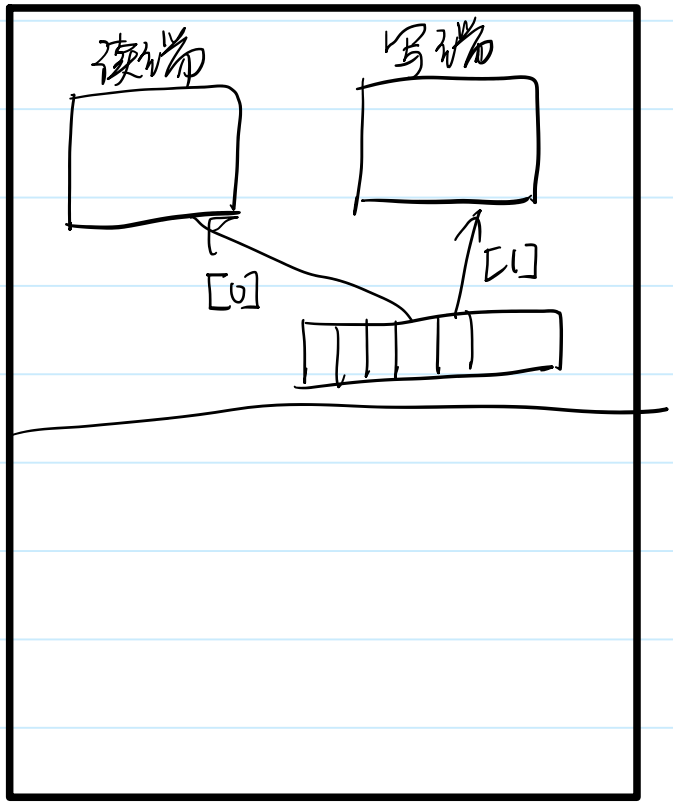
无名管道的系统调用

2023年8月12日 9:26

```
int pipe(int pipefd[2]);
```

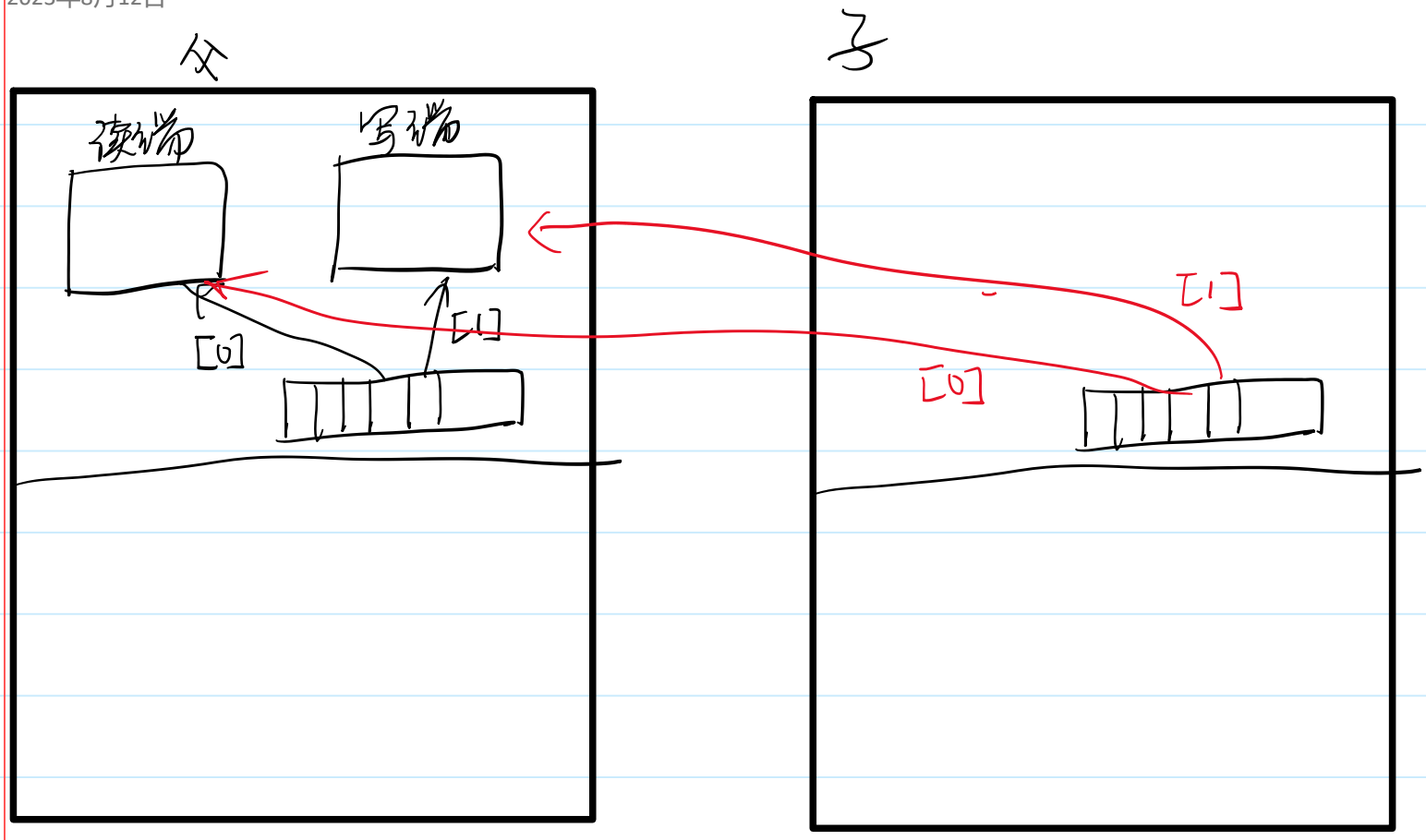
不 2在内核上开的，接个作用




```c
int main(int argc, char *argv[])
{
    int fds[2];
    // pipe(fds[2]); fds[2] 是通过fds访问2号（第3个）
    pipe(fds);
    printf("fds[0] = %d, fds[1] =  %d\n", fds[0], fds[1]);
    write(fds[1],"howareyou",9);
    char buf[4096] = {0};
    read(fds[0],buf,sizeof(buf));
    printf("buf = %s\n", buf);
    return 0;
}
```

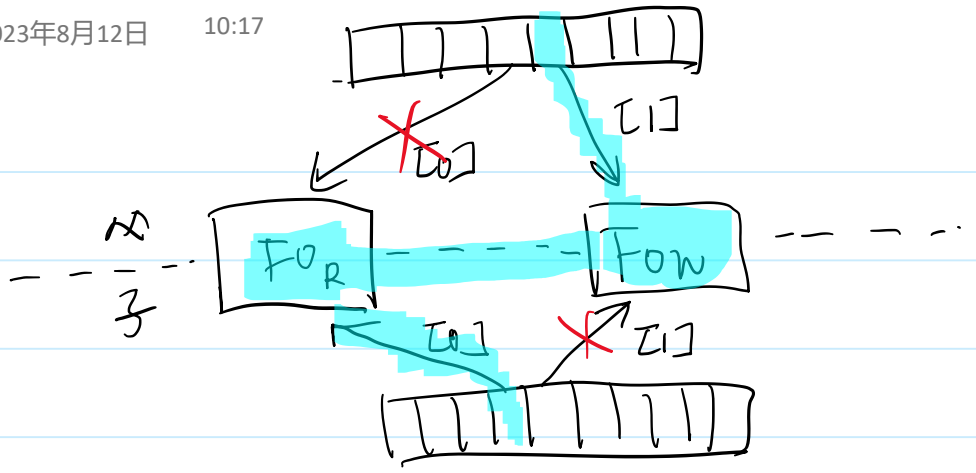
先调用pipe 再fork

2023年8月12日 10:15



半双工通信

2023年8月12日 10:17



单工？ 父→子

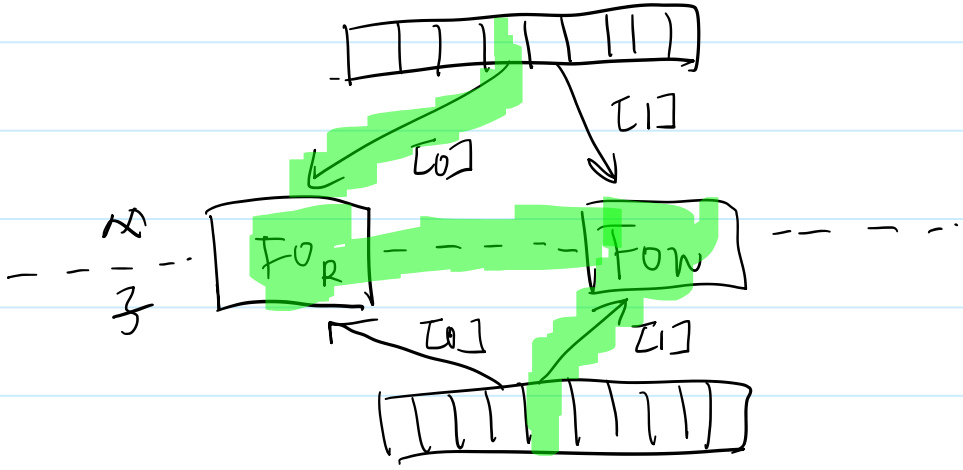
使用无名管道的一般流程

① pipe

② fork

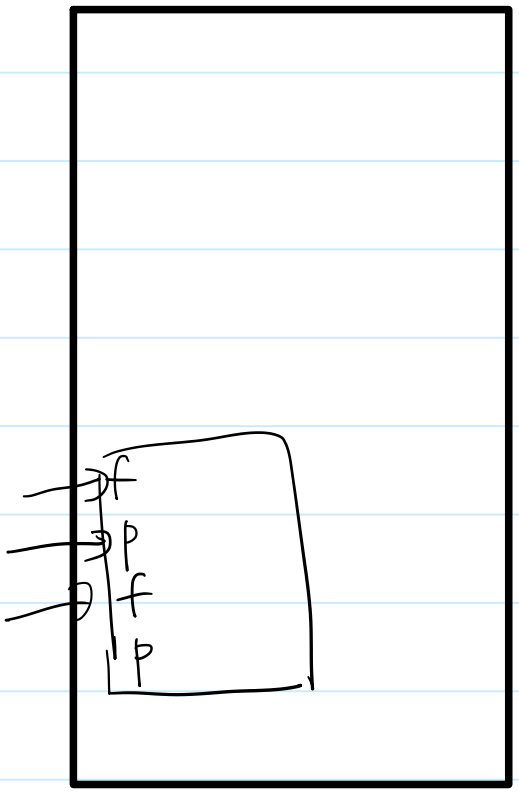
③ 父：close 一端

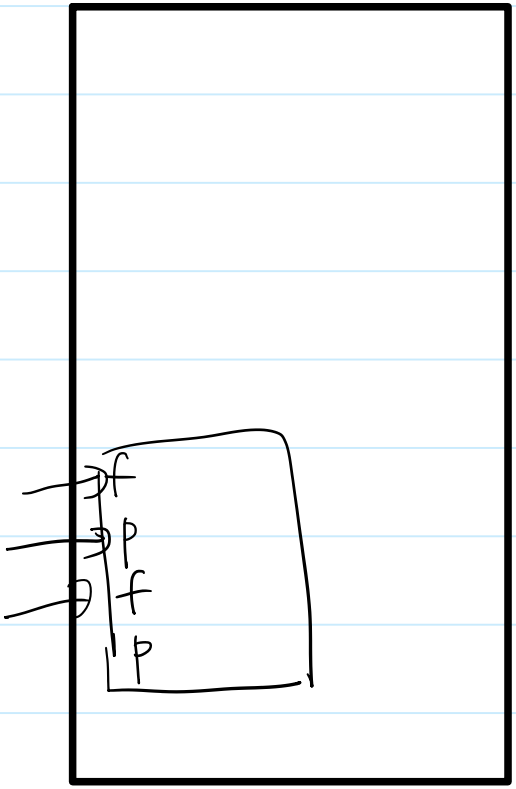
子：close 另一端

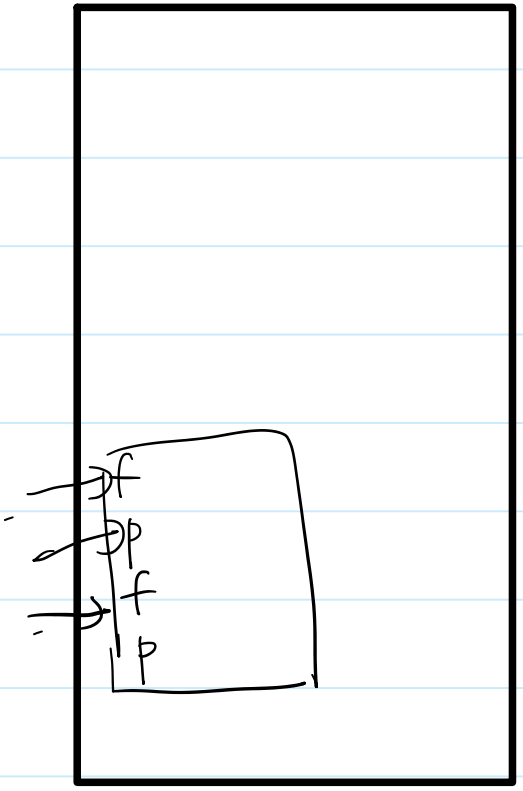


2023年8月12日 11:03

```
int main()
{
    fork();
    printf("-\n");
    fork();
    printf("-\n");
    return 0;
}
```




f
p
f
p


f
p
f
p

共享内存

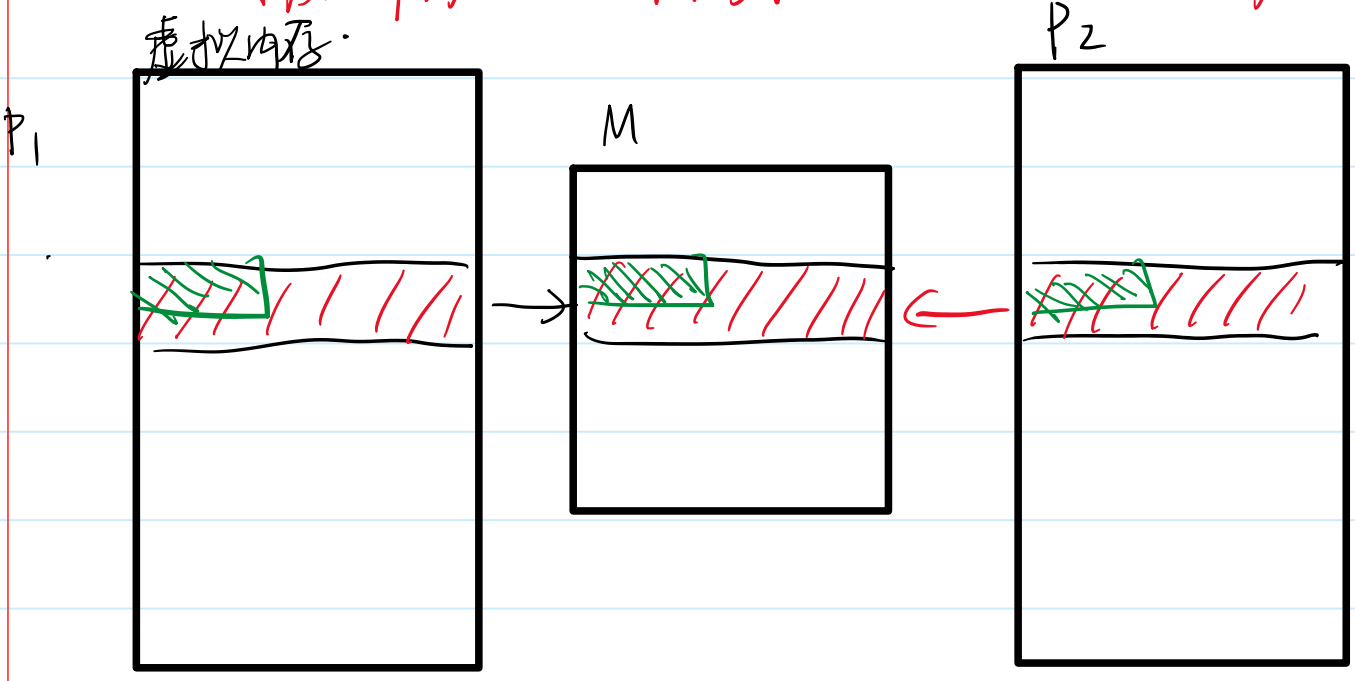
2023年8月12日 11:12

管道效率低 ①. 进程很多，管道 $O(n^2)$

② 阻塞.

共享内存最快的IPC.

共享内存部分的内存隔离性：让不同进程的虚拟页，映射到同一物理页



传递信息的时候不需要拷贝.

使用共享内存需要关注的事情

2023年8月12日 11:19

① 让OS分配物理内存

② 将这个共享的物理内存加载到进程地址空间

③ 进程访问/分配的内存.

分配共享的物理内存

2023年8月12日 11:23

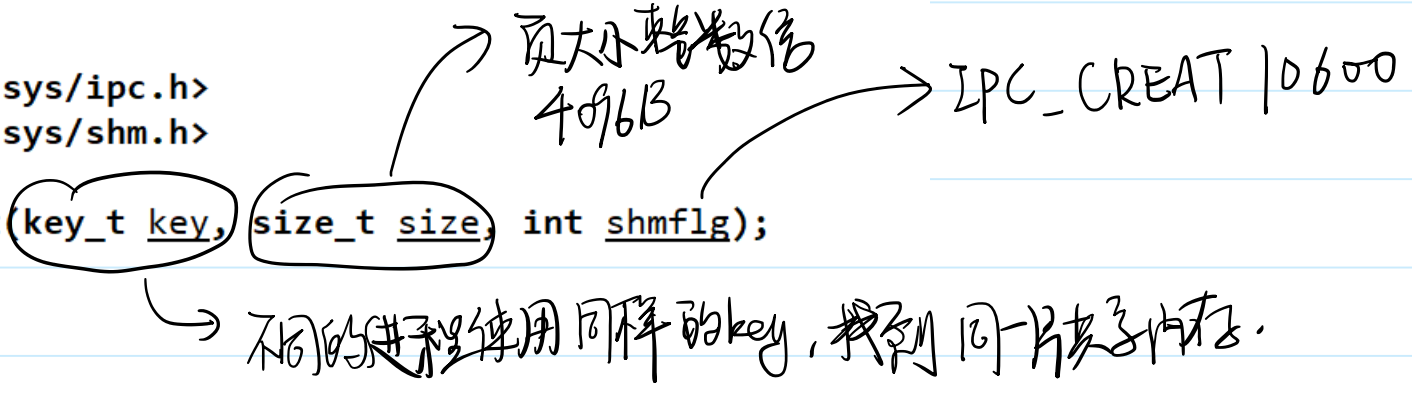
NAME

shmget - allocates a System V shared memory segment

SYNOPSIS

```
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg);
```



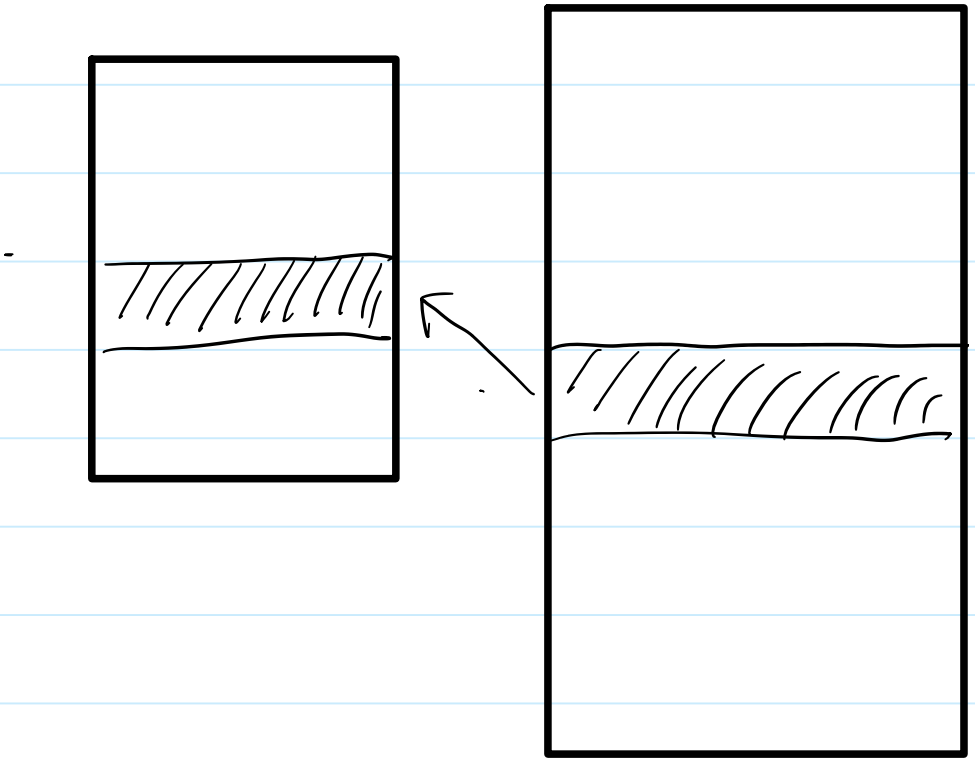
分配虚拟内存 attach.

2023年8月12日 11:30

shmget的返回值 NULL 0

```
void *shmat(int shmid, const void *shmaddr, int shmflg);
```

首地址



释放虚拟内存

2023年8月12日 11:33

```
int shmdt(const void *shmaddr);
```

detach.

2023年8月12日 11:36

```
liao:LinuxDay15$ ipcs

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 6          liao       600        524288     2          dest
0x00000000 10         liao       600        4194304    2          dest
0x00000000 11         liao       600        524288     2          dest
0x00000000 14         liao       600        524288     2          dest

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
```

使用共享内存

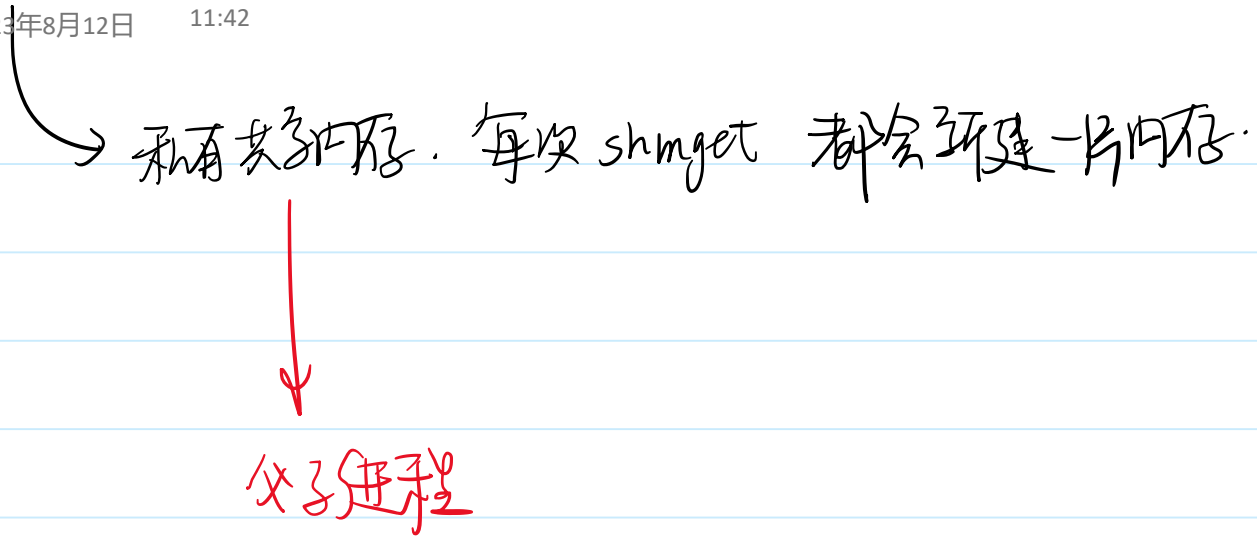
2023年8月12日 11:41

```c
int main(int argc, char *argv[])
{
    key_t key = 0x1234;
    int shmid = shmget(key,4096,IPC_CREAT|0600);
    char *p = (char *)shmat(shmid,NULL,0);
    puts(p);
    return 0;
}
```

```c
int main(int argc, char *argv[])
{
    key_t key = 0x1234;
    int shmid = shmget(key,4096,IPC_CREAT|0600);
    char *p = (char *)shmat(shmid,NULL,0);
    strcpy(p,"howareyou");
    return 0;
}
```

key如果填0

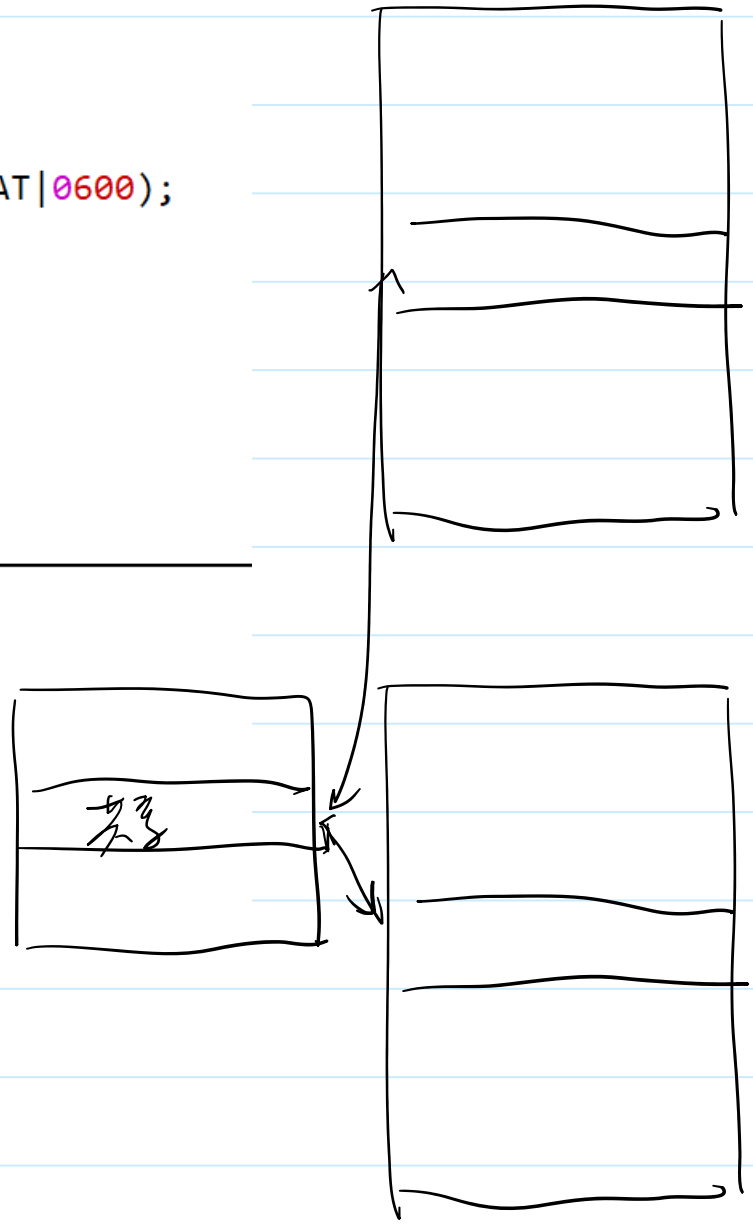
2023年8月12日 11:42



2023年8月12日 11:49

```c
int main(int argc, char *argv[])
{
    int shmid = shmget(IPC_PRIVATE,4096,IPC_CREAT|0600);
    int *p = (int *)shmat(shmid,NULL,0);
    p[0] = 0;
    if(fork()){
        for(int i = 0; i < 100; ++i){
            ++p[0];
        }
        wait(NULL);
        printf("p[0] = %d\n",p[0]);
    }
    else{
        for(int i = 0; i < 100; ++i){
            ++p[0];
        }

    }
    return 0;
}
```



```
liao:LinuxDay15$ ./4_shm_private
p[0] = 200
```

2023年8月12日 11:53

```c
#define NUM 1000000
int main(int argc, char *argv[])
{
    int shmid = shmget(IPC_PRIVATE,4096,IPC_CREAT|0600);
    int *p = (int *)shmat(shmid,NULL,0);
    p[0] = 0;
    if(fork()){
        for(int i = 0; i < NUM; ++i){
            ++p[0];
        }
        wait(NULL);
        printf("p[0] = %d\n",p[0]);
    }
    else{
        for(int i = 0; i < NUM; ++i){
            ++p[0];
        }

    }
    return 0;
}
```

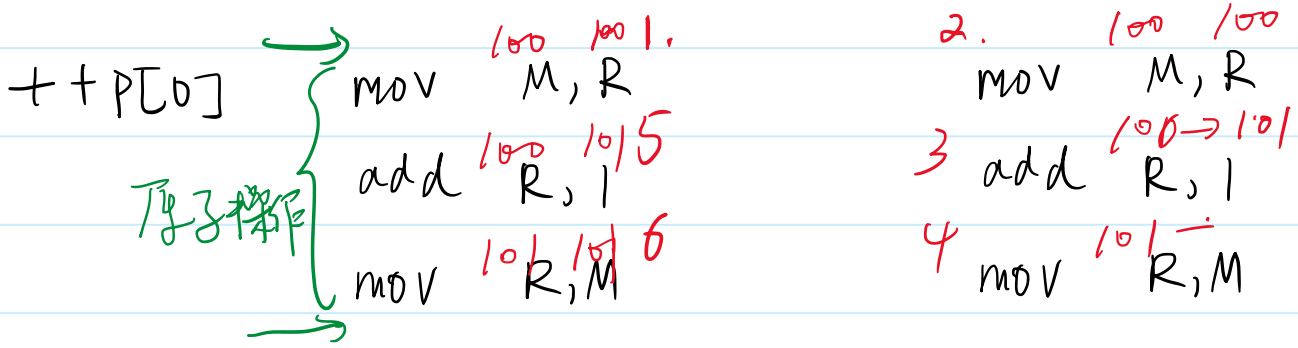
```
liao:LinuxDay15$ ./4_shm_private
p[0] = 174705
liao:LinuxDay15$ taskset -c 0 ./4_shm_private
p[0] = 200000

liao:LinuxDay15$ ./4_shm_private
p[0] = 12764303
liao:LinuxDay15$ taskset -c 0 ./4_shm_private
p[0] = 20000000
liao:LinuxDay15$ taskset -c 0 ./4_shm_private
p[0] = 16379511
liao:LinuxDay15$ taskset -c 0 ./4_shm_private
p[0] = 14644834
```

单核运行

竞争条件 race condition

2023年8月12日 11:54



两个并发的执行流访问共享资源。

```
p0:
wants_to_enter[0] ← true
while wants_to_enter[1] {
    if turn ≠ 0 {
        wants_to_enter[0] ← false
        while turn ≠ 0 {
            // busy wait
        }
        wants_to_enter[0] ← true
    }
}

// critical section p[0]++ 写在这个位置
...
turn ← 1
wants_to_enter[0] ← false
```

加锁

解锁