Assignment-3

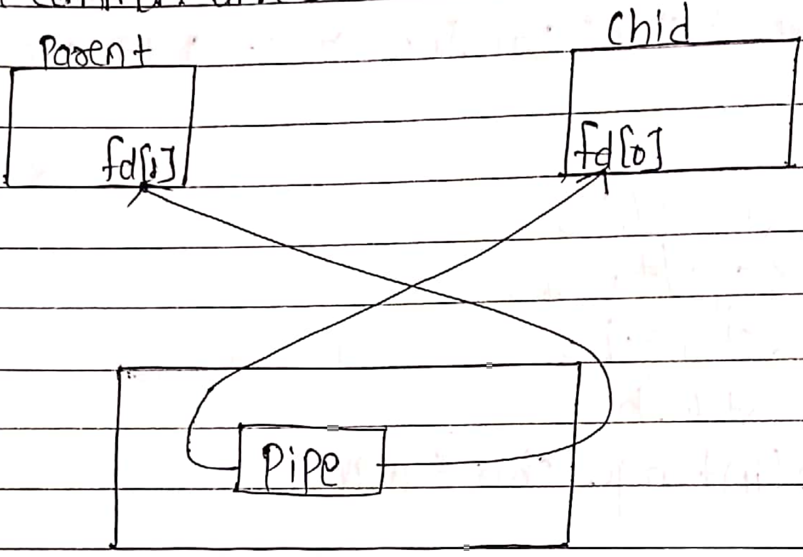
① what are pipes? what are limitations of pipes? write a program to create a pipe between parent & child.

Ans:- Pipes are the oldest form of unix system IPC and are provided by all UNIX systems.

limitations.

i. They have been half duplex.

ii. Pipes can be used only between processes that have a common ancestor.



pipe from parent to child

```
#include <unistd.h>
#include <stdio.h>
# include <stdlib.h>
int main(void)
 {
   int n ;
   int fd[2];
   Pid_t Pid;
   char line [MAXLINE];
```

```
if (pipe(fd)<0 {
    perror("pipe error");
if(pid =fork()) <0) {
    perror ("fork error");
} else if(pid>0) {
    close(fd[0]);
    write(fd[1], "hello world\n", 12);
} else {
    close (fd[1]);
    n=read (fd[0], line, MAXLINE);
    write (STDOUT_FIFENO, line, n);
} exit(0);
  }
```

② what are differences between fork and vfork functions.

	fork	v fork
i.	The system call, chid and parent process have separate memory space.	The system call, chid and parent process share same address space.
ii.	The child process and parent process gets executed simultaneously.	once child process is executed then parent process start its execution.
iii.	The fork() system call uses copy-on-writes as an alternative.	while vfork() system calls does not use copy-on-write.
iv.	page of one process is not affected by page of other process.	page of one process is affected by page of other process.
v.	There is wastage of address space.	There is no wastage of address space.

③ Explain memory layout of C Program. Also discuss different memory allocation used in unix.

Ans:- Memory layout of C Program are:-

(i) Text segment:-
The machine instructions that the CPU executes.
It is text segment is read-only.

Initialized data segmented:-
usually called simply the data segment, containing variable that are specifically initialized in the Program. For example:-

```
int maxcount = 99;
```

uninitialized data segment: often called the "bss" segment, named after an ancient assembler operator that stood for "block started by symbol.".
Ex:-

```
long sum [1000];
```

Stack:- where automatic variables are stored, along with information that is saved each time a function.

Heap:-
where dynamic memory allocation usually takes places. Historically, the heap has been located between the uninitialized data and the stack.

memory allocation:-
- C specifies three functions for memory allocation:

malloc:-which allocates a specified number of bytes of memory
calloc:- which allocates space for a specified number of object of a specified size.
realloc:- which increase or decrease the size of a previously allocated area.

(4) what are device file API? Illustrate the same with a program.

Ans:- Device files are used to interface physical device with application program.
A process with superuser privileges to create a device file must call the mknod API.
The prototype is:

```
      # include <unistd.h>
      # include <sys/stat.h>
int mknod (const char* path-name, mode_t mode, int deviceid);
```

Program,

```
  int main(int argc, char* argv[])
  {
    if(argc!=4)
    {
      cout <<"usage:" << argv[0] <<"<files> major_no>
                                   <minor-no>";

      return 0;
  }
  int major = atoi(argv[2], minor = atoi(argv[3]);
  (void) mknod (argv[i]), S_IFCHR |S_IRWXU |S_IRWXG|
  S_IRWXO, (major<<8) |(minor);
```

```
int (rc =1, fd = open (argv[1], O_RWDR | O-NONBLOCK |
O-NoCTTY;
    Char buf [256];
    while (rc && fd! = -1)
        if(rc = read (fd, buf, sizeof (buf)) >0)
     perror ("read");
   else if(rc)
        cout << buf << endl;
   else
       cout << buf << endl;
   Close(fd);
   }
```

⑤ Explain setuid and setgid functions:-

Ans:- Every Process has a set of resource limit, Some of which can be queried and changed by the getrlimit and setrlimit functions.

```
#include <sys/resource.h>
  int getrlimit(int resource, struct rlimit *rlptr);
  int setrlimit(int resource, const struct rlimit *rlptr);
```

Both return: 0 if ok, nonzero on error.

Each call to these two function specifies a single resource and pointer to the following structure:

```
   struct rlimit
   {
   rlim-t rlim_cur    //soft limit : current limit
```

```
rlim_t rlim_max; // hard limit: maximum value for
        rlim_cur
}
```

⑥ write short notes on
(i) message queues
(ii) semaphores.

(i) message queues:-
A message queue is a linked list of messages stored within the kernel and identified by a message queue identifier.
we can the message queue just a queue and its identifier a queue ID.
Msgget :- A new queue created
msgsnd :- new messages are added to the end of the queue.
msgget :- syntax

$\boxed{\text{int msgget(key\_t key, int flag);}}$

```
Struct msquid_ds {
        msgqum_t   msg_qnum;
        msglent_t  msg_qbytes;
        pid_t      msg_lspid;
```

Semaphores:-
A semaphore is counter used to provide access to a shared data object for multiple processes.

To obtained a shared resource, a process needs to do the following :

1. Test the Semaphore that control the resource.
2. If the value of Semaphore is positive, then the Process can use three resource.
   In this case, the Process decrements the semaphores values by 1. Indicating that it has used one unit of the resource.
3. If the value of the semphores is 0, the Process goes to sleep until the semaphores value is greater than 0. when the Process wakes up, it return to step 1.
   A common form of semphore is called binary semaphore. It controls a single resource, and its value is initialized to 1.

⑦ What are FIFO? how to create a FIFO. Explain with a diagram the client serive communication using FIFO.

Ans:- FIFOs are sometimes called ~~ra~~ named pipes. with FIFOs, however, unrelated processes can exchange data.

Creating a FIFO is similar to creating a file. Indeeded, the pathname for a FIFO exists in the file system.

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

Returns 0 if ok, 1 on error.

Clientserver communication using a FIFO

Another use for FIFOs is the send data between a client & server.

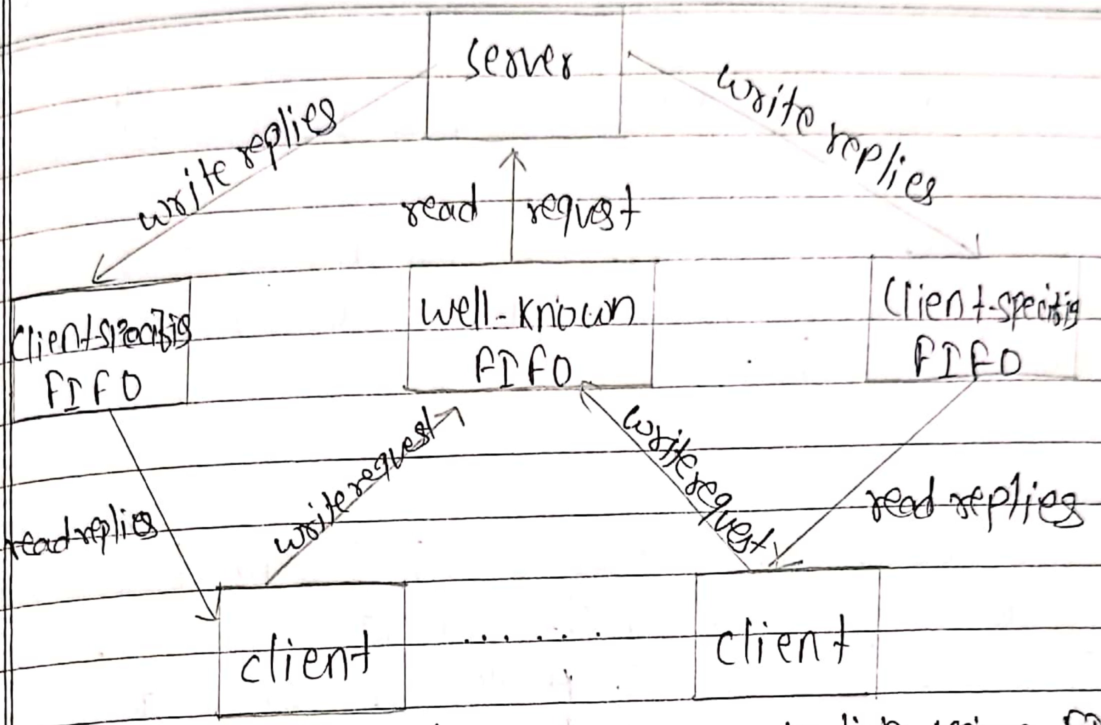


Fig:- client server communication using FIFOs

Since there are multiple writers for the FIFO, the requests 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.

· One solution for each client to send its processID with the request. The server then creates a unique FIFO for each client, using a Pathname based on the clients, using a Pathname based on the client's process ID.

⑧ what are signals? explain with Protype. Describe about signal mask.

Ans:- signals are triggered by events and are posted on a process to notify it that some thing has happened and requires some action.

· The function prototype of the signal is:

```
#include <signal.h>
void (*signal (int signal_num, void (*handler)(int)))(int);
```

- Signal_num is the signal identifier like Sigint or SigTerm.
- handler is the function pointer of a user defined signal handler function.

Signal mask

Each process in unix or POSIX1 System has signal mask that defines which signals are blocked when generated to a process. A blocked signal depends on the recipient process to unblock it and handle it accordingly.

A Process may query or set its signal mask via the sigPromask API:

```
#include <signal.h>.
int sigprocmask (int cmd, const sigset_t* new_mask,
                    sigset_t "old_mask);
```