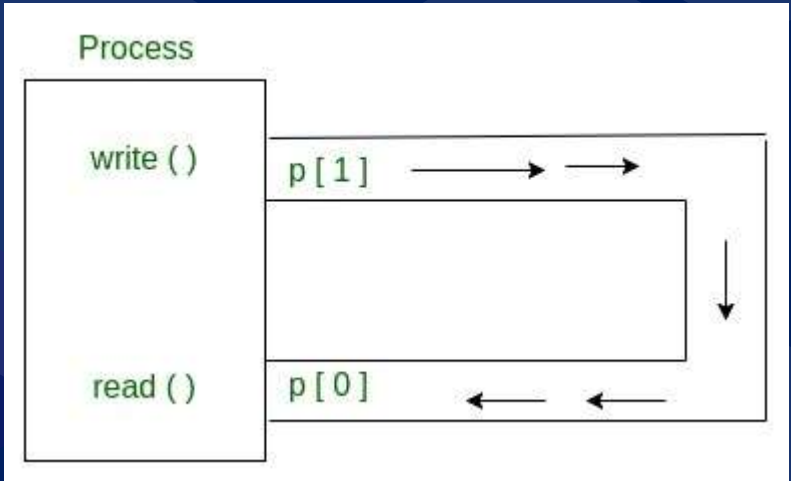
دانشگاه صنعتی امیرکبیر

آزمایشگاه سیستم‌عامل

جلسه هفتم: آشنایی با فراخوانی سیستمی لوله

مدرس :مینا یوسف‌نژاد

pipe یک ساختار داده‌ای است که برای ارتباط بین دو فرایند در سیستم‌های عامل استفاده می‌شود. این ساختار به صورت یک کانال یک‌طرفه عمل می‌کند و داده‌ها را از یک فرایند به فرایند دیگر منتقل می‌کند. لوله‌ها به عنوان یک مکانیزم ساده و کارآمد برای ارتباط بین فرایندهای والد و فرزند، یا بین چندین فرایند مستقل استفاده می‌شوند.



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

pipe(int pipefd[2]): این تابع سیستم برای ایجاد یک لوله استفاده می‌شود و دو توصیفگر فایل (file descriptor) برمی‌گرداند:

- pipefd[0]: سمت خواندن لوله (reading end)
- pipefd[1]: سمت نوشتن لوله (writing end)

فرایندهایی که می‌خواهند داده دریافت کنند، از سمت خواندن استفاده می‌کنند و فرایندهایی که می‌خواهند داده ارسال کنند، از سمت نوشتن استفاده می‌کنند. این فراخوانی در صورت موفقیت صفر و در صورت شکست - 1 برمیگرداند.

تابع open()

این تابع برای باز کردن یک فایل استفاده می‌شود و یک توصیفگر فایل (file descriptor) برمی‌گرداند که برای انجام عملیات بعدی روی فایل مورد استفاده قرار می‌گیرد.

آرگومان‌ها:

pathname: مسیر کامل فایل (absolute path) یا مسیر نسبی relative (path)

flags: مشخص می‌کند که چه عملیاتی روی فایل انجام شود (خواندن، نوشتن، ایجاد و ...)

mode: مجوزهایی که برای فایل و صاحبان مختلف آن در نظر گرفته می‌شود.

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

flags

O_RDONLY: فقط برای خواندن باز می‌شود.

O_WRONLY: فقط برای نوشتن باز می‌شود.

O_RDWR: برای خواندن و نوشتن باز می‌شود.

O_CREAT: اگر فایل وجود نداشته باشد، ایجاد می‌شود.

O_APPEND: داده‌های جدید به انتهای فایل اضافه می‌شوند.

mode: با استفاده از اعداد اکتال یا ثابت‌های تعریف شده در <sys/stat.h> مشخص می‌شود.

هر عدد به سه قسمت تقسیم می‌شود که به ترتیب مجوزهای مالک، گروه و سایر کاربران را مشخص می‌کنند.

هر قسمت شامل سه بیت است که به ترتیب نشان‌دهنده مجوزهای خواندن، نوشتن و اجرا هستند.

مقدار بازگشتی: در صورت موفقیت، توصیفگر فایل باز شده را برمی‌گرداند. در صورت شکست، مقدار -1 را برمی‌گرداند و خطا را می‌توان با errno و perror() بررسی کرد.

تابع close():

این تابع برای بستن یک فایل باز شده استفاده می‌شود. پس از بستن فایل، توصیفگر فایل دیگر معتبر نیست و منابع سیستم آزاد می‌شوند.

آرگومان:

fd: توصیفگر فایلی که می‌خواهیم ببندیم.

مقدار بازگشتی:در صورت موفقیت، صفر را برمی‌گرداند. در صورت شکست، مقدار -1 را برمی‌گرداند و خطا را می‌توان با errno و perror() بررسی کرد.

```
#include<unistd.h>

ssize_t read(int fd, void *buf, size_t count)
```

تابع read():

این تابع برای خواندن داده از یک فایل باز شده استفاده می‌شود.

آرگومان‌ها:

fd: توصیفگر فایل.

buf: آدرس بافری که داده‌های خوانده شده در آن قرار می‌گیرد.

count: حداکثر تعداد بایتی که می‌خواهیم بخوانیم.

مقدار بازگشتی: در صورت موفقیت، تعداد بایت‌های خوانده شده را برمی‌گرداند. در صورت رسیدن به انتهای فایل، صفر را برمی‌گرداند. در صورت خطا، مقدار -1 را برمی‌گرداند و خطا را می‌توان با errno و perror() بررسی کرد.

```c
#include<unistd.h>

ssize_t read(int fd, void *buf, size_t count)
```

تابع write:

این تابع برای نوشتن داده‌ها در یک فایل (یا هر توصیفگر فایل باز دیگری مانند لوله) استفاده می‌شود.

آرگومان‌ها:

fd: توصیفگر فایلی که می‌خواهیم در آن بنویسیم. این توصیفگر توسط توابعی مانند open یا pipe ایجاد می‌شود.

buf: آدرس بافری است که داده‌های مورد نظر برای نوشتن در آن قرار دارند.

count: تعداد بایتی است که می‌خواهیم از بافر buf بنویسیم.

مقدار بازگشتی:در صورت موفقیت، تعداد بایت‌های نوشته شده را برمی‌گرداند. در صورت شکست (مثلاً خطا در نوشتن یا رسیدن به انتهای فایل)، مقدار منفی یک (-1) را برمی‌گرداند.

خطاها:خطاهای رایج شامل پر شدن فضای دیسک، نبود مجوزهای کافی برای نوشتن، یا خطاهای مربوط به توصیفگر فایل هستند.

```
#include<unistd.h>

ssize_t write(int fd, void *buf, size_t count)
```