Паттерны безопасного lock-based кода

Анна Субботина

19 Октября 2015

Условия Бернстайна

- Если для двух данных активностей P и Q:
  - пересечение W(P) и W(Q) пусто,
  - пересечение W(P) с R(Q) пусто,
  - пересечение R(P) и W(Q) пусто,
- тогда выполнение P и Q детерминировано.

Условия Бернстайна

- race condition
- mutual exclusion
- critical section

Lock-переменная

```
shared int lock = 0;

while (some condition) {

        while(lock); lock = 1;          // не атомарны

        // critical section

        lock = 0;

        // remainder section
}
```

Переменная очередности

```
shared int turn = 0;

while (some condition) {
        while(turn != i);
        // critical section
        turn = 1-i;
        // remainder section
}
// не выполнено условие прогресса
```

Флаги готовности

```
shared int ready[2] = {0, 0};

while (some condition) {
    ready[i] = 1;                    // не атомарны => deadlock
    while(ready[1-i]);

    // critical section

    ready[i] = 0;

    // remainder section
}
```

Алгоритм Петерсона

```
shared int ready[2] = {0, 0};
shared int turn;

while (some condition) {

    ready[i] = 1;              // заявил о готовности
    turn =1- i;                // пригласил другой процесс

    while(ready[1-i] && turn == 1-i);

    // critical section

    ready[i] = 0;

    // remainder section

}
```

Алгоритм Булочной

```
shared enum {false, true} choosing[n];
shared int number[n];

while (some condition) {

        choosing[i] = true;
        number[i] = max(number[0], ..., number[n-1]) + 1;          // получаем “талончик”
        choosing[i] = false;

        for(j = 0; j < n; j++){
                while(choosing[j]);
                while(number[j] != 0 && (number[j],j) < (number[i],i)); // обслуживаем по талончикам и именам
        }

        // critical section

        number[i] = 0;

        // remainder section

}
```

ЗАДАНИЕ

- Сравнить работу:
  - pthread_spinlock_t spinlock;
  - pthread_mutex_t mutex;

Вопросы?