UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 6 Examination in Engineering: November 2024

Module Number: EE6206 **Module Name: Operating Systems Programming (C-18)**

[Three Hours]

[Answer **all questions**, each question carries 10 marks.]

Q1 (a) Briefly explain the meaning of following Linux terminal commands?

(i) sudo shutdown
(ii) sudo shutdown -r now
(iii) gnome-session-quit
(iv) cat Doc1.txt>>Doc2.txt
cat Doc2.txt
(v) cat Doc1.txt>Doc2.txt
cat Doc2.txt
(vi) cat Doc1.txt Doc2.txt
(vii) cat >Doc1.txt
ctrl+D
cat >Doc2.txt
ctrl+D
(viii) gedit prog1.c &
(ix) sudo adduser u1
(x) sudo apt-get install gcc **[5 Marks]**

(b) Write Linux console-based commands for the following sequence of operations?

(i) Go to **home** directory. Change the directory to **Documents**. Display the current working directory. Note that **Documents** is a directory inside the home directory.

(ii) Assuming there is a text file known as **txt1.text** inside **Documents** directory; create an exact copy of **txt1.text** as **txt2.text** in the previous (**home**) directory.

(iii) Go back to the **home** directory and copy the **Downloads** directory to inside the **Desktop** directory as **Downloads_copy**. Note that **Downloads and Desktop** are directories inside the home directory.

(iv) Delete **Downloads_copy**, go to the **Home** directory and list the details of **txt2.text** file in short format.

(v) Remove the read privilege of the user group (group) for **txt2.text** and add write privilege to other users (others).

(vi) Create a directory named **mydir2** inside **Downloads** directory. Go to that directory. After that create two empty text files as **Doc1.txt, Doc2.txt** inside **mydir2** using,

(I) **touch** command

(II) **cat** command

(vii) Assuming both **Doc1.txt** and **Doc2.txt** has content inside them, copy the content of **Doc1.txt** to **Doc2.txt** and display **Doc2.txt**.

(viii) Observe the statistics of **Doc1.txt.** Change the access time of **Doc1.txt.** Change the modification time of **Doc1.txt**

(ix) Assume there is a C program file called **my_program.c.** Write instructions to compile and run an object file with same name as **my_program.c.** Note that **my_program.c** contains header files for mathematical computations and multi-threading. **[5 Marks]**

Q2 (a) (i) State the meaning of non-recursive functions. **[1 Mark]**

(ii) State two (2) similarities between structures and unions. **[1 Mark]**

(iii) Consider the code snippet given in Listing Q2(a). State whether function1 implements pass by reference, or pass by value, or both. **[1.Mark]**

(iv) Write a C program to print 5 user input single precision floating point numbers to the console using the knowledge on arrays and **for loops** using a function called "**custom_func**". Include all header files in your program. **[2 Marks]**

(b) (i) State the corresponding function used to copy the content (bytes) from a memory buffer to a file in each of Low level I/O and High level I/O models. **[1 Mark]**

(ii) An incomplete code segment to copy doc1.txt into another file doc2.txt with essential error handling using low level file I/O functions is given in Listing Q2(b). Complete the missing lines of the code A to J. **[4 Marks]**

Q3 (a) (i) State what is meant by swap area. **[1 Mark]**

(ii) State two (2) differences and two (2) similarities between pipes and message queues. **[1 Mark]**

(iii) State the meaning of the following statement.

*heap_v = (int *)malloc(4);* **[1 Mark]**

(iv) State the two file descriptors associated with a pipe. **[1 Mark]**

(v) Consider the following function.

```
pid_t        waitpid(pid_t pid, int * status, int options)
```

Explain the behavior of the waitpid(pid, NULL, 0) under following cases for pid: -1, positive integer representing process ID of a child. **[1 Mark]**

(vi) A message queue is defined as follows.

```
struct msg
 {
     long type;
     char text[50];
 };
```

Briefly explain the purpose of the argument *type*. **[1 Mark]**

(b) Complete the missing lines A-J in the C program given in Listing Q3(b) to implement the following scenario.

Create a pipe to transfer the message "**message from parent**" from parent process to child process and display using terminal in the child process? **[4 Marks]**

Q4 (a) (i) Define what is a thread and a process separately. **[1 Mark]**

(ii) Specify two (2) differences and two (2) similarities between threads and processes. **[1 Mark]**

(iii) State 3 ways that can be used for inter process communication and 2 techniques of thread synchronization. **[1 Mark]**

(iv) Define simultaneous multi-threading. **[1 Mark]**

(v) Briefly explain what is meant by race conditions in hyper-threading. **[1 Mark]**

(vi) Briefly explain how a Mutex and condition variables can be used to avoid race conditions. **[1 Mark]**

(b) Complete the missing lines A-J in the incomplete C program given in Listing Q4(b) to implement the following scenario.

Write a program to create three threads. One thread runs a function which writes "Hello ", thread 2 should write "my name is " and thread3 should write "Khan". Serialize these threads using semaphores to get the correct result? **[4 Marks]**

Q5 (a) (i) State what is meant by SOCK_STREAM and SOCK_DGRAM **[1 Mark]**

(ii) List down the 7 attributes of a socket. **[1 Mark]**

(iii) State and briefly explain the little-endian byte order. **[1 Mark]**

(iv) The *socket( )* function called in the client is defined as follows.
int socket(int domain, int service_type, int protocol)
Briefly explain the socket function using the arguments given. **[1 Mark]**

(v) Briefly explain the meaning of the following two commands,
(I) *shutdown(socket_fd,* SHUT_RDWR)
(II) route -n **[1 Mark]**

(vi) State and briefly explain the server only functions in the correct sequence which they should be called in a server machine in client-server communication. **[1 Mark]**

(b) (i) Complete the missing lines A-J in the incomplete C program given in Listing Q5(b) to implement the following scenario.

Create a client program to connect to the **echo** service of a remote machine to send string **"HELLO"** and display the returned string on screen. Include error handling for sending only. The dotted decimal notation of the IPv4 address of the remote machine is **193.34.76.92.** The socket's domain is internet and its service type is stream. After receiving the reply from the remote machine, the client program must make sure that the socket is disabled for reading only for any process which has been connected to it. **[4 Marks]**

```
void function1(int x, int * y)
{
        x = x + *y;

}
int main( )
{
        function1(val1,&val2);
        return (0);
}
```

Listing Q2(a): Code snippet to implement a function.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
-----(A)-----
#include <sys/stat.h>
-----(B)-----
int main()
{
        int fd1;
        -----(C)-----
        if(fd1 == -1)

        {
                perror("doc1.txt:");
                printf("Error occured while opening doc1. Error number is %d \n",
                errno);
                -----(D)-----
        }
        else
        {
                printf("Successfully opened file 1");
                -----(E)-----
                int ret_val1;
                -----(F)-----
                if(ret_val1 == -1)
                {
                     -----(G)-----
                        printf("Error occured while reading from doc1. Error number is
                        %d \n", errno);
                        exit(1);
                }
                else
                {
                        printf("Read from file successfully");
                        -----(H)-----
                        int fd2;
                fd2 = open("doc2.txt",O_RDWR|O_CREAT|O_TRUNC,0666);
                if(fd2 == -1)
                {
                        perror("doc2.txt:");
                        printf("Error occured while creating doc2. Error number
                        is %d \n", errno);
                        exit(2);
                }
                else
                {
                        int ret_val2;

                        ------(I)------
                        if(ret_val2 == -1)
                        {
                                perror("doc2.txt:");
                                printf("Error occured while writing to doc2. Error
                                number is %d \n", errno);
                                exit(3);
                        }
                        ------(J)------
                        close(fd2);
                }
        }
    }
    return(0);
}
```

Listing Q2(b): Incomplete code snippet to Copy doc1.txt into another file doc2.txt with possible error handling using low level file I/O.

```
------(A)------
#include <stdlib.h>
#include <unistd.h>
------(B)------
#include <sys/wait.h>
int main()
{
    ------(C)------
if(pipe(pipe_fd) == -1)
{
        perror("pipe");
    -----D-----
}
else
{
        pid_t PID;
        {
        -----E-----
                case -1:
                        -----F-----
                        exit(0);
                        break;
                case 0:
                        -----G-----
                        sleep(2);
                        char buf[50];
                        -----H-----
                        printf("This is child process. Received message is %s
                        \n", buf);
                        exit(0);
                        break;
                default:
                        close(pipe_fd[0]);
                        -----I-----
                        write(pipe_fd[1],msg,sizeof(msg));
                        exit(0);
                        break;
        }
}
    -----J-----
}
```

Listing Q3(b): Incomplete code for inter-process communication using pipes.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/msg.h>
#include <string.h>

-----A-----

-----B-----
void * thread_function1(void * argumes);

-----C-----
void * thread_function3(void * argumes);

const char * name1 = "/sem1_name";

-----D-----
sem_t * sem1;
sem_t * sem2;
int main()
{
        sem1 = sem_open(name1,O_CREAT,0666,0);

        -----E-----
        pthread_t tid1,tid2,tid3;

        pthread_create(&tid1,NULL,thread_function1, NULL);

         -----F-----
        pthread_create(&tid3,NULL,thread_function3, NULL);
        pthread_join(tid1,NULL);
        pthread_join(tid2,NULL);

        -----G-----
        sem_close(sem1);
        sem_close(sem2);
        sem_unlink(name1);
        sem_unlink(name2);
        return(0);
}
```

```
void * thread_function1(void * argumes)
{
        printf("Hello ");
        -----(H)-----
        pthread_exit(NULL);
}
void * thread_function2(void * argumes)
{
        sem_wait(sem1);
        printf("my name is ");

        -----(I)-----
        pthread_exit(NULL);
}
void * thread_function3(void * argumes)
{

        -----(J)-----
        printf("Khan ");
        pthread_exit(NULL);
}
```

Listing Q4(b): Incomplete code for thread serialization.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/stat.h>
#include <fcntl.h>
----------(A)----------
#include <sys/types.h>
#include <netinet/in.h>
----------(B)----------
#include <string.h>
int main()
{
        int socket_fd;
        ---------------(C)-----------
        struct sockaddr_in remote_ad;
        remote_ad.sin_family = AF_INET;
        remote_ad.sin_port = htons(7);
        ----------------(D)---------------;
        memset(&remote_ad.sin_zero,'\0',sizeof(remote_ad.sin_zero));
        ------------------(E)---------------------;

        char buf[5] = --------(F)-----------
        int x;
        ---------(G)---------
        if(x > 0)
        {
                sleep(4);
                ---------(H)-------
                read(socket_fd, buf2, sizeof(buf2));
                printf("\nReceived: %s",buf2);
        }
        if(x == -1)
        {
                ---------(I)-----------
                printf("%d",errno);
                exit(0);

        }
        if(x == 0)
        {
                printf("nothing sent");

        }
        sleep(5);
        ----------------(J)--------------
        exit(0);
        return(0);

}
```

Listing Q5(b): Incomplete code to echo a message.