实验四 共享内存实验

一、实验目的

1. 初步了解 Linux 系统中共享内存的使用方法。
2. 学会使用共享内存实现多进程之间的通信。

二、实验内容

编写一个（C/JAVA）程序，使用 Linux 中的共享内存机制，完成“石头、剪子、布”的游戏，要求使用三个进程分别模拟 2 个选手和 1 个裁判。

三、背景介绍

进程间通信（Interprocess Communication，IPC）实现了进程之间同步和交换数据的功能。本实验要求完成的是一个或几个用户态的进程，依靠内核提供 的进程间通信的机制，完成几个用户进程之间的通信。通常，在 Linux 中提供了多种进程间通信的机制，包括管道、信号、消息队列、共享内存、信号量、套接字等。下面重点介绍共享内存机制。

共享内存是指将同一块内存区映射到共享它的不同进程的地址空间中，共享内存是在进程之间共享和传递数据的一种简单但非常有效的方式。进程间的通信只需要对共享的内存区域进行操作，数据不再需要通过内核就可以在不同的进程间复制。

共享内存也是最高效的一种进程间通信方式，因为进程可以直接读写内存，这避免了对数据的各种不必要的复制。另外，进程之间在使用共享内存时，数据将一直保存在共享内存中，直到解除映射、通信完毕才会写回文件，从而达到高效通信的目的。但主要问题在于，当两个或多个进程使用共享内存进行通信时，系统内核并未对共享内存的访问提供同步机制，这容易造成不同进程在同时读写同一共享内存时数据不一致问题，因此程序员需要依靠某种同步机制（如互斥锁、信号量等）来同步进程对共享内存的访问。

在 Linux 中，每个进程的虚拟内存被分为多个页面，并且每个进程都会维护一个从内存地址到虚拟页面的映射关系（即页表）。尽管每个进程都有自己的内存地址，但不同的地址可以同时将同一内存页面映射到自己的地址空间，从而达到共享内存的目的。

Linux 有两种共享内存机制——POSIX 共享内存和 System V 共享内存，它们都是通过 tmpfs（一种基于内存的文件系统，该文件系统的目录为/dev/shm）实现的。但 POSIX 共享内存是通过用户空间挂载的 tmpfs 文件系统实现的，而 System V 共享内存则是通过内核本身的 tmpfs 文件系统实现的。两者的区别在于：System V 共享内存是持久化的，只要机器不重启或不显式销毁，该共享内存就一直存在；而 POSIX 共享内存不是持久化的，如果进程被关闭，映射也将随之失效（事先映射到文件上的情况除外）。

需要注意的是，无论使用哪种共享内存机制，都必须注意数据存取的同步。通常，信号量被用于实现共享数据存取的同步，此外也可以通过 shmctl()函数（如 SHM_LOCK、SHM_UNLOCK 等）设置共享内存的某些标志位来实现共享数据存取的同步。

1．POSIX 共享内存操作函数

POSIX 共享内存使用内存映射机制 mmap 来实现。mmap()系统调用使得进程之间通过映射同一个普通文件实现了内存共享。普通文件在被映射到进程地址空间后，进程就可以像访问普通内存一样对文件进行操作，而不必调用 read()、write()等函数。

具体使用时，除包括头文件 sys/mman.h 外，主要涉及两个步骤。

（1）创建一个新的共享内存区或打开一个已存在的共享内存区。

int shm_open(const char *name, int oflag, mode_t mode);

（2）把该共享内存区映射到调用进程的地址空间。

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

至此，就可以像操作普通内存一样操作共享内存了，可以使用 memcpy()、memset()等函数对共享内存进行操作。共享内存在被使用过程中，其大小可以通过调用 ftruncate()进行修改。

int ftruncate(int df, off_t length)

当打开一个已存在的共享内存区时，可以通过调用 stat()函数来获取有关对象的信息。

int stat(const char *path, struct stat *buf);

当需要结束对共享内存的使用时，可以执行以下步骤。

（1）解除当前进程对这块共享内存的映射。

int munmap(void *addr, size_t length);

（2）从内核中清除共享内存。

int shm_unlink(const char *name);

2．System V 共享内存操作函数

System V 共享内存通过系统调用 shmget()来创建或打开一个 IPC 共享内存区，此外还将在特殊文件系统 shm 中创建或打开一个同名文件，新建的文件不属于任何进程，但任何进程都可以访问该共享内存区。一般情况下，特殊文件系统 shm 不能使用 read()、write()函数进行访问，但可以直接采用访问内存的方式对其进行访问。

System V 共享内存主要涉及以下几个 API 函数，使用时须包含头文件 sys/ipc.h 和 sys/shm.h。

int shmget(key_t key, size_t size, int shmflg); //创建共享内存

//把共享内存映射到当前进程的地址空间

void *shmat(int shm_id, const void *shm_addr, int shmflg);

int shmdt(const void *shmaddr); //从当前进程分离共享内存

int shmctl(int shm_id, int command, struct shmid_ds *buf); //控制共享内存

四、实验步骤

本实验可以创建三个进程，其中，一个进程为裁判进程，另外两个进程为选手进程。可以将“石头、剪子、布”这三招定义为三个整型值。胜负关系：石头〉剪子〉布〉石头。选手进程按照某种策略（例如，随机产生）出招，交给裁判进程判断大小。

裁判进程将对手的出招和胜负结果通知选手。比赛可以采取多盘（＞１００盘）定胜负，由裁判宣布最后结果。每次出招由裁判限定时间，超时判负。

每盘结果可以存放在文件或其他数据结构中。比赛结束，可以打印每盘的胜负情况和总的结果。

1．设计表示“石头、剪子、布”的数据结构，以及它们之间的大小规则。

2．设计比赛结果的存放方式。

3．选择 IPC 的方法。

4．根据你所选择的 IPC 方法，创建对应的 IPC 资源。
5．完成选手进程。
6． 完成裁判进程。
－－以下要求选作：
7．决出班级的前三甲，与另外班级的前三甲比赛，决出年级冠军。
8．如果有兴趣，再把这个实验改造成网络版。即在设计时就要考虑ＩＰＣ层的封装。

五、实验结果

1. 实验数据结构：

2. 实验大小规则&存放方式：

3. 实验所选择的 IPC 方法和理由：

4. 如果选择消息队列机制，描述消息缓冲区结构：

5. 如何创建 IPC 资源？

6. 程序主要流程或关键算法：

六、实验总结