fifo process1 (WRITER)

```
#include<stdio.h>       // For printf(), standard I/O
#include<fcntl.h>       // For open() and file control options
#include<sys/stat.h>    // For mkfifo() — creating named pipes
#include<unistd.h>      // For read(), write(), close(), unlink()

int main()
{
   int fd, fd1;                          // file descriptors for two FIFOs
   char *myfifo = "myfifo";                  // name of first FIFO (writer → reader)
   char *myfifo1 = "myfifo1";                // name of second FIFO (reader → writer)
   char buff[1024];                      // buffer for reading/writing data

   // Step 1: Create the two FIFOs (named pipes)
   mkfifo(myfifo, 0666);                 // Create FIFO for writing
   mkfifo(myfifo1, 0777);                // Create FIFO for reading back the response

   // Step 2: Open first FIFO for writing
   fd = open(myfifo, O_WRONLY);             // open() returns a file descriptor to the FIFO
                                 // O_WRONLY means write-only mode

   // Step 3: Write message into FIFO (sent to reader process)
   write(fd, "Welcome...\nTo PES MCOE\nPune.\nMaharashtra\nIndia.",
        sizeof("Welcome...\nTo PES MCOE\nPune.\nMaharashtra\nIndia."));

   // Step 4: Close the write FIFO
   close(fd);

   // Step 5: Remove (unlink) the FIFO file from filesystem
```

```
    unlink(myfifo);                     // cleans up myfifo once message sent

    // Step 6: Open second FIFO for reading response
    fd1 = open(myfifo1, O_RDONLY);          // open second pipe for reading data sent back

    // Step 7: Read message from second FIFO
    read(fd1, buff, sizeof(buff));          // read() copies data from FIFO into buff

    // Step 8: Display the received message
    printf("%s", buff);

    // Step 9: Cleanup — remove second FIFO and close file descriptor
    unlink(myfifo1);
    close(fd1);

    return 0;
}
```

To make it user input:

Replace it with: #write(fd,"Welcome...\nTo PES MCOE\nPune.\nMaharashtra\nIndia.", sizeof(...));

```
printf("Enter a message: ");
fgets(buff, sizeof(buff), stdin);
write(fd, buff, strlen(buff)+1);
```

READER CODE

```
#include<stdio.h>       // For printf(), FILE operations
#include<fcntl.h>       // For open() function and flags like O_RDONLY, O_WRONLY
#include<sys/stat.h>    // For mkfifo(), permissions
#include<unistd.h>      // For read(), write(), close()
#include<string.h>      // For strlen()
#include<stdlib.h>      // For standard library functions

int main()
{
   int fd, fd1;                    // File descriptors for FIFOs
   char *myfifo = "myfifo";           // FIFO 1: Writer → Reader
   char *myfifo1 = "myfifo1";          // FIFO 2: Reader → Writer
   char buff[1024];                 // Buffer to store data read from FIFO
   int chars = 0, words = 0, lines = 0; // Counters for analysis
   FILE *fp;                        // File pointer to handle output.txt

   // Step 1: Open first FIFO to read message from Writer
   fd = open(myfifo, O_RDONLY);       // Opens FIFO in read-only mode
   read(fd, buff, sizeof(buff));      // Reads data from Writer into buff
   close(fd);                      // Close FIFO after reading
```

```
    // Step 2: Analyze data (count characters, words, and lines)
    for (int i = 0; buff[i] != '\0'; i++) {
        chars++;                        // Count each character
        if (buff[i] == ' ' || buff[i] == '\n')
            words++;                    // Increment word count at space/newline
        if (buff[i] == '\n')
            lines++;                    // Increment line count at newline
    }

    // Step 3: Write results into a text file
    fp = fopen("output.txt", "w");      // Create or open file for writing
    fprintf(fp, "Characters: %d\nWords: %d\nLines: %d\n", chars, words, lines);
    fclose(fp);                         // Close the file

    // Step 4: Send results back to Writer through second FIFO
    fd1 = open(myfifo1, O_WRONLY);      // Open second FIFO for writing
    fp = fopen("output.txt", "r");      // Open the result file for reading
    char line[256];                     // Temporary buffer to hold file lines

    // Read file content line by line and send it
    while (fgets(line, sizeof(line), fp))
        write(fd1, line, strlen(line));

    // Close file and FIFO after sending
    fclose(fp);
    close(fd1);

    return 0;
}
```

first run reader the writer