操作系统原理及Linux内核

西安邮电大学

进程间通信

什么是进程间通信?

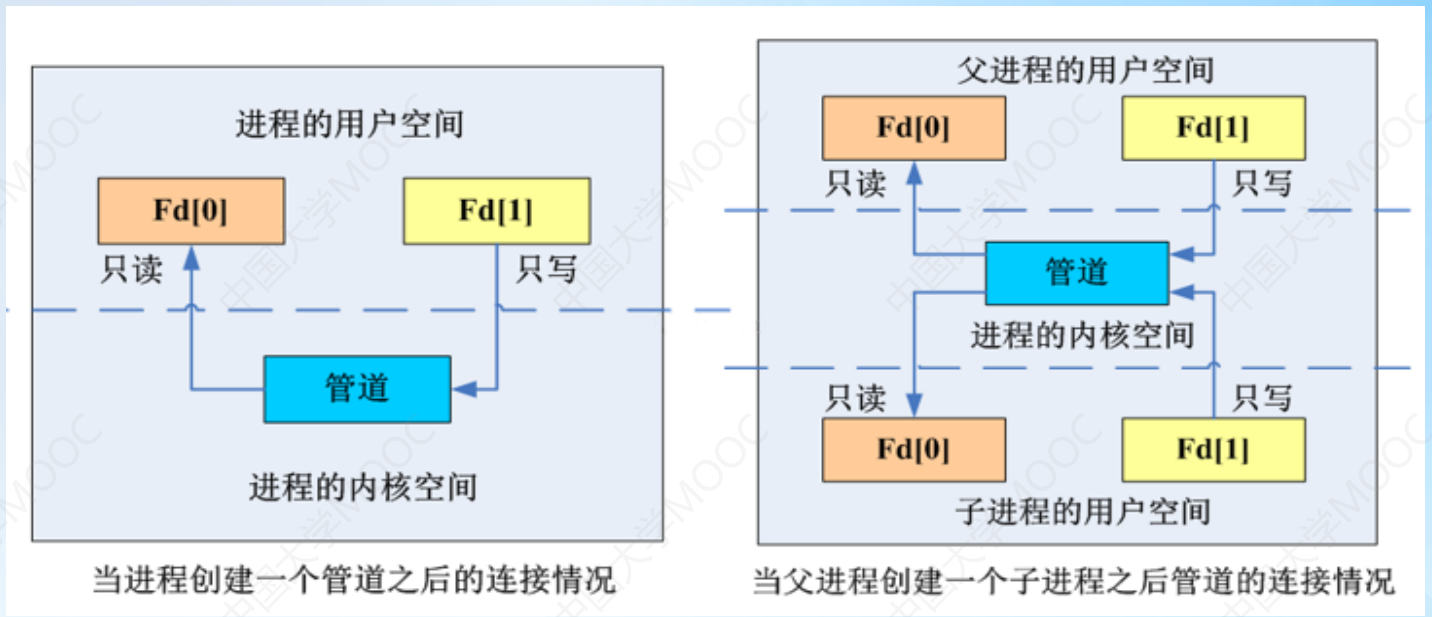
什么是进程间通信（IPC）？

进程间主要通信方式

1. 管道通信（pipe）
2. 共享内存（Shm）
3. 消息队列（Msg）
4. 套接字（Socket）

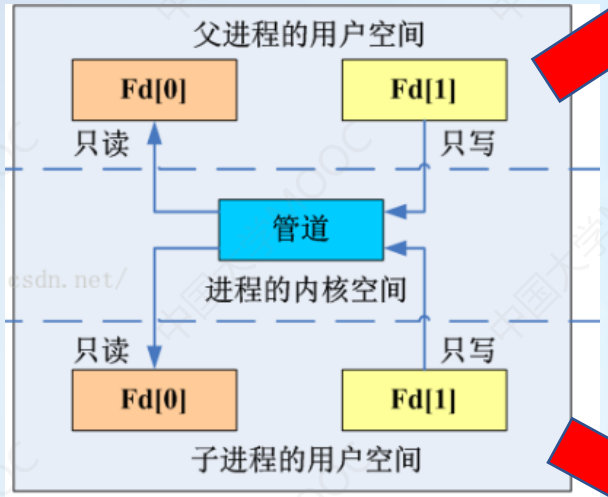
管道通信

发送进程写　　接收进程读



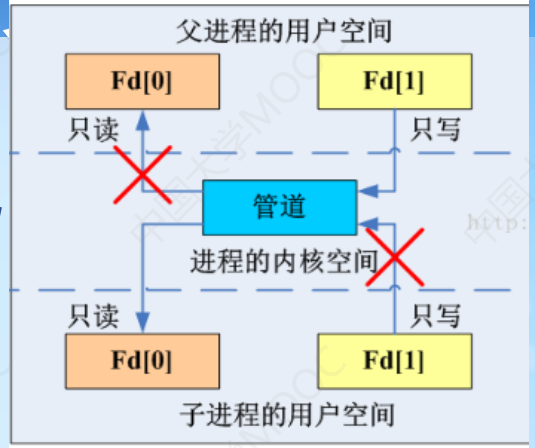
图（a）　　图（b）

使用管道进行



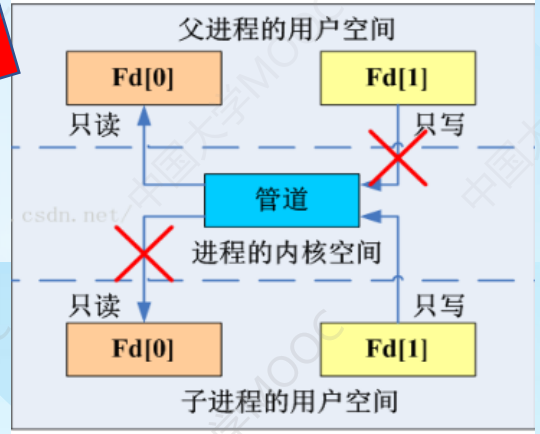
当父进程创建一个子进程之后管道的连接情况

父进程写、子进程读



当确定连接关系之后管道的连接情况

子进程写、父进程读



当确定连接关系之后管道的连接情况

管道通信

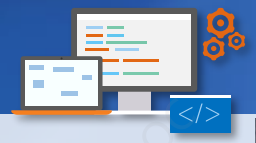
1 互斥 读写操作必须互斥进行

2 同步 读写进程等待与唤醒

3 对方是否存在 只有判断对方存在时才能进行通信

管道通信

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>
//往管道写
void write_to_pipe(int fd)
{
    char *message = "hello pipe!\n";
    write(fd,message,strlen(message)+1);
}
//从管道读
void read_from_pipe(int fd)
{
    char message[100];
    read(fd,message,100);
    printf("read from pipe:%s",message);
}
```

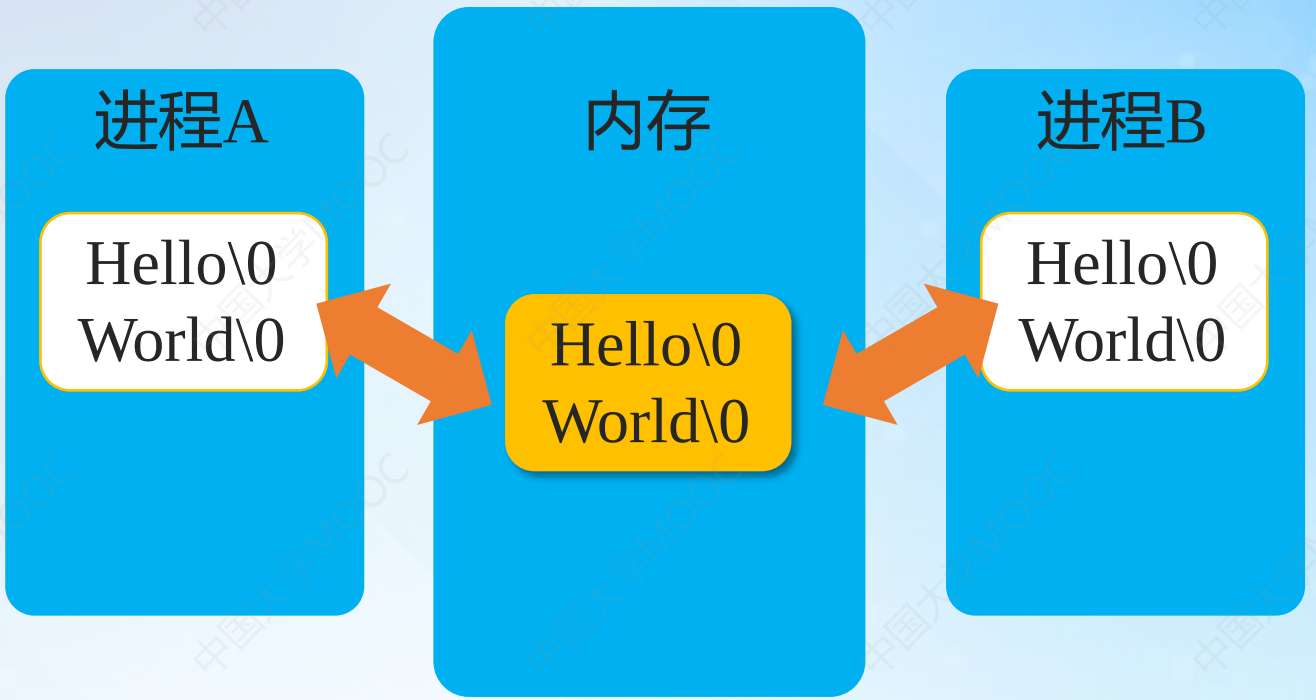


Linux管道通信简单举例

```
19 int main(){
20     int fd[2];    //定义文件描述符
21     pid_t pid;    //进程pid
22     int stat_val;  //返回值
23     if(pipe(fd)){ //创建管道
24         printf("creat pipe falied!\n");
25         exit(1);
26     }
27     pid = fork();  //创建子进程
28     switch(pid){
29             case -1:
30                 printf("fork error!\n");
31                 exit(1);
32             case 0:
33                 close(fd[1]);  //关闭写端
34                 read_from_pipe(fd[0]);
35                 exit(0);
36             default :
37                 close(fd[0]); //关闭读端
38                 write_to_pipe(fd[1]);
39                 wait(&stat_val);
40                 exit(0);
41     }
42     return 0;
43 }
```

共享内存(SHM)

共享内存是两个进程可以直接共享访问同一块内存区域。



Linux中共享内存的编程实现

获得共享内存 shmget()

获取共享内存对象的ID

映射共享内存 shmat()

将共享内存映射至本进程
虚拟内存空间的某个区域

在此使用映射的内存

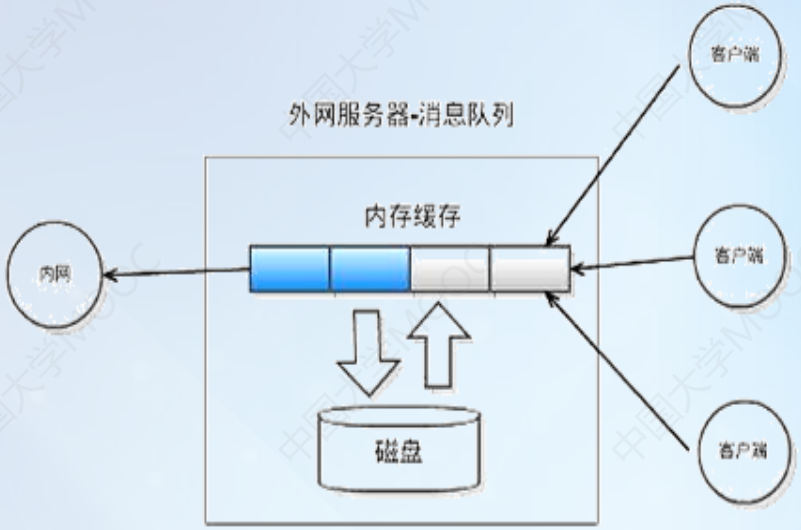
解除内存 shmdt()

当不再使用时，解除映射
关系

控制共享内存 shmctl()

当没有进程再需要这块共享
内存时，可以删除它

消息传递通信(MSG)



Message 以格式化的消息(message)为通信单位

send 发送：当要进行消息传递时执行send（）

receive 接受：当接收者要接收消息时执行recieve（）

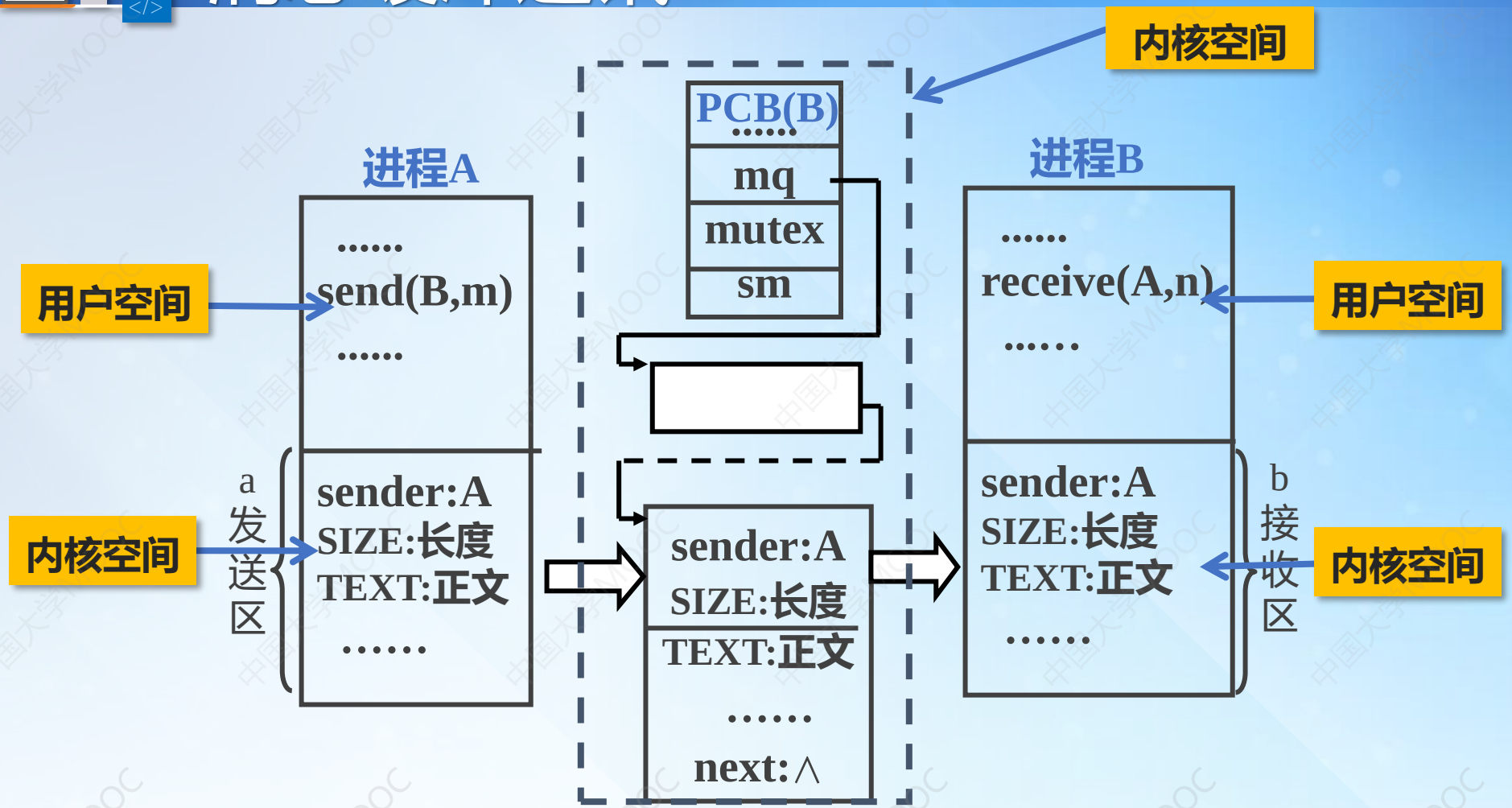
直接通信 接发送消息指定对方的名字

间接通信 接发送消息指定一个邮箱

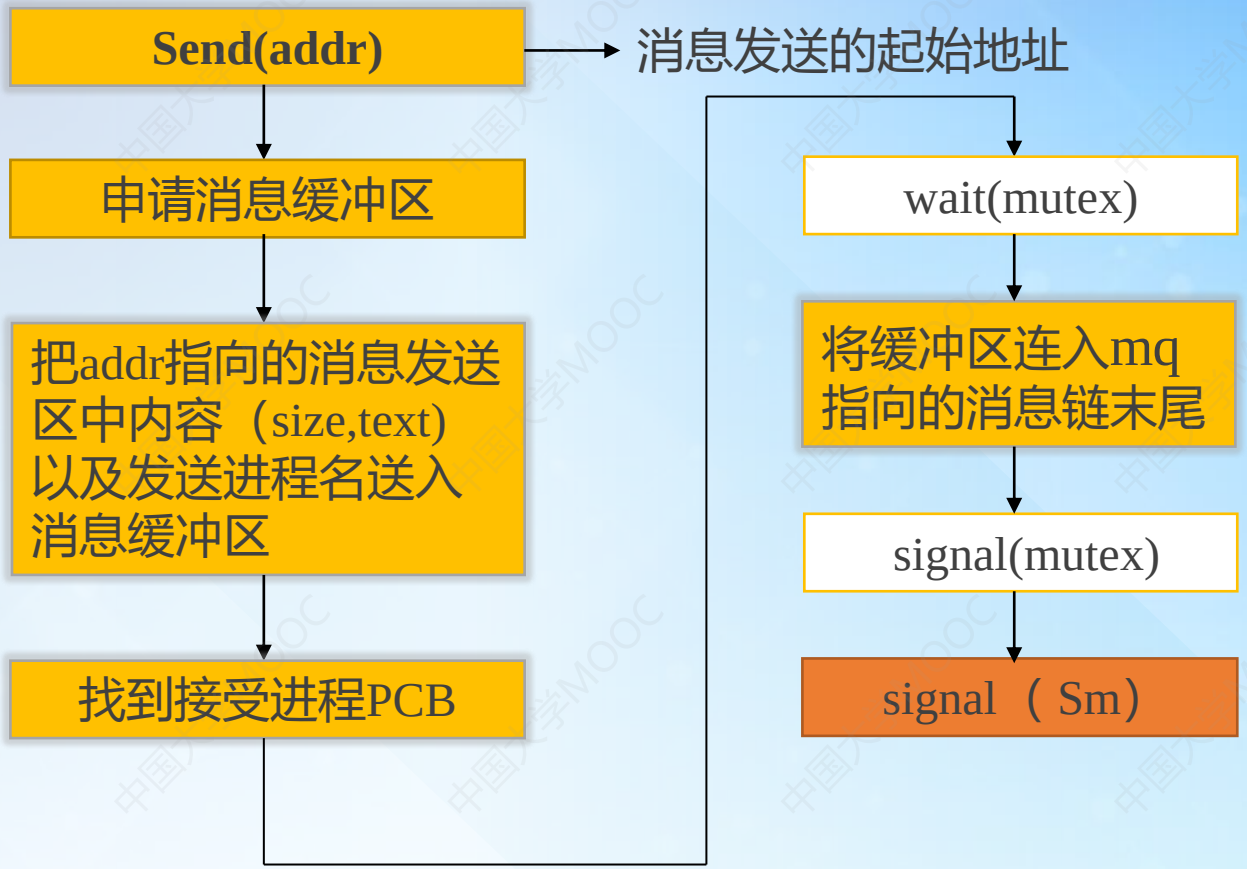
Linux中消息队列的相关函数

函数名	功能
msgget（）	创建和访问一个消息队列
msgsnd（）	把一条消息添加到消息队列中
msgrcv（）	从一个消息队列接受消息
msgctl（）	消息队列的控制函数

消息缓冲通讯


内核空间
PCB(B)
.....
mq
mutex
sm
进程A
.....
send(B,m)
.....
用户空间
a
发送区
sender:A
SIZE:长度
TEXT:正文
.....
内核空间
sender:A
SIZE:长度
TEXT:正文
.....
next:∧
进程B
.....
receive(A,n)
.....
用户空间
sender:A
SIZE:长度
TEXT:正文
.....
b
接收区
内核空间

消息队列Send原语流程图



用wait-signal操作来实现Send原语

```
Procedure Send(rec, a); { rec为接收进程, a为发送区}
Begin
        Getbuf(a.size, i); {从自由区中申请空缓冲区i}
        i.sender:=a.sender; {把消息从a处copy到缓冲区i}
        i.size:=a.size;
         i.text:=a.text;
         i.next:=0;
        GetID(PCB set, rec.j); {获接收进程内部标识符j}
        wait(j.mutex);
        insert(j.mq , i); {把缓冲区i挂到进程rec消息链尾}
        signal(j.mutex);
        signal(j.sm);
END
```

小结

进程间通信

- 管道
- 共享内存
- 消息队列
- 套接字