Операционные системы

Взаимоисключения

Олег Французов

2017

Гонки

- Гонки – ситуация, когда конечный результат выполнения зависит от того, в какой конкретно последовательности произойдут операции в независимых процессах

Гонки

```
x += 1
```

```
x += 1
```

Гонки

```
mov ax, x
inc ax
mov x, ax
```

```
mov ax, x
inc ax
mov x, ax
```

Гонки

```
mov ax, x
inc ax
mov x, ax
```

```
mov ax, x
inc ax
mov x, ax
```

Гонки

- Возможны даже в ситуации, когда один участник пишет, а второй читает (!)

Критическая секция

- Часть программы, в которой производятся логически связанные манипуляции с разделяемыми данными
  - с конкретными разделяемыми данными
- Взаимное исключение (mutual exclusion) – более 1 процесса не может находиться в критической секции по одним и тем же данным

Требования к взаимному исключению

- 2+ процессов не могут находиться в критических секциях по одним и тем же данным
- Нет предположений о скорости выполнения процессов, количестве процессоров в системе
- Процесс вне критических секций не может быть причиной блокировки

Требования к взаимному исключению

- Недопустимо вечное ожидание
- Недопустимо активное ожидание – расход процессорного времени в ожидании входа в критическую секцию

Блокировочная переменная

```
while(s == 0) {} /* пустой цикл, пока нельзя входить
                    в критическую секцию */
s = 0;           /* запретили доступ другим процессам */

section(); /* ... работа с разделяемыми данными ... */

s = 1;           /* разрешили доступ */
```

- while (s == 0) {}
  s = 0
  не атомарно

Запрет внешних прерываний

- Только для кратковременных секций
- Только для данных в ОЗУ
- Только для 1 процессора
- Ведет к зависанию
- Разрешать запрет прерываний процессам опасно

Чередование

```
for(;;) {
    while(turn != 0) {}
    section();
    turn = 1;
    noncritical_job();
}
```

```
for(;;) {
    while(turn != 1) {}
    section();
    turn = 0;
    noncritical_job();
}
```

- Один процесс блокирует другой даже тогда, когда в первом еще не выполняется критическая секция

Алгоритм Петерсона

```
void enter_section() {
    interested[0] = TRUE;
    who_waits = 0;
    while(who_waits==0 &&
          interested[1]) {}
}
void leave_section() {
    interested[0] = FALSE;
}
```

```
void enter_section() {
    interested[1] = TRUE;
    who_waits = 1;
    while(who_waits==1 &&
          interested[0]) {}
}
void leave_section() {
    interested[1] = FALSE;
}
```

- Активное ожидание

Поддержка взаимо-исключения на уровне ОС

- Заблокировать процесс
- Разбудить процесс

- Поручить ОС
- ОС может запрещать прерывания

Мьютексы

```
/* s         - мьютекс
   lock(s) - возвращает true, если открыт
   unlock(s)
*/

while (!lock(s)) {}
section();
unlock(s);
```

Мьютексы

```
/* s          - мьютекс
   lock(s) - блокируется, если закрыт
   unlock(s)
*/

lock(s);
section();
unlock(s);
```

Семафоры Дейкстры

```
/* s         - семафор (>=0)
   down(s) - блокируется, если s == 0
   up(s)
                счетчик доступных ресурсов
*/

lock(s);          down(s);
section();        section();
unlock(s);        up(s);
```

Q & A