linux共享内存-应用编程-专题讲座

written by 王保明

# 1共享内存基本概念

|  |
| --- |
| **共享内存**   * 共享内存区是最快的IPC形式。一旦这样的内存映射到共享它的进程的地址空间，这些进程间数据传递不再涉及到内核，换句话说是进程不再通过执行进入内核的系统调用来传递彼此的数据。 |
| **用管道或者消息队列传递数据**   * 内核为每个IPC对象维护一个数据结构 |
| **用共享内存传递数据** |

# 2System V共享内存API

## 共享内存数据结构及基本API

|  |
| --- |
| struct shmid\_ds {  struct ipc\_perm shm\_perm; /\* Ownership and permissions \*/  size\_t shm\_segsz; /\* Size of segment (bytes) \*/  time\_t shm\_atime; /\* Last attach time \*/  time\_t shm\_dtime; /\* Last detach time \*/  time\_t shm\_ctime; /\* Last change time \*/  pid\_t shm\_cpid; /\* PID of creator \*/  pid\_t shm\_lpid; /\* PID of last shmat(2)/shmdt(2) \*/  shmatt\_t shm\_nattch; /\* No. of current attaches \*/  ...  }; |
| **共享内存函数**   * #include <sys/ipc.h> * #include <sys/shm.h> * int shmget(key\_t key, size\_t size, int shmflg); * void \*shmat(int shmid, const void \*shmaddr, int shmflg); * int shmdt(const void \*shmaddr); * int shmctl(int shmid, int cmd, struct shmid\_ds \*buf); |
| **shmget函数**   * 功能：用来创建共享内存 * 原型   + int shmget(key\_t key, size\_t size, int shmflg); * 参数   + key:这个共享内存段名字   + size:共享内存大小   + shmflg:由九个权限标志构成，它们的用法和创建文件时使用的mode模式标志是一样的 * 返回值：成功返回一个非负整数，即该共享内存段的标识码；失败返回-1 |
| **shmat函数**   * 功能：将共享内存段连接到进程地址空间 * 原型   + void \*shmat(int shmid, const void \*shmaddr, int shmflg); * 参数   + shmid: 共享内存标识   + shmaddr:指定连接的地址   + shmflg:它的两个可能取值是SHM\_RND和SHM\_RDONLY * 返回值：成功返回一个指针，指向共享内存第一个节；失败返回-1 |
| **shmat函数（续）**   * shmaddr为NULL，核心自动选择一个地址 * shmaddr不为NULL且shmflg无SHM\_RND标记，则以shmaddr为连接地址。 * shmaddr不为NULL且shmflg设置了SHM\_RND标记，则连接的地址会自动向下调整为SHMLBA的整数倍。公式：shmaddr - (shmaddr % SHMLBA) * shmflg=SHM\_RDONLY，表示连接操作用来只读共享内存 |
| **shmdt函数**   * 功能：将共享内存段与当前进程脱离 * 原型   + int shmdt(const void \*shmaddr); * 参数   + shmaddr: 由shmat所返回的指针 * 返回值：成功返回0；失败返回-1 * 注意：将共享内存段与当前进程脱离不等于删除共享内存段 |
| **shmctl函数**   * 功能：用于控制共享内存 * 原型   + int shmctl(int shmid, int cmd, struct shmid\_ds \*buf); * 参数   + shmid:由shmget返回的共享内存标识码   + cmd:将要采取的动作（有三个可取值）   + buf:指向一个保存着共享内存的模式状态和访问权限的数据结构 * 返回值：成功返回0；失败返回-1 |
| **shmctl函数（续）** |

# 3共享内存API示例

|  |
| --- |
| 共享内存查看工具及共享内存命令 |
| //  //key shmid owner perms bytes nattch status  //0x00002234 131073 wbm01 666 68 0  //  int main(int argc, char \*argv[])  {  int ret = 0;  int shmid;  //相当于打开文件，文件不存  shmid = shmget(0x2234, sizeof(Teacher), IPC\_CREAT | 0666);  if (shmid == -1)  {  perror("shmget err");  return errno;  }  printf("shmid:%d \n", shmid);  Teacher \*p = NULL;  p = shmat(shmid, NULL, 0);  if (p == (void \*)-1 )  {  perror("shmget err");  return errno;  }    strcpy(p->name, "aaaa");  p->age = 33;    shmdt(p);    printf("键入1 删除共享内存，其他不删除\n");    int num;  scanf("%d", &num);  if (num == 1)  {  ret = shmctl(shmid, IPC\_RMID, NULL);  if (ret < 0)  {  perror("rm errr\n");  }  }  return 0;  } |
| int main(int argc, char \*argv[])  {  int ret = 0;  int shmid;  //相当于打开文件，文件不存  //shmid = shmget(0x2234, sizeof(Teacher), IPC\_CREAT |IPC\_EXCL | 0666);  shmid = shmget(0x2234, 0, 0);  if (shmid == -1)  {  perror("shmget err");  return errno;  }  printf("shmid:%d \n", shmid);  Teacher \*p = NULL;  p = shmat(shmid, NULL, 0);  if (p == (void \*)-1 )  {  perror("shmget err");  return errno;  }    printf("name:%s\n", p->name);  printf("age:%d \n", p->age);  shmdt(p);    //  //key shmid owner perms bytes nattch status  //0x00002234 131073 wbm01 666 68 0  //    printf("键入1 程序暂停，其他退出\n");    int num;  scanf("%d", &num);  if (num == 1)  {  pause();  }    return 0;  } |

删除共享内存：

ipcrm shm shmid

解除引用 unmap (shmdt) -> 删除 shmctl

进程退出：

解除引用 unmap

使用了引用计数，不像只有打开关闭状态的api如close，如果引用计数不为1的共享内存被删除，将会使引用计数减一并key变为0，当所有使用共享内存的进程都退出时，共享内存才能删掉。

如果共享内存被别的进程占用，删除共享内存，则不会立马删除，出现一个现象，key值变为0即为私有，若其他进程再想从原来key获取对应的共享内存shmid会报错。只有，连接共享内存的所有进程都死掉，linux才会把共享内存删除。