Многопоточное программирование

Введение

План лекции

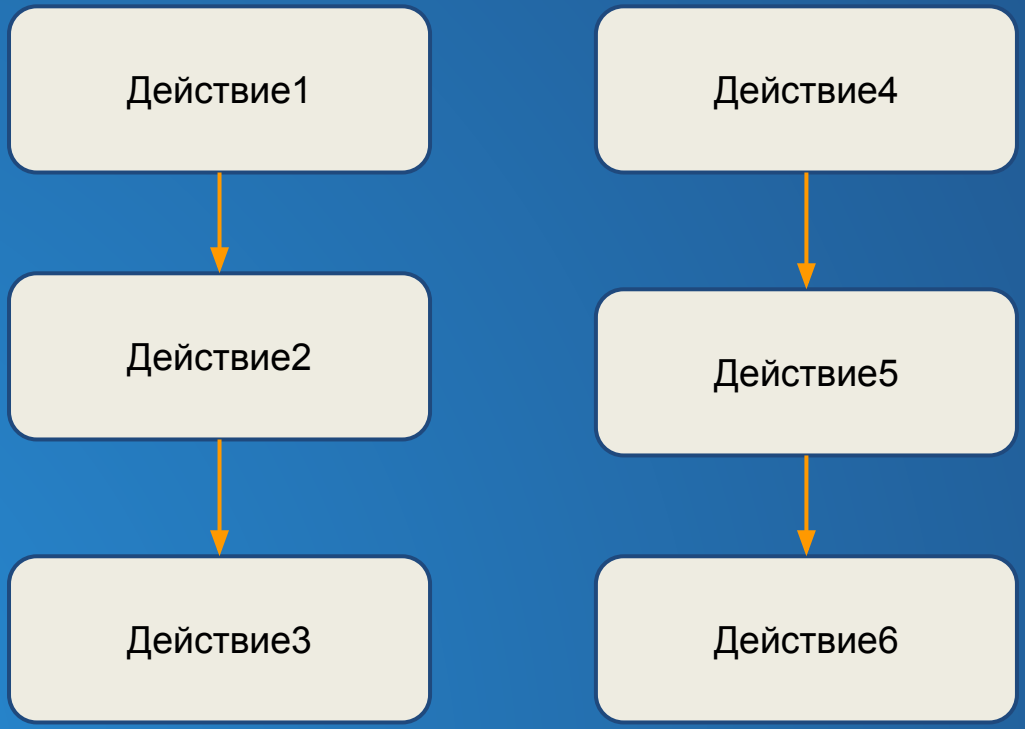
- Введение в курс
- Последовательны и параллельные программы
- Параллельные вычислительные системы

Последовательная программа

Последовательная программа

- Не оптимальное использование аппаратных средств
- Использование традиционных средств программирования
- Инвариантность в возможностям параллелизма аппаратных средств и платформ

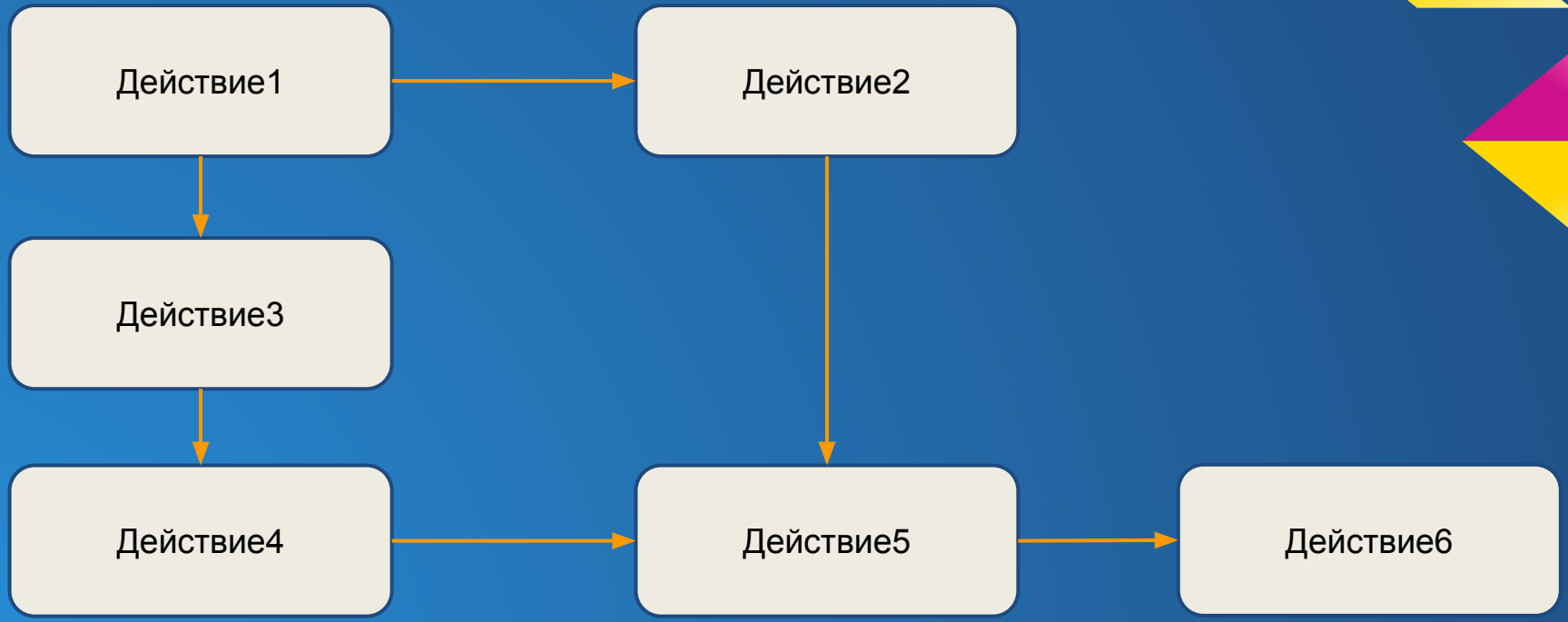
Чисто параллельная программа



Чисто параллельная программа

- Оптимизация использования аппаратных средств
- Использования средств программирования не используемых для последовательных программ
- Повышенная сложность проектирования и разработки
- Ориентация на архитектурные возможности аппаратных средств

Реальные параллельные программы



Особенности разработки

- Управление параллельно выполняющимися действиями
- Обеспечение общими ресурсами
- Необходимость исправлять характерны ошибки (взаимные блокировки, гонки, ...)
- Необходимость обеспечения масштабируемости и балансировки аппаратной загрузки

Большие задачи

Моделирование климатической системы

- 10000000000000000 операций
- Расчет 100 лет за 10 минут

Ядерное оружие

Расшифровка генома человека

Геометрия самолета

Сеточные модели

Параллельные вычисления

Использование нескольких процессоров для:

- Решения задач за меньшее время
- Решения больших задач, чем на одно процессоре

Параллельные вычисления

Использование нескольких процессоров для:

- Решения задач за меньшее время
- Решения больших задач, чем на одно процессоре

Создание параллельного алгоритма

- Поиск параллелизма в последовательном алгоритме
- Декомпозиция задачи
- Определение подзадач и их зависимостей

Параллельные вычисления

Использование нескольких процессоров для:

- Решения задач за меньшее время
- Решения больших задач, чем на одно процессоре

Создание параллельного алгоритма

- Поиск параллелизма в последовательном алгоритме
- Декомпозиция задачи
- Определение подзадач и их зависимостей

Реализация параллельной задачи

- Распределение подзадач между процессами
- Организация взаимодействия процессов

Режимы выполнение независимых частей программы

Многозадачный режим
- Псевдопараллельный
- Разделение времени

Режимы выполнение независимых частей программы

Многозадачный режим
- Псевдопараллельный
- Разделение времени

Параллельное выполнение
- Несколько процессоров
- Конвейерные и векторные устройства

Режимы выполнение независимых частей программы

Многозадачный режим
- Псевдопараллельный
- Разделение времени

Параллельное выполнение
- Несколько процессоров
- Конвейерные и векторные устройства

Распределенные вычисление
- Несколько устройств
- Временные задержки при передачи данных по линиям связи

Пример

Численное вычисление интеграла

$$\int_a^b f(x)\,dx \approx h\Big(f(a)/2 + \sum_{j=1}^{n-1} f(a+jh) + f(b)/2\Big),$$

где $h = (b-a)/n$, а параметр n задает точность вычислений.

П

Ч

```
int eteration_count = 100000000; // количество итераций
int a = 0;                       // левая граница интегрирования
int b = 1;                       // правая граница интегрирования

/*
линейная функция для интеграции
*/
double liner(double x){
    return x;
}

/*
 Вычисляем интеграл функциии func  на отрезке [a,b] с числом разбиений n
*/
double integrate(const double a, const double b, const int n){
    double step = (b - a) / n;

    double result = (liner(a) + liner(b))*step / 2;
    for (int i = 0; i < n; ++i){
        result += liner(a + i* step)*step;
    }

    return result;
}
```

ений.

Пример. Последовательный алгоритм.

```
int _tmain(int argc, _TCHAR* argv[])
{
    time_t start_time;
    time_t finish_time;


    time(&start_time);
    double total = integrate(a, b, eteration_count);
    time(&finish_time);

    std::cout << "Result is: " << total << " Operation time: " << difftime(finish_time, start_time) << std::endl;

    getchar();

    return 0;
}
```

Пример. Параллельный алгоритм

```
std::mutex mutex;
double result;

void thread_integrate(int a, int b, int step_count){

    double part_integral = integrate(a, b, step_count);
    std::lock_guard<std::mutex> guard(mutex);
    result += part_integral;
}
```

Пример. Параллельный алгоритм

```
int thread_count = 4;

if (argc == 2)
    thread_count = _wtoi(argv[1]);

std::vector<std::thread> threads;
double step_length = ((double)(b - a))/thread_count;

time_t start_time;
time_t finish_time;
time(&start_time);

for (int i = 0; i < thread_count; ++i){
    threads.push_back(std::thread(thread_integrate, a + step_length*i,
                                  a + step_length*(i + 1), eteration_count / thread_count));
}

for (auto it = threads.begin(); it != threads.end(); ++it){
    if ((*it).joinable())
        (*it).join();
}

time(&finish_time);
```

Пример

Приложение	Standart	Thread 1	Thread 4	Thread 8	Thread 32
Время	22	22	6	6	6

Вопросы