# Управление потоками

# Пример

В традиционных операционных системах у каждого процесса есть адресное пространство и единственный поток управления. Нередко возникают ситуации, когда неплохо было бы иметь несколько потоков управления в одном и том же адресном пространстве, выполняемых квазипараллельно, как будто они являются чуть ли не обособлен-ли не обособлен-ли не обособленными процессами

# Применение потоков

Во многих приложениях одновременно происходит несколько действий, часть которых может периодически быть заблокированной. Модель программирования упрощается за счет разделения такого приложения на несколько последовательных потоков, выполняемых в квазипараллельном режиме.

# Применение потоков

Вторым аргументом в пользу потоков является легкость (то есть быстрота) их создания и ликвидации по сравнению с более «тяжеловесными» процессами. Во многих системах создание потоков осуществляется в 10–100 раз быстрее, чем создание процессов.

# Применение потоков

Когда потоки работают в рамках одного центрального процессора, они не приносят никакого прироста производительности, но когда проводятся значительные вычисления, а также значительная часть времени тратится на ожидание ввода-вывода, наличие потоков позволяет этим действиям перекрываться по времени, ускоряя работу приложения.

# Применение потоков

Потоки весьма полезны для систем, где есть реальная возможность параллельных вычислений.

# Процессы и потоки

**Процессы** представляют собой экземпляры программ, выполняющихся в ОС. Каждый процесс имеет собственное виртуальное адресное пространство, набор ресурсов (таких как файлы, память и т.д.) и один или несколько потоков.

**Потоки** являются единицами исполнения внутри процесса. Они делятся общим адресным пространством и ресурсами своего родительского процесса, но каждый поток имеет свой собственный стек и состояние выполнения. Именно потоки выполняют конкретные задачи и занимают процессорное время.

# Процессы и потоки

В операционных системах планируются потоки, а не процессы. Планировщик операционной системы распределяет процессорное время между различными потоками выполнения, обеспечивая их эффективное исполнение.

# Пример

1.пользователь удаляет одно предложение на первой странице 800-страничного документа

2.пользователь хочет внести еще одну поправку на 600-й

3.текстовый процессор вынужден немедленно переформатировать всю книгу вплоть до 600-й страницы

# Пример

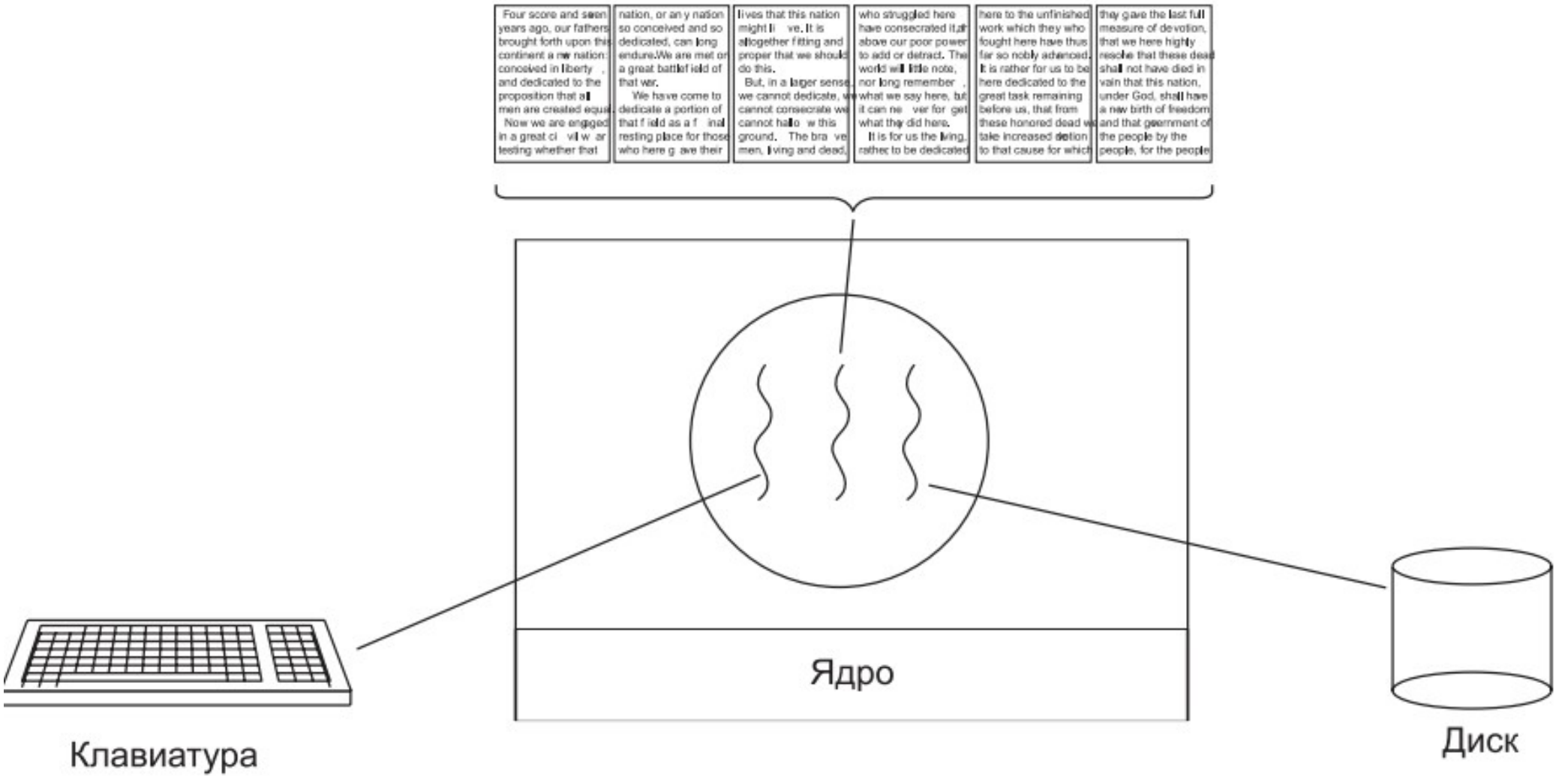
Перед отображением 600-й страницы может произойти существенная задержка, вызывающая недовольство пользователя.

# Распоточить программу

Один из потоков взаимодействует с пользователем

Другой занимается переформатированием в фоновом режиме

Третий поток может заниматься созданием резервных копий на диске, не мешая первым двум


Four score and seven
years ago, our fathers
brought forth upon this
continent a new nation:
conceived in liberty ,
and dedicated to the
proposition that all
men are created equal.
Now we are engaged
in a great ci vil w ar
testing whether that
nation, or an y nation
so conceived and so
dedicated, can long
endure.We are met on
a great battlef ield of
that war.
We have come to
dedicate a portion of
that f ield as a f inal
resting place for those
who here g ave their
li ves that this nation
might li ve. It is
altogether f itting and
proper that we should
do this.
But, in a larger sense,
we cannot dedicate, we
cannot consecrate we
cannot hallo w this
ground. The bra ve
men, l iving and dead,
who struggled here
have consecrated it,f ar
above our poor power
to add or detract. The
world will little note,
nor long remember ,
what we say here, but
it can ne ver for get
what they did here.
It is for us the living,
rather to be dedicated
here to the unfinished
work which they who
fought here have thus
far so nobly advanced.
It is rather for us to be
here dedicated to the
great task remaining
before us, that from
these honored dead we
take increased devotion
to that cause for which
they gave the last full
measure of de votion,
that we here highly
resolve that these dead
shall not have died in
vain that this nation,
under God, shall have
a new birth of freedom
and that government of
the people by the
people, for the people
Ядро
Клавиатура
Диск

# По-другому

Если бы программа была рассчитана на работу только одного потока, то с начала создания резервной копии на диске и до его завершения игнорировались бы команды с клавиатуры или мыши. Пользователь ощущал бы это как слабую производительность.

# Разница между процессами и потоками

Процессы используются для группировки ресурсов в единое образование, а потоки являются «сущностью», распределяемой для выполнения на центральном процессоре.

# Потоки

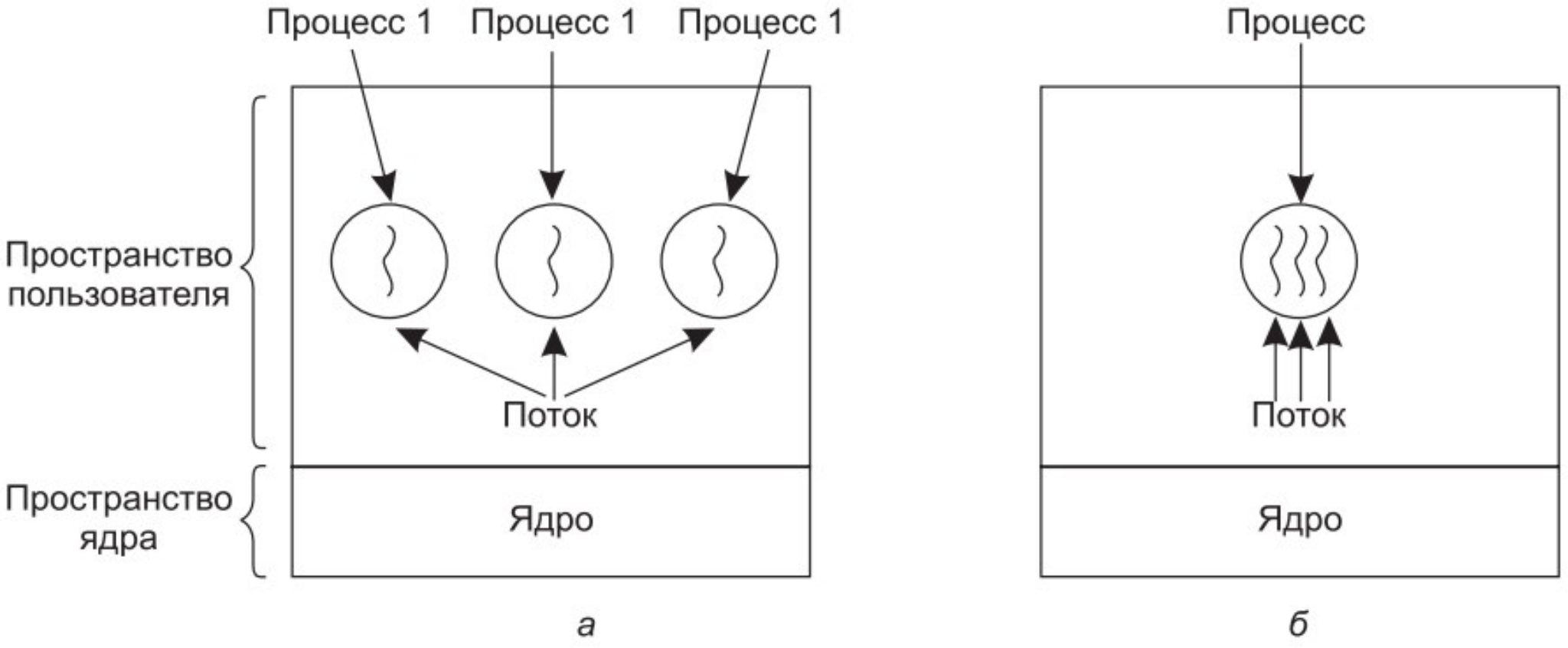
Потоки добавляют к модели процесса возможность реализации нескольких, в большой степени независимых друг от друга, выполняемых задач в единой среде процесса.

# Процессы и потоки

Наличие нескольких потоков, выполняемых параллельно в рамках одного процесса, является аналогией наличия нескольких процессов, выполняемых параллельно на одном компьютере. В первом случае потоки используют единое адресное пространство и другие ресурсы. А в последнем случае процессы используют общую физическую память, диски, принтеры и другие ресурсы.

# Многопоточный режим

Некоторые центральные процессоры обладают непосредственной аппаратной поддержкой многопоточного режима и проводят переключение потоков за наносекунды


Процесс 1
Процесс 1
Процесс 1
Пространство пользователя
Поток
Пространство ядра
Ядро
а
Процесс
Поток
Ядро
б

# Многопоточность на C#

Поток представляет некоторую часть кода программы. При выполнении программы каждому потоку выделяется определенный квант времени. И при помощи многопоточности мы можем выделить в приложении несколько потоков, которые будут выполнять различные задачи одновременно.

# Многопоточность на C#

Если у нас, допустим, графическое приложение, которое посылает запрос к какому-нибудь серверу или считывает и обрабатывает огромный файл, то без многопоточности у нас бы блокировался графический интерфейс на время выполнения задачи. А благодаря потокам мы можем выделить отправку запроса или любую другую задачу, которая может долго обрабатываться, в отдельный поток.

# Многопоточность на C#

Основной функционал для использования потоков в приложении сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток - класс Thread.

# Thread

ExecutionContext: контекст, в котором выполняется поток

IsAlive: указывает, работает ли поток в текущий момент

IsBackground: указывает, является ли поток фоновым

Name: содержит имя потока

ManagedThreadId: возвращает ID текущего потока

# Thread

Priority: хранит приоритет потока - значение перечисления ThreadPriority:

- Lowest
- BelowNormal
- Normal
- AboveNormal
- Highest

# Thread

ThreadState: возвращает состояние потока (Aborted, AbortRequested, Background, Running, Stopped, StopRequested,Suspended, SuspendRequested, Unstarted, WaitSleepJoin)

# Пример

```
Thread currentThread = Thread.CurrentThread;
Console.WriteLine($"Имя потока: {currentThread.Name}");
currentThread.Name = "Метод Main";
Console.WriteLine($"Имя потока: {currentThread.Name}");

Console.WriteLine($"Запущен ли поток: {currentThread.IsAlive}");
Console.WriteLine($"Id потока: {currentThread.ManagedThreadId}");
Console.WriteLine($"Приоритет потока: {currentThread.Priority}");
Console.WriteLine($"Статус потока: {currentThread.ThreadState}");
```

```
Имя потока:
Имя потока: Метод Main
Запущен ли поток: True
Id потока: 1
Приоритет потока: Normal
Статус потока: Running
```

# Задание

- Получить информацию и текущем потоке: ID, приоритет, состояние, запущен или нет
- Задать произвольное имя потоку, проверить изменилось ли имя
- Изменить приоритет потока на самый высокий