Inter-process Communication (IPC)

1. Pipes (Named and Unnammed)

1.1 Usage of pipe as a symbol |:

```
somrita@somrita-H110M-H:~/Handson/OS/IPC$ ls -l | grep nmd
-rw-rw-r-- 1 somrita somrita  251 Nov 21 17:00 pipe_nmd_crt.c
-rw-rw-r-- 1 somrita somrita  533 Nov 22 11:30 pipe_nmd_read.c
-rw-rw-r-- 1 somrita somrita  588 Nov 22 11:41 pipe_nmd_write.c
-rw-rw-r-- 1 somrita somrita  472 Nov 21 16:25 pipe_unnmd0.c
-rw-rw-r-- 1 somrita somrita  720 Nov 21 16:42 pipe_unnmd1.c
somrita@somrita-H110M-H:~/Handson/OS/IPC$
```

1.2 Creating a pipe from the Command Line Interface:

```
$ mkfifo filename
```

1.3 Some Example Programs

Q1. Write a c program (single process) to create an **unnamed pipe** and pass data through that pipe.

Note: Include header file unistd.h for the built in functions pipe().

```
int pipe(int file_descriptor[2]);
```

Pipe is passed (a pointer to) an array of two integer file descriptors. It fills the array with two new file descriptors and returns a zero. The two file descriptors returned are connected in a special way. Any data written to file_descriptor[1] can be read back from file_descriptor[0]. The data is processed in a first in, first out basis, usually abbreviated to FIFO. This means that if you write the bytes 1, 2, 3 to file_descriptor[1], reading from file_descriptor[0] will produce 1, 2, 3.

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main()
{
      int res,pipes[2];
      char data[1024];
      char buffer[1024];

      printf("enter data\n");
      scanf("%s", data);

      memset(buffer, '\0', sizeof(buffer));

      if(pipe(pipes) == 0)
      {
            res=write(pipes[1], data, strlen(data));
            printf("Wrote %d bytes\n",res);

            res=read(pipes[0], buffer,1024);
```

```
            printf("Read %d bytes:\n%s\n",res,buffer);

            exit(EXIT_SUCCESS);
      }

      exit(EXIT_FAILURE);
}
```

Q2. Write a c program which will create a child process using fork (), and parent process and Child process will pass data through an **unnamed pipe.**

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main()
{
      int res;
      int pipes[2];
      char msg[1024];
      char buffer[BUFSIZ+1];
      pid_t pid;

      printf("Enter Data\n");
      scanf("%s",msg);

      memset(buffer,'\0',sizeof(buffer));
      if(pipe(pipes) == 0)
      {
            pid = fork();
            if(pid == -1)
            {
                  fprintf(stderr, "Fork failure");
                  exit(EXIT_FAILURE);
            }
            if(pid==0)
            {
                  res = read(pipes[0], buffer, BUFSIZ);
                  printf("processid %d Read %d bytes: %s from parent
process %d\n",getpid(),res, buffer,getppid());
                  exit(EXIT_SUCCESS);
            }
            else
            {
                  res = write(pipes[1], msg, strlen(msg));
                  printf("Processid %d Wrote %d bytes\n",getpid(),res);
                  sleep(10);
            }
      }
      exit(EXIT_SUCCESS);
}
```

Q3. Write a program which will write on pipe and another program to

read data from that pipe

Note: Include header file fcntl.h for the built in functions write() and read (), and also for the macros O_WRONLY and O_RDONLY

Program for Writing into the pipe

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
{
      int pipe_fd;
      int res;
      int s=0;
      char buffer[1024];

      printf("Enter data\n");
      fgets(buffer,1024,stdin);

      res = mkfifo("fifo3", 0777);
      printf("Process %d opening FIFO O_WRONLY\n", getpid());

      pipe_fd=open("fifo3",O_WRONLY);
      printf("Process %d result %d\n", getpid(), pipe_fd);

      res=write(pipe_fd, buffer, sizeof(buffer));

      (void)close(pipe_fd);
      unlink("fifo3");
      printf("Process %d finished\n", getpid());

      exit(EXIT_SUCCESS);
}
```

Program for reading from the pipe

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
int main()
{
      int fd;
      int r,l;
      char buffer[1024];

      printf("Process %d opening FIFO O_RDONLY\n", getpid());
      fd=open("fifo3",O_RDONLY);
      printf("Process %d result %d\n",getpid(),fd);
```

```
	r=read(fd, buffer, sizeof(buffer));
	l=strlen(buffer)-1;

	(void)close(fd);
	unlink("fifo3");

	printf("Process %d finished, with reading %s of %d bytes\n",
getpid(), buffer, l);

	exit(EXIT_SUCCESS);
}
```