| **Name** | **Dishant Modh** |
| --- | --- |
| **Roll Number** | **IT076** |

**EXPERIMENT 6**

**Aim:** Inter process communication (POSIX-IPC) using pipe

Study system call: pipe()

Problem

1. Create a pipe. Write a message in that pipe and read the content of the pipe. (Single process reads and writes in pipe).
2. Create a parent-child process and using pipe make a two-way communication system among them.
3. Client server application using pipes: Client sends file name to server and server sends file content or error as reply to client’s request.

**Tools:** Linux OS, gcc compiler

**Procedure:**

* **Details of System Calls:**

| NAME |
| --- |
|  | pipe - create pipe |
|  |  |
|  | SYNOPSIS |
|  | #include <unistd.h> |
|  |  |
|  | int pipe(int filedes[2]); |
|  |  |
|  | DESCRIPTION |
|  | pipe() creates a pair of file descriptors, pointing to a pipe inode, |
|  | and places them in the array pointed to by filedes. filedes[0] is for |
|  | reading, filedes[1] is for writing. |
|  |  |
|  | RETURN VALUE |
|  | On success, zero is returned. On error, -1 is returned, and errno is |
|  | set appropriately. |

1. **Create a pipe. Write a message in that pipe and read the content of the pipe. (Single process reads and writes in pipe).**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#define MAX\_SIZE 20

int main()

{

int p1[2];

int x = pipe(p1);

if(x==0)

{

char \*str = "I am Dishant Modh";

char buff[MAX\_SIZE];

write(p1[1],str,MAX\_SIZE);

sleep(2);

read(p1[0],buff,MAX\_SIZE);

printf("%s\n",buff);

}

else

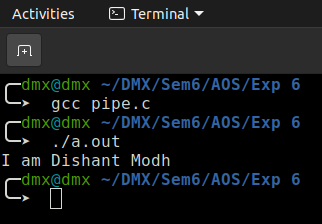
{

printf("Pipe Error!\n");

}

return 0;

}



1. **Create a parent-child process and using pipe make a two-way communication system among them.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#define MAX\_SIZE 20

int main()

{

int p1[2],p2[2];

int x = pipe(p1);

int y = pipe(p2);

if(x==0)

{

printf("Pipe1 created successfully!\n");

}

else

{

printf("Pipe1 Error!\n");

exit(0);

}

if(y==0)

{

printf("Pipe2 created successfully!\n");

}

else

{

printf("Pipe2 Error!\n");

exit(0);

}

char \*str1 = "From Pipe...";

pid\_t child;

child = fork();

char \*str2 = "Hola";

char buff[MAX\_SIZE];

char buff1[MAX\_SIZE];

if (child > 0)

{

close(p1[0]);

write(p1[1],str1,MAX\_SIZE);

close(p1[1]);

wait(NULL);

printf("Pipe2:\n");

close(p2[1]);

read(p2[0],buff1,MAX\_SIZE);

close(p2[0]);

printf("%s\n",buff1);

}

else

{

printf("Pipe1:\n");

close(p1[1]);

read(p1[0],buff,MAX\_SIZE);

close(p1[0]);

printf("%s\n",buff);

strcat(buff,str2);

close(p2[0]);

write(p2[1],buff,MAX\_SIZE);

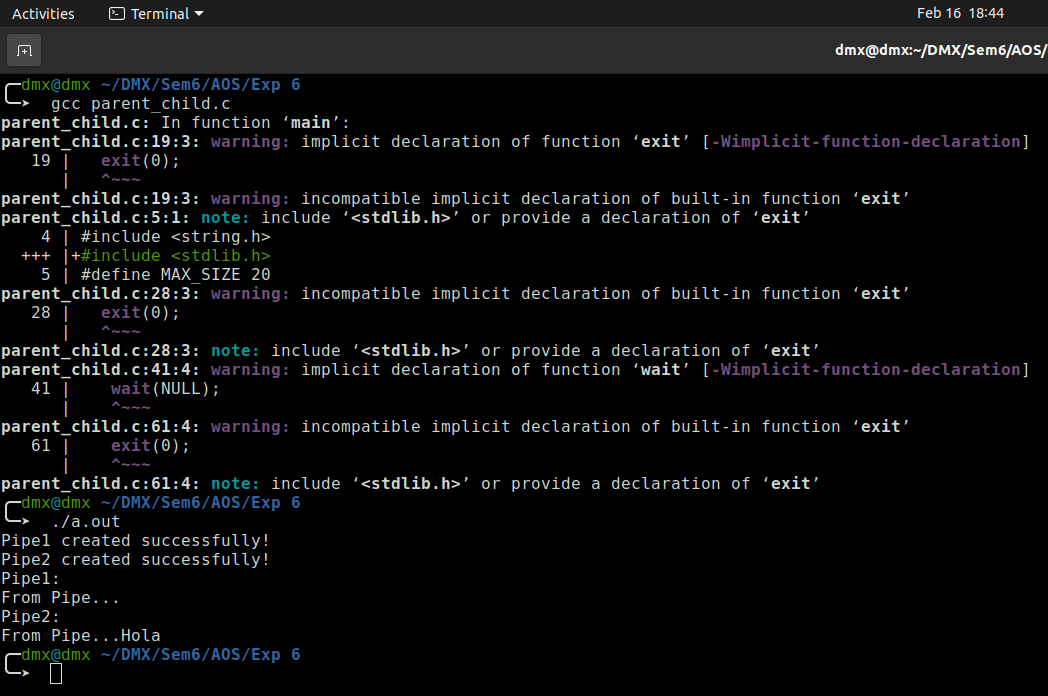
close(p2[1]);

exit(0);

}

return 0;

}



1. **Client server application using pipes: Client sends file name to server and server sends file content or error as reply to client’s request.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#define MAX\_SIZE 50

int main()

{

int p1[2],p2[2];

int x = pipe(p1);

int y = pipe(p2);

if(x==0)

{

printf("Pipe1 created successfully!\n");

}

else

{

printf("Pipe1 Error!\n");

exit(0);

}

if(y==0)

{

printf("Pipe2 created successfully!\n");

}

else

{

printf("Pipe2 Error!\n");

exit(0);

}

char \*str1="file1.txt";

//printf("Enter file name: ");

//scanf("%s",&str1);

pid\_t child;

child = fork();

char buff[MAX\_SIZE];

char buff1[MAX\_SIZE];

int nbytes;

if (child > 0)

{

close(p1[0]);

write(p1[1],str1,MAX\_SIZE);

close(p1[1]);

wait(NULL);

close(p2[1]);

read(p2[0],buff1,MAX\_SIZE);

close(p2[0]);

printf("%s\n",buff1);

}

else

{

close(p1[1]);

read(p1[0],buff,MAX\_SIZE);

close(p1[0]);

int fd;

fd = open(buff,O\_RDWR);

if(fd > 0)

{

nbytes=read(fd,buff1,MAX\_SIZE);

close(p2[0]);

write(p2[1],buff1,nbytes);

close(p2[1]);

exit(0);

}

else

{

printf("File does not exit!\n");

exit(0);

}

}

return 0;

}

