```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
        char str[50];
        int fd[2];
        char str2[50];

        pipe(fd);
        printf("Wpisz tekst: ");
        scanf("%s", str);

        pid_t pid = fork();

        if (pid == 0)
        {
                close(fd[1]);
                read(fd[0], str2, sizeof(str2));

                str2[0] = 'X';
                printf("%s\n", str2);
                close(fd[0]);
        }
        else
        {
                close(fd[0]);
                write(fd[1], str, strlen(str) + 1);
                close(fd[1]);

                wait(NULL);
        }

        return 0;
}
```

char str[50] - tworzy tablicę znaków o długości 50 bajtów; to jest tekst, który użytkownik wpisuje w terminalu

int fd[2] - fd to tablica z dwoma deskryptorami plików:

1. fd[0] - koniec do odczytu z potoku

2. fd[1] - koniec do zapisu do potoku

char str2[50] - druga tablica znaków, tutaj proces potomny zapisze dane, które odczyta z potoku

pipe(fd) - tworzy potok(pipe), który pozwala przesyłać dane z jednego procesu do drugiego

printf("Wpisz tekst: ") - wypisuje w terminalu komunikat: "Wpisz tekst: "

scanf("%s", str) - odczytuje tekst wpisany przez użytkownika i zapisuje go do zmiennej str.

pid_t pid = fork() - wywołanie fork() tworzy nowy proces - proces potomny

Proces potomny:

if (pid == 0) - jeśli pid == 0, to znaczy, że jesteśmy w procesie potomnym

close(fd[1]) - proces potomny nie musi nic zapisywać do potoku -> zamyka koniec do zapisu

read(fd[0], str2, sizeof(str2)) - odczytuje dane z potoku i zapisuje je do str2

str2[0] = 'X' - zamienia pierwszy znak w odczytanym tekście na 'X'

printf("%s\n", str2) - wypisuje zmieniony tekst

close(fd[0]) - zamyka koniec do odczytu potoku

Proces rodzica:

close(fd[0]) - proces rodzica nie odczytuje z potoku -> zamyka koniec do odczytu

write(fd[1], str, strlen(str) + 1) - zapisuje tekst użytkownika do potoku

close(fd[1]) - zamyka koniec zapisu, aby proces potomny wiedział, że dane się skończyły

wait(NULL) - czeka, aż proces potomny zakończy działanie.

Podsumowując,

Proces rodzica:

1. bierze wpisany tekst
2. zapisuje go do potoku (pipe)

Proces potomny:

1. odczytuje tekst z potoku, a nie z terminala
2. zmienia pierwszy znak na 'X'
3. wypisuje zmieniony tekst

Wynik działania:

```
karolina@karolina:~/ProgSys/C/lab4.$ make
mkdir -p build
gcc -c src/main.c -o build/main.o
gcc build/main.o -o build/hello
karolina@karolina:~/ProgSys/C/lab4.$ ./build/hello
Wpisz tekst: main
Xain
```