Знакомство

Знакомство

- **Огнёв Роман Андреевич**
- Связаться: tg **@roa2321**
- Канал с актуальной информацией: tg **@ibks_os_2024**

Цели занятий

—получить представление о структуре ОС

—освоить основные принципы функционирования ОС

—научиться читать исходный код в крупных проектах

—научиться вносить собственные правки в исходный код, отлаживать и исправлять ошибки

План занятий

Курс состоит из 5 лабораторных работ:

- ядро ОС, базовые механизмы (работы 1-3)
- пользовательские программы (работы 4-5)
- файловая система (4-5)

Для сдачи ЛР необходимо:

1) выполнить задание ЛР
2) написать отчет о проделанной работе
3) ответить на вопросы при сдаче

На выполнение одной работы дается 2-4 недели

Отчет

- Необходим для сдачи работы

- Должен включать:
  - Обязательные пункты, которые описаны в задании к ЛР
  - Детальный ход работы, уникальный для каждого студента (какие изменения были внесены, зачем, какие были сложности, ...)

Лабораторная работа

- Весь материал (исходные коды, задания, дополнительный материал) на сайте - ibks.spbstu.ru:3508
- На сайте выполняется автоматическое тестирование и проверка на антиплагиат
- У каждого студента свой вариант задания

Расписание

Виды занятий:

– теоретический материал по ЛР

– практическая работа

– дополнительный материал по курсу

– прием лабораторных работ

PintOS

- Учебная операционная система, которая была разработана Стэнфордским университетом для курса «Принципы построения операционных систем» в 2004 году
- Реализует основные компоненты реальных операционных систем, такие как:
  - процессы ядра
  - загрузка и запуск пользовательских программ
  - файловая система
  - виртуальная память

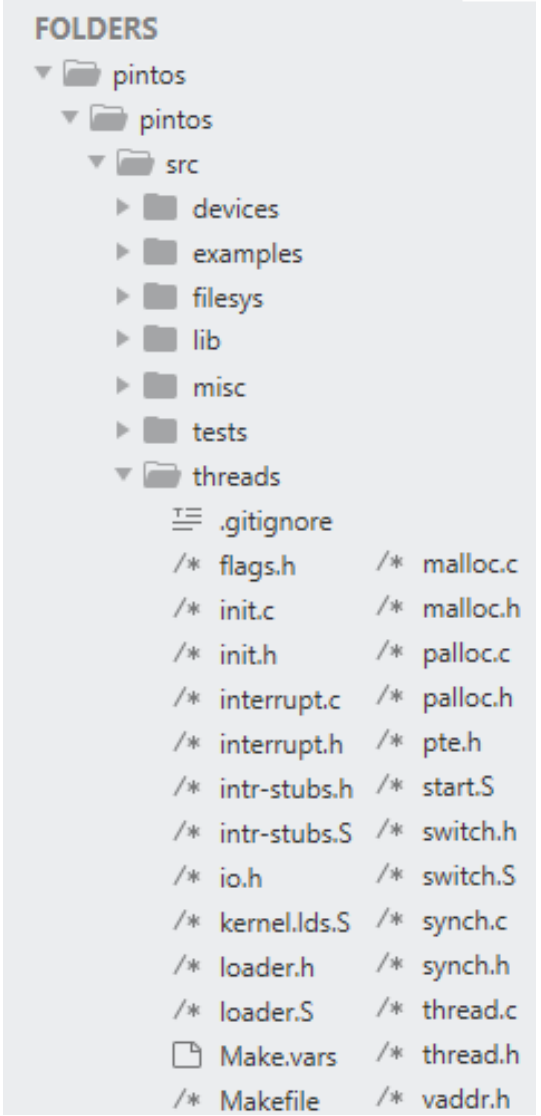
Основные преимущества

- **Модульность** - разделена на набор модулей, которые позволяют студентам изучать различные аспекты операционных систем по отдельности
- **Простота**
- **Документация и комментарии к функциям**
- **Покрытие ключевых тем**
- **Расширяемость** – возможность добавлять собственные функции и возможности

Первая работа
Системный таймер

Чтение исходных кодов

- Изучить рабочую директорию *src/threads* в несколько итераций:
  1) Названия директорий, расширений
  2) Названия файлов
  3) Поверхостное чтение содержимого ключевых файлов
  4) Выяснение связи между директориями и файлами



Потоки

- Поток (или процесс) представляет собой основную единицу выполнения программного кода

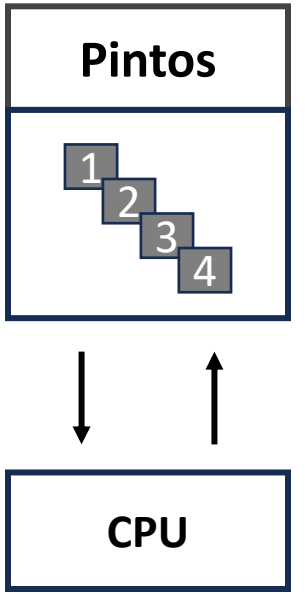
- Основной структурой данных ОС Pintos, предназначенной для работы с процессами, является структура struct thread

```
struct thread
{
      tid_t tid;
      enum thread_status status;
      char name[16];
      uint8_t *stack;
      int priority;
      struct list_elem allelem;
      struct list_elem elem;

      #ifdef USERPROG
          uint32_t *pagedir;
      #endif
      unsigned magic;
};
```

Планирование потоков

- Потоки исполняются на процессоре
- Каждый поток может исполняться неопределенное количество процессорных тиков (ticks)
- ОС использует механизм планирования процессов, чтобы обеспечить каждый поток процессорным временем
- Основной принцип – никто не должен быть обделен



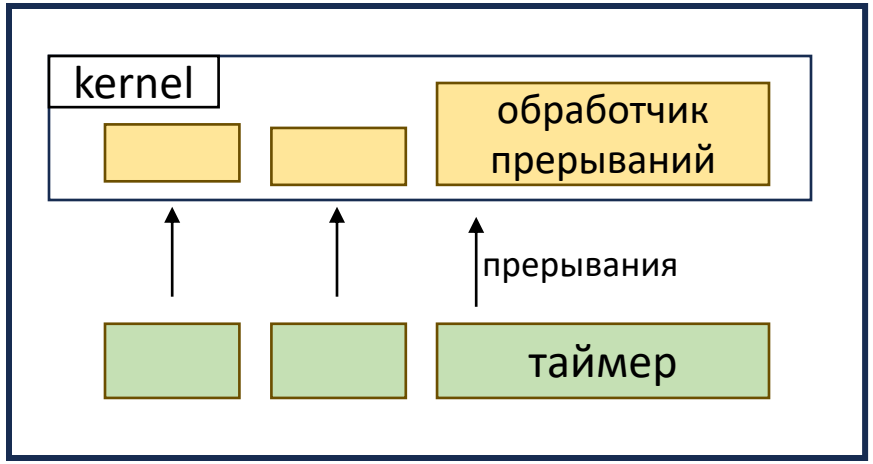

```
/* States in a thread's life cycle. */
enum thread_status
  {
    THREAD_RUNNING,     /* Running thread. */
    THREAD_READY,       /* Not running but ready to run. */
    THREAD_BLOCKED,     /* Waiting for an event to trigger. */
    THREAD_DYING        /* About to be destroyed. */
  };
```

В процессе выполнения поток может переходить из одного состояния в другое

Системный таймер

Системный таймер (или аппаратный таймер) - это важное устройство, используемое в системах для измерения времени и генерации событий в определенные моменты времени

Системный таймер является частью подсистемы управления процессами



Активное ожидание

Функция *timer_sleep(),* реализованная в *devices/timer.c*, приостанавливает выполнение текущего процесса на заданное число тактов таймера *ticks*, в цикле while() проверяя текущее значение ticks и вызывая функцию *thread_yield()*

```
void timer_sleep (int64_t ticks)
{
  int64_t start = timer_ticks ();
  ASSERT (intr_get_level () == INTR_ON);

  /*АКТИВНОЕ ОЖИДАНИЕ*/
  while (timer_elapsed (start) < ticks)

    thread_yield ();
}
```

Реализация содержит в себе цикл активного ожидания
Активное ожидание – это состояние процесса, при котором он многократно проверяет истинность некоторого условия, например, такого как доступность некоторых ресурсов или состояние других процессов

Использование активного ожидания приводит к бесполезному расходованию процессорного времени и снижению общей производительности системы
При некоторых стратегиях планирования времени ЦП в целом корректный алгоритм с активным ожиданием может привести к тупику

Основные этапы работы

В рамках работы необходимо модифицировать системный таймер (файлы devices/timer.[c|h]

Подготовительная часть

1) Изучить механизм работы процессов (threads)
   thread_create, thread_block / thread_unblock, thread_tick, schedule, thread_yield
2) Изучить принцип работы планировщика
3) Разобраться с механизмом работы таймера
4) Понять их взаимосвязь

В рамках **основной части** необходимо устранить недостатки текущей реализации системного таймера

Чтобы избавиться от проблемы «активного ожидания» необходимо решить несколько задач:

1) Как и где хранить информацию о «спящих» процессах (способ хранения – массив или список, зависит от варианта)
2) Найти способ «приостановить» поток, чтобы он не расходовал ресурсы процессора (никак не исполнялся)
3) Определить оптимальный способ «пробуждения» «спящего» потока – в строго определенного время