

Interprozesskommunikation mit Named Pipes (FIFO)

Aufgabenstellung:

Es sollen 3 Programm-Module entstehen:

- 1) Ein "**Start-Programm**", welches zwei Sohn-Prozesse generiert
- 2) Ein "**Sende-Programm**", welches Nachrichten über ein Xterm in die Named Pipe einschreibt.
- 3) Ein "**Empfangs-Programm**", welches die Nachrichten aus der Named Pipe auf ein Xterm ausliest.

Benutzen Sie als Rohlinge die "Lückentext-Programme" *pip_main.c*, *pip_send.c* und *pip_empf.c*.

Anleitung zu 1):

Das Start-Programm, der Vater-Prozeß, hat folgende Aufgaben:

- a) Erstellen der Named Pipe mit dem Namen "**FIFO**"
- b) Starten der zwei Kommunikationsprozesse. Hierzu wird neben dem Befehl **fork()** ausserdem der Befehl **execlp()** benötigt. Mit dem Befehl **execlp()** kann ein Xterm geöffnet (evt. positioniert) werden und gleichzeitig innerhalb dieses Xterm das Sende- (Empfangs-) Programm gestartet werden.
Programmbeispiel zum Start eines Programmes in einem Xterm:

```
execlp( "xterm", "xterm", "-geometry", "50x25+0+0",  
"-e", "Programmname", NULL );
```

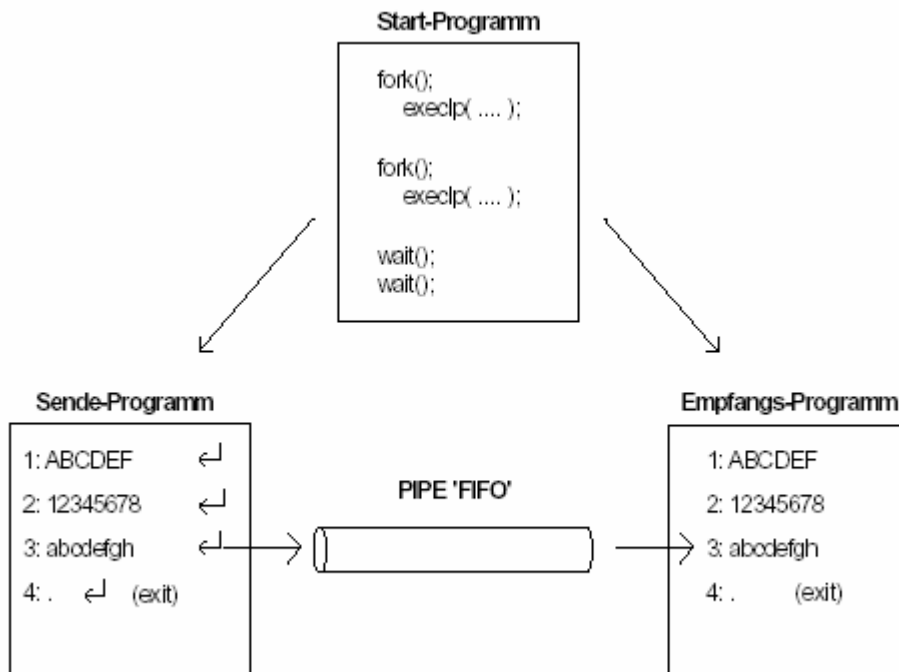
Die Angaben Geometry bestimmen die Positionierung des Xterms auf dem Bildschirm. 100x50 entspricht der Größe des Fensters (in Zeichen und Zeilen, also abhängig von der gewählten Schriftgröße). Die erste +0 ist die x-Koordinate und bedeutet 'Abstand vom linken Bildschirmrand' in Pixeln (-0 bedeutet rechter Bildschirmrand). Die zweite +0 ist die Y-Koordinate und bedeutet 'Abstand vom oberen Bildschirmrand'.
- c) Warten auf das Ende der beiden Sohnprozesse
- d) Löschen der Named Pipe mit dem Namen "**FIFO**"

Anleitung zu 2)

Das Sende-Programm muß zuerst die Named-Pipe öffnen. Nun soll in einer Schleife am Bildschirm eine Zeile eingegeben werden können und nach Beendigung der Zeile durch Return dieser Text in die Pipe eingeschrieben werden. Als Abbruchkriterium der Schleife soll ein Punkt (.) gelten. Hierauf wird die Pipe geschlossen und das Programm beendet.

Anleitung zu 3)

Das Empfangs-Programm muß ebenfalls zuerst die **Named-Pipe** öffnen. Nun wird in einer Schleife auf eine Nachricht in der Pipe gewartet. Ist eine vorhanden, soll diese ausgegeben werden. Wird ein Punkt übertragen, so soll ebenfalls wie oben das Programm beendet werden.



Start-Programm (pip_main.c)

```
/* *****  
/* Startprogramm: Laboruebung PIPES */  
/* *****  
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/wait.h>  
#include <sys/stat.h>  
#include <sys/types.h>  
main( )  
{  
    int pid_1, pid_2, ret_pid;  
    short i;  
    int status;  
    /* Erstellen der Pipe */  
    if( ..... < 0 )  
    {  
        perror("Fehler beim Erstellen der Pipe");  
        exit(-1);  
    }  
    /* Erstellen eines Sohn-Prozesses fuer den Empfaenger */  
    if( ... )  
    {  
        /* Starten des Empfaengers als Xterm */  
        if( ... < 0 )  
        {  
            perror("Fehler Oeffnen Xterm fuer Empfaenger-Prozess");  
            exit(-1);  
        }  
    }  
    /* Erstellen eines Sohn-Prozesses fuer den Sender */  
    if( ... )  
    {  
        /* Starten des Senders als Xterm */  
        if( ... < 0 )  
        {  
            perror("Fehler Oeffnen Xterm fuer Sender-Prozess");  
            exit(-1);  
        }  
    }  
    printf("Sohn-Prozess mit PID %d generiert\n", pid_1 );  
    printf("Sohn-Prozess mit PID %d generiert\n\n", pid_2 );  
    for( i=0; i<2; i++ )  
    {  
        if(( ret_pid = wait( &status )) < 0 )  
        {  
            perror("Fehler beim Warten auf Ende Sohn-Prozess");  
            exit(-1);  
        }  
        printf("Sohn-Prozess mit PID %d wurde beendet\n", ret_pid );  
        printf("Status: %x\n", status >> 8 );  
    }  
  
    if( unlink("FIFO") < 0 ) /* Loeschen einer Datei (hier PIPE) */  
    {  
        perror("Fehler beim Loeschen der Pipe 'FIFO'");  
        exit(-1);  
    }  
    exit(0);  
}
```

Sende-Programm (pip_send.c)

```
/*
*****
*/
/*
*****
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#define ABRUCH "."
#define MAXLEN 256
main( )
{
    int fd;
    int length;
    char nachricht[MAXLEN];
    /* Oeffnen der Pipe */
    if( ... < 0 )
    {
        perror("Fehler beim Oeffnen der Pipe");
        exit(-1);
    }
    printf("Verbindung aufgenommen!\n");
    do
    {
        printf("Eingabe: ");
        gets( nachricht );
        length=strlen( nachricht ) + 1;
        /* Schreiben einer Nachricht in die Pipe */
        if( ... < 0 )
        {
            perror("Fehler beim Schreiben auf die Pipe");
            exit(-1);
        }
    } while( strcmp( nachricht, ABRUCH ) );
    /* Schliessen der Pipe */
    if( ... < 0 )
    {
        perror("Fehler beim Schliessen der Pipe");
        exit(-1);
    }
    exit(0);
}
```

Empfangs-Programm (pip_empf.c)

```
/* **** */
/* Empfangsprogramm: Laboruebung PIPES */
/* **** */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#define ABBRUCH "."
#define MAXLEN 256
main( )
{
    int fd;
    char nachricht[MAXLEN];
    /* Oeffnen der Pipe */
    if( ... < 0 )
    {
        perror("Fehler beim Oeffnen der Pipe");
        exit(-1);
    }
    printf("Verbindung aufgenommen!\n");
    do
    {
        /* Lesen einer Nachricht aus der Pipe */
        if( ... < 0 )
        {
            perror("Fehler beim Lesen von der Pipe");
            exit(-1);
        }
        printf("Ausgabe: %s\n",nachricht);
    } while( strcmp( nachricht, ABBRUCH ));
    /* Schliessen der Pipe */
    if( ... < 0 )
    {
        perror("Fehler beim Schliessen der Pipe");
        exit(-1);
    }
    exit(0);
}
```