1. **Особенности ОС с экзоядром. Достоинства и недостатки.**

Вместо клонирования настоящей машины, как это делается в виртуальных машинах, существует иная стратегия, которая заключается в их разделении, иными словами, в предоставлении каждому пользователю подмножества ресурсов. При этом одна виртуальная машина может получить дисковые блоки от 0 до 1023, другая может получить блоки от 1024 до 2047, и т. д. Самый нижний уровень, работающий в режиме ядра, — это программа под названием экзоядро. Его задача состоит в распределении ресурсов между виртуальными машинами и затем в отслеживании попыток их использования, чтобы ни одна из машин не пыталась использовать чужие ресурсы. Каждая виртуальная машина может запускать свою собственную операционную систему, как на VM/370 и на Pentium в режиме виртуальных машин 8086, с тем отличием, что каждая машина ограничена использованием тех ресурсов, которые она запросила и которые были ей предоставлены. Преимущество схемы экзоядра заключается в том, что она исключает уровень отображения. При других методах работы каждая виртуальная машина считает, что она имеет свой собственный диск с нумерацией блоков от 0 до некоторого максимума. Поэтому монитор виртуальных машин должен вести таблицы преобразования адресов на диске (и всех других ресурсов). При использовании экзоядра необходимость в таком переназначении отпадает. Экзоядру нужно лишь отслеживать, какой виртуальной машине какие ресурсы были переданы. Такой подход имеет еще одно преимущество: он отделяет многозадачность (в экзоядре) от пользовательской операционной системы (в пространстве пользователя) с меньшими затратами, так как для этого ему необходимо всего лишь не допускать вмешательства одной виртуальной машины в работу другой.

1. **Состояния процесса. Характеристика состояний.**

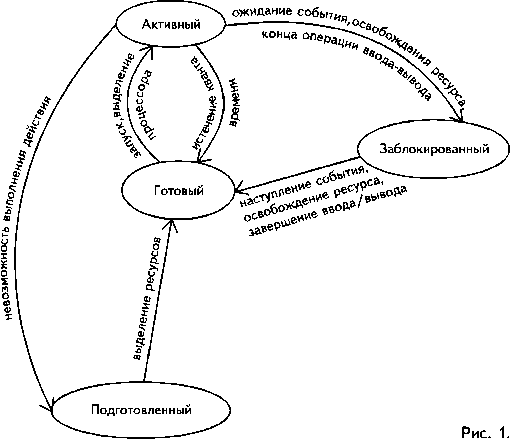
Во время своего существования процесс может находиться в четырех состояниях: ***подготовленном, готовом, активном и заблокированном*** (рис.2.10). В каждый конкретный момент времени процесс находится в одном из этих состояний, которое фиксируется в блоке управления процессом (PCB).

Процесс находится в **подготовленном** состоянии, если он находится в системе (как правило на внешнем носителе информации), но ему не выделены ресурсы системы.

Процесс находится в **готовом** состоянии или активизирован, если ему уже выделены ресурсы (программа, необходимая для выполнения, находится в оперативной памяти), однако ему не выделено время процессора для выполнения (процессор занят другим процессом).

Процесс, которому выделяется время процессора, переводиться в **активное** состояние или состояние выполнения.

Во время выполнения процесса ему могут понадобиться дополнительные ресурсы, но для их получения необходимо некоторое время. Т.е. процесс должен ожидать наступления некоторого события или окончания выполнения конкретной операции (например, ввода/вывода для получения/выдачи данных). Такой процесс переводится в **заблокированное** состояние.



Готов – стоит в очереди – готов для обработки( у меня все сеть)

Активизирован – активизирован процессором.

Системные задачи формируются и активизируются при загрузке системы.

Подготовленный процесс – еще не задача, не готова к выполнению, но имеет ресурсы(не все)

Если процесс нашей задачи пытается активизировать действия, связанные с О.С., то задача входит в заблокированное состояние.

Заблокирован до тех пор, пока системная оп-ция, которая была активизирована, не выполнится и пришлет сообщение о выполнении (событие)

Если во время выполнения задачи оказалось, что задачу не возможно продолжать выполнять из-за отсутствия ресурса, то она приостанавливается и переходит в «режим ядра». Если ресурсы найдены, то задача переходит в состояние заблокированной .

Нельзя сразу перейти из активизированного в заблокированный «режим ядра» - это шлюз.

Если система считает, что ко-во активизированных процессов у нее слишком большее, она может выбрать некоторые процессы по своему усмотрению. Решает она по показателю эффективности работы системы.

1. **Последовательность операция при загрузке активного процесса в память с виртуальной организацией.**

**?????????????????????**

1. **Обход тупиков**

Алгоритм "банкира".

При использовании алгоритма "банкира" подразумевается, что :

1. системе заранее известно количество имеющихся ресурсов;
2. система знает, сколько ресурсов может максимально потребоваться процессу;
3. число пользователей (процессов), обращающихся к ресурсам, известно и постоянно;
4. система знает, сколько ресурсов занято в данный момент времени этими процессами (в общем и каждым из них);
5. процесс может потребовать ресурсов не больше, чем имеется в системе.

В алгоритме "банкира" введено понятие надежного состояния. Текущее состояние вычислительной машины называется **надежным**, если ОС может обеспечить всем текущим пользователям (процессам) завершение их заданий в течение конечного времени. Иначе состояние считается **ненадежным**. Надежное состояние ⎯ это состояние, при котором общая ситуация с ресурсами такова, что все пользователи имеют возможность со временем завершить свою работу. Ненадежное состояние может со временем привести к тупику.

Система удовлетворяет только те запросы, при которых ее состояние остается надежным, т.е. при запросе ресурса система определяет сможет ли она завершить хотя бы один процесс, после этого выделения.

Пример решения задачи выделения ресурсов по алгоритму "банкира".

Имеем 6 ресурсов и 6 процессов (рис .2.17.). В таблице показано состояние занятости ресурсов на данный момент (1 свободный ресурс).



Рис. 2.17

Эта таблица отражает надежное состояние, т.к. существует такая последовательность выделений ⎯ освобождений ресурсов, при которой все процессы за конечное время будут завершены.

Недостатки алгоритма банкира:

1. Если количество ресурсов большое, то становится достаточно сложно вычислить надежное состояние.
2. Достаточно сложно спланировать, какое количество ресурсов может понадобиться системе.
3. Число работающих пользователей (процессов) остается постоянным.
4. Пользователи должны заранее указывать максимальную потребность в ресурсах, однако, потребность может определяться динамически по мере выполнения процесса.
5. Пользователь всегда заказывает ресурсов больше, чем ему может потребоваться.
6. Алгоритм требует, чтобы процессы возвращали выделенные им ресурсы в течение некоторого конечного времени. Однако, для систем реального времени требуются более конкретные гарантии. Т.е. в системе должно быть задано максимальное время захвата всех ресурсов процессом (через это время ресурсы должны быть освобождены).

**Графический метод** обхода тупика (рис. 2.18).

Имеются 2 процесса P1 и P2 и каждому из них для выполнения требуются по 2 ресурса: Д ⎯ диск , П ⎯ процессор. Если P1 захватил процессор , а P2 ⎯ диск, то возникает тупиковая ситуация. Однако, если P1 захватил П и Д и завершил свое выполнение, то P2 захватывает эти же ресурсы и заканчивает выполнение ⎯ беступиковое взаимодействие.

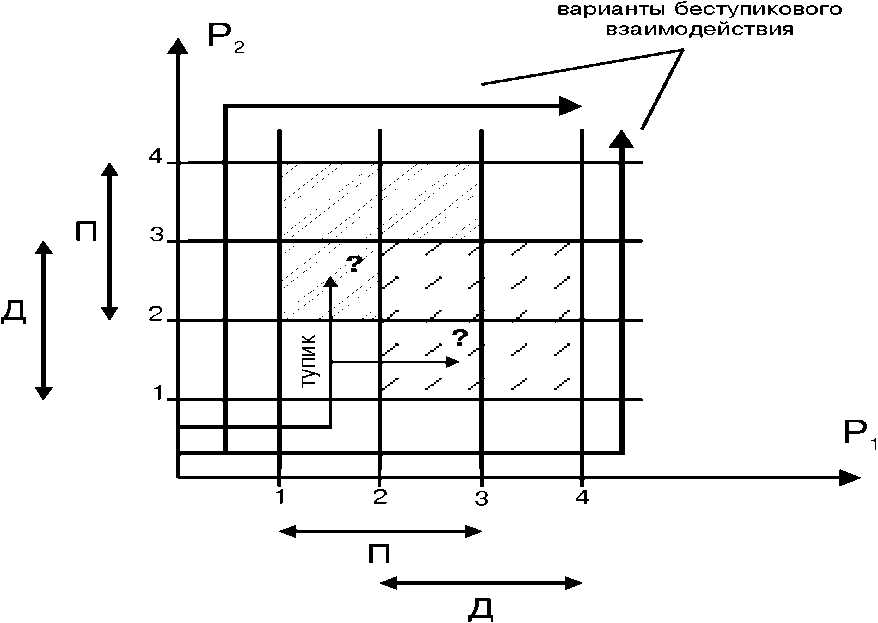


Рис. 2.18

1. **Виды сигналов**

В системе реализован механизм управление сигналами.

1-я фаза: генерация, отправка

2-я фаза: доставка и обработка.

Между фазами время может быть очень большое (при делении на «0» нужно быстро обрабатывать)

Виды сигналов:

- зависит от того, кто и какой процесс его гегегегенерировал, гиги, и кто является получателем.

- особые ситуации (деление на «0», превышение мантиссы).

- терминальное прерывание, источник-пользователь с кривыми руками.

- другой процесс;

- системы управления зданиями (фоновыми и текущими задачами ).

- сигналы квоты (прерывание квоты или приближение к ней).

- сигнал – уведомление (сигнал о поступлении в систему).

- сигнал Alarm (00 счетчику таймера)