National University of Computer and Emerging Sciences

FAST School of Computing Spring-2023 Islamabad Campus

Question 1 [Multiple choice questions]

1. What is the main difference between a device controller and a device driver?
   a. A device controller is a software component that controls the device, while a device driver is a hardware component that communicates with the device.
   b. A device driver is a software component that controls the device, while a device controller is a hardware component that communicates with the device.
   c. A device controller is responsible for managing the device's resources, while a device driver is responsible for providing an interface between the operating system and the device controller.
   d. A device driver is responsible for managing the device's resources, while a device controller is responsible for providing an interface between the operating system and the device driver.
   e. None of the above

2. Which of the following statements is true regarding Interrupts without I/O and Interrupt-driven I/O?
   a. Interrupts without I/O are used to handle I/O operations, while Interrupt-driven I/O is used for non-I/O operations.
   b. Interrupts without I/O require more hardware resources than Interrupt-driven I/O.
   c. Interrupts without I/O are less efficient than Interrupt-driven I/O.
   d. Interrupts without I/O require the CPU to continuously check for events, while Interrupt-driven I/O allows the CPU to perform other tasks while waiting for events.
   e. None of the above.

3. Which of the following statements is true regarding Interrupt-driven I/O and Direct Memory Access (DMA)?
   a. Interrupt-driven I/O transfers data directly between I/O device and memory, while DMA transfers data between CPU and memory.
   b. Interrupt-driven I/O requires CPU involvement for each data transfer, while DMA does not require CPU involvement.
   c. Interrupt-driven I/O is faster than DMA.
   d. All of the above.
   e. None of the above.

4. Which of the following statements is true regarding spatial locality and temporal locality?
   a. Spatial locality refers to the tendency of a computer system to access data that is near to the data it has just accessed, while temporal locality refers to the tendency to access the same data repeatedly over time.
   b. Spatial locality refers to the tendency of a computer system to access data that is far from the data it has just accessed, while temporal locality refers to the tendency to access the same data repeatedly over time.
   c. Spatial locality and temporal locality both refer to the tendency to access the same data repeatedly over time.
   d. Spatial locality and temporal locality both refer to the tendency to access data that is near to the data it has just accessed.
   e. None of the above

5. A microkernel is a kernel ____.
   a. containing many components that are optimized to reduce resident memory size

b. that is compressed before loading in order to reduce its resident memory size
c. that is compiled to produce the smallest size possible when stored to disk
d. that is stripped of all nonessential components
e. All of the above

6. What happens when an interrupt is triggered during a process execution?
a. The interrupt handler is executed after the process completes its current instruction.
b. The interrupt handler is executed immediately, interrupting the current process execution.
c. The interrupt handler waits for the current process to finish before it is executed.
d. The interrupt handler is never executed during a process execution.
e. None of the above.

7. A process that has terminated, but whose parent has not yet called wait(), is known as a _____ process.
a. Zombie
b. orphan
c. terminated
d. main
e. None of the above

8. The _____ process is assigned as the parent to orphan processes.
a. Zombie
b. orphan
c. init
d. main
e. None of the above

9. Which of the following statements is true?
a. Shared memory is typically faster than message passing.
b. Message passing is typically faster than shared memory.
c. Message passing is most useful for exchanging large amounts of data.
d. Shared memory is far more common in operating systems than message passing.
e. All of the above

10. Which of the following statements is true?
a. Shared memory is typically faster than message passing.
b. Message passing is typically faster than shared memory.
c. Message passing is most useful for exchanging large amounts of data.
d. Shared memory is far more common in operating systems than message passing.
e. None of the above

11. In a(n) ____ temporary queue, the sender must always block until the recipient receives the message.
a. zero capacity
b. fixed capacity
c. variable capacity
d. bounded capacity
e. unbounded capacity

National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2023** **Islamabad Campus**

12. A process may transition to the Ready state by which of the following actions?
    a. Completion of an I/O event
    b. Completion of allocated time slice on the CPU
    c. Awaiting its turn on the CPU
    d. Newly-admitted process
    e. All of the above

13. The ____ of a process contains temporary data such as function parameters, return addresses, and local variables.
    a. text section
    b. data section
    c. program counter
    d. stack
    e. None of the above

14. A process control block ____.
    a. includes information on the process's state
    b. stores the address of the next instruction to be processed by a different process
    c. determines which process is to be executed next
    d. is an example of a process queue
    e. All of the above

15. The _____ refers to the number of processes in memory.
    a. process count
    b. process affinity
    c. degree of multiprocessing
    d. degree of multiprogramming
    e. None of the above

16. When a child process is created, which of the following is a possibility in terms of the execution or address space of the child process?
    a. The child process runs concurrently with the parent.
    b. The child process has a new program loaded into it.
    c. The child is a duplicate of the parent.
    d. The child gets a new process id.
    e. b & c only

17. A process is:
    a. Copy of a program.
    b. The same as a program.
    c. A program after compilation.
    d. Both b and c

e. None of the Above

18. Which of the following system call can be used to create a named pipe?
    a. mkfifo
    b. pipe
    c. get_named_pipe
    d. make_pipe
    e. None of the above

19. A named pipe is different from simple pipe in that:
    a. It is bidirectional instead of unidirectional
    b. It can be used for communication between processes on different machines
    c. It has a name in the filesystem
    d. both b & c
    e. none of the above

20. Which of the following segment stores the constants?
    a. data
    b. stack
    c. heap
    d. text
    e. b or c

21. What will happen to a zombie process if its parents get terminated without calling wait?
    a. It will always be zombie until we restart the system
    b. It will be adopted by *init* and then it will continue its execution
    c. It will always stay orphan
    d. Both a & c
    e. None of the above

22. Which of the following is true about BIOS
    a. Very useful part of hardware that can be used to carry IO operation
    b. buffer to efficiently store and access the data
    c. it is a very important software
    d. both a & b
    e. None of the above

23. Which of the following can be used for communication between two processes on two different machines having no parent child relationship?
    a. Pipe
    b. Named pipe
    c. Command line arguments
    d. Both a & b
    e. None of the above

24. In programmed I/O, the CPU is:
    a. Available for other tasks while waiting for I/O operations to complete

b. Not available for other tasks while waiting for I/O operations to complete
c. Available for other tasks after the I/O operation starts
d. Not available for other tasks after the I/O operation completes
e. Both a & c

25. In programmed I/O, the transfer of data between the CPU and the I/O device is performed by:
a. The I/O device
b. The CPU
c. The DMA controller
d. The operating system
e. None of the above

26. Which of the following program saves context of one process and reloads the context of another?
a. Context switcher
b. Switcher
c. Dispatcher
d. Scheduler
e. Both a & c

27. Which of the following cache is closest to memory?
a. L1
b. C1
c. L2
d. L3
e. M2

28. Which of the following requires full path to the executable?
a. execl
b. execv
c. execlp
d. Both a & b
e. execvp

29. What will happen when a child process, executing a new program using exec, calls exit.
a. It will become orphan
b. Parent process will receive its status
c. It will become zombie if its parent has not called exit
d. Both b & c
e. Its exit status will be lost

30. What is the maximum value of status that can be returned by exit(status) in Unix.
a. Maximum value that can be stored in 4 bytes signed
b. Maximum value that can be stored in 4 bytes unsigned
c. Maximum value that can be stored in 1 byte unsigned
d. Maximum value that can be stored in 1 byte signed

e. None of the above

31. What is the role of a driver in the context of a device controller?
    a. It provides power to the device
    b. acts as an interface between the hardware and the operating system
    c. stores data for the device
    d. It physically connects the device to the computer
    e. Both a & c

32. Which of the following is not true about processes?
    a. A process can communicate with processes running on different machines
    b. A process can create other processes using fork
    c. A program can create multiple processes
    d. A process can be created through more than one programs
    e. Both c & d

33. What can be done if a named pipe is not needed anymore
    a. It will be deleted automatically if no process is using it
    b. It will be deleted automatically after execution of communicating processes.
    c. We have to manually remove it from the file system
    d. We have to use unlink system call to remove it
    e. Both c & d

34. What will happen if we run the program below:

```
int main() {

  int fd = open("in.txt", O_RDWR | O_CREAT | O_TRUNC, 0666);

  dup2(fd, STDOUT_FILENO);

  printf("Writing %d\n",fd);

  close(fd);

  return 0;
}
```

    a. It will read the content of in.txt and display it on terminal
    b. It will display Writing followed by the file descriptor value on terminal
    c. It will display Writing followed by the file descriptor value on in.txt
    d. both a & b
    e. both a & c

35. What will happen if we run the program below:

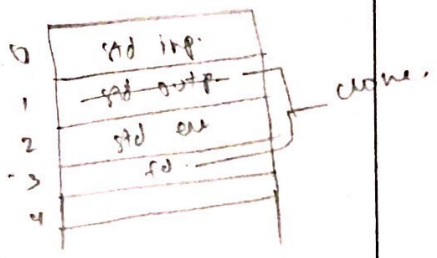
```
int main() {

  int fd = open("in.txt", O_RDWR | O_CREAT | O_TRUNC, 0666);
```

```
dup(fd);

printf("Writing %d\n",fd);

close(fd);

return 0;
}
```

a. It will read the content of in.txt and display it on terminal
b. It will display Writing followed by the file descriptor value on terminal
c. It will display Writing followed by the file descriptor value on in.txt
d. both a & b
e. both a & c

36. What will happen if we run the program below:

```
int main() {

  int fd = open("in.txt", O_RDWR | O_CREAT | O_TRUNC, 0666);

  close(1);

  dup(fd);

  printf("Writing %d\n",fd);

  close(fd);

  return 0;
}
```



a. It will read the content of in.txt and display it on terminal
b. It will display Writing followed by the file descriptor value on terminal
c. It will display Writing followed by the file descriptor value on in.txt
d. both a & b
e. both a & c

37. What will happen if we run the program below:

```
int main()
    {

        int pfds[2];

        char buff[5];
        int test = 10;

        pipe(pfds);
```

```
        write(pfds[1], test, 5);

        read(pfds[0], buf, 5);

        printf("read %s\n", buf);

    }
```

a. program will display "test"
b. program will display the data read from pipe
c. program will display "read "followed by 10
d. program will display read followed by "test"
e. None of the above

38. What will be the output of the following code snippet. Assume there are no syntax errors, and the program executes correctly.

```
#include    <unistd.h>
#include    <sys/wait.h>

//    W(A)  means write(1,    "A",  sizeof      "A")
#define     W(x)  write(1,    #x,   sizeof      #x)

int   main()
{
    W(A);
    Int child =   fork();
    W(B);
    If (child)
        wait(NULL);
    W(C);
}
```

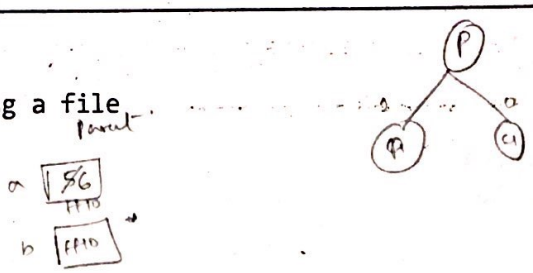
a. ABBCC
b. ABCBC
c. ABBCC or ABCBC
d. None of above

Consider the following code snippet. A parent process P that has forked a child process C in the program below.

After the new process is forked, suppose that the parent process is scheduled first, before the child process. Once the parent resumes after fork, it closes the file descriptor and changes the value of a variable as shown above. Assume that the child process is scheduled for the first time only after the parent completes these two changes.

Answer the following questions.

```
01: int a = 5;
02: int *b = &a;
03: int fd = open(...) //opening a file
04: int ret = fork();
05: if(ret >0) {
06:     close(fd);
07:     *b = 6;
08:     ...
09: }
10: else if(ret==0) {
11:     printf("a=%d\n", a);
12:     read(fd, something);
13: }
```



39. What is the value of the variable a as printed in the child process, when it is scheduled next?
    a. 5
    b. 6
    c. Value of b is same in both processes.
    d. None of above

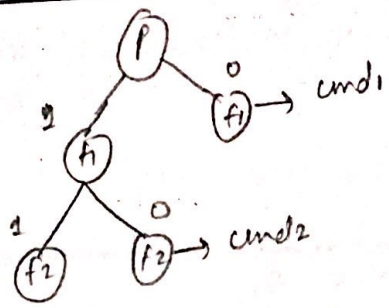
40. What will be the result of the execution of line 12, i.e., read ?
    a. Error: File is closed
    b. Segmentation fault
    c. Read will succeed.
    d. Read will not be able to fetch data from file

Answer should be c
Reference:
https://stackoverflow.com/questions/10198541/parent-child-process-close-file-descriptor

41. Consider a process P1 that forks P2, P2 forks P3, and P3 forks P4. P1 and P2 continue to execute while P3 terminates. Now, when P4 terminates, which process must wait for P4?
    a. P2
    b. P1
    c. Both P1 and P2
    d. init

42. Consider the following code snippet.

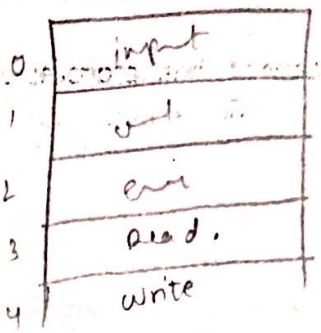
```
int rc1 = fork();
if(rc1 == 0) {
   exec(cmd1);
}
else {
   int rc2 = fork();
   if(rc2 == 0) {
      exec(cmd2);
   }
   else {
      wait();
      wait();
   }
}
```



43. Do the two commands cmd1 and cmd2 execute serially (one after the other) or in parallel?
    a. Serial
    b. Parallel
    c. Serial if one CPU, otherwise parallel
    d. None of above

44. Indicate how you would modify the code above to change the mode of execution from serial to parallel or vice versa.
    a. Using the wait system call
    b. Passing parameter to exec system call
    c. Changing the if/else structure of the program
    d. Changes at OS is required.

```
01: int fds[2];
02: pipe(fds);
03: pid = fork();
04: if (pid == 0) {
05:     dup2(pfd[1], 1);
06:
07:     if (execlp("ls", "ls", NULL) == -1)
08:         perror("execlp ls");
09: }
10: else {
11:     if (fork() == 0) {
12:         dup2(pfd[0], 0);
13:
14:         if (execlp("wc", "wc", "-l", NULL) == -1)
15:             perror("execlp wc");
16:     }
17:     else {
18:
19:
20:     }
21:
22: }
```



45. Which functionality is implemented by the above code snippet?
    a. ls | wc -l
    b. wc -l | ls
    c. Two separate processes wc and ls will execute.
    d. Pipe will not be used by any process

46. Select the correct option for the line 06 of the code snippet.
    a. close(pfd[0]);
    b. close(pfd[1]);
    c. close(pfd[0]); close(pfd[1]);
    d. No code is required.

47. Select the correct option for the line 13 of the code snippet.
    a. close(pfd[0]);
    b. close(pfd[1]);
    c. close(pfd[0]); close(pfd[1]);
    d. No code is required.

48. Select the correct option for the line 18 of the code snippet.
    a. close(pfd[0]);
    b. close(pfd[1]);
    c. close(pfd[0]); close(pfd[1]);
    d. No code is required.

49. Select the correct option for the line 19 of the code snippet.
    a. wait(NULL);
    b. wait(NULL); wait(NULL);
    c. wait statement can only be written at line 21
    d. No code is required.

50. Consider a simple linux shell implementing the command sleep 100. Which of the following is an accurate (ordered) list of system calls invoked by the shell from the time the user enters this command to the time the shell comes back and asks the user for the next input?
    a. sleep
    b. fork-sleep
    c. exec-wait
    d. fork-exec-wait

Answer: d