系统调用函数说明、参数值及定义

1. fork( )

创建一个新进程。

int fork( )

其中返回 int 取值意义如下：

0：创建子进程，从子进程返回的 id 值

大于 0：从父进程返回的子进程 id 值

－1：创建失败

2. lockf(files,function,size)：

用作锁定文件的某些段或者整个文件，本函数适用的头文件为：

```
#include <unistd.h>
```

参数定义：

```
int lockf(files,function,size)
int files,function;
long size;
```

其中：files 是文件描述符；function 是锁定和解锁；1 表示锁定，0 表示解锁。Size 是锁定或解锁的字节数，若用 0，表示从文件的当前位置到文件尾。

3. msgget(key,flag)：

获得一个消息的描述符，该描述符指定一个消息队列以便用于其他系统调用。

该函数使用头文件如下：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

参数定义

```
int msgget(key,flag)
key_tkey;
int flag;
```

语法格式：msgqid＝msgget(key,flag)

其中：

msgqid 是该系统调用返回的描述符，失败则返回－1；

flag 本身由操作允许权和控制命令值相“或”得到。

如：IPC_CREAT |0400　　是否该队列应被创建；

IPC_EXCL |0400　　是否该队列的创建映是互斥的；等。

4. msgsnd(id,msgp,size,flag)：

发送一消息。

该函数使用头文件如下：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

参数定义：

```
int msgsnd(id,msgp,size,flag)
int id,size,flag;
struct msgbuf * msgp;
```

其中：id 是返回消息队列的描述符；msgp 是指向用户存储区的一个构造体指针，size 指示由 msgp 指向的数据结构中字符数组的长度；即消息的长度。这个数组的最大值由 MSGMAX 系统可调用参数来确定。flag 规定当核心用尽内部缓冲空间时应执行的动作；若在标志 flag 中未设置 IPC_NOWAIT 位，则当该消息队列中的字节数超过一最大值时，或系统范围的消息数超过某一最大值时，调用 msgsnd 进程睡眠。若是设置 IPC_NOWAIT，则在次情况下，msgsnd 立即返回。

5. msgrcv(id,msgp,size,type,flag)：

接受一消息。

该函数调用使用头文件如下：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

参数定义：

```
int msgrcv(id,msgp,size,flag)
int id,size,type,flag;
struct msgbuf * msgq;
struct msgbuf{long mtype;chat mtext[];};
```

语法格式：

```
count=msgrcv(id,msgp,size,type,flag)
```

其中：id 是消息描述符，msgp 是用来存放欲接收消息的拥护数据结构的地址；size 是 msgp 中数据数组的大小；type 是用户要读的消息类型：

type 为 0：接收该队列的第一个消息；

type 为正：接收类型 type 的第一个消息；

type 为负：接收小于或等于 type 绝对值的最低类型的第一个消息。

Flag 规定倘若该队列无消息，核心应当做什么事，如果此时设置了 IPC_NOWAIT 标志，则立即返回，若在 flag 中设置了 MSG_NOERROR，且所接收的消息大小大于 size，核心截断所接受的消息。

count 是返回消息正文的字节数。

6. msgctl(id,cmd,buf)：

查询一个消息描述符的状态，设置它的状态及删除一个消息描述符。

调用该函数使用头文件如下：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

参数定义：

```
int msgctl(id,cmd,buf)
int id,cmd;
struct msqid_ds * buf;
```

其中：函数调用成功时返回 0，调用不成功时返回－1。

id 用来识别该消息的描述符；cmd 规定命令的类型。

IPC_STAT 将与 id 相关联的消息队列首标读入 buf。

IPC_SET 为这个消息序列设置有效的用户和小组标识及操作允许权和字节的数量。

IPC_RMID 删除 id 的消息队列。

buf 是含有控制参数或查询结果的用户数据结构的地址。

附：msgid_ds 结构定义如下：

```
struct msgid_ds
{struct ipc_perm msg_perm;        /* 许可权结构 */
short pad1[7];                    /* 由系统使用 */
ushort onsg_qnum;                 /* 队列上消息数 */
ushort msg_qbytes;                /* 队列上最大字节数 */
ushort msg_lspid;                 /* 最后发送消息的 PID */
ushort msg_lrpid;                 /* 最后接收消息的 PID */
time_t msg_stime;                 /* 最后发送消息的时间 */
time_t msg_rtime;                 /* 最后接收消息的时间 */
```

```
time_t msg_ctime;                    /* 最后更改时间 */
};
struct ipc_perm
{ushort uid;                         /* 当前用户 id */
ushort gid;                          /* 当前进程组 id */
ushort cuid;                         /* 创建用户 id */
ushort cgid;                         /* 创建进程组 id */
ushort mode;                         /* 存取许可权 */
{short pad1;long pad2}               /* 由系统使用 */
};
```

7. shmget(key,size,flag):

获得一个共享存储区。

该函数使用头文件如下:

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

语法格式:

```
shmid=shmget(key,size,flag)
```

参数定义:

```
int shmget(key,size,flag)
key_t key;
int size,flag;
```

其中:size 是存储区的字节数,key 和 flag 与系统调用 msgget 中的参数含义相同。

附:

操作允许权	八进制数
用户可读	00400
用户可写	00200
小组可读	00040
小组可写	00020
其他可读	00004
其他可写	00002

控制命令	值
IPC_CREAT	0001000
IPC_EXCL	0002000

如:shmid=shmget(key,size,(IPC_CREAT|0400));

创建一个关键字为 key,长度为 size 的共享存储区。

8. shmat(id,addr,flag):

从逻辑上将一个共享存储区附接到进程的虚拟地址空间上。

该函数调用使用头文件如下：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

参数定义：

```
char * shmat(id,addr,flag)
int id,flag;
char * addr;
```

语法格式：virtaddr=shmat(id,addr,flag)

其中：id 是共享存储区的标识符，addr 是用户要使用共享存储区附接的虚地址，若 addr 是 0，系统选择一个适当的地址来附接该共享区。flag 规定对此区的读写权限，以及系统是否应对用户规定的地址做舍入操作。如果 flag 中设置了 shm_rnd 即表示操作系统在必要时舍去这个地址。如果设置了 shm_rdonly，即表示只允许读操作。viraddr 是附接的虚地址。

9. shmdt(addr):

把一个共享存储区从指定进程的虚地址空间断开。

调用该函数使用头文件：

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/mhm.h>
```

参数定义：

```
int shmdt(addr)
char * addr
```

其中，当调用成功时，返回 0 值，调用不成功，返回－1，addr 是系统调用 shmat 所返回的地址。

10. shmctl(id,cmd,buf):

对与共享存储区关联的各种参数进行操作，从而对共享存储区进行控制。

调用该函数使用头文件：

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

参数定义：

```
int shmctl(id,cmd,buf)
```

```
int id,cmd;
struct shmid_ds * buf;
```

其中：调用成功时返回0，否则返回－1。id 为被共享存储区的标识符。cmd 规定操作的类型。规定如下：

IPC_STAT： 返回包含在指定的 shmid 相关数据结构中的状态信息，并且把它放置在用户存储区中的 * but 指针所指的数据结构中。执行此命令的进程必须有读取允许权。

IPC_SET： 对于指定的 shmid，为它设置有效用户和小组标识和操作存取权。

IPC_RMID： 删除指定的 shmid 以及与它相关的共享存储区的数据结构。

SHM_LOCK： 在内存中锁定指定的共享存储区，必须是超级用户才可以进行此项操作。

Buf 是一个用户级数据结构地址。

附：

```
shmid_ds
{struct ipc_perm shm_perm;          /* 许可权结构； */
int shm_segsz;                      /* 段大小； */
int padl;                           /* 由系统使用； */
ushort shm_lpid;                    /* 最后操作的进程 id； */
ushort shm_cpid;                    /* 创建者的进程 id； */
ushort shm_nattch;                  /* 当前附界数； */
short pad2;                         /* 由系统使用； */
time_t shm_atime;                   /* 最后附接时间； */
time_t shm_dtime;                   /* 最后断接时间； */
time_t shm_ctime;                   /* 最后修改时间； */
}
```

11. signal(sig,function)：

允许调用进程控制软中断信号的处理。

头文件为：

```
#include<signal.h>
```

参数定义：

```
signal(sig,function)
int sig;
void ( * func)( );
```

其中：sig 的值是：

SIGHVP	挂起
SIGINT	键盘按 delete 键或 break 键
SIGQUIT	键盘按 quit 键

SIGILL	非法指令
SIGIOT	IOT 指令
SIGEMT	EMT 指令
SIGFPE	浮点运算溢出
SIGKILL	要求终止进程
SIGBUS	总线错
SIGSEGV	段违例
SIGSYS	系统调用参数错
SIGPIPE	向无读者管道上写
SIGALRM	闹钟
SIGTERM	软件终结
SIGUSR1	用户定义信号
SIGUSR2	第二个用户定义信号
SIGCLD	子进程死
SIGPWR	电源故障

function 的解释如下：

SIG_DFL：缺省操作。对除 SIGPWR 和 SIGCLD 外所有信号的缺省操作是进程终结对信号 SIGQUIT，SIGILL，SIGTRAP，SIGIOT，SIGEMT，SIGFPE，SIGBUS，SIGSEGV 和 SIGSYS 它产生一内存映像文件。

SIG_IGN：忽视该信号的出现。

Function：在该进程中的一个函数地址，在核心返回用户态时，它以软中断信号的序号作为参数调用该函数，对除了信号 SIGILL，SIGTRAP 和 SIGPWR 以外的信号，核心自动地重新设置软中断信号处理程序的值为 SIG_DFL，一个进程不能捕获 SIGKILL 信号。

Linux 操作系统是一个向用户开放源码的免费的类 UNIX 操作系统。它为在校学生学习操作系统课程提供了一个看得见摸得着的范例。对于学生正确理解，掌握操作系统的基本知识具有重要意义。鉴于此，本操作系统课程涉及的实验均在 Linux 环境下进行。

这就要求大家：

(1) 熟悉 Linux 的操作和开发环境；

(2) 具有 C 语言知识(Linux 操作系统大约 90%的源码是用 C 语言编写)。

如果想安装 Linux 系统，可以利用 Linux 安装光盘进行安装，或向下列网址获取源码进行安装：

ftp://ftp.pk.edu.cn

ftp://ftp.lib.pku.edu.cn

ftp://ftp.ncic.cn.cn