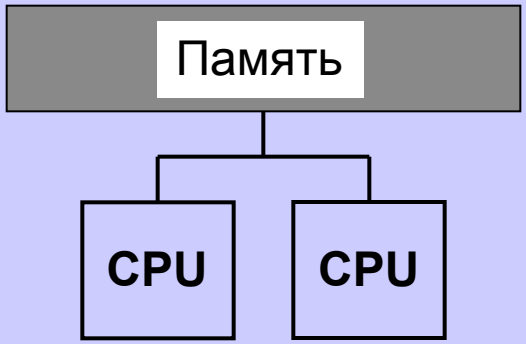
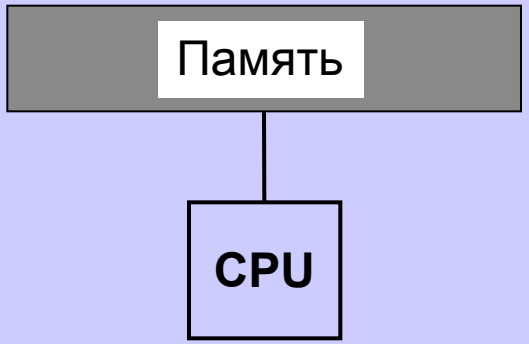
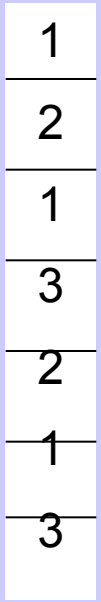
Многопоточное программирование

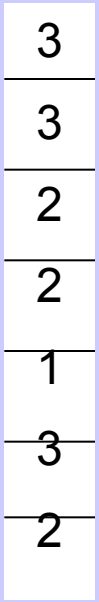
Киреев С.
Отдел МО ВВС ИВМиМГ

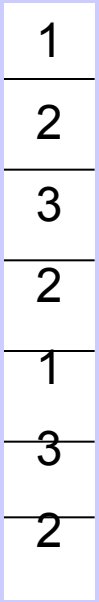
Процессы и потоки











Процессы и потоки

Преимущества потоков:

- Быстрое переключение между потоками
- Простая организация взаимодействия – общая память

Недостатки потоков:

- Некорректное использование данных одним потоком отражается на всех других
- Необходимость в синхронизации при доступе к общим данным
- Используемые библиотеки должны поддерживать многопоточность

Разработка многопоточной программы

Потоки:

- Решают задачу над общими данными
- Взаимодействуют через общую память
- Упорядочивают взаимодействие путем синхронизации

Ключ к созданию корректной параллельной программы – правильная синхронизация процессов и потоков.

Средства создания многопоточных программ

- Библиотеки потоков
  - Posix Threads
  - Windows Threads
  - …
- OpenMP
- Распараллеливающие компиляторы

Разработка многопоточной программы

(WinAPI, Pthreads)

- Все потоки описываются в виде отдельных функций
- Первичный поток выполняет функцию main
- Новые потоки могут запускаться выполняющимися потоками

Управление потоками

- Создание потока
- Завершение потока
- Приостановка потока
- Возобновление потока
- Переключение между потоками

Не завершайте поток вручную, пусть завершится функция потока.

Способы передачи данных между потоками

- Передача числа через параметр функции потока
- Передача указателя на объект через параметр функции потока
- Работа с глобальными переменными

Синхронизация потоков

Необходима при:

- Совместном использовании ресурса (атомарные операции)
- Уведомлении потоков о некотором событии

Средства синхронизации потоков

Windows Threads	**Posix Threads**
Interlocked-функции	
Критические секции Мьютексы	Мьютексы
События	Условные переменные
Семафоры	Семафоры

Средства синхронизации потоков в Windows

В пользовательском режиме:

- Interlocked-функции
- Критические секции

С использованием объектов ядра:

- Процессы, потоки
- События
- Семафоры
- Мьютексы
- Таймеры

Interlocked-функции

Функции атомарного доступа к переменным:

- Присваивание целого числа
- Присваивание указателя
- Условное присваивание целого числа
- Условное присваивание указателя
- Прибавление целого числа
- Инкремент целого числа
- Декремент целого числа

Критические секции

- Используются для взаимоисключающего доступа к ресурсу
- Обеспечивают атомарное исполнение участка кода
- Операции:
  - Вход в критическую секцию (ожидание)
  - Выход из критической секции
- В начале ожидания используют спин-блокировку
- При длительном ожидании используют мьютекс

Объекты ядра

- Объект может быть «занят» или «свободен»
- Операции:
  - Создание объекта ядра
  - Получение доступа к существующему объекту ядра
  - Удаление объекта ядра
  - …

Синхронизация с помощью объектов ядра

- Синхронизация осуществляется с помощью wait-функций, ожидающих освобождения одного или нескольких объектов:
  - WaitForSingleObject
  - WaitForMultipleObjects
  - …

События

- Используются для уведомления потоков о некотором событии
- Операции
  - Перевести в свободное состояние
  - Перевести в занятое состояние
  - Ждать освобождения (wait-функция)
- Типы событий
  - С автосбросом: просыпается 1 ожидающий поток
  - Со сбросом вручную: просыпаются все ожидающие потоки

Семафоры

- Используются для учета некоторого числа ресурсов
- Содержат счетчик доступных ресурсов (i=0…max)
- Операции
  - Занять ресурс: i=i-1 (wait-функция)
  - Освободить ресурс: i=i+1

Мьютексы

- Используются для взаимоисключающего доступа к ресурсу
- Операции:
  - Занять мьютекс (wait-функция)
  - Освободить мьютекс

Таймеры

- Используются для уведомления о наступлении определенного времени
- Операции
  - Установить таймер
  - Ждать срабатывания таймера
  - Сбросить таймер