内容回顾

2017年5月23日 8:55

进程控制

fork，getpid，getppid

exec族函数

进程回收

wait、waitpid

学习目标

2017年5月22日 9:23

1. 熟练使用pipe进行父子进程间通信
2. 熟练使用pipe进行兄弟进程间通信
3. 熟练使用fifo进行无血缘关系的进程间通信
4. 熟练掌握mmap函数的使用
5. 掌握mmap创建匿名映射区的方法
6. 使用mmap进行有血缘关系的进程间通信
7. 使用mmap进行无血缘关系的进程间通信

IPC概念

2017年5月22日 9:30

IPC：InterProcess Communication 进程间通信 ，通过内核提供的缓冲区进行数据交换的机制。

IPC通信的方式有几种：

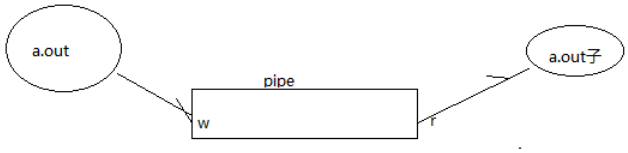
- pipe 管道 -- 最简单
- fifo 有名管道
- mmap 文件映射共享IO -- 速度最快
- 本地socket 最稳定
- 信号 携带信息量最小
- 共享内存
- 消息队列

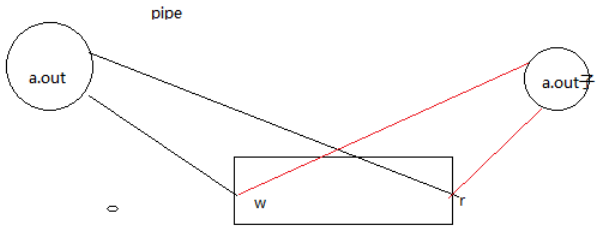
PIPE通信

2017年5月22日 9:31

常见的通信方式：单工（广播），半双工（对讲机），全双工（打电话）

管道：半双工通信





管道函数：

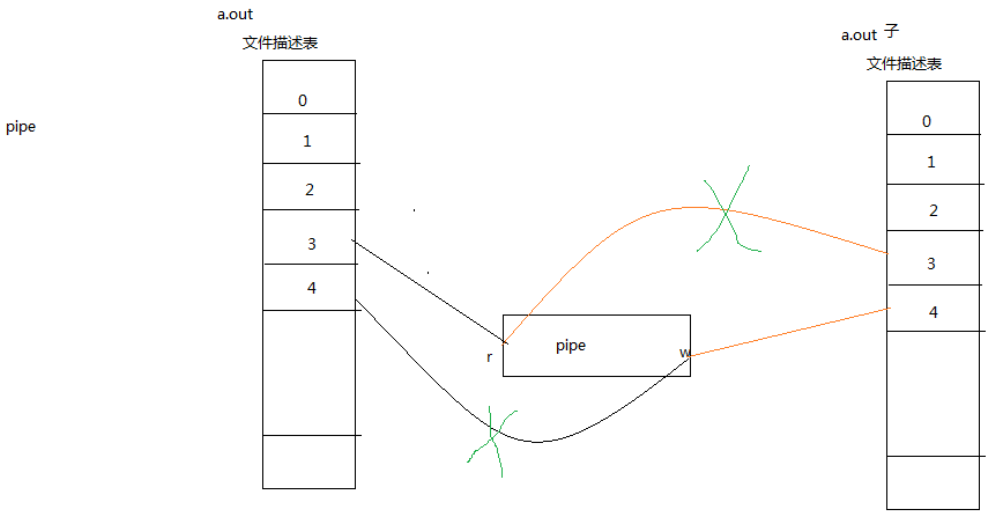
int pipe(int pipefd[2]);

- pipefd 读写文件描述符 ，0-代表读，1-代表写
- 返回值：失败返回-1，成功返回0

父子进程实现pipe通信，实现ps aux|grep bash 功能

代码的问题：

父进程认为还有写端存在，就有可能还有人给发数据，继续等待。



读管道：

- 写端全部关闭 -- read读到0，相当于读到文件末尾
- 写端没有全部关闭
  - 有数据 -- read 读到数据
  - 没有数据 -- read 阻塞 fcntl函数可以更改非阻塞

写管道：

- 读端全部关闭 ---- ？产生一个信号SIGPIPE，程序异常终止
- 读端未全部关闭
  - 管道已满 -- write阻塞 --如果要显示现象，读端一直不读，写端狂写。
  - 管道未满 -- write正常写入

计算管道大小 512*8

long fpathconf(int fd, int name);

ulimit -a

优点：

- 简单

缺点：

- 只能有血缘关系的进程通信
- 父子进程单方向通信，如果需要双向通信，需要创建多根管道

实现兄弟进程间通信，ps aux|grep bash

FIFO通信

2017年5月22日 9:31

FIFO 有名管道，实现无血缘关系进程通信。

- 创建一个管道的伪文件
  - mkfifo myfifo 命令创建
  - 也可以用函数 int mkfifo(const char *pathname, mode_t mode);
- 内核会针对fifo文件开辟一个缓冲区，操作fifo文件，可以操作缓冲区，实现进程间通信--实际上就是文件读写

FIFOs

Opening the read or write end of a FIFO blocks until the other end is also opened (by another process or thread). See fifo(7) for further details.

open注意事项，打开fifo文件的时候，read端会阻塞等待write端open，write端同理，也会阻塞等待另外一端打开

```
//fifo_w.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main(int argc,char * argv[])
{
   if(argc != 2){
      printf("./a.out fifoname\n");
      return -1;
   }
```

```
    // 当前目录有一个 myfifo 文件
    //打开fifo文件
    printf("begin open ....\n");
    int fd = open(argv[1],O_WRONLY);
    printf("end open ....\n");
    //写
    char buf[256];
    int num = 1;
    while(1){
        memset(buf,0x00,sizeof(buf));
        sprintf(buf,"xiaoming%04d",num++);
        write(fd,buf,strlen(buf));
        sleep(1);
        //循环写
    }
    //关闭描述符
    close(fd);
    return 0;
}

//fifo_r
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main(int argc,char *argv[])
{
    if(argc != 2){
        printf("./a.out fifoname\n");
        return -1;
    }
    printf("begin oepn read...\n");
    int fd = open(argv[1],O_RDONLY);
    printf("end oepn read...\n");
```

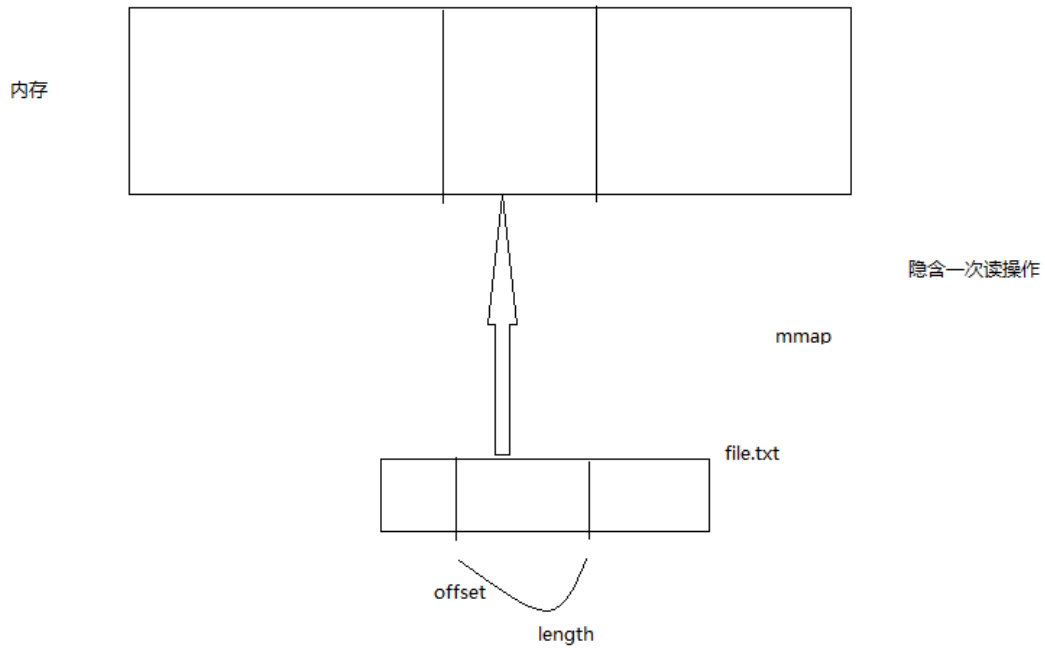
```
    char buf[256];
    int ret;
    while(1){
        //循环读
        memset(buf,0x00,sizeof(buf));
        ret = read(fd,buf,sizeof(buf));
        if(ret > 0){
            printf("read:%s\n",buf);
        }
    }

    close(fd);
    return 0;
}
```

man 7 fifo

查看fifo信息

mmap共享映射区

2017年5月22日 9:31



创建映射区

void *mmap(void *addr, size_t length, int prot, int flags,int fd, off_t offset);

- addr 传NULL
- length 映射区的长度
- prot
  - PROT_READ 可读
  - PROT_WRITE 可写
- flags
  - MAP_SHARED 共享的，对内存的修改会影响到源文件
  - MAP_PRIVATE 私有的
- fd 文件描述符，open打开一个文件
- offset 偏移量
- 返回值
  - 成功 返回 可用的内存首地址
  - 失败 返回 MAP_FAILED

释放映射区

int munmap(void *addr, size_t length);

- addr 传mmap的返回值

- length mmap创建的长度
- 返回值

mmap九问：

1. 如果更改mem变量的地址，释放的时候munmap，传入mem还能成功吗？不能！！
2. 如果对mem越界操作会怎么样？文件的大小对映射区操作有影响，尽量避免。
3. 如果文件偏移量随便填个数会怎么样？ offset必须是 4k的整数倍
4. 如果文件描述符先关闭，对mmap映射有没有影响？没有影响
5. open的时候，可以新创建一个文件来创建映射区吗？不可以用大小为0的文件
6. open文件选择O_WRONLY，可以吗？ 不可以： Permission denied
7. 当选择MAP_SHARED的时候，open文件选择O_RDONLY，prot可以选择PROT_READ|PROT_WRITE吗？Permission denied ，SHARED的时候，映射区的权限 <= open文件的权限
8. mmap什么情况下会报错？很多情况
9. 如果不判断返回值会怎么样？ 会死的很难堪！！

用mmap 实现父子进程之间通信

匿名映射：

MAP_ANON，ANONYMOUS 这两个宏在有些unix系统没有

可以使用/dev/zero文件，打开

/dev/zero 聚宝盆 ，可以随意映射

/dev/null 无底洞 ，一般错误信息重定向到这个文件中

用mmap支持无血缘关系进程通信

如果进程要通信，flags 必须设为 MAP_SHARED

作业

2017年5月22日 9:26

1.通过命名管道传输数据，进程A和进程B，进程A将一个文件（MP3）发送给进程B

2.实现多进程拷贝文件

…

2017年5月22日 10:50

ubuntu上的游戏

http://blog.csdn.net/suwu150/article/details/52714396

管道容量：管道容量分为pipi capacity 和 pipe_buf .这两者的区别在于pipe_buf定义的是内核管道缓冲区的大小，这个值的大小是由内核设定的，这个值仅需一条命令就可以查到；而pipe capacity指的是管道的最大值，即容量，是内核内存中的一个缓冲区。

pipe_buf：命令：ulimit -a

信号的概念

2017年5月23日 17:20

信号的特点：简单，不能带大量信息，满足特定条件发生

信号的机制：进程B发送给进程A，内核产生信号，内核处理。

信号的产生：

- 按键产生 ctrl+c ctrl+z ctrl+\
- 调用函数 kill raise abort
- 定时器 alarm ，setitimer
- 命令产生 kill
- 硬件异常 段错误，浮点型错误，总线错误，SIGPIPE

信号的状态：

- 产生
- 递达 信号到达并且处理完
- 未决 信号被阻塞了

信号的默认处理方式：

- 忽略
- 执行默认动作
- 捕获

信号的4要素：

- 编号
- 事件
- 名称
- 默认处理动作
  - 忽略
  - 终止

- 终止+core
- 暂停
- 继续