CS/CE 232/324-L4 Operating Systems – Fall 2023

Instructor: Dr. Munzir Zafar

Date: 29th August, 2023

Time = 10 minutes **Quiz 01** **Max Points: 100**

For each of the following questions, select only one correct answer from the multiple choices provided:

[Question 1] System calls are NOT:

A. An interface
B. Standard functions provided by an OS
C. Calls made by the system's physical components (I/O devices, memory, etc).
D. Used by application programs
E. To allow access to resources of the system

[Question 2] Virtualizing the CPU does NOT:

A. Allow many programs to run at once
B. Create an illusion that system has a very large number of CPUs
C. Give illusion to each program that it has its own independent CPU
D. Create multiple physical CPUs

[Question 3] Virtualizing memory does NOT:

A. Create separate physical memories dedicated to each program
B. Create an illusion that each program has its dedicated address space
C. Allow different programs to use the same address for their independent local variables
D. Map virtual addresses of the program to physical addresses in the memory

[Question 4] Two concurrently running threads share a counter. Each thread loads the variable, increments its value, and stores it back to memory. The threads will run correctly if:

A. Loading and incrementing occurs atomically (all at once) in each thread
B. Incrementing and storing occurs atomically in each thread
C. Loading, incrementing, and storing occur atomically in each thread
D. Without the need for any group of instructions to run atomically (i.e. without loading to be executed together with incrementing and storing, and so on)

[Question 5] The file system does NOT:

A. Figure out where on disk the new data will reside
B. Ensure that files are organized in the physical storage exactly like they appear to the user
C. Maintain data structures of its own
D. Have to keep track of where data resides on the disk through various structures

Program "cpu.c"

```
1  #include
2  #include
3  #include
4  #include
5  #include "common.h"
6
7  int
8  main(int argc, char *argv[])
9  {
10     if (argc != 2) {
11         fprintf(stderr, "usage: cpu \n");
12         exit(1);
13     }
14     char *str = argv[1];
15     while (1) {
16         Spin(1);
17         printf("%s\n", str);
18     }
19     return 0;
20 }
```

[Question 6] In "cpu.c", the statement on line 16, does the following:

A. Spin the hard disk by 1 block
B. Make the CPU rotate through all the running processes once before executing this process again
C. Wait for input of a single special character from the console
D. Wait for 1 second before proceeding forward

[Question 7] In "cpu.c", the statement on line 15 does the following:

A. Create an infinite while loop
B. Create a while loop that runs for a single iteration
C. Create a while loop that never executes
D. Creates a variant of an "if" statement whose condition is always true

Program "mem.c"

```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "common.h"
5
6  int
7  main(int argc, char *argv[])
8  {
9      int *p = malloc(sizeof(int));
10     assert(p != NULL);
11     printf("(%d) address pointed to by p:
%p\n",
12             getpid(), p);
13     *p = 0;
14     while (1) {
15         Spin(1);
16         *p = *p + 1;
17         printf("(%d) p: %d\n", getpid(), *p);
18     }
19     return 0;
20 }
```

[Question 8] In program "mem.c", the malloc() function on line 9, does this:

A. Locks the integer variable p to prevent access by other threads
B. Dynamically allocates memory the size of an integer
C. Statically allocates memory the size of an integer
D. Locks the integer pointer p to prevent access by other threads

[Question 9] In program "mem.c", line 13 does this:

A. Store 0 in the pointer p
B. Store 0 in the memory pointed by p
C. Make the size of memory allocated for p go to 0
D. Tell the compiler that the address of variable p is 0

[Question 10] In "mem.c", the getpid() function (lines 12 and 17) does this:

A. Returns the ID of the variable p maintained by the OS
B. Returns the ID of the program location in the file structure of the OS
C. Returns the ID of the process that is created as this program runs
D. Returns the ID of the pointer p in the physical address space