Practical – 6

Input and Output Redirection

STDIN_FILENO -> standard input file
STDOUT_FILENO -> standard output file

Integer value	Name
0	Standard input (stdin)
1	Standard output (stdout)
2	Standard error (stderr)

Each process has its own file descriptor table. The process passes the file descriptor to the kernel through a system call, and the kernel will access the file on behalf of the process. The process itself cannot read or write the file descriptor table directly.

open system call.

This allocates resources associated to the file (the file descriptor), and returns. The process will use to refer to that file instead of standard i/p or o/p file. In some cases the open is performed by the first access.

```
close(STDOUT_FILENO)
open("f1.txt",O_WRONLY.);
printf(""standard output redirection"); -> this will written to f1.txt instead of standard
 o/p file
```

dup() : these system calls create a copy of the file descriptor .

it creates an alias for the provided file descriptor. dup always uses the smallest available file descriptor. Thus, if we called dup first thing in our program, then you could write to standard output by using file descriptor 3 (dup uses 3 because 0, 1, and 2 are already taken by default).

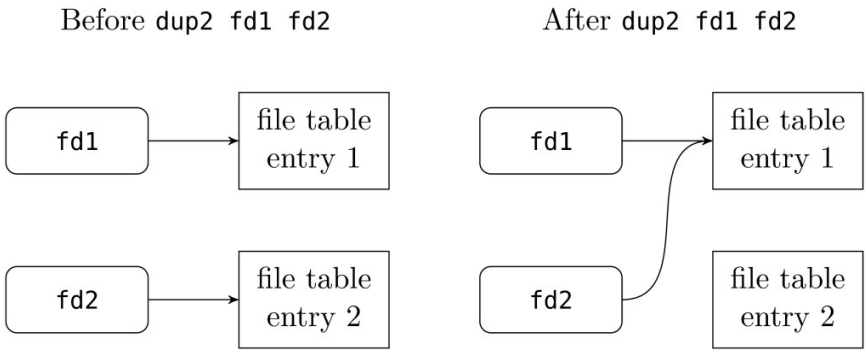
```
#include <unistd.h>
Syntax  : int     dup(int filediscriptor)
```

dup2() : The dup2() system call is similar to dup() but the basic difference between them is that instead of using the lowest-numbered unused file descriptor, it uses the descriptor number specified by the user.

```
#include<unistd.h>
Syntax :  int dup2(int fildes, int fildes2);
```

int fildes : The source file descriptor. This remains open after the call to dup2.
int fildes2 : The destination file descriptor. This file descriptor will point to the same file as filedes after this call returns.

return value: dup2 returns the value of the second parameter (fildes2) upon success. A negative return value means that an error occurred.



Program -1

```
//This program shows standard input and output redirection
//First argument is input file, second is output file and remaining arguments
//as command line to be executed

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

int main(int argc, char *argv[])
{
        int fd1, fd2;

        fd1=open(argv[1],O_RDONLY);
        fd2=open(argv[2],O_WRONLY|O_TRUNC|O_CREAT,0644);

        close(STDIN_FILENO);
        dup(fd1);

        close(STDOUT_FILENO);
        dup(fd2);

        execvp(argv[3],&argv[3]);

        printf("Command failed...\n");
        close(fd1);
        close(fd2);
        exit(0);
}

./a.out in.txt out.txt  wc  -l
```

Program – 2

//This program perform standard input and output redirection but using dup2()

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


int main(int argc, char *argv[])
{
        int fd1,fd2,exitstatus;

        switch(fork())
        {
          case 0:
                //Child process
                fd1=open(argv[1],O_RDONLY);
                fd2=open(argv[2],O_WRONLY|O_TRUNC|O_CREAT,0644);
                dup2(fd1,0);
                dup2(fd2,1);
                execvp(argv[3],&argv[3]);
                exit(1);
          case -1:
                printf("Fork error\n");
                exit(2);
          default:
                //Parent process
                wait(&exitstatus);
                printf("Exit status : %d\n",WEXITSTATUS(exitstatus));
        }
}
```

$ ./a.out in.txt out.txt wc –l

Program 3

```
// Program to illustrate dup()
// program for create a duplicate file.
#include<stdio.h>
#include <unistd.h>
#include <fcntl.h>

int main()
{
        // open() returns a file descriptor file_desc to 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;
}
```