**Процессы, потоки.**

1. **Ресурсы ОС (перечислить).** К основным ресурсам могут быть отнесены процессоры, память, внешние устройства, данные и программы.

2. **Процесс (задача)** – заявка к операционной системе на потребление

ресурсов вычислительной системы необходимых для функционирования

приложения.

3. **Поток.** Понятию «поток» соответствует последовательный переход

процессора от одной команды программы к другой. ОС распределяет

процессорное время между потоками.

4. **Исходный код-**текст [компьютерной программы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) на каком-либо языке [программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) или [языке разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8), который может быть прочтён человеком. В обобщённом смысле это любые входные данные для [транслятора](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Исходный код [транслируется](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в исполняемый код целиком до запуска программы при помощи [компилятора](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), или может исполняться сразу при помощи [интерпретатора](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

5. **Компилятор-**это программа, которая преобразует исходные тексты программ, написанные на языке программирования высокого уровня, в программу на машинном языке, «понятную» компьютеру.

6. **Объектный модуль** программы получается в результате трансляции исходного текста модуля. В состав объектного модуля программы помещается оттранслированный код модуля, информация для редактора связей, позволяющая объединять модули в единую программу, и отладочная информация (переменные, константы, метки и их адреса). Для объектного модуля по умолчанию используется имя файла, совпадающее с именем файла исходного модуля и с расширением obj.

7. **Редактор связей-**(также компоновщик, линкер от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) link editor, linker) программа, позволяющая осуществить связывание отдельных оттранслированных программных модулей в единую программу.

8. **Загрузочный модуль**- это готовая программа в машинных кодах, которая может быть записана во внутреннюю память микроконтроллера. Может храниться на компьютере в бинарном или HEX формате.

9. **Загрузчик -** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) loader) в [информатике](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), [программа](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), отвечающая за загрузку [исполнимых файлов](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D0%BC%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C) и запуск соответствующих новых [процессов](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Обычно является частью [операционной системы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), но может быть и самостоятельной программой**-**к примеру, позволяющей операционной системе запускать программы, [скомпилированные](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F) для других операционных систем

10. **Исполняемый код** -система команд (язык) конкретной [вычислительной машины](http://dic.academic.ru/dic.nsf/ruwiki/616) (машинный язык), который [интерпретируется](http://dic.academic.ru/dic.nsf/ruwiki/15224) непосредственно [микропроцессором](http://dic.academic.ru/dic.nsf/ruwiki/1187537) или [микропрограммами](http://dic.academic.ru/dic.nsf/ruwiki/224644) данной вычислительной машины.

11. **Состояние процесса (перечислить)** Существует три возможных состояния процесса:

**-** *Выполнение* (процесс использует процессор)

**-** *Готовность* (процесс временно приостановлен, чтобы позволить выполняться другому процессу).

**-** *Блокировка/ожидание* (процесс не может быть запущен прежде, чем произойдет некое внешнее событие).

12. **Блок управления процессом (PCB)** Для реализации модели процессов операционная система поддерживает таблицу (массив структур), называемую таблицей процессов, в которой для каждого процесса имеется по одной записи, эти записи называются *блоками управление процессом.* Записи таблицы содержат информацию о состоянии процесса, счетчике команд, указателе стека, распределение памяти, состоянии открытых файлов, использовании и распределении ресурсов, аварийных и прочих сигналах. А также всю остальную информацию, которую необходимо сохранить при переключении из состояния *выполнения* в состояние *готовности*, чтобы позже процесс мог быть запущен снова так, как будто он никогда не останавливался.

13. **Планировщики для диспетчеризации процессов-** часть ОС отвечающая за обеспечение поочередного доступа процессов к одному процессору.

*-долговременный планировщик* (планировщик задания)

Долговременный планировщик решает, какие задачи или процессы будут добавлены в очередь процессов, готовых к выполнению; то есть, когда производится попытка запуска процесса, долговременный планировщик или добавляет новый процесс в очередь готовых процессов (допускает к выполнению), или откладывает это действие. Таким образом, долговременный планировщик решает, какие процессы будут выполняться одновременно, тем самым контролируя степень параллелизма и пропорцию между процессами, интенсивно выполняющими ввод-вывод, и процессами, интенсивно использующими процессор. Обычно в настольных компьютерах не применяется долговременный планировщик и новые процессы допускаются к выполнению автоматически. Но данный планировщик очень важен для систем реального времени, так как при чрезмерной нагрузке системы параллельно выполняющимися процессами время отклика системы может стать больше требуемого, что недопустимо.

*-кратковременный планировщик* (планировщик процессора)

Планировщик на этом уровне решает, какие из готовых и загруженных в память процессов будут запущены на ЦПУ после прерывания (по времени, операции ввода-вывода, вызову операционной системы или другому сигналу). Решения на этом уровне приходится принимать очень часто (как минимум, каждый временной отрезок). Также планировщик может поддерживать или не поддерживать вытесняющую многозадачность (то есть иметь возможность прервать исполнение какого-либо процесса).

*-планировщик откачки/подкачки процессов* (среднесрочный планировщик)

Во всех системах с [виртуальной памятью](http://www.turbo-ballet.com/kult777/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) среднесрочный планировщик временно перемещает (выгружает) процессы из основной памяти во вторичную (например, на жёсткий диск), и наоборот. Эти действия называются подкачкой или свопингом. Среднесрочный планировщик может принять решение выгрузить процесс из основной памяти если:

* процесс был неактивен некоторое время;
* процесс имеет низкий приоритет;
* процесс часто вызывает ошибки страниц (page fault);
* процесс занимает большое количество основной памяти, а системе требуется свободная память для других целей (например, чтобы удовлетворить запрос выделения памяти для другого процесса).

Процесс будет возвращён в основную память, когда будет доступно необходимое количество свободной памяти или когда процесс выйдет из режима ожидания (в этом случае планировщик выгрузит из основной памяти другой процесс для освобождения основной памяти).

Во многих современных системах, поддерживающих отображение виртуального адресного пространства на вторичную память, отличную от файла подкачки, среднесрочный планировщик может одновременно играть роль и долговременного планировщика, рассматривая новые процессы как процессы, которые были выгружены из основной памяти. Таким образом система может подгружать в основную память программный код только тогда, когда он понадобится процессу для выполнения (это называется загрузкой по требованию или «[ленивой загрузкой](http://www.turbo-ballet.com/kult777/%D0%9E%D1%82%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B8%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)»)

14. **Очереди процессов.** Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме всего прочего, содержит по крайней мере один указатель на другой дескриптор, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать процессы, переводить процессы из одного состояния в другое.

15. **Однопоточные языки программирования**

16. **Многопоточные языки программирования**

С#, Java (c 1995 г.), .NET (с 2000 г.)

17. **Модели многопоточности (перечислить)**

Существуют различные модели многопоточности – способы отображения пользовательских потоков в потоки ядра. Теоретически возможны (и на практике реализованы) следующие модели многопоточности:

- *Модель много / один* (many-to-one) – отображение нескольких пользовательских потоков в один и тот же поток ядра. Используется в операционных системах, не поддерживающих множественные системные потоки (например, с целью экономии памяти). Данная модель изображена на [рис. 10.2](http://www.intuit.ru/studies/courses/641/497/lecture/11284?page=2#image.10.2).

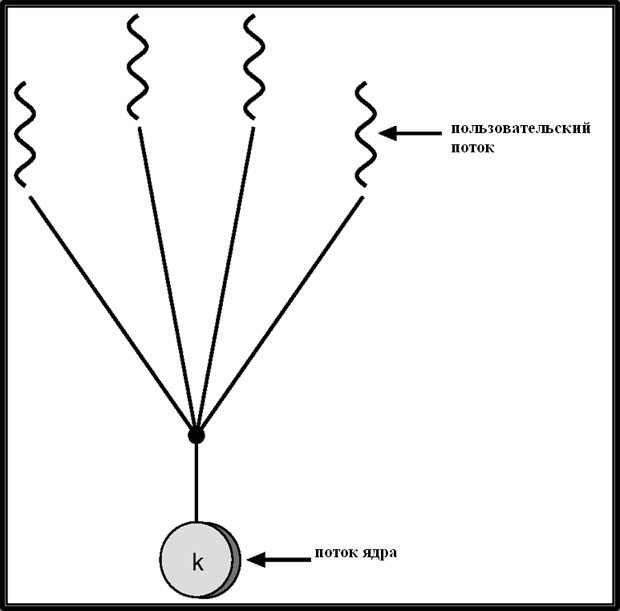


Рис. 10.2. Схема модели многопоточности "много / один".

*- Модель один / один* (one-to-one) – взаимно-однозначное отображение каждого пользовательского потока в определенныйпоток ядра. Примеры ОС, использующих данную модель, - Windows 95/98/NT/2000/XP/2003/2008/7; OS/2. Данная модель изображена на [рис. 10.3](http://www.intuit.ru/studies/courses/641/497/lecture/11284?page=2#image.10.3).

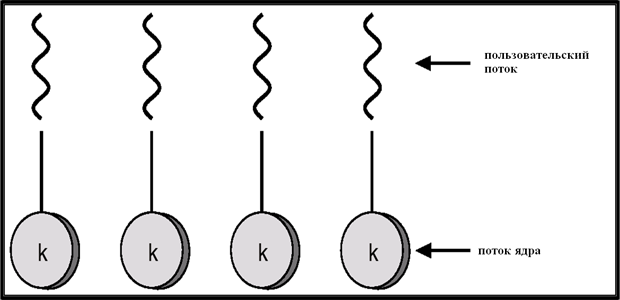


Рис. 10.3. Схема модели многопоточности "один / один".

*- Модель много / много* (many-to-many) – модель, допускающая отображение нескольких пользовательских потоков в несколько системных потоков. Такая модель позволяет ОС создавать большое число системных потоков. Характерным примером ОС, использующей подобную модель, является ОС Solaris, а также Windows NT / 2000 / XP / 2003 / 2008 / 7 с пакетомThreadFiber. Данная модель изображена на [рис. 10.4](http://www.intuit.ru/studies/courses/641/497/lecture/11284?page=2#image.10.4).

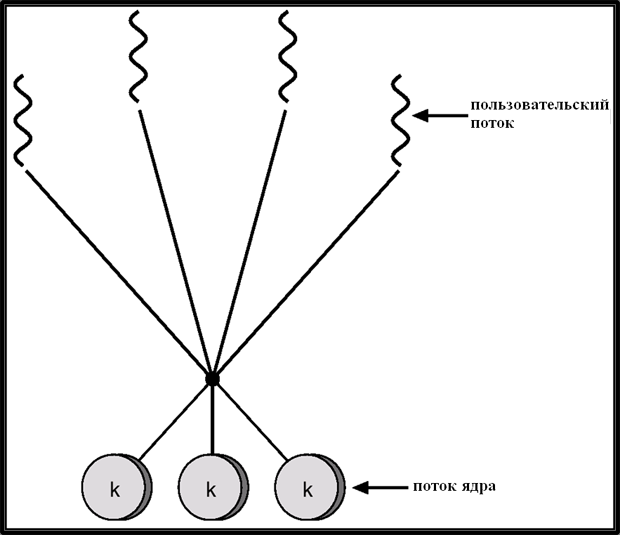


Рис. 10.4. Схема модели многопоточности "много / много".