any PID, as it is a kernel thread, which anyways do not have PIDs associated.
- ◉ It does not itself handle any signals, but allows child processes to do so.

Which of the following is NOT true about most process scheduling schemes:

- ☑ May take into consideration potential race conditions.
- ☑ Takes into consideration blocking system calls.
- ☑ The task scheduler also needs to take care of process memory allocation, so as to improve task scheduling efficiency, upon process creation.
- ☐ Usually makes use of process priorities to allocate appropriate timeslices to various processes.
- ☐ None of the options

FIFOs differ from sockets in the following ways:

- ○ FIFOs work through message passing schemes while sockets communicate via byte streams.
- ○

Read and write to a FIFO cannot be pre-empted as it involves pre-empting and context switching two processes, unlike the case of sockets where the messages are queued.

Bytes are queued in a FIFO, unlike that in a socket, where unread messages are lost.

Because Bytes are queued in a FIFO, unlike that in a socket, where unread messages are lost. FIFOs cannot be used for real time communication, while sockets can.

None of the options

Which of the following cannot be used while modifying or adding code to the Linux kernel?

C data types

structure

Library function

Macros

What of the following is one of the disadvantages of having a monolithic kernel, like Linux?

Slow performance

Requires assembly programming

Lack of support for isolation of execution among modules

Unsuitable for embedded systems

Suppose a new Intel processor adds an additional opcode which the Linux kernel wants to use. How would it be able to use it?

Using a C statement

Using a system call

Using inline assembly

- (●) Cannot use it until compiler supports

Suppose you have an array of size 10. You try to access position 15 in C. What behavior do you expect?

- ( ) Segmentation fault
- ( ) Return of garbage value
- ( ) Process would abort
- (●) Either Segmentation fault or return of garbage value

Which of the following system calls returns control from the function only in case of failure?

- ( ) Fork
- ( ) Clone
- (●) Exec
- ( ) Wait

Suppose you want to run a (a) 32-bit executable program on a 64-bit Intel processor, and (b) a 64-bit executable program on a 32-bit Intel processor. Which of the following is correct, without using any simulator or emulator?

- ( ) Both (a) and (b) are impossible
- (●) (a) is possible, but (b) is impossible
- ( ) (a) is impossible, but (b) is possible
- ( ) Both (a) and (b) are possible