Read-Me
OS Assignment -3
Answer-2

Ashwin Sheoran
2020288

1. Unix Domain Sockets
   In process P1 50 random strings are created using srand(time(0)); the socket identifier is defined by using predefined struct identifier Sockaddr_un, in this program, the socket identifier is P1. The first member of this struct is the address family, The address family is defined as AF_UNIX which is used for local communication by instructions name.sun family = AF_UNIX; the second member of this struct is the path of socket special file. The path of the socket special sile is attached to the socket identifier by using strncpy instructions. 1st unlink is used to remove the IIITD.txt file if it already exists.
   The socket is created by socket(AF_UNIX, SOCK STREAM O) The first argument of this instruction is the address family, the second argument defines the type of socket here the socket is a stream socket. The return value of the instruction is an integer that is used for binding the socket with its identifier next is the bind system call which assigns the address specified by P1 the socket with instructions the system understands for which socket data is transferred. Next is the listen to system call which defines how many processes can communicate with process p1 here the value is one which means only one process can communicate with P1, the accept system call will return an identifier here it is P1 data socket. All data and communication will take place using P1 data _socket identifying write system call is used to write data from the string rdnm[i][j], to the socket, the process P1 gets locked at the read system call which reads data from the socket send by process P1 into the buffer, after reading the data form the process P2 will send back the highest index value of received strings to P1 by using the write system call.
2. FIFOs
   Process P1 creates 50 random strings and the size of each string is 5 character random strings and are created using inbuilt functions srand(time(0)); The strings are stored in a character array named “randarr”
   mkfifo(“fifo_p1”,0666); creates a FIFO special file “fifo_p1” with read-write permission to all.
   If make “fifo_p1” is successful the positive integer is returned and on failure “-1” is returned.
   Open(“fifo_p1”, O_WRONLY); this opens the “fifo_p1” in write-only mode, if open is successful then a positive integer is returned and on failure “-1” is returned.
   The value returned by the above open sys call is a file discriptor which will be used for further read write operations
   write(fd , randarr[i][j], sizeof(randarr[i][j]); This write syscall will write the string from “randarr[i][j]” to file descriptor “fd” .
   Now close(fd); will close the file descriptor.

Now execute the process P2, P2 opens the “fifo_p1” special file and read data from “fifo_p1” by executing.
read(fd, buff[1], sizeof(buff[a])); where the first argument is a name of a file descriptor and the second argument is the buffer into which string is stored and the third argument is the size of the buffer. After reading five consecutive strings file is closed. and the file is again opened in the write-only mode for sending the highest index value from the group of received strings. After returning the index value, the file is closed. This index value is read by read(id, &a, sizeof(int)); system call in process P1 and finally file is closed in P1.and “fifo_p1” special file was removed by unlink(“fifo_p1”);

3. Message Passing queues
   The message passing queues are used for transferring messages from one process to another process. Here P1 wites into the message queue, P2 reads from the message queue and sends an acknowledgement to P1. The message queues are identified by their key values and the message in the message queue is identified by its message type value.
   P1 executes it generates fifty random strings and the size of each string is 5 characters. The random strings are generated by using a system called srand(time(0)).
   A Struct mesg_buffer is created to define the message sent in the message queue. A variable “msgid” is created to hold the value of, message queue. Now to identify a message queue a key is generated using ftok(“IIITD.txt”,’ A’);, IIITD.txt file was created using system(); syscall and ‘A’ is an identifier. msgget(Key, 0666 | IPC_CREAT); is used to create a message queue. In msgget(); the first argument is key to identifying the message queue and the second argument
   IPC_CREAT is used to create a message queue along with permission.
   Strncpy command is used to copy the value of rdnm[i][j] to the defined message queue. The msgsnd(msgid, &message, sizeof(rdnm[i][j]), 0); The system call is used to send message to P2, mesg_type=1; is used to set the type value of the message which is 1 here,
   The msgid is used to describe the message queue containing the message to be transferred and the size of (rdnm[i][j]), defines the size of the message, now process P1 get blocked at msgrcv(); system call.
   Now P2 is executed and the message is received by msgrcv(msgid, &message, sizeof(message), 1, 0); first argument msgid is used to identify message queue, the second argument gives the address of the buffer where data to be stored, the Third argument defines the size of the message and the fourth argument defines the type of message. The msgrcv(); is a blocking system call that blocks the execution of the process till something is written in the queue. Now the greatest index value from the group of five received strings is sent to P1 in the same message queue as the message type 2. The index value in P1 is received by msgrcv (msgid, &message, sizeof(message), 2, 0); In this syscall, the first argument identifies the message queue, the second argument identifies the buffer from which message to be read thrice argument identifies the size of the message and fourth argument identifies the type of message to be received which is “2”.Finally msgctl(msgid, IPC_RMID, NULL); is used to

remove queue and unlink(“IIITD.txt”); is used to unlink IIITD.txt file, which was used in the key.