1. Назначение - Система управления данными

Файловая система - это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами.

В широком смысле понятие "файловая система" включает:
совокупность всех файлов на диске,
наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске,
комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

2. Оптимизация обработки потока запросов к файловой системе.

Принципы повышения эффективности управления данными заключается в оптимизации ввода вывода – то есть оптимизация обращения к диску(оптимизация времени поиска и времени передачи) и оптимизации размещения файлов.

Управления данными с точки зрения ОС – оптимизация размещения файлов.

FCFS- первый пришел, первый обслуживается

SSTF- поиск с наименьшим временем

SCAN-сканирующий алгоритм (головка двигается внутри ,по дороге обслуживает все запросы)

CSCAN- запросы обрабатываются от внешней дорожки к внутренней до конца,потом перескок на внешнюю

дорожку

N-Step_Scan- точно так же, как предидущий, но после начала движения все запросы ставятся в отдельную

очередь

SLTF- с наименьшим временем ожидания первого(обрабатываются запросы под головкой в рамках всего

цилиндра)

Схема Эшенбаха – пытается обработать только одну дорожку.

ТУТ ХЕРНЯ!!!
Mount Options

`noatime , nodiratime`

This is one of the quickest and easiest performance gains. This mount option tells the system not to update inode access times. This is a good option for web servers, news servers or other uses with high access file systems.

`commit`
This file system option controls how often the file system is told to sync data and meta data. The default value is 5 seconds, but you can extend this for a performance gain. The downside is that if your system loses power or crashes without writing out data, you could lose up that time value's worth of data. The values you set here are entirely up to you based on the performance of your system.

3. TSS Структура и назначение

TSS(task status segment)- это специальный системный сегмент (состояния задачи), (содержит состояние регистров процессора и регистра флагов).Когда задача(процесс) неактивна в нём хранится вся информация про значения всех РОНов и системных регистров.В TSS находится вся информация, необходимая для восстановления задачи после прерывания(переключение задач предусматривает сохранение TSS для старой задачи в соответствующие регистры). Место расположение TSS для каждой задачи это GDT.

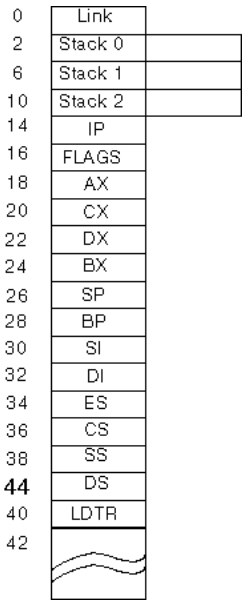

Рис. 14. Формат сегмента состояния задачи TSS.

Сегмент TSS адресуется процессором при помощи 16-битного регистра TR (Task Register), содержащего селектор дескриптора TSS, находящегося в глобальной таблице дескрипторов GDT (рис. 15).

Рис. 15. Дескриптор сегмента состояния задачи TSS.

Поле доступа содержит бит B - бит занятости. Если задача активна, этот бит устанавливается процессором в 1.

Операционная система для каждой задачи создаёт свой TSS. Перед тем как переключиться на выполнение новой задачи, процессор сохраняет контекст старой задачи в её сегменте TSS.

Что же конкретно записывается в TSS при переключении задачи?

Записывается содержимое регистров общего назначения AX, BX, CX, DX, регистров SP, BP, SI, DI, сегментных регистров ES, CS, SS, DS, содержимое указателя команд IP и регистра флажков FLAGS. Кроме того, сохраняется содержимое регистра LDTR, определяющего локальное адресное пространство задачи.

Дополнительно при переключении задачи в область TSS со смещением 44 операционная система может записать любую информацию, которая относится к данной задаче. Эта область процессором не считывается и никак не модифицируется.

Поле Link представляет собой поле обратной связи и используется для организации вложенных вызовов задач. Это поле мы рассмотрим в следующем разделе.

Поля Stack 0, Stack 1, Stack 2 хранят логические адреса (селектор:смещение) отдельных для каждого кольца защиты стеков. Эти поля используются при межсегментных вызовах через вентили вызова.

Для обеспечения защиты данных процессор назначает отдельные стеки для каждого кольца защиты. Когда задача вызывает подпрограмму из другого кольца через вентиль вызова, процессор вначале загружает указатель стека SS:SP адресом нового стека, взятого из соответствующего поля TSS.

Затем в новый стек копируется содержимое регистров SS:SP задачи (т.е. адрес вершины старого стека задачи). После этого в новый стек копируются параметры, количество которых задано в вентиле вызова и адрес возврата.

Таким образом, при вызове привилегированного модуля через вентиль вызова менее привилегированная программа не может передать в стеке больше параметров, чем это определено операционной системой для данного модуля.

Включение адресов стеков в TSS позволяет разделить стеки задач и обеспечивает их автоматическое переключение при переключении задач.

4. Целостность файловой системы

Важный аспект надежной работы файловой системы - контроль ее целостности. В результате файловых операций блоки диска могут считываться в память, модифицироваться и затем записываться на диск. Причем многие файловые операции затрагивают сразу несколько объектов файловой системы. Например, копирование файла предполагает выделение ему блоков диска, формирование индексного узла, изменение содержимого каталога и т. д. В течение короткого периода времени между этими шагами информация в файловой системе оказывается несогласованной.

И если вследствие непредсказуемого останова системы на диске будут сохранены изменения только для части этих объектов (нарушена атомарность файловой операции), файловая система на диске может быть оставлена в несовместимом состоянии. В результате могут возникнуть нарушения логики работы с данными, например появиться "потерянные" блоки диска, которые не принадлежат ни одному файлу и в то же время помечены как занятые, или, наоборот, блоки, помеченные как свободные, но в то же время занятые (на них есть ссылка в индексном узле) или другие нарушения.

В современных ОС предусмотрены меры, которые позволяют свести к минимуму ущерб от порчи файловой системы и затем полностью или частично восстановить ее целостность.

1. В чем причина появления несогласованности файловой системы

Сбой в системе прежеде чем кэши будут сброшены на диск. Особенно важно если это служебная инфа(i-node, зап о свободном месте…)

2. Как решается задача согласованности данных.

Прогой(scandisk,fsck). Используется избыточность ФС – копии.

5. Взаимодействие контроллера прерываний с ОС

Для того чтобы процессор не дожидался состояния готовности устройства ввода-вывода в цикле, а мог выполнять в это время другую работу, необходимо, чтобы устройство само умело сигнализировать процессору о своей готовности. Технический механизм, который позволяет внешним устройствам оповещать процессор о завершении команды вывода или команды ввода, получил название механизма прерываний.

В простейшем случае для реализации механизма прерываний необходимо к имеющимся у нас шинам локальной магистрали добавить еще одну линию, соединяющую процессор и

устройства ввода-вывода – линию прерываний. По завершении выполнения операции внешнее устройство выставляет на эту линию специальный сигнал, по которому процессор после выполнения очередной команды (или после завершения очередной итерации при выполнении цепочечных команд, т. е. команд, повторяющихся циклически со сдвигом по памяти) изменяет свое поведение. Вместо выполнения очередной команды из потока команд он частично сохраняет содержимое своих регистров и переходит на выполнение программы обработки прерывания, расположенной по заранее оговоренному адресу. При наличии только одной линии прерываний процессор при выполнении этой программы должен опросить состояние всех устройств ввода-вывода, чтобы определить, от какого именно устройства пришло прерывание (polling прерываний!), выполнить необходимые действия (например, вывести в это устройство очередную порцию информации или перевести соответствующий процесс из состояния **ожидание** в состояние **готовность**) и сообщить устройству, что прерывание обработано (снять прерывание).

В большинстве современных компьютеров процессор стараются полностью освободить от необходимости опроса внешних устройств, в том числе и от определения с помощью опроса устройства, сгенерировавшего сигнал прерывания. Устройства сообщают о своей готовности процессору не напрямую, а через специальный контроллер прерываний, при этом для общения с процессором он может использовать не одну линию, а целую шину прерываний. Каждому устройству присваивается свой номер прерывания, который при возникновении прерывания контроллер прерывания заносит в свой регистр состояния и, возможно, после распознавания процессором сигнала прерывания и получения от него специального запроса выставляет на шину прерываний или шину данных для чтения процессором. Номер прерывания обычно служит индексом в специальной таблице прерываний, хранящейся по адресу, задаваемому при инициализации вычислительной системы, и содержащей адреса программ обработки прерываний – **векторы** прерываний. Для распределения устройств по номерам прерываний необходимо, чтобы от каждого устройства к контроллеру прерываний шла специальная линия, соответствующая одному номеру прерывания. При наличии множества устройств такое подключение становится невозможным, и на один проводник (один номер прерывания) подключается несколько устройств. В этом случае процессор при обработке прерывания все равно вынужден заниматься опросом устройств для определения устройства, выдавшего прерывание, но в существенно меньшем объеме. Обычно при установке в систему нового устройства ввода-вывода требуется аппаратно или программно определить, каким будет номер прерывания, вырабатываемый этим устройством.

Рассматривая кооперацию процессов и взаимоисключения, мы говорили о существовании критических секций внутри ядра операционной системы, при выполнении которых необходимо исключить всякие прерывания от внешних устройств. Для запрещения прерываний, а точнее, для невосприимчивости процессора к внешним прерываниям обычно существуют специальные команды, которые могут маскировать (запрещать) все или некоторые из прерываний устройств ввода-вывода. В то же время определенные кризисные ситуации в вычислительной системе (например, неустранимый сбой в работе оперативной памяти) должны требовать ее немедленной реакции. Такие ситуации вызывают прерывания, которые невозможно замаскировать или запретить и которые поступают в процессор по специальной линии шины прерываний, называемой линией немаскируемых прерываний (NMI – Non-Maskable Interrupt).

Не все внешние устройства являются одинаково важными с точки зрения вычислительной системы ("все животные равны, но некоторые равнее других"). Соответственно, некоторые прерывания являются более существенными, чем другие. **Контроллер прерываний обычно позволяет устанавливать приоритеты для прерываний от внешних устройств.** При почти одновременном возникновении прерываний от нескольких устройств (во время выполнения одной и той же команды процессора) процессору сообщается номер наиболее приоритетного прерывания для его обслуживания в первую очередь. Менее приоритетное прерывание при этом не пропадает, о нем процессору будет доложено после обработки более приоритетного прерывания. Более того, при обработке возникшего прерывания процессор может получить сообщение о возникновении прерывания с более высоким приоритетом и переключиться на его обработку.

Механизм обработки прерываний, по которому процессор прекращает выполнение команд в обычном режиме и, частично сохранив свое состояние, отвлекается на выполнение других действий, оказался настолько удобен, что зачастую разработчики процессоров используют его и для других целей. Хотя эти случаи и не относятся к операциям ввода-вывода, мы вынуждены упомянуть их здесь, для того чтобы их не путали с прерываниями. Похожим образом процессор обрабатывает исключительные ситуации и программные прерывания.

Для внешних прерываний характерны следующие особенности.

- Внешнее прерывание обнаруживается процессором между выполнением команд (или между итерациями в случае выполнения цепочечных команд).
- Процессор при переходе на обработку прерывания сохраняет часть своего состояния перед выполнением **следующей** команды.
- Прерывания происходят **асинхронно** с работой процессора и **непредсказуемо**, программист никоим образом не может предугадать, в каком именно месте работы программы произойдет прерывание.

Исключительные ситуации возникают во время выполнения процессором команды. К их числу относятся ситуации переполнения, деления на ноль, обращения к отсутствующей странице памяти (см. часть III) и т. д. Для исключительных ситуаций характерно следующее.

- Исключительные ситуации обнаруживаются процессором во время выполнения команд.
- Процессор при переходе на выполнение обработки исключительной ситуации сохраняет часть своего состояния перед выполнением текущей команды.
- Исключительные ситуации возникают **синхронно** с работой процессора, но **непредсказуемо** для программиста, если только тот специально не заставил процессор делить некоторое число на ноль.

Программные прерывания возникают после выполнения специальных команд, как правило, для выполнения привилегированных действий внутри системных вызовов. Программные прерывания имеют следующие свойства.

- Программное прерывание происходит в результате выполнения специальной команды.
- Процессор при выполнении программного прерывания сохраняет свое состояние перед выполнением **следующей** команды.
- Программные прерывания, естественно, возникают синхронно с работой процессора и **абсолютно предсказуемы** программистом.

Надо сказать, что реализация похожих механизмов обработки внешних прерываний, исключительных ситуаций и программных прерываний лежит целиком на совести разработчиков процессоров. Существуют вычислительные системы, где все три ситуации обрабатываются по-разному.