Lab07: More on PIPES

Objectives

1. **Understanding dup() and dup2()**
2. **Understanding named pipe**

What is dup() and dup2()?

```
#include <unistd.h>
int dup(int file_descriptor);
int dup2(int file_descriptor_one, int file_descriptor_two);
```

The purpose of the dup call is to open a new file descriptor, a little like the open call. The difference is that the new file descriptor created by dup refers to the same file (or pipe) as an existing file descriptor. In the case of dup, the new file descriptor is always the lowest number available, and in the case of dup2 it's the same as, or the first available descriptor greater than, the parameter file_descriptor_two.

Example # 1 Using dup

```
#include<stdio.h>
#include <unistd.h>
#include <fcntl.h>

int main()
{
      // open() returns a file descriptor file_desc to a
      // the file "dup.txt" here"

      int file_desc = open("dup.txt", O_WRONLY | O_APPEND);

      if(file_desc < 0)
            printf("Error opening the file\n");

      // dup() will create the copy of file_desc as the copy_desc
      // then both can be used interchangeably.

      int copy_desc = dup(file_desc);

      // write() will write the given string into the file
      // referred by the file descriptors

      write(copy_desc,"This will be output to the file named dup.txt\n",
46);

      write(file_desc,"This will also be output to the file named
dup.txt\n", 51);
```

```
	return 0;
}
```

Example # 2 Using dup 2

```
#include<unistd.h>
#include<stdio.h>
#include<fcntl.h>

int main()
{
	int file_desc = open("tricky.txt",O_WRONLY | O_APPEND);

	// here the newfd is the file descriptor of stdout (i.e. 1)
	dup2(file_desc, 1) ;

	// All the printf statements will be written in the file
	// "tricky.txt"
	printf("I will be printed in the file tricky.txt\n");

return 0;
}
```

Named Pipe : FIFO

You can create named pipes from the command line and from within a program. Historically, the command- line program for creating them was mknod:

$ mknod filename p

However, the mknod command may not be available on all UNIX-like systems. The preferred command-line method is to use

$ mkfifo filename

#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *filename, mode_t mode);
int mknod(const char *filename, mode_t mode | S_IFIFO, (dev_t) 0);

Example # 3 Creating Name Pipe

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

int main()
{
    int res = mkfifo("/tmp/my_fifo", 0777);
    if (res == 0)
        printf("FIFO created\n");
    exit(EXIT_SUCCESS);
}
```

Accessing FIFO file

```
$ cat < /tmp/my_fifo &
$ echo “Hello World” > /tmp/my_fifo
```

Task :

Use the named pipe to communicate between two programmes. Send and receive message to each other.