Assignment - III

1) Define signals. Categorize the ways in which a process can handle the signals.

Signals are software interrupts. Signals provide a way of handling asynchronous events: a user at a terminal typing the interrupt key to stop a program or the next program in a pipeline terminating prematurely.

Name	Description	Default action
SIGABRT	abnormal termination (abort)	terminate + core
SIGALRM	timer expired (alarm)	terminate.

When a signal is sent to a process. it is pending on the process to handle it. The process can react to pending signals in one of 3 ways.

* Accept the default action of the signal, which for most signals will terminate the process.
* Ignore the signal. The signal will be discarded & it has no affect whatsover on the recipient. process.
* Invoke a user defined function. The function is known as a signal handler routine & the signal is said to be caught when this function is called.

Q7 Write a program to check whether SIGINT signal is in present in a process signal mask & adds it to the mask if it is not there. It should clear SEGSGV signal from the process signal mask.

```
#include<stdio.h>
#include<signal.h>
int main()
{
    sigset_t sigmask;
    sigemptyset (& sigmask);

    if (sigprocmask( 0,0, & sigmask)==-1)
    {
        perror("sigprocmask");
        exit (1);
    }
    else sigaddset(& sigmask, SIGINT);
    sigdelset (& sigmask, SIGSEGV);
    if (sigprocmask(SIG_SETMASK, & sigmask,0)==-1)
    perror("sigpromask");
}
```

4) What is FIFO? Explain how it is used in IPC. Discuss with an example the client server communication using FIFO.

FIFO's are sometimes called named pipes. Pipes can't be used only b/w related processes when a common ancestor has created the pipe.

```
#include <sys/stat.h>
int mkfifo (const char *pathname, mode-t mode);
```

Returns: 0 if OK, 1 on error.

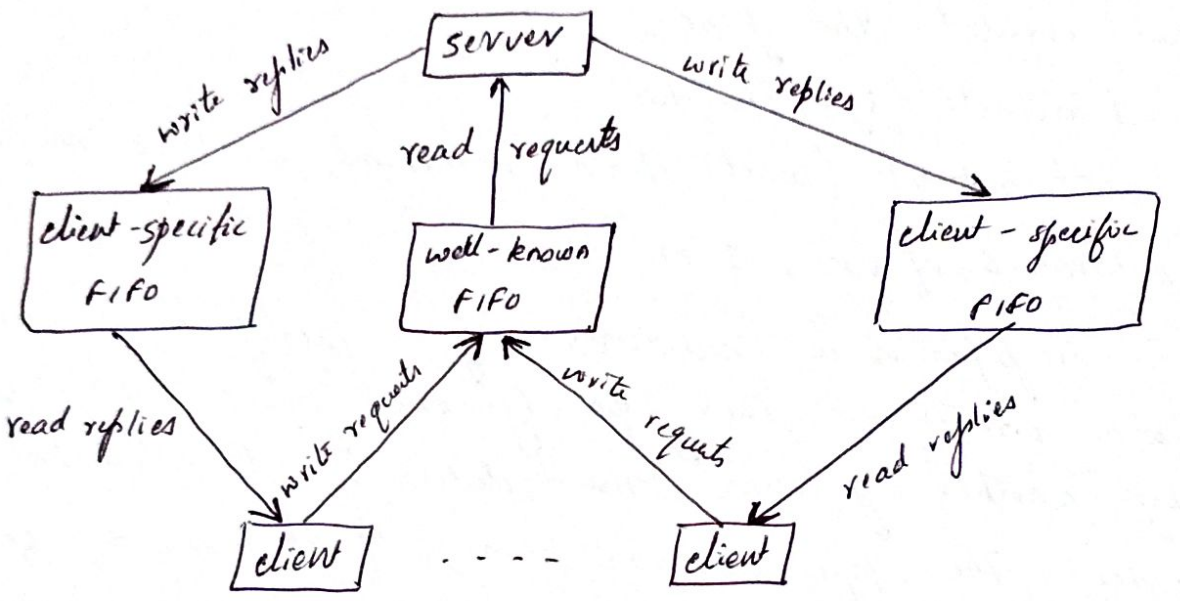
A pipe is a mechanism for interprocess communication data written to the pipe by one process can be read by another process. The data is handled in a FIFO order. The pipe has no name; it is created for one use & both ends must be inherited from the single process which created the pipe.

Example: Client-server communication using a FIFO

* FIFO's can be used to send a data b/w client & a server of a server ie contacted by many clients, each client can write its request to a FIFO that the server creates since there are multiple writers for the FIFO, the request sent by the clients to the server need to be less than PIPE-BUF bytes in size.
* This prevents any interleaving of the client writes. But the problem using FIFO is how to send replies back from the server to each client.

☒ A single FIFO can't be used, one soln is for each client to send its process ID with the request

☒ The server also must catch SIGPIPE, since it's possible for a client to send a request & terminate before reading the response.



3) Define message queue. Discuss how it is useful in inter-process communication.

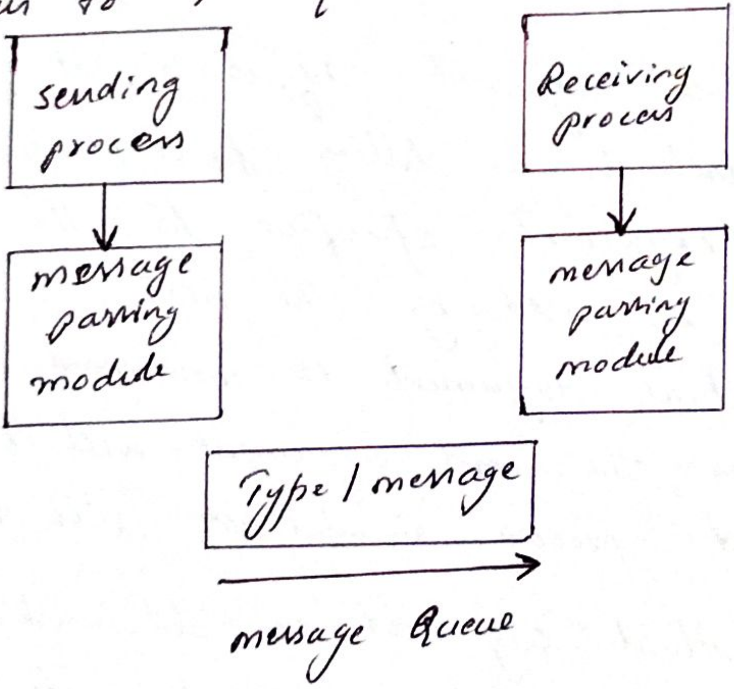
A message queue is a linked list of messages stored within the kernel & identified by a message queue identifier. We'll call the message queue just a queue & its identifier a queue ID.

A new queue is created or an existing queue opened by msgget().

New messages are added to the end of a queue by msgsnd().

Every message has a +ve long int type. field. a non-ve length of the actual data bytes, all of

which are specified to msgsnd()
When the message is added to a queue by msgrcvl don't have to fetch the messages in a FIFO order. instead fetch messages based on their type field.
All processes can exchange info. through access to a common system message queue. The sending process places a message onto a queue which can be read by another process.
Each message is given an identification or type so that processes can select the appropriate message.
Process must share a common key in order to gain access to the queue in the first place.



5) What is signal mask of a process?
Explain sigprocmask function along with its prototype.

A process initially inherits the parent's signal mask when it is created, but any pending signals for the parent process are not passed on.
A process may query or set its signal mask via the sigprocmask API.

```
#include <signal.h>
int sigprocmask(int cmd, const sigset_t *new-mask, sigset_t
                                              *old-mask);
```

Returns: 0 if ok, 1 on error.

The new-mask argument defines a set of signals to be set or reset in a calling process signal mask, & the cmd argument specifies how the new-mask value is to be used by the API.

If the actual argument to new-mask argument is a NULL pointer, the cmd argument will be ignored, & the current process signal mask will not be altered.

If the actual argument to old-mask argument is a NULL pointer, no previous signal mask will be returned.

The sigset_t contains a collection of bit flags.