Assignment 7A

By Yash Kale (33233)

Aim :

Full duplex communication between two independent processes. First process accepts sentences and writes on one pipe to be read by second process and second process counts number of characters, number of words and number of lines in accepted sentences, writes this output in a text file and writes the contents of the file on second pipe to be read by first process and displays on standard output.

Theory :

FIFOs

A first-in, first-out (FIFO) file is a pipe that has a name in the filesystem. Any process can open or close the FIFO; the processes on either end of the pipe need not be related to each other. FIFOs are also called named pipes

You can make a FIFO using the mkfifo command. Specify the path to the FIFO on the command line. For example, create a FIFO in /tmp/fifo by invoking this:

```
% mkfifo /tmp/fifo
% ls -l /tmp/fifo
prw-rw-rw-
1 samuel          users          0 Jan 16 14:04 /tmp/fifo
```

The first character of the output from ls is p, indicating that this file is actually a FIFO (named pipe). In one window, read from the FIFO by invoking the following:

```
% cat < /tmp/fifo
```

In a second window, write to the FIFO by invoking this:

```
% cat > /tmp/fifo
```

Then type in some lines of text. Each time you press Enter, the line of text is sent through the FIFO and appears in the first window. Close the FIFO by pressing Ctrl+D in the second window. Remove the FIFO with this line:

```
% rm /tmp/fifo
```

Creating a FIFO

Create a FIFO programmatically using the mkfifo function.The first argument is the path at which to create the FIFO; the second parameter specifies the pipe's owner, group, and world permissions, and a pipe must have a reader and a writer, the permissions must

include both read and write permissions. If the pipe cannot be created (for instance, if a file with that name already exists), mkfifo returns –1. Include<sys/types.h> and <sys/stat.h> if you call mkfifo.

Accessing a FIFO

Access a FIFO just like an ordinary file. To communicate through a FIFO, one program must open it for writing, and another program must open it for reading. Either low-level I/O functions like open, write, read, close or C library I/O functions (fopen, fprintf, fscanf, fclose, and soon) may be used.
For example, to write a buffer of data to a FIFO using low-level I/O routines, you could use this code:

```
intfd = open (fifo_path, O_WRONLY);

write (fd, data, data_length);

close (fd);
```

To read a string from the FIFO using C library I/O functions, you could use this code:

```
FILE* fifo = fopen (fifo_path, "r");

fscanf (fifo, "%s", buffer);

fclose (fifo);
```

A FIFO can have multiple readers or multiple writers. Bytes from each writer are written atomically up to a maximum size of PIPE_BUF (4KB on Linux). Chunks from simultaneous writers can be interleaved. Similar rules apply to simultaneous reads.

Output :

```
inferno080@inferno080-Swift-SF314-54:~/OSL/Assignment 7$ gcc 7a.c
inferno080@inferno080-Swift-SF314-54:~/OSL/Assignment 7$ ./a.out

Enter the String:An apple a day keeps doctor away
#

The contents of file are An apple a day keeps doctor away


 No of Words: 8
 No of Characters: 26
 No of Lines: 2
inferno080@inferno080-Swift-SF314-54:~/OSL/Assignment 7$ S
```

33233

Conclusion :

Thus, we studied inter process communication using FIFOs.