FH | JOANNEUM
University of Applied Sciences

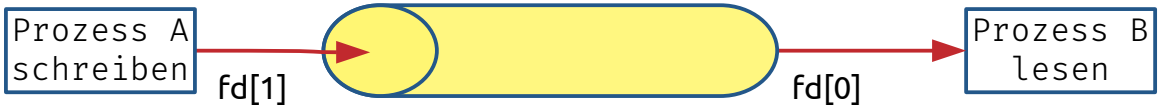
Systemnahe Programmierung
SS 2024

Pipes

Pipes

Was ist eine Pipe?

Eine Pipe ist ein unidirektionaler Kommunikationskanal zwischen Prozessen.



Pipes können nur von Prozessen eingerichtet werden, die gemeinsame Vorfahren besitzen. Normalerweise zwischen Eltern- und Kind-Prozessen.

Der Zugriff erfolgt wie auf Dateien, mit Low-Level Dateideskriptoren.
Die Funktion `pipe()` erzeugt eine Pipe und liefert 2 Filedeskriptoren, die jeweils zum Lesen bzw. Schreiben geöffnet sind:

```
#include <unistd.h>
int pipe( int fd[2]);

fd[0] ist der Filedeskriptor zum Lesen.
fd[1] ist der Filedeskriptor zum Schreiben.
```

Pipes

Pipe verwenden

Die Filedeskriptoren, die durch Pipe angelegt werden können jetzt auf 2 Arten verwendet werden:

1. Elternprozess schreibt, Kindprozess liest
2. Kindprozess schreibt, Elternprozess liest

Wichtig ist, dass jeder Prozess zu Beginn jenen Filedeskriptor schließt, den er nicht benötigt.

Anmerkungen

- Versucht ein Prozess in das „Lesende“ zu schreiben, so wird SIGPIPE ausgelöst, die Default-Aktion ist dann das Programm zu beenden.
- Der geschriebene Inhalt wird gepuffert, bis er auf der Leseseite ausgelesen wird.
- Schließen des Filedeskriptors auf schreibenden Seite erzeugt EOF auf der Leseseite.
- Die maximale Anzahl der Daten, die in eine Pipe geschickt werden können, sind mit der Konstante PIPE_BUF (in limits.h) definiert.

Pipes

Pipe einfacher erzeugen

`popen()` erzeugt eine Pipe, führt `fork()` aus und startet ein Programm mit `exec()` ...
Der aufrufende und der neue Prozess sind dann mit der Pipe verbunden.
Als Rückgabewert erhalten wir einen `stdio` Stream der abhängig von `type` zum Schreiben oder Lesen geöffnet wurde.

```
#include <stdio.h>
FILE *popen(const char *command, const char *type);
```

`command` ... Programm das ausgeführt und mit der Pipe verbunden werden soll.
`type` ... `"r"` (lesen) oder `"w"` (schreiben)

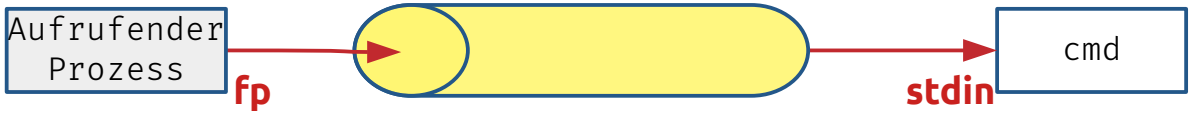
Nach Beendigung ist der erhaltene Stream zu schließen mit:

```
int pclose(FILE *stream);
```

Pipes

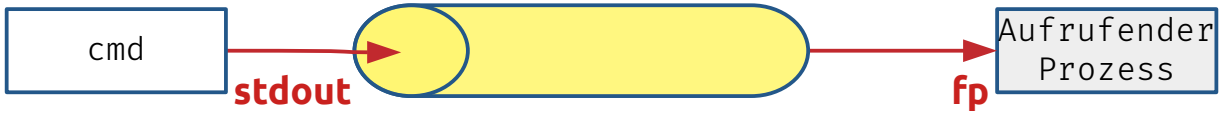
popen()

```
fp = popen(cmd, "w");
```



Der aufrufende Prozess schreibt über `fp` in die Pipe, wo das Programm `"cmd"` es als stdin zur Verfügung gestellt wird.

```
fp = popen(cmd, "r");
```



Der `stdout` Stream des Programms `"cmd"` wird in die Pipe geschrieben und über `fp` dem aufrufenden Prozess zur Verfügung gestellt.

Pipes

Beispiel

Pipe zu `"ls -l"` aufmachen und den Output mit Zeilennummer ausgeben:

```
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>

#define BUFFER_SIZE 256

int main(int argc, char *argv[]) {
  char buffer[BUFFER_SIZE];
  FILE *fp = popen("ls -l", "r");
  if (fp == NULL) {
    perror("Pipe");
    exit(EXIT_FAILURE);
  }
  int i=0;
  while (fgets(buffer, BUFFER_SIZE, fp) != NULL) {
    printf("%d %s", ++i, buffer);
  }

  pclose(fp);
}
```