

Операционные Системы

Потоки исполнения

January 5, 2019

Поток исполнения

- ▶ Поток исполнения - это код и его состояние (а. к. а. контекст)

Поток исполнения

- ▶ Поток исполнения - это код и его состояние (а. к. а. контекст)
 - ▶ код - набор инструкций в памяти, на который указывает регистр *RIP*;

Поток исполнения

- ▶ Поток исполнения - это код и его состояние (а. к. а. контекст)
 - ▶ код - набор инструкций в памяти, на который указывает регистр *RIP*;
 - ▶ контекст потока включает значения регистров и память.

Потоки и процессы

- ▶ Поток работает в контексте некоторого процесса

Потоки и процессы

- ▶ Поток работает в контексте некоторого процесса
 - ▶ т. е. поток "живет" в логическом адресном пространстве процесса;

Потоки и процессы

- ▶ Поток работает в контексте некоторого процесса
 - ▶ т. е. поток "живет" в логическом адресном пространстве процесса;
 - ▶ несколько потоков могут работать в рамках одного процесса;

Потоки и процессы

- ▶ Поток работает в контексте некоторого процесса
 - ▶ т. е. поток "живет" в логическом адресном пространстве процесса;
 - ▶ несколько потоков могут работать в рамках одного процесса;
 - ▶ процесс имеет как минимум один поток.

Стек потока

- ▶ Каждый поток исполнения имеет свой собственный стек

Стек потока

- ▶ Каждый поток исполнения имеет свой собственный стек
 - ▶ стек хранит адреса возвратов и локальные переменные;

Стек потока

- ▶ Каждый поток исполнения имеет свой собственный стек
 - ▶ стек хранит адреса возвратов и локальные переменные;
 - ▶ для процессора стек - место в памяти, куда указывает *RSP*.

Многопоточность

- ▶ В системе могут *одновременно* работать несколько потоков исполнения

Многопоточность

- ▶ В системе могут *одновременно* работать несколько потоков исполнения
 - ▶ на нескольких ядрах процессора;

Многопоточность

- ▶ В системе могут *одновременно* работать несколько потоков исполнения
 - ▶ на нескольких ядрах процессора;
 - ▶ на одном ядре, создавая *иллюзию* одновременной работы.

Многопоточность



Переключение между потоками

- ▶ Для переключения между потоками необходимо:

Переключение между потоками

- ▶ Для переключения между потоками необходимо:
 - ▶ сохранить контекст исполняемого потока;

Переключение между потоками

- ▶ Для переключения между потоками необходимо:
 - ▶ сохранить контекст исполняемого потока;
 - ▶ восстановить контекст потока, на который мы переключаемся.

Пример переключения для x86

```
.text
switch_threads:
    pushq %rbx
    pushq %rbp
    pushq %r12
    pushq %r13
    pushq %r14
    pushq %r15
    pushfq

    movq %rsp, (%rdi)
    movq %rsi, %rsp

    popfq
    popq %r15
    popq %r14
    popq %r13
    popq %r12
    popq %rbp
    popq %rbx

    retq
```

C API

- ▶ `void switch_threads(void **prev, void *
↪ next);`
- ▶ по завершении функции `*prev` будет указывать на сохраненный контекст;

C API

- ▶ `void switch_threads(void **prev, void *
↪ next);`
- ▶ по завершении функции `*prev` будет указывать на сохраненный контекст;
- ▶ *next* указывает на сохраненный контекст потока, на который мы переключаемся.

switch_threads

```
.text
switch_threads:
    /* save contex on stack */
    pushq %rbx
    pushq %rbp
    pushq %r12
    pushq %r13
    pushq %r14
    pushq %r15
    pushfq

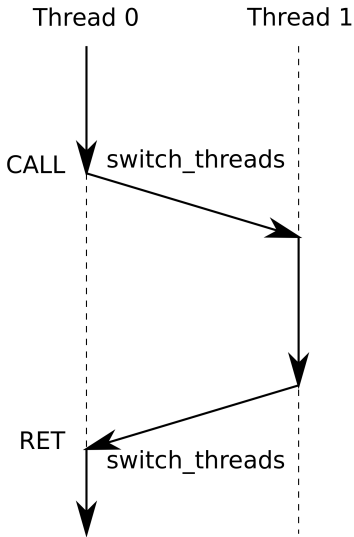
    /* rdi - the first argument */
    movq %rsp, (%rdi)

    /* rsi - the second argument */
    movq %rsi, %rsp

    /* restore from stack */
    popfq
    popq %r15
    popq %r14
    popq %r13
    popq %r12
    popq %rbp
    popq %rbx

    retq /* ! */
```

Переключение потоков



Создание нового потока

- ▶ Как создать новый поток и переключиться на него в первый раз?
 - ▶ нам нужно выделить место для хранения указателя на контекст;

Создание нового потока

- ▶ Как создать новый поток и переключиться на него в первый раз?
 - ▶ нам нужно выделить место для хранения указателя на контекст;
 - ▶ нам нужно выделить место под стек нового потока;

Создание нового потока

- ▶ Как создать новый поток и переключиться на него в первый раз?
 - ▶ нам нужно выделить место для хранения указателя на контекст;
 - ▶ нам нужно выделить место под стек нового потока;
 - ▶ нам нужно сохранить на стеке начальный контекст и сохранить указатель на него.

Начальный контекст

```
struct switch_frame {  
    uint64_t rflags;  
    uint64_t r15;  
    uint64_t r14;  
    uint64_t r13;  
    uint64_t r12;  
    uint64_t rbp;  
    uint64_t rbx;  
    uint64_t rip;  
} __attribute__((packed));
```

Кооперативная многозадачность

- ▶ Невытесняющая (кооперативная) многозадачность
 - ▶ поток должен сам вызвать функцию переключения;

Кооперативная многозадачность

- ▶ Невытесняющая (кооперативная) многозадачность
 - ▶ поток должен сам вызвать функцию переключения;
 - ▶ что если в коде содержится ошибка?

Кооперативная многозадачность

- ▶ Невытесняющая (кооперативная) многозадачность
 - ▶ поток должен сам вызвать функцию переключения;
 - ▶ что если в коде содержится ошибка?
 - ▶ или мы обращаемся к библиотеке, которая выполняет долгую операцию?

Вытесняющая многозадачность

- ▶ Вытесняющая (preemptive) многозадачность
 - ▶ поток снимается ОС с CPU "силой", по истечении *кванта* времени;

Вытесняющая многозадачность

- ▶ Вытесняющая (preemptive) многозадачность
 - ▶ поток снимается ОС с CPU "силой", по истечении *кванта* времени;
 - ▶ синхронизация потоков при этом усложняется;

Вытесняющая многозадачность

- ▶ Вытесняющая (preemptive) многозадачность
 - ▶ поток снимается ОС с CPU "силой", по истечении *кванта* времени;
 - ▶ синхронизация потоков при этом усложняется;
 - ▶ как организовать вытесняющую многозадачность?

Сново о прерываниях

- ▶ Обработчик прерывания "прерывает" исполняемый код

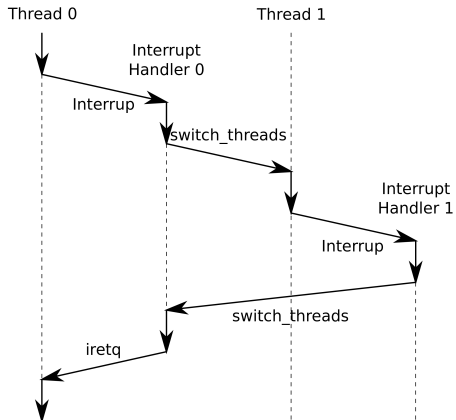
Сново о прерываниях

- ▶ Обработчик прерывания "прерывает" исполняемый код
 - ▶ но обработчик работает в контексте прерванного потока;

Сново о прерываниях

- ▶ Обработчик прерывания "прерывает" исполняемый код
 - ▶ но обработчик работает в контексте прерванного потока;
 - ▶ функцию переключения контекста можно вызвать от имени потока из обработчика прерываний.

Вытесняющая многозадачность



Таймер

- ▶ Таймер может генерировать прерывания с заданной периодичностью
 - ▶ Programmable Interval Timer (PIT, intel 8253) - IBM PC;

Таймер

- ▶ Таймер может генерировать прерывания с заданной периодичностью
 - ▶ Programmable Interval Timer (PIT, intel 8253) - IBM PC;
 - ▶ High Precision Event Timer (HPET);

Таймер

- ▶ Таймер может генерировать прерывания с заданной периодичностью
 - ▶ Programmable Interval Timer (PIT, intel 8253) - IBM PC;
 - ▶ High Precision Event Timer (HPET);
 - ▶ Local APIC Timer.

Планирование потоков

- ▶ Планировщик (scheduler) - компонент ОС, который определяет

Планирование потоков

- ▶ Планировщик (scheduler) - компонент ОС, который определяет
 - ▶ когда переключаться с потока;

Планирование потоков

- ▶ Планировщик (scheduler) - компонент ОС, который определяет
 - ▶ когда переключаться с потока;
 - ▶ на какой поток переключаться.

Простое планирование

- ▶ Рассмотрим простейшую задачу планирования

Простое планирование

- ▶ Рассмотрим простейшую задачу планирования
 - ▶ все задачи известны заранее;

Простое планирование

- ▶ Рассмотрим простейшую задачу планирования
 - ▶ все задачи известны заранее;
 - ▶ про каждую задачу известно, сколько времени она займет;

Простое планирование

- ▶ Рассмотрим простейшую задачу планирования
 - ▶ все задачи известны заранее;
 - ▶ про каждую задачу известно, сколько времени она займет;
 - ▶ задачи работают без переключений;

Простое планирование

- ▶ Рассмотрим простейшую задачу планирования
 - ▶ все задачи известны заранее;
 - ▶ про каждую задачу известно, сколько времени она займет;
 - ▶ задачи работают без переключений;
 - ▶ т. е. нам осталось только определить порядок.

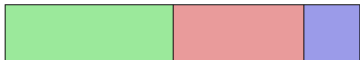
Пропускная способность

 A, 9с

 B, 7с

 C, 3с

Schedule



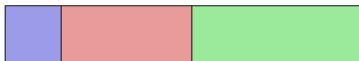
Пропускная способность

 A, 9с

 B, 7с

 C, 3с

Schedule



Среднее время ожидания

- ▶ Пусть все задачи принадлежат разным пользователям

Среднее время ожидания

- ▶ Пусть все задачи принадлежат разным пользователям
 - ▶ пользователю важно, сколько ему нужно ждать завершения его задачи;


Среднее время ожидания

- ▶ Пусть все задачи принадлежат разным пользователям
 - ▶ пользователю важно, сколько ему нужно ждать завершения его задачи;
 - ▶ давайте в качестве метрики использовать среднее время ожидания.

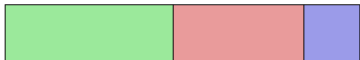
Среднее время ожидания

 A, 9с

 B, 7с

 C, 3с

Schedule



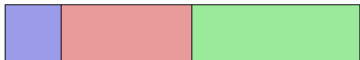
Среднее время ожидания

 A, 9с

 B, 7с

 C, 3с

Schedule



Динамическое создание задач

- ▶ Зачастую все задачи не известны заранее

Динамическое создание задач

- ▶ Зачастую все задачи не известны заранее
 - ▶ задачи могут создаваться в произвольные моменты времени;

Динамическое создание задач

- ▶ Зачастую все задачи не известны заранее
 - ▶ задачи могут создаваться в произвольные моменты времени;
 - ▶ каждая вновь появившаяся задача может изменить решение планировщика.

IO операции

- ▶ Задачи могут давать команды устройствам или ждать каких-то событий:

IO операции

- ▶ Задачи могут давать команды устройствам или ждать каких-то событий:
 - ▶ запись/чтение на/с HDD (порядка нескольких мс);

IO операции

- ▶ Задачи могут давать команды устройствам или ждать каких-то событий:
 - ▶ запись/чтение на/с HDD (порядка нескольких мс);
 - ▶ ждать входящих соединений по сети;

IO операции

- ▶ Задачи могут давать команды устройствам или ждать каких-то событий:
 - ▶ запись/чтение на/с HDD (порядка нескольких мс);
 - ▶ ждать входящих соединений по сети;
 - ▶ ждать, пока пользователь нажмет на клавишу.

IO операции

- ▶ Пока задача ждет завершения IO операции, можно забрать у нее CPU

IO операции

- ▶ Пока задача ждет завершения IO операции, можно забрать у нее CPU
 - ▶ время ожидания может быть большим (1мс - очень много для CPU);

IO операции

- ▶ Пока задача ждет завершения IO операции, можно забрать у нее CPU
 - ▶ время ожидания может быть большим (1мс - очень много для CPU);
 - ▶ утилизация CPU - сколько времени CPU делал полезную работу.

Утилизация



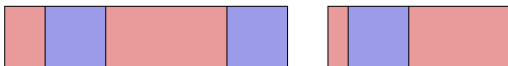
Schedule



Утилизация



Schedule



Информация о задаче

- ▶ Зачастую время работы задачи и расписание ее IO не известны

Информация о задаче

- ▶ Зачастую время работы задачи и расписание ее IO не известны
 - ▶ задачи могут влиять друг на друга или зависеть от внешних обстоятельств;

Информация о задаче

- ▶ Зачастую время работы задачи и расписание ее IO не известны
 - ▶ задачи могут влиять друг на друга или зависеть от внешних обстоятельств;
 - ▶ мы можем оценивать эти параметры и классифицировать задачи.

IO-bounded и CPU-bounded

- ▶ IO-bounded задачи - много IO, но мало вычислений:

IO-bounded и CPU-bounded

- ▶ IO-bounded задачи - много IO, но мало вычислений:
 - ▶ например, текстовый редактор;
 - ▶ вообще приложения, ожидающие ввода пользователя.

IO-bounded и CPU-bounded

- ▶ IO-bounded задачи - много IO, но мало вычислений:
 - ▶ например, текстовый редактор;
 - ▶ вообще приложения, ожидающие ввода пользователя.
- ▶ CPU-bounded задачи - много вычислений, но мало IO:

IO-bounded и CPU-bounded

- ▶ IO-bounded задачи - много IO, но мало вычислений:
 - ▶ например, текстовый редактор;
 - ▶ вообще приложения, ожидающие ввода пользователя.
- ▶ CPU-bounded задачи - много вычислений, но мало IO:
 - ▶ например, научные вычисления;
 - ▶ компиляция программ.

Round Robin

- ▶ Round Robin - выдаем потокам квант времени на CPU по очереди:

Round Robin

- ▶ Round Robin - выдаем потокам квант времени на CPU по очереди:
 - ▶ поток, отработавший свой квант, встает в конец очереди;

Round Robin

- ▶ Round Robin - выдаем потокам квант времени на CPU по очереди:
 - ▶ поток, отработавший свой квант, встает в конец очереди;
 - ▶ каждый новый поток встает в конец очереди;

Round Robin

- ▶ Round Robin - выдаем потокам квант времени на CPU по очереди:
 - ▶ поток, отработавший свой квант, встает в конец очереди;
 - ▶ каждый новый поток встает в конец очереди;
 - ▶ потоки, дождавшиеся завершения IO, встают в конец очереди;

Round Robin

- ▶ Round Robin - выдаем потокам квант времени на CPU по очереди:
 - ▶ поток, отработавший свой квант, встает в конец очереди;
 - ▶ каждый новый поток встает в конец очереди;
 - ▶ потоки, дождавшиеся завершения IO, встают в конец очереди;
 - ▶ CPU отдается потоку в начале очереди.

Достоинства Round Robin

- ▶ К списку достоинств Round Robin можно отнести:




Достоинства Round Robin

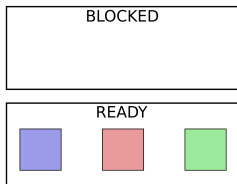
- ▶ К списку достоинств Round Robin можно отнести:
 - ▶ подход очень прост;

Достоинства Round Robin

- ▶ К списку достоинств Round Robin можно отнести:
 - ▶ подход очень прост;
 - ▶ время ожидания CPU ограничено - никто не голодает.




Round Robin

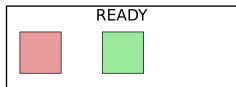
  CPU-bounded
 IO-bounded



Schedule

Round Robin




  CPU-bounded
 IO-bounded

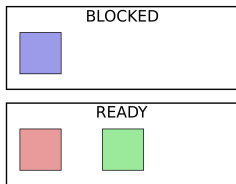


Schedule



Round Robin




  CPU-bounded
 IO-bounded



Schedule



Round Robin




  CPU-bounded
 IO-bounded

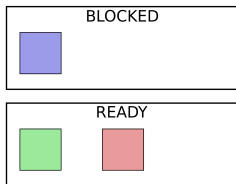


Schedule



Round Robin

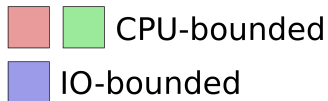
  CPU-bounded
 IO-bounded



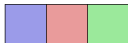
Schedule



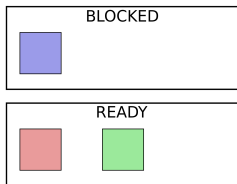
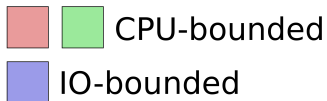
Round Robin



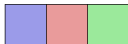
Schedule






Round Robin

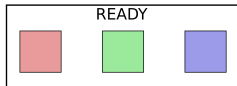
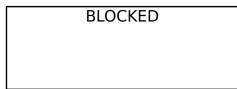


Schedule



Round Robin




  CPU-bounded
 IO-bounded

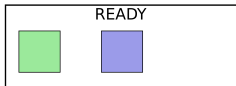


Schedule

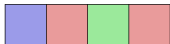


Round Robin




  CPU-bounded
 IO-bounded

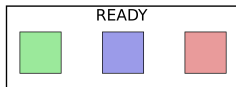


Schedule

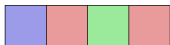


Round Robin




  CPU-bounded
 IO-bounded

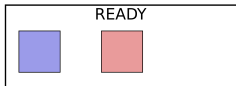


Schedule



Round Robin

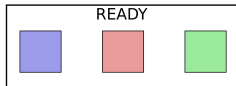
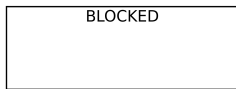
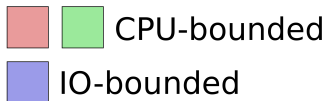
  CPU-bounded
 IO-bounded



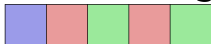
Schedule






Round Robin

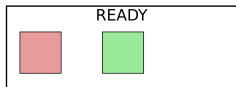


Schedule

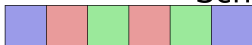


Round Robin

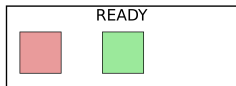
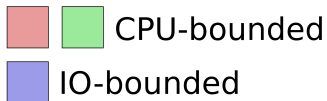
  CPU-bounded
 IO-bounded



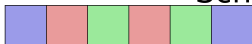
Schedule



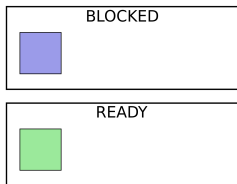
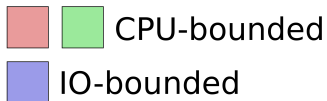
Round Robin



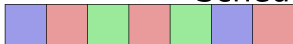
Schedule



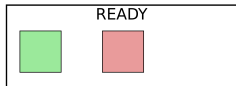
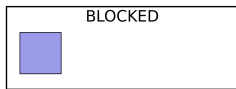
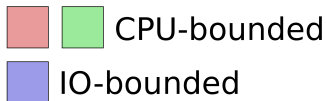
Round Robin



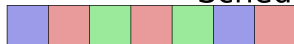
Schedule



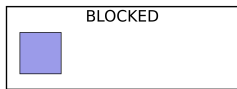
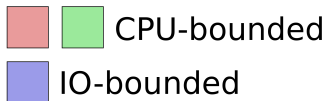
Round Robin



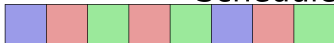
Schedule



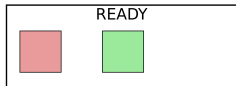
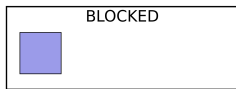
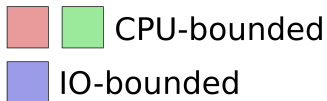
Round Robin



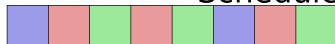
Schedule



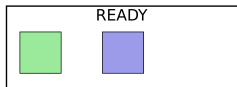
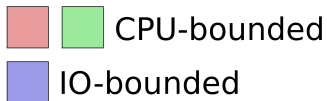
Round Robin



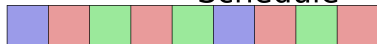
Schedule






Round Robin

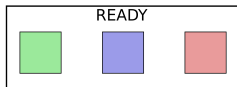


Schedule

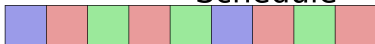


Round Robin

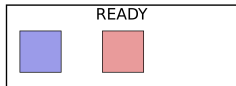
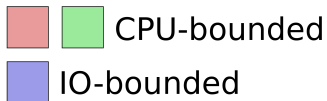
  CPU-bounded
 IO-bounded



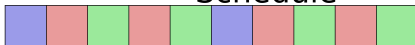
Schedule






Round Robin

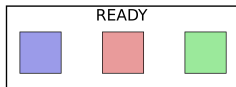


Schedule

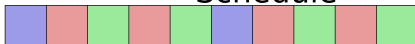


Round Robin

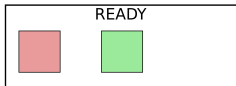
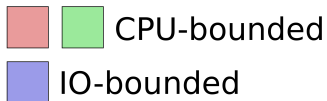
  CPU-bounded
 IO-bounded



Schedule



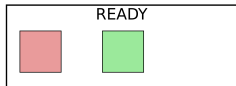
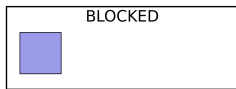
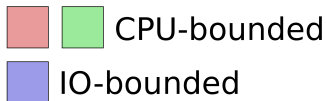
Round Robin



Schedule



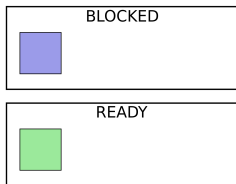
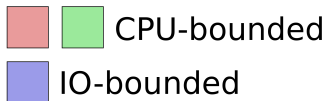
Round Robin



Schedule



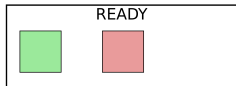
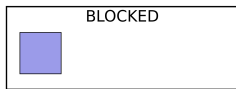
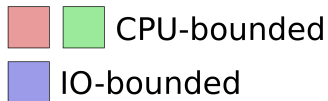
Round Robin



Schedule



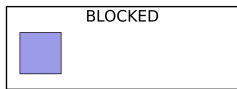
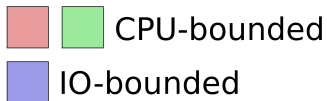
Round Robin



Schedule



Round Robin



Schedule



Выбор кванта времени

- ▶ Из каких соображений стоит выбирать квант времени?

Выбор кванта времени

- ▶ Из каких соображений стоит выбирать квант времени?
 - ▶ чем больше квант

Выбор кванта времени

- ▶ Из каких соображений стоит выбирать квант времени?
 - ▶ чем больше квант
 - ▶ тем меньше доля времени на переключение;
 - ▶ тем больше время отклика;

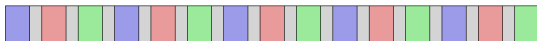
Выбор кванта времени

- ▶ Из каких соображений стоит выбирать квант времени?
 - ▶ чем больше квант
 - ▶ тем меньше доля времени на переключение;
 - ▶ тем больше время отклика;
 - ▶ чем меньше квант

Выбор кванта времени

- ▶ Из каких соображений стоит выбирать квант времени?
 - ▶ чем больше квант
 - ▶ тем меньше доля времени на переключение;
 - ▶ тем больше время отклика;
 - ▶ чем меньше квант
 - ▶ тем больше доля времени на переключение;
 - ▶ тем меньше время отклика.

Выбор кванта времени



Планировщик Windows

- ▶ Потокам в Windows назначен приоритет
 - ▶ приоритет состоит из класса и приоритета внутри класса.

Планировщик Windows

- ▶ Потокам в Windows назначен приоритет
 - ▶ приоритет состоит из класса и приоритета внутри класса.
- ▶ На исполнение выбирается поток с наивысшим приоритетом

Планировщик Windows

- ▶ Потокам в Windows назначен приоритет
 - ▶ приоритет состоит из класса и приоритета внутри класса.
- ▶ На исполнение выбирается поток с наивысшим приоритетом
 - ▶ для потоков с равными приоритетами используется RR.

Priority Boost

- ▶ Чтобы избежать неограниченного голодания потоков, Windows иногда повышает им приоритет:

Priority Boost

- ▶ Чтобы избежать неограниченного голодания потоков, Windows иногда повышает им приоритет:
 - ▶ если поток отвечает за видимую часть UI;

Priority Boost

- ▶ Чтобы избежать неограниченного голодания потоков, Windows иногда повышает им приоритет:
 - ▶ если поток отвечает за видимую часть UI;
 - ▶ при получении ввода или завершении операции IO;

Priority Boost

- ▶ Чтобы избежать неограниченного голодания потоков, Windows иногда повышает им приоритет:
 - ▶ если поток отвечает за видимую часть UI;
 - ▶ при получении ввода или завершении операции IO;
 - ▶ для "случайно" выбранных потоков.

Планировщик Linux (один из)

- ▶ Completely Fair Scheduler (CFS) - честный планировщик:

Планировщик Linux (один из)

- ▶ Completely Fair Scheduler (CFS) - честный планировщик:
 - ▶ для каждого потока поддерживается "виртуальное время";

Планировщик Linux (один из)

- ▶ Completely Fair Scheduler (CFS) - честный планировщик:
 - ▶ для каждого потока поддерживается "виртуальное время";
 - ▶ "виртуальное время" увеличивается, когда поток работает;

Планировщик Linux (один из)

- ▶ Completely Fair Scheduler (CFS) - честный планировщик:
 - ▶ для каждого потока поддерживается "виртуальное время";
 - ▶ "виртуальное время" увеличивается, когда поток работает;
 - ▶ CPU отдаем потоку с наименьшим "виртуальным временем".