4.1 进程间通讯 - 无名管道_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.1 进程间通讯 – 无名管道涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

全部开发者教程

物联网 / 嵌入式工程师

第 24 周 stm32 芯片 – 智能硬件项目实战与企业笔试

未命名节

第 25 周 大厂必备 – linux 内核与文件系统移植

未命名节

第 26 周 嵌入式开发 – 系统移植 – bootloader、yocto

未命名节

第 27 周 嵌入式底层核心技能 – Linux 设备驱动初级

未命名节

第 28 周 嵌入式底层核心技能 – Linux 设备驱动中级

未命名节

第 29 周 嵌入式底层核心技能 – Linux 设备驱动高级 1

未命名节

第 30 周 嵌入式底层核心技能 – Linux 设备驱动高级 2

未命名节

第 31 周 嵌入式人工智能必备 – Python

未命名节

第 32 周 智能家居项目实战之客户端功能开发

未命名节

第 33 周 智能家居项目实战之网关端功能开发

未命名节

第 34 周 智能家居项目实战之设备端功能开发

未命名节

第 35 周 物联网 / 嵌入式项目答辩和就业指导

未命名节

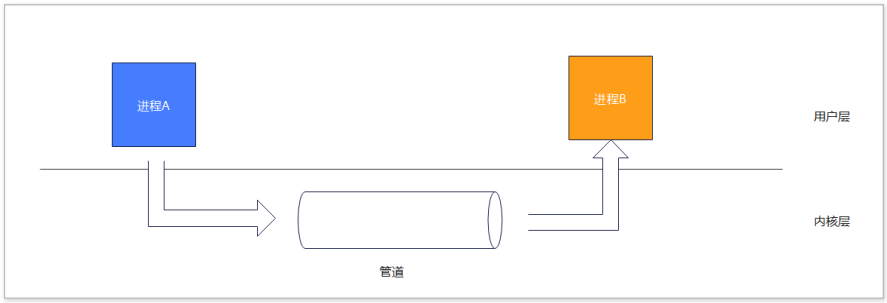
第 36 周 独立开发阶段 – 三大热门领域项目

未命名节

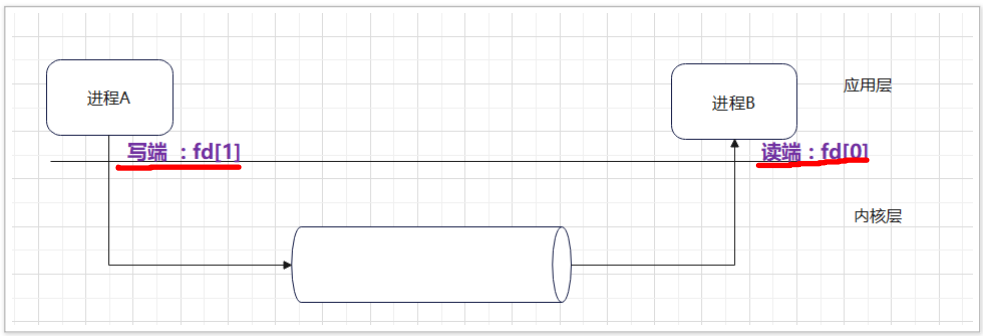
- linux 下的进程通信手段基本上是从 Unix 平台上的进程通信手段继承而来的。
- 每个进程都有自己独立的地址空间, 当两个不同进程需要进行交互时, 就需要使用进程间通讯
- 进程间通讯分为单个计算机的进程间通讯与局域网的计算机的进程间通讯
- 进程间通讯方式有 管道, 信号, 消息队列, 共享内存, 网络

pipe mkfifo

- 管道分为 无名管道 与 有名管道
  - 无名管道用于父子进程之间通讯
  - 有名管道用于任意进程之间通讯
- 管道的本质是在内存建立一段缓冲区，由操作系统内核来负责创建与管理, 具体通讯模型如下:



- 无名管道的特点:
  - 无名管道属于单向通讯
  - 无名管道只能用于 父子进程通讯
  - 无名管道发送端叫做 写端, 接收端叫做 读端
  - 无名管道读端与写端抽象成两个文件进行操作，在无名管道创建成功之后，则会返回读端与写端的文件描述符

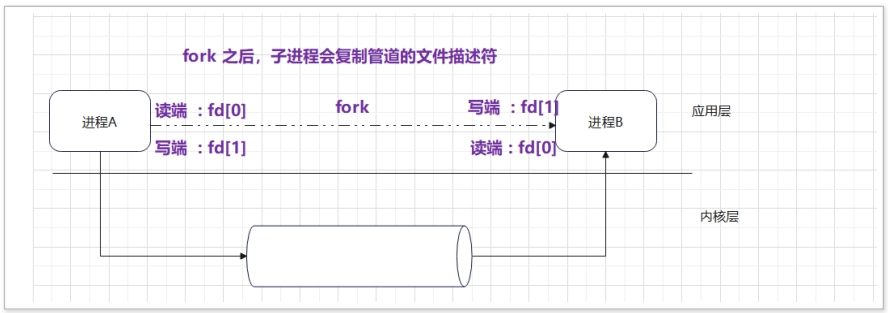


- 创建无名管道需要调用 pipe() 函数
  - 函数头文件
  - #include <unistd.h>
  - 函数原型
  - int pipe(int pipefd[2]);
  - 函数参数
  - pipefd：用于存储无名管道读端与写端的文件描述符的数组
    - pipefd[0]：读端文件描述符
    - pipefd[1]：写端文件描述符
  - 函数返回值:
  - 成功：0
  - 失败： –1, 设置 errno

-

- 示例：创建子进程，父进程通过管道向子进程发送“Hello,pipe”

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


int main(void)
{
    pid_t cpid;
    int ret;
    int pipefd[2];

    ret = pipe(pipefd);
    if (ret == -1){
        perror("[ERROR] pipe(): ");
        exit(EXIT_FAILURE);
    }

    cpid = fork();
    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if (cpid == 0){
        ssize_t rbytes;
        char buffer[64] = {0};
        close(pipefd[1]);

        rbytes = read(pipefd[0],buffer,sizeof(buffer));
        if (rbytes == -1){
            perror("[ERROR] read(): ");
            close(pipefd[0]);
            exit(EXIT_FAILURE);
        }

        printf("buffer : %s\n",buffer);

        close(pipefd[0]);

    }else if (cpid > 0){
        ssize_t wbytes;
        char buffer[] = "hello,pipe.";
        close(pipefd[0]);
        wbytes = write(pipefd[1],buffer,strlen(buffer));
        if (wbytes == -1){
            perror("[ERROR] write(): ");
            wait(NULL);
            close(pipefd[1]);
            exit(EXIT_FAILURE);
        }

        close(pipefd[1]);
        wait(NULL);
    }

return 0;
}
```



- 无名管道的特点
    - 当管道为空时，读管道会阻塞读进程
    - 当管道的写端被关闭了，从管道中读取剩余数据后，read 函数返回 0

- 在写入管道时，确保不超过 PIPE_BUF 字节的操作是原子的
    - 当写入的数据达到 PIPE_BUF 字节时，write() 会在必要的时候阻塞直到管道中的可用空间足以原子地完成操作
    - 当写入的数据大于 PIPE_BUF 字节时，write() 会尽可能多传输数据以充满这个管道
- 管道的大小是有限的，不能让父 / 子进程同时对管道进行读 / 写操作
- 当一个进程试图想一个管道中写入数据但没有任何进程拥有该管道的打开着的读取描述符，内核向写入进程发送一个 SIGPIPE 信号

- 练习
    - 创建一个子进程，负责循环从管道中读取数据，父进程从键盘读取数据后，写入到管道中，输入“quit”字符串时退出

全文完