Казахстанский филиал МГУ им. М.В. Ломоносова

Операционные системы
лекции 15, 16

- Классические задачи синхронизации процессов.

- Реализация межпроцессного взаимодействия в UNIX

6.11.2010

(С) Корухова Ю.С., 2010

Классические задачи синронизации процессов

Обедающие философы

Задача читателей и писателей

Задача о спящем парикмахере

Обедающие философы

Постановка задачи

Рассматриваем задачу для 5 философов

Вариант решения 1.

```
void philosopher ( int i ) {
  while (1) {
       think ();
       take_fork(i);
       take_fork((i+1)%N);
       eat();
       put_fork(i);
       put_fork((i+1)%N) ;
  }
}
```

Вариант решения 2.

```
void philosopher ( int i ) {
  while (1) {
      think ();
      while(1){
        while (take_fork(i)==0);
        if (take_fork((i+1)%N)) break;
          put_fork(i);
      }
      eat();
      put_fork(i);
      put_fork((i+1)%N) ;
  }
  return;
}
```

Вариант решения 3.

```
void philosopher ( int i ) {
  while (1) {
       think ();
       down(s,1);
       take_fork(i);
       take_fork((i+1)%N);
       eat();
       put_fork(i);
       put_fork((i+1)%N) ;
       up(s,1);
 }
}
```

Вариант решения 4.

```
#define N 5
#define LEFT (i+N-1)%N
#define RIGHT (i+1)%N
#define THINKING 0
#define EATING 1
#define HUNGRY 2
int state[N];
semaphore mutex = 1;
semaphore s[N];

void philosopher ( int i ) {
  while (1) {
       think ();
       take_forks(i);
       eat();
       put_forks((i+1)%N) ;
 }
}
```

```
void test(int i){
   if (state[i]==HUNGRY && state[LEFT]!=EATING &&
                           state[RIGHT]!=EATING){
      state[i]=EATING;
       up(s[i],1);
   }
}
```

```
void take_forks(int i){
   down(mutex,1);
   state[i]=HUNGRY;
   test(i);
   up(mutex,1);
   down(s[i]);
}
```

```
void put_forks(int i){
   down(mutex,1)
   state[i]=THINKING;
   test(LEFT);
   test(RIGHT);
   up(mutex,1)
}
```

Задача читателей и писателей

Постановка задачи, ограничения доступа к базе данных

ВАРИАНТ РЕШЕНИЯ 1

```
int rc=0;
semaphore db = 1;
void reader(void){
  while(1){
   rc=rc+1;
   if (rc==1) down(db,1);
   read_dbase();
   rc=rc-1;
   if (rc==0) up(db,1);
  /*using read data */
}
void writer(void){
  while(1){ down(db,1); write_dbase(); up(db,1);}
}
/* ЧТО НЕВЕРНО В ЭТОМ РЕШЕНИИ? */
```

Задача читателей и писателей

```
ВАРИАНТ РЕШЕНИЯ 2
semaphore mutex = 1;
int rc=0;
semaphore db = 1;
void reader(void){
  while(1){
   down(mutex,1);
   rc=rc+1;
   if (rc==1) down(db,1);
   up(mutex,1);
   read_dbase();
   down(mutex,1);
   rc=rc-1;
   if (rc==0) up(db,1);
   up(mutex,1);
  /*using read data */
}
```

Задача о спящем парикмахере

```
#define CHAIRS 5
semaphore mutex = 1;
semaphore customers = 0;
semaphore barbers = 0;
int waiting =0;
void barber(void){
  while(1){
   down(customers,1)
   down(mutex,1);
   --waiting;
   up(barbers,1);
   up(mutex,1);
   haircut();
  }
}
```

```
void customer(void){
 down(mutex,1);
 if (waiting<CHAIRS){
   ++waiting;
   up(customers,1);
   up(mutex,1);
   down(barbers,1);
   get_haircut();
 }
 else
 up(mutex,1);
}
```

Средства взаимодействия процессов UNIX System V IPC

Разделяемая память
Массив семафоров
Очередь сообщений

Для именования разделяемых объектов исп. КЛЮЧ

```
#include <sys/types.h>
#include <sys/ipc.h>
key_t ftok(const char *filename, char proj);
```

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

```
int main(int argc, char **argv){
  key_t key;
  int shmid;
  char *a;

  key = ftok("/tmp/f1",'1');
  shmid = shmget(key, 100, 0666|IPC_CREAT);
  a = shmat(shmid, NULL, 0);

  /* работа с ресурсом */

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