Chapter Pipe and Dup System Call

1. Named Pipe

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>

char buffer_write[]="Hello";
char buffer_read[512];
int main(){
int fdp[2];
pipe(fdp); //Create an unnamed pipe
write(fdp[1],buffer_write,6); //Writing to pipe by process
close(fdp[1]); //close write end
read(fdp[0], buffer_read,6); //reading from pipe by the same process
close(fdp[0]);//close read end of pipe
printf("Success\n");
return 0;
}
```

2.Example- Named Pipe

```
#include<sys/wait.h>
#include <unistd.h>
#include<stdio.h>
```

```
char buffer_write[]="Hello";
char buffer_read[512];

int main(){

int fdp[2];

pipe(fdp); //Create an unnamed pipe

pid_t pid;
pid=fork();

if(pid==0){
printf("Child will write to pipe..\n");
printf("Child id=%d\n",getpid());

close(fdp[0]);//Close read end
write(fdp[1],buffer_write,6); //Writing to pipe by process
close(fdp[1]);

}
else{
wait(0);//wait(NULL) //man wait //man 2 wait
printf("Parent will read from pipe..\n");
printf("Parent id=%d\n",getppid());

close(fdp[1]); //Write end closed
read(fdp[0], buffer_read,6); //reading from pipe by the same process
```

```
close(fdp[0]); //Now close read end

}
printf("Success\n");

return 0;

}
```

```
/* Output

root@ubuntu:~/Desktop/shell_scripts# gcc pipe2.c -o pipe2
root@ubuntu:~/Desktop/shell_scripts# ./pipe2
Child will write to pipe..
Child id=7842
Success
Parent will read from pipe..
Parent id=4151
Success
*/
```

Example-3 Named pipes

1. They are special files

```
root@ubuntu:~/Desktop/shell_scripts# mknod example_pipe    p
root@ubuntu:~/Desktop/shell_scripts# ls -l example_pipe
```

prw-r--r-- 1 root root 0 Aug 28 16:41 example_pipe

4.Example:

Create Writer Process First

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include<unistd.h>
#include<stdlib.h>
#include<error.h>

char buffer_write[]="Hello";

//Writer Process
void main(){
int fdw;

system("ls -l pipe_u1");

//OPen two file descriptors

fdw = open ( "pipe_u1", O_WRONLY );


///Write to pipe:

write (fdw, buffer_write, 6 ); //6 bytes for int
```

```
close (fdw);
printf("Success\n");
}//main

/*

1. Writer process produces data items in buffer
2. Run this process along with the reader process
root@ubuntu:~/Desktop/shell_scripts# ./pipe5
prw------- 1 root root 0 Aug 28 17:36 pipe_u1
Success
3. Run reader process
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include<unistd.h>
#include<stdlib.h>
#include<error.h>

//Reader process
char buffer_read[20];

void main(){
int fdr;
system("ls -l pipe_u1");
```

```
///Read
fdr = open ( "pipe_u1", O_RDONLY );
read(fdr,buffer_read,6);
printf("%s\n",buffer_read);
close(fdr);



}//main


/*
1.Writer process must be executed first
2. Then execute this process
root@ubuntu:~/Desktop/shell_scripts# gcc pipe6.c -o pipe6
root@ubuntu:~/Desktop/shell_scripts# ./pipe6
prw------- 1 root root 0 Aug 28 17:36 pipe_u1
Hello
*/
```

References:

[1] https://www.gnu.org/software/libc/manual/html_node/Creating-a-Pipe.html

[2] https://linuxhint.com/pipe_system_call_c/

[3] https://www.cs.uml.edu/~fredm/courses/91.308-spr05/files/pipes.html

[4] https://profile.iiita.ac.in/bibhas.ghoshal/lab_files/System%20calls%20for%20files%20and%20directories%20in%20Linux.html