|  |  |  |
| --- | --- | --- |
| |  | | --- | | **INSTYTUT TELEINFORMATYKI I AUTOMATYKI**  **Wydział Cybernetyki WAT** | | |
| |  | | --- | | Przedmiot: SYSTEMY OPERACYJNE  SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO NR 2 | | |
| **Temat**: komunikacja międzyprocesowa – I (Pipe i FIFO) | |
| **Wykonał:**  xxxxx | **Data wykonania ćwiczenia:**  xxxxx  **Prowadzący ćwiczenie:**  xxxxx |

1. **Treść zadania laboratoryjnego**

Obraz zawierający tekst, gazeta, zrzut ekranu, dokument

Opis wygenerowany automatycznie

1. **Opis rozwiązania, komentarze, wnioski**

Program:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <fcntl.h>

#include <dirent.h>

#include <sys/stat.h>

#include <string.h>

#define FIFO "my\_fifo"

#define FIFO2 "my\_fifo2"

int main(int argc, char \*\*argv){

    FILE \*input;

    FILE \*out1, \*out2;

    if(argc!=2){

        fprintf(stderr, "Wrong number of arguments, excpected: 2\n");

        return 1;

    }

    if((input=fopen(argv[1],"r")) == NULL){

        fprintf(stderr, "Can't open file \"%s\"\n", argv[1]);

        return 1;

    }

    if((out1=fopen("K1.txt","w"))==NULL){

        fprintf(stderr, "Can't open file \"K1.txt\"\n");

        return 1;

    }

    fclose(out1);

    if((out2=fopen("K2.txt","w"))==NULL){

        fprintf(stderr, "Can't open file \"K1.txt\"\n");

        return 1;

    }

    fclose(out2);

    umask(0);

    mkfifo(FIFO,0666);

    mkfifo(FIFO2,0666);

    char in[1], out[1];

    int fifo, fifo2, a, b, c, loop, i, l=1, k=0;

    if((a=fork())==0){

        printf("Parent process (PID: %d)\n", getpid());

        fifo=open(FIFO, O\_WRONLY);

        fifo2=open(FIFO2, O\_WRONLY);

        while(!feof(input)){

            if(feof(input))

                break;

            in[0]=getc(input);

            write(fifo, in, 1);

            write(fifo2, in, 1);

        }

        close(fifo);

        close(fifo2);

        unlink(FIFO);

        unlink(FIFO2);

    }

    else if((b=fork())==0){

        printf("Child process K1 (PID: %d)\n", getpid());

        fifo=open(FIFO, O\_RDONLY);

        i=1;

        while(1){

            loop=read(fifo, out, 1);

            if((int)\*out==-1)

                break;

            if(loop>0){

                out1=fopen("K1.txt","a");

                if(l){

                    fprintf(out1, "%d %c",i , \*out);

                    l=0;

                    i++;

                }else fprintf(out1, "%c", \*out);

                if(\*out=='\n')

                    l=1;

                fclose(out1);

            }

        }

        close(fifo);

    }

    else if((c=fork())==0){

        printf("Child process K2 (PID: %d)\n", getpid());

        fifo2=open(FIFO2, O\_RDONLY);

        i=1;

        l=1;

        while(1){

            loop=read(fifo2, out, 1);

            if((int)\*out==-1)

                break;

            if(loop>0){

                k++;

                out2=fopen("K2.txt","a");

                if(l){

                    l=0;

                    i++;

                }else if(i%2!=0)

                    fprintf(out2, "%c", \*out);

                if(\*out=='\n') l=1;

                fclose(out2);

            }

        }

        out2=fopen("K2.txt","a");

        fprintf(out2, "Rows: %d\tChars: %d\n",i-1,k-(i-1));

        fclose(out2);

        close(fifo2);

    }

    fclose(input);

    return 0;

}

Kolejność działań:

1. Deklaracja kolejek, otwarcie plików
2. Proces macierzysty: pobranie do kolejek linii tekstu z pliku
3. Proces potomny K1: wypisanie ponumerowanych linii do pliku „K1.txt”
4. Proces potomny K2: wypisanie co drugiej linii do pliku „K2.txt”
5. Zamknięcie plików, odłączenie kolejek, zakończenie programu

Prawidłowa imprementacja kolejki:

1. Zerowanie maski - umask(0)
2. Tworzenie kolejki – mkfifo(FIFO,0666)
   1. (tu powstaje plik z zawartością kolejki, drugi argument to prawa dostępu dla do pliku z kolejką)
3. Otwarcie kolejki do zapisu danych - fifo=open(FIFO, O\_WRONLY);
4. Wpisywanie danych - write(fifo, in, 1);
5. Zamknięcie kolejki - close(fifo);
6. Otwarcie kolejki do odczytu danych - fifo=open(FIFO, O\_RDONLY);
7. Odczytywanie danych - read(fifo, out, 1);
8. Zamknięcie kolejki - close(fifo);
9. **Wyniki uruchomienia programu**

Zawartość pliku x:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Wywołanie programu:

Obraz zawierający tekst

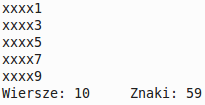
Opis wygenerowany automatycznie

Zawartość pliku „K1.txt”:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Zawartość pliku „K2.txt”:



Wywołanie programu z nieodpowiednią ilością argumentów:





Wywołanie programu bez istniejącego pliku x:

