DEPARTMENT OF INFORMATION TECHNOLOGY

IT 253 Operating Systems Lab

LAB3: 07/03/2023

Evaluation: 10 Marks

Objective

To understand the Inter process communication – FIFO or named pipes

Inter process communication (IPC) -FIFO or named pipes

1. Read these Basics for understanding IPC: named pipes:

Pipes were meant for communication between related processes. But pipes can not be used network for communication. For that we can use Named Pipes. Even though this works for related processes, it gives no meaning to use the named pipes for related process communication. We can use single named pipe that can be used for two-way communication (communication between the server and the client, plus the client and the server at the same time) as Named Pipe supports bi-directional communication. Another name for named pipe is **FIFO (First-In-First-Out)**. The system call (mknod()) is used to create a named pipe, which is a kind of a special file.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int mknod(const char *pathname, mode_t mode, dev_t dev);
```

This system call would create a special file or file system node such as ordinary file, device file, or FIFO. The arguments to the system call are pathname, mode and dev. The pathname along with the attributes of mode and device information. The pathname is relative, if the directory is not specified it would be created in the current directory. The mode specified is the mode of file which specifies the file type such as the type of file and the file mode as mentioned in the following tables. The dev field is to specify device information such as major and minor device numbers.

File Type	Description	File Type	Description
S_IFBLK	block special	S_IFREG	Regular file
S_IFCHR	character special	S_IFDIR	Directory
S_IFIFO	FIFO special	S_IFLNK	Symbolic Link

File Mode	Description	File Mode	Description
S_IRWXU	Read, write, execute/search by owner	S_IWGRP	Write permission, group
S_IRUSR	Read permission, owner	S_IXGRP	Execute/search permission, group
S_IWUSR	Write permission, owner	S_IRWXO	Read, write, execute/search by others
S_IXUSR	Execute/search permission, owner	S_IROTH	Read permission, others
S_IRWXG	Read, write, execute/search by group	S_IWOTH	Write permission, others
S_IRGRP	Read permission, group	S_IXOTH	Execute/search permission, others

File mode can also be represented in octal notation such as 0XYZ, where X represents owner, Y represents group, and Z represents others. The value of X, Y or Z can range from 0 to 7. The values for read, write and execute are 4, 2, 1 respectively. If needed in combination of read, write and execute, then add the values accordingly.

Say, if we mention, 0640, then this means read and write (4 + 2 = 6) for owner, read (4) for group and no permissions (0) for others. This call would return zero on success and -1 in case of failure. To know the cause of failure, check with errno variable or perror() function.

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode)
```

This library function creates a FIFO special file, which is used for named pipe. The arguments to this function is file name and mode. The file name can be either absolute path or relative path. If full path name (or absolute path) is not given, the file would be created in the current folder of the executing process. The file mode information is as described in **mknod()** system call.

This call would return zero on success and -1 in case of failure. To know the cause of failure, check with errno variable or perror() function.

One Way Communication

Let us consider a program of running the server on one terminal and running the client on another terminal. The program would only perform one-way communication. The client accepts the user input and sends the message to the server, the server prints the message on the output. The process is continued until the user enters the string “end”.

Let us understand this with an example −

Step 1− Create two processes, one is fifoserver and another one is fifoclient.

Step 2− Server process performs the following −

- •Creates a named pipe (using system call mknod()) with name “MYFIFO”, if not created.
- •Opens the named pipe for read only purposes.
- •Here, created FIFO with permissions of read and write for Owner. Read for Group and no permissions for Others.
- •Waits infinitely for message from the Client.

•If the message received from the client is not “end”, prints the message. If the message is “end”, closes the fifo and ends the process.

Step 3− Client process performs the following −

•Opens the named pipe for write only purposes.

•Accepts the string from the user.

•Checks, if the user enters “end” or other than “end”. Either way, it sends a message to the server. However, if the string is “end”, this closes the FIFO and also ends the process.

•Repeats infinitely until the user enters string “end”.

Two Way Communication

Named pipe is meant for communication between two or more unrelated processes and can also have bi-directional communication. For bi-directional communication i.e., the client sending message to the server and the server receiving the message and sending back another message to the client using the same named pipe.

Procedure for two way communication

Step 1− Create two processes, one is fifoserver and another one is fifoclient.

Step 2− Server process performs the following −

•Creates a named pipe (using library function mkfifo()) with name “fifo_twoway” in /tmp directory, if not created.

•Opens the named pipe for read and write purposes.

•Here, created FIFO with permissions of read and write for Owner. Read for Group and no permissions for Others.

•Waits infinitely for a message from the client.

• Perform read/write operation to FIFO.

Step 3− Client process performs the following −

•Opens the named pipe for read and write purposes.

•Accepts string from the user.

•Perform read and write operation.

2. Execute the following program where a client sends message to server using named PIPE. Observe the result write your observation along with screenshots of result. [Marks : 2 Marks]

```c
/* Filename: fifoserver.c */
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "MYFIFO"
int main() {
   int fd;
   char readbuf[80];
   char end[10];
   int to_end;
   int read_bytes;

   /* Create the FIFO if it does not exist */
   mknod(FIFO_FILE, S_IFIFO|0640, 0);
   strcpy(end, "end");
   while(1) {
      fd = open(FIFO_FILE, O_RDONLY);
      read_bytes = read(fd, readbuf, sizeof(readbuf));
      readbuf[read_bytes] = '\0';
      printf("Received string: \"%s\" and length is %d\n", readbuf, (int)strlen(readbuf));
      to_end = strcmp(readbuf, end);
      if (to_end == 0) {
         close(fd);
         break;
      }
   }
   return 0;
}
```

Above program is fifoserver.c

compile as $gcc fifoserver.c -o ser

$./ser

```c
/* Filename: fifoclient.c */
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "MYFIFO"
int main() {
   int fd;
   int end_process;
   int stringlen;
   char readbuf[80];
   char end_str[5];
   printf("FIFO_CLIENT: Send messages, infinitely, to end enter \"end\"\n");
   fd = open(FIFO_FILE, O_CREAT|O_WRONLY);
   strcpy(end_str, "end");

   while (1) {
      printf("Enter string: ");
      fgets(readbuf, sizeof(readbuf), stdin);
      stringlen = strlen(readbuf);
      readbuf[stringlen - 1] = '\0';
      end_process = strcmp(readbuf, end_str);

      //printf("end_process is %d\n", end_process);
      if (end_process != 0) {
         write(fd, readbuf, strlen(readbuf));
         printf("Sent string: \"%s\" and string length is %d\n", readbuf, (int)strlen(readbuf));
      } else {
         write(fd, readbuf, strlen(readbuf));
         printf("Sent string: \"%s\" and string length is %d\n", readbuf, (int)strlen(readbuf));
         close(fd);
         break;
      }
   }
   return 0;
}
```

Above program is fifoclient.c

compile as $gcc fifoclient.c -o cli

$./cli

The client sends typed message to server.

2. Modify the above program to send the message by parent process to the child process. At server side make all the received characters to capital letters and send it back to client. The client prints the received message from server. [Note: The file which you are uploading should also contain source code of this program] [Marks : 4 Marks]

```c
/*  server */
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FIFO_FILE "/tmp/fifo_twoway"
....
.....
/* Create the FIFO if it does not exist */
    mkfifo(FIFO_FILE, S_IFIFO|0640);
.....
...
/* open file in FIFO mode and use fd as descriptor for read and write */
 fd = open(FIFO_FILE, O_RDWR);

….. .
```

```c
/*  client*/
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FIFO_FILE "/tmp/fifo_twoway"

/* no need to create in client once again*/
.....

/* open file in FIFO mode and use fd as descriptor for read and write */
fd = open(FIFO_FILE, O_CREAT|O_RDWR);

....
```

3. A client program reads a file name from the standard input and sends it to server program. The server program opens the file if it is present, reads the content of file and sends it back to client program. If file is not present, then the server sends an error message. Implement the client server program using inter process communication: Named Pipe or FIFO. [Marks : 4 Marks]

[Note: The file which you are uploading should also contain source code of this program]
