**实 验 报 告**

**课程名称** 操作系统

**实验项目** 内存管理

**实验仪器** 计算机

**系 别** 计算机科学与技术

**专 业**  计算机科学与技术

**班级/学号** 计科1606/2016012620

**学生姓名**  闫明

**实验日期**  19-1-1

**成 绩 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师**  牛欣源

1. **实验目的**

内存管理是操作系统的主要功能之一，本实验需要完成两个并发进程通过共享存储器机制对文件的修改。通过本次实验，能够使学生进一步熟练使用Linux内存管理相关函数和系统调用进行编程，进而掌握内存管理的基本功能和通过共享存储器方式实现进程间通信的基本原理。

1. **实验环境**

操作系统：Ubuntu 18.01.1 LTS

编译器：gcc编辑器

文本编辑器：vim

1. **实验内容**

1. 打开当前目录下的文件f1，使用系统调用mmap()创建共享存储区，大小为一个页面，将文件f1映射进内存，地址返回到src。

2．使用malloc()函数申请一个内存块，地址返回到dst。申请成功后，使用bcopy()函数将共享存储区的内容复制到dst。

3．使用fork()调用创建一个新进程，之后父进程阻塞自己。

4．子进程使用memccpy()将文件f1中W之后内容的地址复制到temp，将temp中前2个字符用@替换。之后将修改后的dst的内容使用memmove()复制回src，关闭文件f1，释放dst。

5．父进程使用memset()将共享存储区src中前2个字符用QQ替换。最后显示经2次修改后的f1的内容。

1. **设计与实现**

**算法思路：**

打开f1，调用mmap创建共享区，地址返回给src，申请一个内存块，地址返回dst，然后将src内容复制给dst。创建子进程，父进程阻塞，子进程将W之后地址复制到temp，修改前两个字符，然后把修改后的内容移送到src中，关闭f1，唤醒父进程，修改前两个字符。

**关键代码：**

pid=fork();

    if(pid==0){

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

        子进程使用memccpy()将文件f1中W之后内容的地址复制到temp，

        将temp中前2个字符用@替换。之后将修改后的dst的内容使用memmove()复制回src，

        关闭文件f1，释放dst。

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        char \*temp=0;

        if(!(temp=(char \*)memccpy(dst,src,dst\_Char,page\_size))){

            printf("/t/t/tWarnning!Can't find the char W in src.\n");

            free(dst);

            dst=0;

            \_exit(0);

        }

        printf("\t\t\tThe addr pointed by temp is 0x%x. Its contect is:\n%s\n",temp,strlen(temp)==0?"NULL":temp);

        temp[0]=tem[1]='@';

        printf("\t\t\tAfter replacing with 2 @, the dst is:\n%s\n",dst);

        memmove(src,dst,strlen(dst));

        printf("\t\t\tPID=%d,PPID=%d: After move dst to src, the src is:\n%s\n",getpid(),getppid(),src);

        close(fd);

        free(dst);

        dst = 0;

        \_exit(0);

    }else if (pid>0) {

        wait(NULL);

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

            ⑿使用memset()将共享存储区src中前2个字符用QQ替换，并显示替换后的内容(包含PID及PPID);

            ⒀显示经2次修改后的f1的最终内容;

        \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    memset(src,'Q',2);

    }

1. **运行结果**

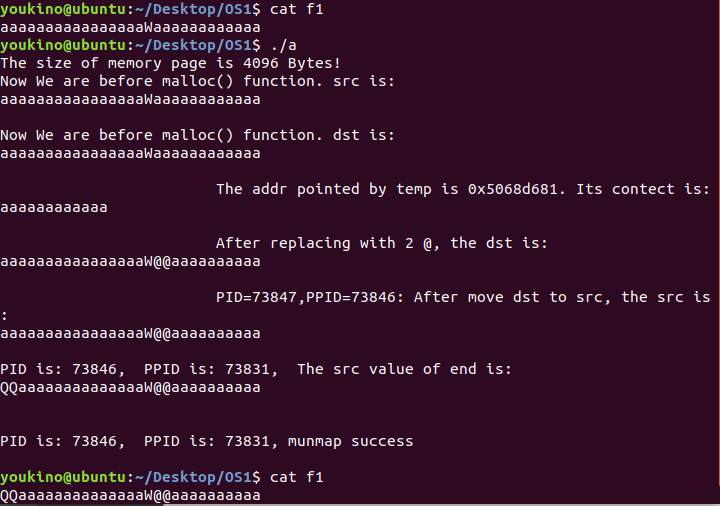


图1 运行结果

运行说明：文件f1原为aaaaaaaaWaaaaa，运行之后前两个字符变为QQ，W之后字符变为@@,其余字符不变

1. **总结**

**实验收获：**

1. 本次实验中了解了内存管理的基本方式，了解了Linux系统中如何开辟使用共享内存，同时了解到了改变共享内存也就是改变了文件内容。
2. 了解了进程之间的通信方式：共享内存方式，这种方式原来只在书上看到过，未曾实现，实现了才大概了解这种方式的原理和优点，共享内存的方式通信非常快，但是相对来说较为危险，因为读内存，开辟内存时，都需要时刻注意释放内存和释放指针，否则会出现内存泄漏或者野指针，这两者都是计算机的重大安全风险的导致者。
3. 同时也了解了内存之间的不同，比如共享区的内容被修改了，文件也就被修改了，但是子进程申请的内存空间中的内容修改了，文件就没有修改，这样的内存之间的不同。
4. **问题**
5. 删除wait（null），输出结果有和不同？

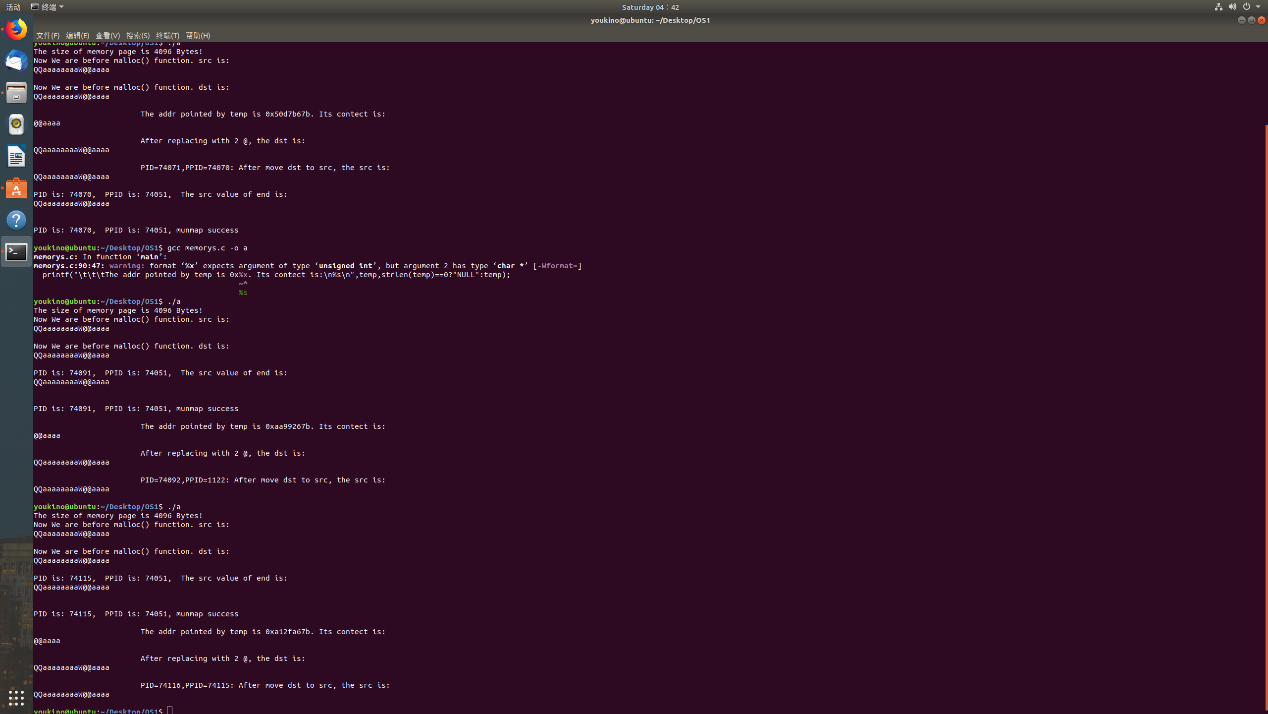


图 删除wait后变化

删除wait（null）后输出语句顺序变化了，原因在于父进程原先阻塞等待子进程实现，现在不阻塞，两者并行了，所以就不会是先输出子进程的语句，而是父子进程的语句交替出现，而且存在一定的随机性，因为进程是存在随机性的。

1. memchr用法

从标记所指的内存区域查找字符ch，可以不用开辟子进程来进行W之后@@的替换，直接用memchr找到W，然后把W内存地址交给temp，使得temp[1]=temp[2]=’@’即可。

1. 说明PID和PPID输出的结果？

该输出结果就是输出父子进程的进程号，可以看到，就相差1位，表现了父子进程的关系。