Занятие № 10

Тема: «Модель процесса. Иерархия процесса. Состояния процесса. Операции над процессами»

Процесс

Процесс – абстрактное понятие, описывающие работу программы.

В современных ОС многозадачность реализована за счет предоставления пользовательской программе процессора на несколько миллисекунд. При условии чередования использования процессора между программами. Все ПО исполняемое на компьютере, а иногда и операционная система, организовано в виде последовательных процессов.

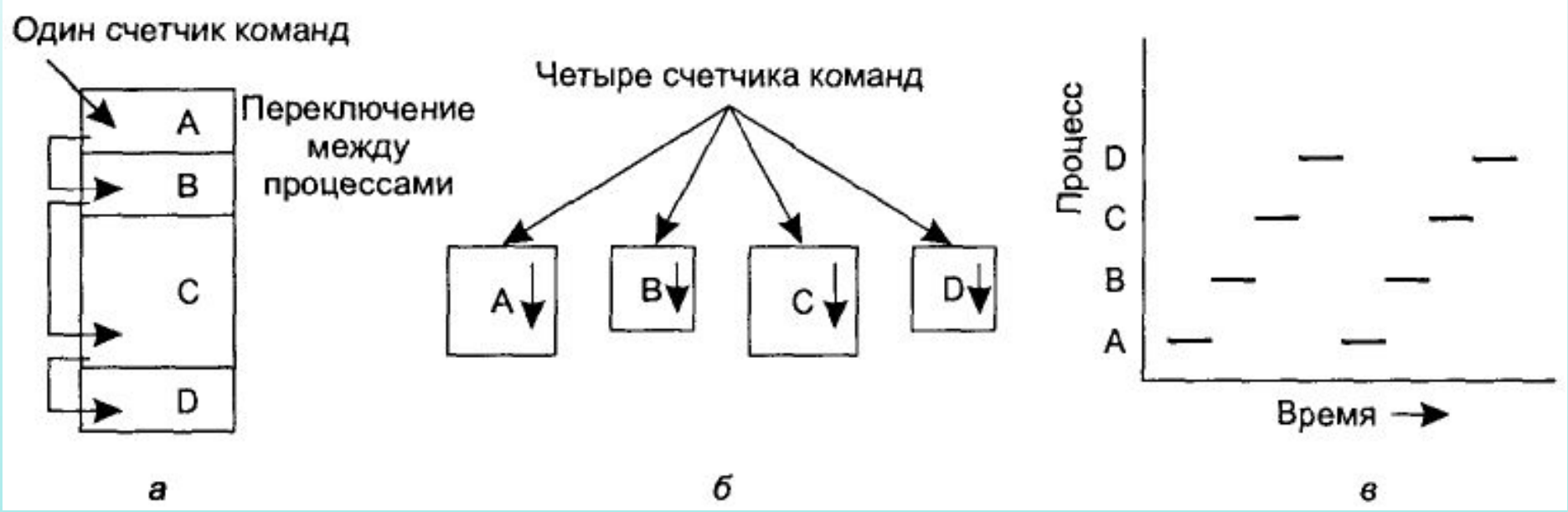
Процессом является выполняемая программа, включая:

текущие значения счетчиков команд

текущие значения регистров

текущие значения переменных

Модель процесса



Четыре программы, работающие в многозадачном режиме

Концептуальная модель четырех независимых друг от друга последовательных процессов

В отдельно взятый момент времени активна только одна программа

Внимание:

- Центральный процессор переключается между процессами, следовательно, скорость вычислений процесса всегда будет разной.
- Процессы не должны программироваться с жестко заданным временем выполнения.
- Планирование процессов.
- Программа и процесс понятия схожие, но разные!
- Если программа запущенная дважды, то ею заняты два процесса

Создание процесса

В универсальных системах определенные способы создания и прекращения процессов по мере необходимости.

Способы создания процессов:

1. Инициализация системы
2. Выполнение работающим процессом системного запроса на создание процесса
3. Запрос пользователя на создание процесса
4. Инициализация пакетного задания

Инициализация операционной системы

При загрузке ОС создается несколько процессов.

Процессы, обеспечивающие взаимодействие с пользователями и выполнение заданий, являются **высокоприоритетными** процессами**.**

Процессы, не связанные с конкретными пользователями, но выполняющими ряд специфических функций, являются **фоновыми** процессами (**демонами**).

Пример:

Получение электронной почты, web-новости, вывод на принтер

Создание процесса (системный вызов)

Новый процесс формируется на основании системного запроса от текущего процесса.

В роли текущего процесса может выступать:

- Процесс, запущенный пользователем;
- Системный процесс;
- Процесс, инициализированный клавиатурой или мышью;
- Процесс, управляющий пакетами.

Создание процесса (системный вызов)

В UNIX существует только один системный запрос: **fork** (**ветвление**). Этот запрос создает дубликат вызываемого процесса.

В Windows же вызов всего одной функции **CreateProcess** интерфейса Win32 управляет и созданием процесса, и запуском в нем нужной программы.

Кроме CreateProcess в Win32 есть около 100 функций для управления процессами и их синхронизации

Завершение процесса

1. Обычный выход (преднамеренно);
2. Выход по ошибке (преднамеренно);
3. Выход по неисправимой ошибке (непреднамеренно);
4. Уничтожение другим процессом (непреднамеренно).

После окончания работы процесс генерирует системный запрос на завершение работы. В UNIX этот системный запрос – **exit**, а в Windows – **ExitProcess**.

Программы, рассчитанные на работу с экраном, также поддерживают преднамеренное завершение работы

Иерархия процессов

В некоторых системах родительский и дочерний процессы остаются связанными между собой определенным образом.

Дочерний процесс также может, в свою очередь, создавать процессы, формируя иерархию процессов. В UNIX процесс, все его «дети» и дальнейшие потомки образуют группу процессов.

Сигнал, посылаемый пользователем с клавиатуры, доставляется всем членам группы, взаимодействующим с клавиатурой в данный момент

Пример иерархии процессов

В образе загрузки присутствует специальный процесс **init**. При запуске этот процесс считывает файл, в котором находится информация о количестве терминалов. Затем процесс разветвляется таким образом, чтобы каждому терминалу соответствовал один процесс.

Процессы ждут, пока какой-нибудь пользователь не войдет в систему. Если пароль правильный, процесс входа в систему запускает оболочку для обработки команд пользователя, которые, в свою очередь, могут запускать процессы. Таким образом, все процессы в системе принадлежат к единому дереву, начинающемуся с процесса init

Иерархия процессов в разных ОС

В Windows не существует понятия иерархии процессов, и все процессы равноправны.

Единственное, в чем проявляется что-то вроде иерархии процессов – создание процесса, в котором родительский процесс получает специальный маркер (так называемый **дескриптор**), позволяющий контролировать дочерний процесс.

Но маркер можно передать другому процессу, нарушая иерархию.

В UNIX это невозможно

Состояние процессов

Несмотря на самостоятельность каждого процесса, наличие собственного счетчика команд и внутреннего состояния, процессам зачастую необходимо взаимодействовать с другими процессами.

Один процесс может генерировать выходную информацию, используемую другими процессами в качестве входной информации.

Пример:

Выходные данные процесса `cat` могут служить входными данными для процесса `grep`.

```
Cat chapter.txt chapter2.txt | grep tree
```

Реализация процессов

Для реализации модели процессов операционная система содержит **таблицу процессов.**

В таблице содержится информация – о состоянии процесса, счетчик команд, указатель стека, распределение памяти, состояние открытых файлов – необходима для переключения в состояние **готовности** или **блокировки**

Состояние блокировки

Возможны два вида блокировки процесса:

1. Процесс блокируется с точки зрения логики приложения (из-за отсутствия входных данных)

2. Процесс блокируется операционной системой (из-за отсутствия ресурсов)

Реализация процессов (**таблица процессов**)

Управление процессом	**Управление памятью**	**Управление файлами**
Регистры	Указатель на текстовый сегмент	Корневой каталог
Счетчик команд	Указатель на сегмент данных	Рабочий каталог
Слово состояния программы	Указатель на сегмент стека	Дескрипторы файла
Указатель стека		Идентификатор пользователя
Состояние процесса		Идентификатор группы
Приоритет		
Параметры планирования		
Идентификатор процесса		
Родительский процесс		
Группа процесса		
Сигналы		
Время начала процесса		
Использованное процессорное время		
Процессорное время дочернего процесса		
Время следующего аварийного сигнала		