```
 GNU nano 7.2                                                main.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(){
        int fd[2];
        char text[50];
        char text2[50];

        pipe(fd);
        pid_t pid = fork();
        if (pid == 0) {
                close(fd[0]);
                printf("enter text: ");
                scanf("%s", text);
                write(fd[1], text, strlen(text)+1);
                close(fd[1]);
        }
        else {
                close(fd[1]);
                read(fd[0], text2, 20);
                text2[0]='X';
                close(fd[0]);
                printf("%s\n", text2);
        }
        return 0;
}
```

stdio.h — do obsługi printf, scanf.
string.h — do pracy z string (np. strlen).
unistd.h — do nowych funkcji takich, jak pipe() i fork().

int fd[2];
tworzymy tablicę fd z dwóch elementów.
fd[0] — do **odczytu danych** z kanału (skąd można je pobrać).
fd[1] — do **zapisu danych** do kanału (gdzie możemy je wysłać).

char text[50];
char text2[50];
text — przechowuje **tekst wpisany przez użytkownika, t**en tekst następnie **jest wysyłany przez pipe** do procesu rodzica. text2 — to tablica znaków, do której

proces rodzica odczytuje dane z pipe, wysłane przez proces potomny, czyli text2 przechowuje tekst **otrzymany od procesu** pipe.

pipe(fd);
tworzymy kanał, którym mogą się komunikować dwa procesy.
fd[1] to miejsce, gdzie proces może wysłać dane.
fd[0] to miejsce, z którego drugi proces może te dane odebrać.

pid_t pid = fork();
fork() tworzy nowy proces. W procesie potomnym zwraca 0, w procesie rodzica zwraca **PID** procesu potomnego, czyli jego unikalny numer.
Jeśli coś pójdzie nie tak, zwraca liczbę ujemną.
pid_t to typ zmiennej do przechowywania numeru procesu.

if (pid == 0) { // sprawdzamy, czy **to proces potomny(0)**.

close(fd[0]);

zamykamy koniec do **odczytu**, bo "dziecko" pisze do pipe.

write(fd[1], text, strlen(text)+1);

wysyłamy tekst przez pipe do procesu rodzica.

close(fd[1]); — zamykamy koniec do **zapisu**, bo "dziecko" skończyło pisać, to jest jak **sygnał dla rodzica**, że teraz może czytać dane z pipe.

Wynik:

```
karyna@karyna-VirtualBox:~/lab2$ make
gcc -c src/main.c -o build/main.o
gcc build/main.o -o build/hello
karyna@karyna-VirtualBox:~/lab2$ ./build/hello
enter text: cyberbezpieczenstwo
Xyberbezpieczenstwo
```