БИЛЕТ № 7

1) Схема функционирования ОС.

Во время загрузки – грузятся только программы необходимые 4/ быстрой реакции системы.

Загрузка sys (активизированы системные задачи и ядро в памяти) Супервизор – совокупность управляющих программ, поддерживающих ядро системы

Табличный метод управления – Состояние ВС -> Таблицы -> Решения.

Ввод система понимает на уровне языка системного ввода. Определение правомерности задачи.U/ библиотеку языка управления заданием. ЕСЛИ задание неверное ТО оно via главный планировщик направляется на(в)зад usaru.

Задание во второй очереди начинают обрабатываться при наличии в системе ресурсов. Система загружает планировщик по транзитной схеме, который обрабатывает входную очередь, для выборки наиболее приоритетного задания для загрузки.

При работе программ системного ввода u/ режим спуллинга – согласование скоростей на входе и выходе.

Инициатор – фиксирует блок управления задачей. Как только есть блок – Процесс готов – ему надо только время проца.

3 уровень – выделяет задачам время прца.
-0зПрога системного вывода работает в режиме спуллинга, ЕСЛИ задания выдают ин-фу на ус-ва системного вывода то создается выходная очередь заданий.

2) PCB.

Для того чтобы операционная система могла выполнять *операции над процессами*, каждый *процесс* представляется в ней некоторой структурой данных. Эта структура содержит информацию, специфическую для данного *процесса*:

- *состояние*, в котором находится *процесс*;
- программный счетчик *процесса* или, другими словами, адрес команды, которая должна быть выполнена для него следующей;
- содержимое регистров процессора;
- данные, необходимые для планирования использования процессора и управления памятью (приоритет *процесса*, размер и расположение адресного пространства и т. д.);
- учетные данные (идентификационный номер *процесса*, какой пользователь инициировал его работу, общее время использования процессора данным *процессом* и т. д.);
- сведения об устройствах ввода-вывода, связанных с *процессом* (например, какие устройства закреплены за *процессом*, таблицу открытых файлов).

Ее состав и строение зависят, конечно, от конкретной операционной системы. Во многих операционных системах информация, характеризующая *процесс*, хранится не в одной, а в нескольких связанных структурах данных. Эти структуры могут иметь различные наименования, содержать дополнительную информацию или, наоборот, лишь часть описанной информации. Для нас это не имеет значения. Для нас важно лишь то, что для любого *процесса*, находящегося в вычислительной системе, вся информация, необходимая для совершения *операций* над ним, доступна операционной системе. Для простоты изложения будем считать, что она хранится в одной структуре данных. Мы будем называть ее *PCB* (Process Control Block) или *блоком управления процессом*. ***Блок управления процессом*** является моделью *процесса* для операционной системы. Любая *операция*, производимая операционной системой над *процессом*, вызывает определенные изменения в *PCB*. В рамках принятой модели *состояний процессов* содержимое *PCB* между *операциями* остается постоянным.

Информацию, для хранения которой предназначен *блок управления процессом*, удобно для дальнейшего изложения разделить на две части. Содержимое всех регистров процессора (включая значение программного счетчика) будем называть *регистровым* контекстом *процесса*, а все остальное – *системным контекстом процесса*. Знания *регистрового* и *системного контекстов процесса* достаточно для того, чтобы управлять его работой в операционной системе, совершая над ним *операции*. Однако этого недостаточно для того, чтобы полностью охарактеризовать *процесс*. Операционную систему не интересует, какими именно вычислениями занимается *процесс*, т. е. какой код и какие данные находятся в его адресном пространстве. С точки зрения пользователя, наоборот, наибольший интерес представляет содержимое адресного пространства *процесса*, возможно, наряду с *регистровым контекстом* определяющее последовательность преобразования данных и полученные результаты. Код и данные, находящиеся в адресном пространстве *процесса*, будем называть его *пользовательским контекстом*. Совокупность *регистрового*, *системного* и *пользовательского контекстов процесса* для краткости принято называть просто *контекстом процесса*. В любой момент времени *процесс* полностью характеризуется своим *контекстом*.

PCB — это системная структура данных, содержащая определенные сведения о процессе и имеющая следующие поля.

1. Уникальный и идентификатор процесса (имя),

2. Текущее состояние процесса.

3. Приоритет процесса.

4. Указатели участка памяти выделяемого программе, подчиненной данному процессу.

5. Указатели выделенных ему ресурсов.

6. Область сохранения регистров

7. Права процесса (список разрешенных операций)

8. Связи зависимости в иерархии процессов (список дочерних процессов, имя родительского процесса)
9. Пусковой адрес программы, подчиненной данному процессу

доп. Инфа
Управление процессами

Выполнение функций ОС, связанных с управлением процессами, осуществляется с помощью специальных структур данных, образующих окружение процесса, среду исполнения или образ процесса. Образ процесса состоит из двух частей: данных режима задачи и режима ядра. Образ процесса в режиме задачи состоит из сегмента кода программы, которая подчинена процессу, данных, стека, библиотек и других структур данных, к которым он может получить непосредственный доступ. Образ процесса в режиме ядра состоит из структур данных, недоступных процессу в режиме задачи, которые используются ядром для управления процессом. Оно содержит различную вспомогательную информацию, необходимую ядру во время работы процесса.

Каждому процессу в ядре операционной системы соответствует блок управления процессом (PCB – process control block). Вход в процесс (фиксация системой процесса) ? это создание его блока управления (PCB), а выход из процесса ? это его уничтожение, т. е. уничтожение его блока управления.

Таким образом, для каждой активизированной задачи система создает свой PCB, в котором, в сжатом виде, содержится используемая при управлении информация о процессе. PCB ? это системная структура данных, содержащая определенные сведения о процессе со следующими полями:

1. Идентификатор процесса (имя);

2. Идентификатор родительского процесса;

3. Текущее состояние процесса (выполнение, приостановлен, сон и т.д.);

4. Приоритет процесса;

5. Флаги, определяющие дополнительную информацию о состоянии процесса;

6. Список сигналов, ожидающих доставки;

7. Список областей памяти выделенной программе, подчиненной данному процессу;

8. Указатели на описание выделенных ему ресурсов;

9. Область сохранения регистров;

10. Права процесса (список разрешенных операций);

11. Пусковой адрес программы, подчиненной данному процессу.

Данные структур PCB для всех процессов, в любой момент времени, должны присутствовать в памяти, хотя остальные структуры данных, включая образ процесса, могут быть перемещены во вторичную память, — область свопинга. Это позволяет ядру иметь под рукой минимальную и достаточную информацию, необходимую для определения местонахождения остальных данных, относящихся к процессу, даже если они отсутствуют в памяти. Структура PCB является отдельной записью в системной таблице процессов. Запись этой таблицы для выполняющегося в настоящий момент времени процесса адресуется содержимым регистра TR.

Когда ОС переключает процессор с процесса на процесс, она использует области сохранения регистров в PCB для запоминания информации, необходимой для рестарта (повторного запуска) каждого процесса с точки прерывания, когда он в следующий раз получит в свое распоряжение процессор. Количество процессов в системе ограничено и определяется самой системой или пользователем во время генерации ОС. Неудачное определение количества одновременно исполняемых программ может привести к снижению полезной эффективности работы системы, т.к. переключение процессов требует выполнения дополнительных операций по сохранению и восстановлению состояния процессов. Блоки управления системных процессов создаются при загрузке системы. Это необходимо, чтобы система выполняла свои функции достаточно быстро, а время реакции ОС было минимальным. Однако, количество блоков управления системными процессами меньше, чем количество самих системных процессов. Это связано с тем, что структура ОС имеет либо оверлейную, либо динамически ? последовательную или параллельные структуры иерархического типа, и нет необходимости создавать отдельные PCB для процессов, которые никогда не будут активизироваться одновременно. При такой организации легко учитывать приоритеты системных процессов, выстроив их по приоритетам заранее при инициализации системы. Блоки управления проблемными (пользовательскими) процессами создаются в процессе активизации процессов динамически, а их приоритеты могут изменяться во время жизненного пути процессов. Все PCB находятся в выделенной системной области памяти.

В каждом PCB есть поле состояния процесса. Все блоки управления системными процессами располагаются в порядке убывания приоритетов и находятся в системной области памяти. Если приоритеты системных блоков можно определить заранее, то для проблемных процессов необходима таблица приоритетов проблемных программ. Каждый блок PCB имеет стандартную структуру, фиксированный размер, точку входа и содержит, как правило, указанную выше информацию и дополнительную информацию для синхронизации процессов. Для синхронизации в PCB используются поля:

1. поля для организации связи (два поля):

1.1. Имя вызванного процесса.

1.2. Имя вызвавшего процесса.

2. поля для организации ожидания (два поля).

2.1. Поле ожидания. Имя процесса, освобождения которого ожидает данный.

2.2. Счетчик ожидания. Число процессов, ожидающих данный.

В поле организации связи у порожденного процесса указывается идентификатор родительского процесса и код возврата, передаваемый родительскому процессу при завершении своих действий потомком. Родительский процесс получает от порожденного процесса свой идентификатор.

В полях организации ожидания, указывается адрес PCB вызываемого процесса, если вызываемый процесс занят. В соответствующем поле занятого процесса записывается число процессов, его ожидающих.

Если процесс А пытается вызвать процесс В, а у процесса В в PCB занята цепочка связей, то есть он является занятым по отношению к другим процессам, тогда адрес процесса В записывается в поле ожидания PCB процесса А, а к содержимому поля счетчика ожидания PCB процесса В добавляется 1. Как только процесс В завершает выполнение своих функций, он передает управление вызывающему процессу следующим образом: В проверяет состояние своего счетчика ожидания, и, если счетчик больше 0, то среди PCB других процессов ищется первый (по приоритету или другим признакам) процесс, в

поле ожидания PCB которого стоит имя ожидаемого процесса, в данном случае B, тогда управление передается этому процессу.

В современных ОС для синхронизации доступа нескольких процессов к разделяемым ресурсам используются семафоры. Они выполняют функции разрешения или запрещения процессу использование того или иного ресурса. Допустим, имеется разделяемый ресурс (файл, программа, разделяемая память, общие переменные…). Необходимо блокировать доступ к ресурсу для других процессов, когда некий процесс производит операцию над ресурсом (например, записывает в файл). Для этого свяжем с данным ресурсом некую целочисленную величину — счетчик (семафор), доступный для всех процессов. Примем, что значение 1 семафора означает доступность ресурса, 0 – его недоступность. Если оно равно 0 — ресурс занят, и операция недопустима, процессу необходимо ждать. Он переходит в режим ожидания или сна (заблокирован), что отражается в поле состояния процесса в PCB. В системную таблицу процессов при этом заносится соответствующий флаг состояния и определяется событие, пробуждающего процесс. События возвещают о доступности того или иного ресурса. Наступление этого события (освобождения ресурса) может ожидать несколько процессов. Поскольку переход из одного состояния в другое действие скорее логическое, то и пробуждаются эти процессы одновременно. Однако это не означает, что один из них сразу начнет выполняться. Это приведет к тому, что их состояние измениться от «сна» к «готов к выполнению», и они могут конкурировать с другими процессами, находящимися в аналогичном состоянии за захват времени процессора, т.е. перехода в состояние «выполнение». Если оно равно 1 – можно работать с ресурсом. Для этого, прежде всего, необходимо заблокировать ресурс, т.е. изменить значение семафора на 0. Для нормальной работы необходимо обеспечить выполнение следующих условий:

1. Значение семафора должно быть доступно различным процессам. Поэтому семафор находится в адресном пространстве ядра.

2. Операция проверки и изменения значения семафора должна быть реализована в виде одной атомарной по отношению к другим процессам операции. Это исключает ситуацию, когда после проверки значения семафора выполнение процесса будет прервано другим процессом, который в свою очередь проверит семафор и изменит его состояние. Для этого все операции над семафорами должны выполняться в режиме ядра.

Таким образом, семафоры являются системным ресурсом, действия над которыми производятся через интерфейс системных вызовов.

Для каждой группы семафоров ядро ОС поддерживает специальную структуру данных. Например, для каждого семафора System V в этой структуре имеются следующие поля:

1. Значение семафора.

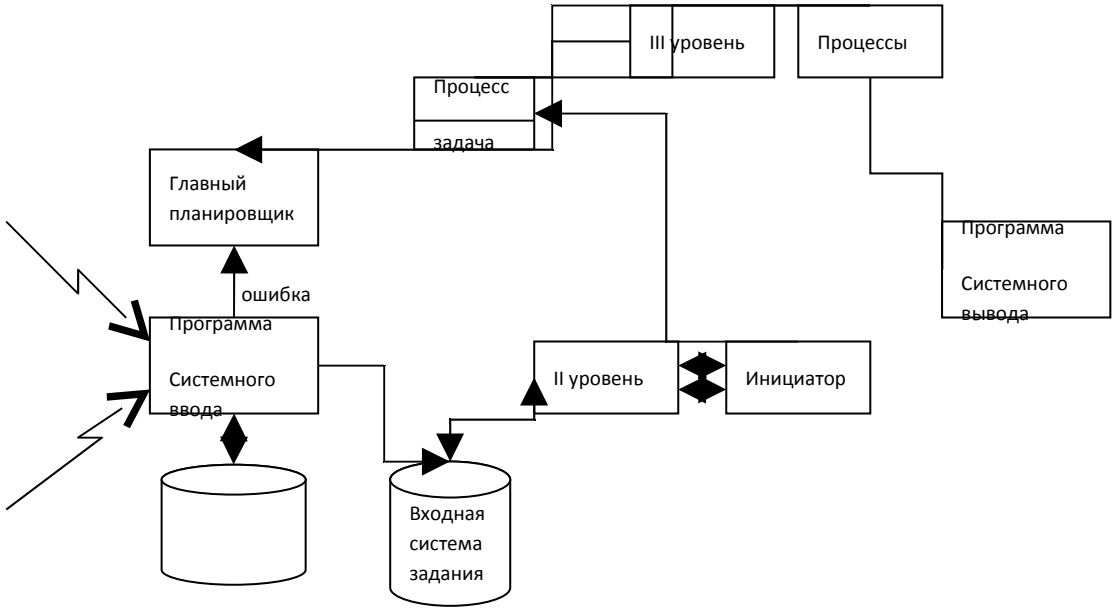
2. Идентификатор процесса, выполнившего последнюю операцию над семафором.

3. Число процессов, ожидающих доступность ресурса ассоциированного с семафором, т.е. ожидающих изменения значения семафора.

Изменение значения семафора с «0» на «1» при наличии процессов, ожидающих доступность ресурса, приводит к выполнению системной операции перевода их в состояние «готовности». Задачу выбора процесса для запуска затем решает планировщик процессов.

3) Процедура ввода задания в ВС. Участие системных программ. Особенности реализации в разных ОС.

Слежение за прохождением заданий от входа до выхода на всех этапах его выполнения



Программа системного ввода – планировщик 1 уровня
Командный интерпритатор берет входное задание
В том случае, если сис+т*-ема считает, что есть ресурс для активизации задания, то активизируется планировщик 2-го уровня, то он выбирает задание с наивысшим приритетом. В том случае, если система считает, что есть ресурс для активизации задания, то активизируется планировщик. Инициатор проверяет наличие ресурсов для выполнения задания. Если ресурсов нет, то задание сбрасывается. Если всё в порядке, то образуется задача или процесс. Как только сформирован ТСВ система должна его обязательно выполнить. При формировании РСВ определяется программа подчиненная задаче, и данные, которые должна выполнить программа. Как только освобождается процессор, всплывает планировщик 3-го уровня, который обрабатывает ТСВ или PSB, ищет наиболее приоритетный процесс, который можно запустить

Система управления заданиями управляет прохождением заданий в ВС и выполняет функции:

1.Предоставление языковых средств управления работами в вычислительной системе.

2.Ввод и интерпретация заданий/команд.

3.Выделение и освобождение необходимых ресурсов.

4.Планирование заданий на выполнение.

5.Сбор и предоставление информации о состоянии заданий.

Существует три основных уровня планирования:

1.Планирование на верхнем уровне или планирование заданий.

На этом уровне осуществляется выбор заданий пользователем для выполнения и их запуск. Выбранные задания становятся готовыми процессами. Эту работу выполняет системный компонент - планировщик заданий.

2.Планирование на нижнем уровне или диспетчирование процессов.

Здесь осуществляется выбор готового процесса для выполнения, то есть предоставления ему ЦП. Выбранный процесс становится активным. Эту работу выполняет системный компонент - диспетчер.

3.Планирование на промежуточном уровне.

На данном уровне определяется, каким процессам будет разрешено состязаться за захват ЦП, то есть быть готовыми, и какие процессы будут кратковременно приостановлены (задержаны) для оптимизации загрузки системы. Промежуточное планирование управляет текущей производительностью вычислительной системы.

Эффективное планирование заданий и процессов является сложной проблемой, поскольку должно учитываться много противоречивых требований, таких как:

- справедливость;

- максимальная пропускная способность;

- приемлемое время ответа для максимального числа интерактивных пользователей;

- предсказуемость (задание должно выполняться примерно за одно время независимо от загрузки вычислительной системы);

- минимум накладных расходов на выполнение планирования;

- сбалансированность использования ресурсов;

- исключение бесконечного откладывания;

- учет приоритетов;

- отдавать предпочтение процессам, занимающие ключевые ресурсы;

- плавно деградировать при увеличении нагрузок.

Для того чтобы реализовать перечисленные требования, механизм планирования должен знать и учитывать следующие факторы:

- является ли процесс обменным (активно использующим операции ввода/вывода) или вычислительным (активно использующим процессор);

-является ли процесс пакетным или диалоговым;

- уровень реактивности интерактивного процесса;

-приоритетность процесса;

- частота прерываний по отсутствию нужной страницы;

- частота прерывания с низкого приоритета на высокий;

- длительность периода ожиданий ЦП процесса;

- суммарное использование времени ЦП и оценочное время, необходимое для завершения.

Для детального выделения действий, выполняемых при планировании, рассмотрим схему прохождения работ через ВС. При этом будем отслеживать прохождение задания с момента его ввода до полного завершения (вывода), фиксируя моменты изменения формы представления задания в ВС (Mi) и интервалы времени преобразования задания из одного состояния в другое (рис 4.2.)

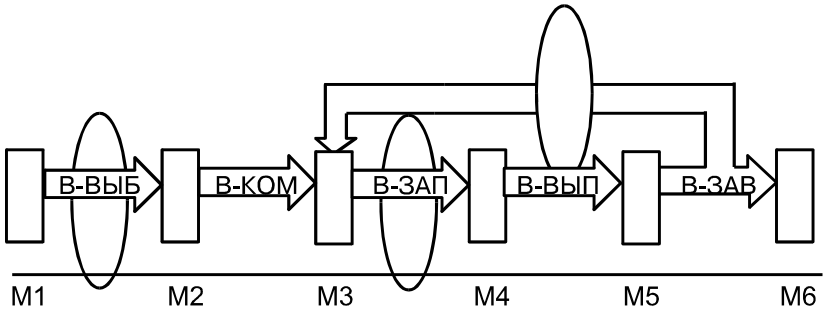


Рис. 4.2

Где:

- М1 — момент запроса: момент фиксации системой заявки/заявок на обслуживание;
- В-ВЫБ — время выборки: время предварительной обработки входного потока заявок;
- М2 — момент выборки: момент принятия решения об активизации заявок или отсрочке их выполнения;
- В-КОМ — время компиляции: время подготовки исходного описания задания-заявки и преобразования его в форму, необходимую для выполнения в параллельной системе;
- М3 — момент компоновки: момент, когда компиляция закончена, и входные задания переводятся в ранг "процессов", для которых ВС должна выделить ресурсы;
- В-ЗАП — время запуска: время, в течение которого выполняются действия, требуемые для запуска (инициирования) готовых процессов, — выделение ресурсов и перевод процессов в активное состояние;
- М4 — момент запуска: момент инициирования (активизации) задачи-процесса после распределения ресурсов (устройств, данных, памяти), который выполняется во время В-ЗАП.
- В-ВЫП — время выполнения: период времени, когда задания в системе активны, то есть выполняются.
- М5 — момент завершения: момент фиксации завершения задания (естественного или аварийного);
- В-ЗАВ — время завершения: время, в течение которого диспетчер ОС определяет, выполнено ли задание или его нужно продолжать выполнять. В последнем случае он ставит задание обратно в очередь на выполнение;
- М6 — момент выхода: момент, когда ОС считает задание полностью выполненным или устанавливает, что система не имеет возможности его дальнейшего выполнения, тогда оно удаляется из системы;
- Эллипс означает выполнение действий планировщика системы.
- заявок;
- требуется новый тип планировщика (назовем его ПТ — планировщик транспортный, рис. 4.17) для распараллеливания заданий, синхронизации процессов по данным — обеспечения поступления требуемых данных и поддержки связей между вычислительными узлами при реализации связи по данным;
- обработанные задания (или их распараллеленные модули) транслятором ОС и ПТ ставятся в новую очередь — буфер БУФ;
- к ПП добавляется функция *адаптирования*, при выполнении которой задания распределяются соответственно особенностям данной системы (например: специальные схемы для систем гиперкуб, транспьютерных систем или дополнительная схема для неоднородной среды).
- к ПН добавляется функция балансирования в случае реконфигурации (отказа некоторых элементов) системы.

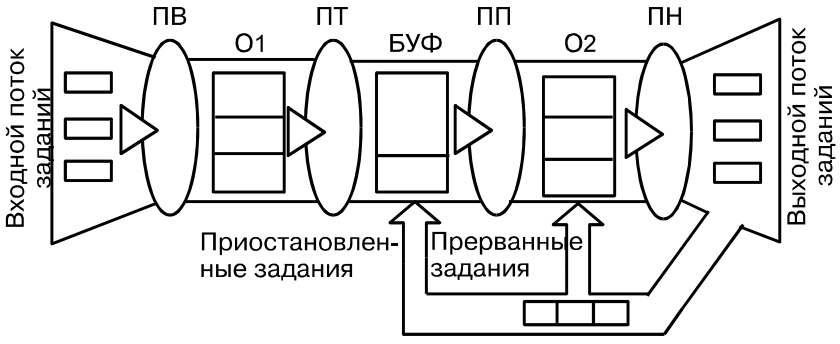


Рис. 4.17. Схема прохождения заданий через ВС

Полную систему планирования в этом случае можно представить в виде семиуровневой модели (табл. 4.1), где каждый уровень часто рассматривают как отдельную задачу. Порядок выполнения уровней может быть изменен в зависимости от особенности системы и целей задачи планирования.

Табл. 4.1

ПВ	1	Предварительное входное планирование исходного потока заявок, претендующих на захват ресурсов вычислительной системы	Задача Ввода
	2	Структурный анализ взаимосвязи входного потока заявок по ресурсам и определение общих ресурсов	Задача анализа
ПТ	3	Структурный анализ заявок и определение возможности распараллеливания каждой работы	Задача распараллеливания
ПП	4	Адаптирование распределения работ соответственно особенностям вычислительной системы	Задача адаптирования
	5	Составление расписания выполнения взаимосвязанных процедур: оптимизация плана по времени решения, количеству используемых ресурсов и количеству пересылок	Задача оптимизации
	6	Планирование потока процессов, претендующих на захват времени процессора/процессоров вычислительной системы	Задача распределения
ПН	7	Выделение времени процессоров ВС активизированным процессам, перераспределение (реконфигурация) работ в вычислительной среде (отказ оборудования)	Задача распределения и перераспре-деления

4) Вход и выход из режима ожидания.

Ответ:

Доп.инфа:

Операции над процессами

Над процессами можно производить следующие операции: создание, уничтожение, восстановление, изменение приоритета, блокирование, активизация (пробуждение), запуск (выбор), приостановка.

1. Создание процесса включает в себя:

1. присвоение имени процессу,

2. включение этого имени в список имен процессов, известных системе,

3. определение начального приоритета,

4. формирование блока управления процессом,

5. выделение начальных ресурсов.

2. Уничтожение процесса означает его удаление из системы. Ресурсы, выделенные этому процессу, возвращаются системе, имя в системных списках стирается, блок управления процессом освобождается.

3. Изменение приоритета означает модификацию значения приоритета в блоке управления данным процессом. Операционная система увеличивает приоритет у процессов, когда предполагается, что они будут находиться в состоянии бесконечного откладывания или уменьшает приоритет процессу, надолго захватившему процессор.

4. Запуск (выбор) процесса, осуществляемый планировщиком, который выделяет процессу время процессора.

5. Приостановленный процесс не может продолжить свое выполнение до тех пор, пока его не активизирует любой другой процесс (кратковременное исключение определенных процессов в периоды пиковой нагрузки). В случае длительной приостановки процесса, его ресурсы должны быть освобождены. Решение об освобождении ресурса зависит от его природы. Основная память должна освобождаться немедленно, а вот принтер может и не быть освобожден. Приостановка процесса обычно производится в следующих случаях:

1. если в системе возникло прерывание от схем контроля (например, произошло переключение на бесперебойный блок питания), то текущие процессы надо приостановить, исправить причину неисправности и возникновения прерывания, а затем активизировать приостановленные процессы;

2. если промежуточные результаты работы процесса вызвали сомнение, то нужно его приостановить, найти ошибку и запустить либо сначала, либо с контрольной точки;

3. если ВС перегружена (в системе активизировано слишком много процессов), что вызвало снижение ее эффективности;

4. если для продолжения нормального выполнения процессу необходимы ресурсы, которые ВС не может ему предоставить.

Инициатором приостановки может быть либо сам процесс, либо другой процесс. В однопроцессорной ВС при однопрограммном режиме работы выполняющийся процесс может приостановить сам себя или быть принудительно остановлен с пульта управления без возможности восстановления, т.к. ни один другой процесс не может выполняться одновременно с ним. В мультипроцессорной системе или при многопрограммном режиме работы любой выполняющийся процесс может быть приостановлен другим процессом (системным, родительским или относящимся к одному и тому же пользователю).

6. Восстановление или активизация процесса — операция подготовки процесса к повторному запуску выполняемая операционной системой, с той точки, в которой он был приостановлен.

5) Блокировка записи.

БОТАНА КОНСПЕКТ

Блокировка записи – при записи в файл запись идёт не в файл, а в буфер. И записывается в буфер до тех пор, пока буфер не будет заполнен. После заполнения буфера идёт сброс на диск. Буфер сбрасывается по закрытию файла.

При реализации блокировки\разблокировки возникает опасность рассогласования файловой системы. Если система обнаружила, что ФС рассогласована подключается программное согласование ФС. Для этого эта программа заполняет два вектора: один векторзаполняется по занятости кластеров в сообтветствии с каталогами.

1010111 есть ссылки\нет ссылок

0101000 занят свободен

Несогласование тогда, когда есть 2 нуля или 2 единицы.

2 нудя – нет ссылки, а он занят, т. Е. потерян кластер.

2 един. – есть ссылка, а он свободен – система автоматич считает такой занятым.

В Юникс системах исп. Не бит карта а счётчик.