Name : Sarthak Pagar

Roll No. : 34

Classs : TE (IT)

Practical : 7A

Statement : [A. FIFOS: 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 onstandard output.](https://drive.google.com/drive/folders/17KgnyquyHBPlCsSThGnqXeKjRs-AGnEs?usp=sharing)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#define FIFO1 "/tmp/fifo1" // First named pipe for communication from parent to child

#define FIFO2 "/tmp/fifo2" // Second named pipe for communication from child to parent

// Function to count characters, words, and lines

void count\_and\_write(const char \*sentence) {

int characters = 0, words = 0, lines = 0;

int inWord = 0; // To track if we're currently in a word

// Count characters, words, and lines

for (int i = 0; i < strlen(sentence); i++) {

if (sentence[i] != '\n') { // Exclude newline from character count

characters++;

}

if (sentence[i] == ' ' || sentence[i] == '\n') {

if (inWord) {

words++; // End of a word

inWord = 0; // Not in a word anymore

}

if (sentence[i] == '\n') {

lines++; // Count lines based on newline

}

} else {

inWord = 1; // We're in a word

}

}

// Count the last word if necessary

if (inWord) {

words++;

}

// Open file to write results

FILE \*fp = fopen("output.txt", "w");

if (fp != NULL) {

fprintf(fp, "Characters: %d\nWords: %d\nLines: %d\n", characters, words, lines);

fclose(fp);

} else {

perror("Failed to open file");

}

}

// Child process to read from FIFO1, count, and write results to FIFO2

void child\_process() {

char sentence[200];

int fd1 = open(FIFO1, O\_RDONLY); // Open FIFO1 for reading

int fd2 = open(FIFO2, O\_WRONLY); // Open FIFO2 for writing

// Read the sentence from the first pipe

read(fd1, sentence, sizeof(sentence));

close(fd1); // Close the read end of FIFO1

// Count characters, words, and lines

count\_and\_write(sentence);

// Open the output file and read its content to send back to the parent

FILE \*fp = fopen("output.txt", "r");

char ch;

while ((ch = fgetc(fp)) != EOF) {

write(fd2, &ch, sizeof(ch)); // Write each character to FIFO2

}

fclose(fp);

close(fd2); // Close the write end of FIFO2

exit(0); // Terminate the child process

}

// Parent process to read input and communicate with the child process

void parent\_process() {

char sentence[200];

char buffer[200];

sentence[0] = '\0'; // Initialize sentence to be an empty string

int fd1 = open(FIFO1, O\_WRONLY); // Open FIFO1 for writing

int fd2 = open(FIFO2, O\_RDONLY); // Open FIFO2 for reading

// Get input from user line by line

printf("Enter multiple lines of text (end input with a blank line):\n");

while (1) {

fgets(buffer, sizeof(buffer), stdin);

// If user enters a blank line (newline only), stop input

if (strcmp(buffer, "\n") == 0) {

break;

}

// Append the buffer to sentence

strcat(sentence, buffer);

}

// Send the full sentence to child through FIFO1

write(fd1, sentence, strlen(sentence) + 1); // +1 to include null terminator

close(fd1); // Close the write end of FIFO1

// Wait for child process to finish

wait(NULL);

// Read results from FIFO2

printf("\nResults received from child process:\n");

char ch;

while (read(fd2, &ch, sizeof(ch)) > 0) {

printf("%c", ch); // Output the content received

}

close(fd2); // Close the read end of FIFO2

}

int main() {

// Create named pipes (FIFOs)

mkfifo(FIFO1, 0666); // Create FIFO1

mkfifo(FIFO2, 0666); // Create FIFO2

pid\_t pid = fork(); // Fork a new process

if (pid < 0) {

perror("Fork failed");

return 1;

} else if (pid == 0) {

// Child process

child\_process();

} else {

// Parent process

parent\_process();

}

// Remove the named pipes

unlink(FIFO1);

unlink(FIFO2);

return 0;

}

Output :-

[Saru1594@localhost 7a]$ gcc exp7a.c

[Saru1594@localhost 7a]$ ./a.out

Enter multiple lines of text (end input with a blank line):

hi

my name is sarthak

this is my 7th practical

Results received from child process:

Characters: 44

Words: 10

Lines: 3