Roll Number: 20234757054

Master of Computer Applications
MCAC-403: Advanced Operating Systems
Unique Paper Code: 223401404
Semester IV
May-June-2022
Year of admission: 2020

Time: Three and half Hours **Max. Marks: 70**

Instructions:

- Attempt all the parts of a question together.
- Attempt any **FIVE** questions.

1. a. Explain the system call with its parameters that allows a process to change the size of its data region. Also, write a code snippet in *C/C++* to explain it. **(4)**

   b. What is the `callout` table, and how is it maintained in the kernel? Also, highlight the significance of a negative value in the `callout` table. **(4)**

   c. Consider the execution of four processes, namely, *A, B, C,* and *D*. Assume that these processes are created simultaneously with an initial priority of 60. The highest user level priority is 60. The clock interrupts the system 60 times a second, the processes make no system calls, and no other process is ready to run. Processes *A* and *B* belong to one group, while *C* and *D* belong to another group. Assume that the kernel schedules process *A* first. If two or more processes in the same group have equal priority, then the kernel picks processes in alphabetical order, i.e., process A will be scheduled first if both *A* and *B* have the same priority. Show the scheduling of these processes for 5 seconds using the principle of fair share scheduling with groups of processes. **(6)**

2. a. What is `SEM_UNDO` flag in `semop` system call and *undo structure* for semaphore in interprocess communication? Also, elaborate insertion/deletion in the *undo structure* with the help of an example. **(4)**

   b. Consider the following code. What are `getpgrp()` and `kill()` system calls used in the following *C* code snippet? Justify your answer. **(4)**

```
#include <signal.h>
main()
{
        register int i;

        setpgrp();
        for (i = 0;  i < 10;  i++)
        {
                if (fork() == 0)
                {

                        if (i & 1)
                                setpgrp();
                        printf("pid = %d pgrp = %d\n", getpid(), getpgrp());
                        pause();
                }
        }
        kill(0, SIGINT);
}
```

c. In a client-server architecture in the network communication system, write down the sequence of system calls at the client and server-side, assuming unreliable communication using *the User Datagram Protocol.* **(6)**

3. a. What is the `copy on write` bit in a page table? Explain its significance during fork system call with the help of an example, assuming a demand paging system. **(4)**

b. Assuming a demand paging system, write the steps carried out by the validity fault handler when a page that caused fault is **(4)**

(i) On the free page list in memory

(ii) Marked as "`demand Zero.`"

c. What is the role of `map` data structure used in swapping? Consider the following `map` data structure. Show the sequence of the changes in it during the following operations, assuming operations occurred sequentially: **(6)**

Address	Units
1	5000

(i) When kernel allocates 150 units

(ii) When kernel allocates 100 units

(iii) When kernel allocated 250 units

(iv) When kernel frees 150 units allocated during operation (i)

(v) When kernel allocates 200 units

4. a. Consider *inode lists* where block 4 is the beginning of the *inode list*. There are four *inodes* per block, a disk block size of 512 bytes, and each block can be addressed using 16 bits. Find the following: **(4)**

   (i) The logical disk block that contains the *inode* 13

   (ii) The block number and offset in the file if a process wants to access byte offset 7000.

b. Consider the following *C* code snippet. Show and explain the status of the kernel data structures after each system call. **(4)**

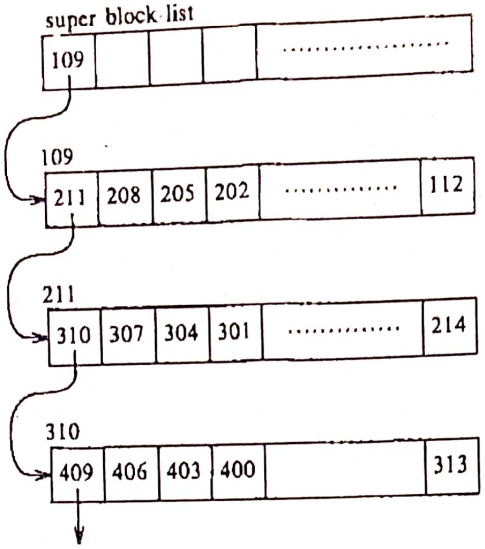
```
#include <fcntl.h>
main(argc, argv)
        int argc;
        char *argv[];
{
        int fd, skval;
        char c;

        if (argc != 2)
                exit();
        fd = open(argv[1], O_RDONLY);
        if (fd == -1)
                exit();
        while ((skval = read(fd, &c, 1)) == 1)
        {
                printf("char %c\n", c);
                skval = lseek(fd, 1023L, 1);
                printf("new seek val %d\n", skval);
        }
        close(fd);
}
```

c. What is the significance of `remembered` *inode* in assigning/freeing an *inode*? When will its value be modified? Consider the following configuration of the super block list for allocation/freeing of disk blocks. Show the changes in the configuration, assuming the requests are satisfied sequentially, when **(6)**

   (i) A disk block is allocated

   (ii) A disk block is allocated

   (iii) A disk block number 1000 is freed



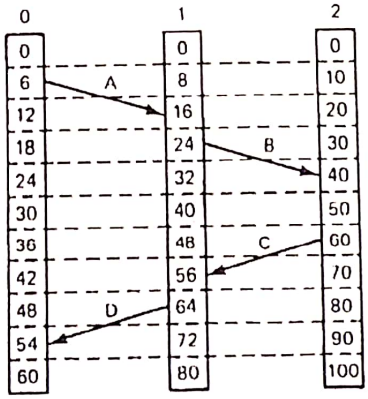
5. a. How Remote Procedure Call is different from the conventional procedure call. Also, draw the diagram to represent the state of stacks in the both schemes. **(4)**

b. Differentiate the following in context with the client-server model in distributive systems: **(4)**

(i) Blocking and non-blocking primitives

(ii) Buffered and unbuffered primitives

c. Explain Lampart's clock synchronization algorithm in the distributed systems. Consider the following three processes, each with its clock. Apply Lampart's clock synchronization algorithm to correct the clocks. **(6)**



6. a. Differentiate between bus-based multiprocessors and switched-based multiprocessors. Also, draw diagrams of their architecture. **(4)**

b. Explain the working principle of Page Stealer with its state diagram and when it'll be invoked. Consider the following page table where aging is shown. Assume that the threshold value is 3. Show the changes in the page **(4)**

state and time (last referenced) after one clock cycle, assuming the requested page is shown with a grey shade.

Page State	Time (Last Referenced)
In memory	0
	1
	2
	2
	3
	1

c. What is an unnamed pipe? Explain with the help of an example. In the following C code snippet, explain the statement highlighted in the box. (6)

```
#include <fcntl.h>
char string[] = "hello";
main(argc, argv)
        int argc;
        char *argv[];
{
        int fd;
        char buf[256];
        mknod("fifo", 010777, 0);
        if (argc == 2)
                fd = open("fifo", O_WRONLY);
        else
                fd = open("fifo", O_RDONLY);
        for (;;)
                if (argc == 2)
                        write(fd, string, 6);
                else
                        read(fd, buf, 6);
}
```