Казахстанский филиал МГУ им. М.В. Ломоносова

Операционные системы

лекция 17

- Реализация межпроцессного взаимодействия IPC
  разделяемая память,
  массив семафоров,
  очередь сообщений

8.11.2010

(C) Корухова Ю.С., 2010

Принципы работы с разделяемыми ресурсами

– Доступ к ресурсу — по ключу.

– Если с ресурсом работает группа родственных процессов, используется IPC_PRIVATE

– Для работы используется id ресура, который получен как результат функции

....get (ключ,....., флаги, ...)

Последние 9 битов в параметре флаги определяют доступ к ресурсу (например, 0600

– Операции над ресурсом в целом (в частности, удаление)

....ctl (id, ..., команда, ...)

команда IPC_RMID - удаление ресурса

Разделяемая память

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

int shmget (key_t key, int size, int shmemflg)

char *shmat(int shmid, char *shmaddr, int shmflg)

int shmdt(char *shmaddr)

int shmctl(int shmid, int cmd, struct shmid_ds *buf)
```

Пример: работа с разделяемой памятью

```
int main(){
  key_t key;
  int shmid;
  char *a;
key =ftok("/tmp/f1",'1');
shmid = shmget(key, 256,
        0666|IPC_CREAT);
a = shmat(shmid,NULL,0);

*a='*';
strcpy(a+1,"info");

shmdt(a);
shmctl(shmid,IPC_RMID,0);
  return 0;
}
```

Пример: работа с разделяемой памятью

```
int main(){
  key_t key;
  int shmid;
  char *a;
key =ftok("/tmp/f1",'1');
shmid = shmget(key, 256,
        0666|IPC_CREAT);
a = shmat(shmid,NULL,0);

*a='*';
strcpy(a+1,"info");

shmdt(a);
shmctl(shmid,IPC_RMID,0);
  return 0;
}
```

```
int main(){
  key_t key;
  int shmid;
  char *a,b,buf[256];
key =ftok("/tmp/f1",'1');
shmid = shmget(key, 256,
        0666|IPC_CREAT);
a = shmat(shmid,NULL,0);

b=*a;
strcpy(buf,a+1);

shmdt(a);
shmctl(shmid,IPC_RMID,0);
  return 0;
}
```

Массив семафоров

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

int semget (key_t key, int N, int flag)

int semop(int semid,struct sembuf * op,size_t nops)

struct sembuf {
    short sem_num;    /* номер семафора в массиве */
   short sem_op;      /* операция */
   short sem_flg;     /* флаги операции */
}
```

Последовательность операций, заданная в op, выполняется атомарно.

```
int semctl(int semid, int semnum, int cmd,
                                    union semun arg);
```

Использование массива семафоров

С каждым критическим ресурсом связывается семафор.
Вариант 1:Значение семафора >0 - ресурс доступен, значение семафора ==0 – ресурс недоступен.
Вариант 2: Значение семафора ==0 - ресурс доступен,иначе - ресурс недоступен, нужно ждать обнуления семафора.

Реализация взаимного исключения (вариант2):

```
struct sembuf lock[2] = {{0,0,0},{0,1,0}};
struct sembuf unlock[1] {{0,-1,0}};

key = ftok("/home/student/server.c",'1');
id = semget(key,1,0666|IPC_CREAT);

semop(id,lock,2);
/* критическая секция */
semop(id, unlock,1);

semctl(id, 0, IPC_RMID, 0);
```

Очередь сообщений

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

int msgget (key_t key, int flags);

int msgctl(int msqid, int cmd, struct msqid_ds *buf);

int msgsnd(int msqid, struct msgbuf *msgp,
           size_t msgsz, int flags);

struct msgbuf{
long mtype;
char mtext[100];
};

ssize_t msgrcv(int msqid, struct msgbuf *msgp,
           size_t msgsz, long msgtyp, int msgflg);
```

Пример использования очереди сообщений : взаимодействие клиент - сервер

```
#define N 256
int main(){ /*сервер*/
struct {long mtype; char mtext [N];} mto;
struct {long mtype; long mtext;} mfrom;

key_t key;
int mesid;
key = ftok("server.c",'1');
mesid = msgget (key, 0666 | IPC_CREAT);
while(1){
  if (msgrcv(mesid, &mfrom, sizeof(long),1,0)<=0)
    continue;
  mto.mtype = mfrom.mtext;
  strcpy( mto.mtext, "Message for client");
  msgsnd (mesid, &mto, N, 0);
}
/*msgctl (mesid, IPC_RMID, 0); return 0;*/
}
```

Пример использования очереди сообщений : взаимодействие клиент - сервер

```
#define N 256
int main(){ /*клиент*/
struct {long mtype; long mtext;} mto;
struct {long mtype; char mtext [N];} mfrom;
key_t key;
int mesid;
long pid;

key = ftok("server.c",'1');
mesid = msgget (key, 0666 | IPC_CREAT);
pid=getpid();

mto.mtype = 1;
mto.mtext = pid;
msgsnd(mesid, &mto, sizeof(long),0);
while( msgrcv(mesid, &mfrom, N, pid,0)<=0) ;
 printf("%s \n",mfrom.mtext);
return 0;
}
```