Лекция 2

Последовательность выполнения команд процессором определяется логикой программы и результатами выполнения предыдущих команд. Существуют ситуации когда процессор должен приостановить выполнение текущей программы для того чтобы срочно отреагировать наиболее срочный запрос. Для этого существует прерывание.

Прерывание (аппаратное) – это сигналы при поступлении которых нормальная последовательность выполнения программ может быть прервана. При этом система запоминает информацию необходимую для возобновления работы прерванной программы и передает управление подпрограмме обработки прерываний. После завершения обработки управление возвращается прерванной программе.

Прерывание делится на 3 основных типа:

1. Прерывание аппаратное и периферийных устройств
2. Внутреннее (исключение)
3. Программное

В современных ОС обработку всех типов прерываний берет на себя сама система и это влияет на все системы. Типы прерываний разнообразны каждый из них требует особой обработки поэтому большинство процессоров поддерживают векторное прерывание. То есть каждая разновидность прерывания имеет свой номер. Этот номер используется как индекс в массиве который хранит для всех прерываний так называемые векторы прерываний. Каждый вектор содержит адрес соответствующий адрес команд обработки прерываний и доп информацию(дескриптор прерываний). При возникновении прерывания аппаратура по номеру прерывания по номеру определяет адрес подпрограммы обработки и вызывает ее предварительно сохраняется информация которая нужно для последующего возврата к выполнению прерванной программы. Чтобы наиболее ответственный участки программ выполнялись без прерываний система может временной запретить прием большинства прерываний. Этот запрет устанавливается автоматически при начале обработки полученного прерывания. Запрет прерывания должен сохраняться лишь в течении короткого интервалов времени не дольше нескольким мс иначе следубщие прерывания могут остаться необработанными и приводит к ошибкам системы.

Прерывания от периферийных устройств. Этот тип прерываний используется для управления операциями ввода вывода. Устройства генерируют сигнал прерывания в 1 из двух случаев: при переходе в состояние готовности, при возникновении ошибки выполнения операции.

Состояние готовности – это такое состояние устройства, в котором оно готово принять и выполнить команды от процессора. Для устройства ввода готовность означает наличие в устройстве данных который могут быть переданы в процессор. Для устройства вывода готовность это возможность принять процессором данные которые нужно вывести. Для служат сигнал прерывания.

Ошибка операции тоже требует вмешательства системы или пользователя (отсутствие бумаги в принтере, ошибка чтения диска…) Не каждое устройство генерирует прерывание, например монитор не выдает прерываний (может принять данные для отображения). Линии запроса прерывания от различных устройств поступают на входы схемы контроллера прерывания. Каждому устройству в контроллере присваивается номер запроса для вызова подпрограммы обработка. Контроллер формирует запрос на прерывание, он передается в ЦП. Запрос содержит номер вектора прерывания, который определяется на основе программы обработки прерываний. Контроллером может заблокировать передачу менее приоритетных прерываний. В периферийных устройствах жестко определяют номера контроллеров. Число подключаемых устройств ограничивается числом входов контроллера.

Исключения или внутри аппаратное прерывание это сигналы которые сообщают о внештатных ситуациях возникающих при выполнении команд (деление на 0). Обработка исключений не может быть запрещена. Результат обработки исключений либо аварийное завершение программы либо попытка продолжить выполнение но предварительно попытавшись устранить ошибку. Они вызываются выполнением специальной команды программы Но обрабатываются также как остальные типы прерываний. Остальные типы программного прерывания это особый случай вызова подпрограммы но при это вместо адреса подпрограммы указывается номер прерывания обработчик которого должен быть вызван. В большинстве операционных систем программные прерывания исспользуются для перехода из режима пользователя в режим ядра при вызове системных функций.

Синхронные и асинхронные прерывания:

исключения и программные прерывания объединяются общим понятием синхронное прерывание. Он могут возникнуть только при выполнении определенных команд. Они как бы синхронизированы с выполнением текущей программы. А вот прерывания периферийных устройств являются асинхронными, они могут произойти в произвольный момент вне зависимости от текущей программы.

Режимы работы процессора:

Как минимум 2 режима работы:

Привилегированный (режим ядра)

Непривилегированный (режим задач, пользователя)

Программы работающие в привилегированном режиме имеют неограниченный ко всем ресурсам пк. В непривилегированном режиме возможности программы ограничены, не может выполнять запрет разрешения и запрета прерывания и тд. Аппаратное разграничение возможностей это необходимое условие для надежной защиты данных в многопользовательской системе.

Программа пользователя работает в непривилегированном режиме. Не могут произвольно переключаться в режим ядра и выполняться в нем. Переход из режима пользователя в режим ядра только в случае обработки прерывания аппаратного или команды программы. Выбранный номер прерывания и другие параметры определяют какая из системных функций будет выполнена, после выполнения необходимых действий в режиме ядра система возвращает управление программе пользователя переключая процессор в непривилегированный режим. Важная возможность обеспечиваемая различением режимов работы процессора это ограничение доступа пользовательских процессов к областям памяти. Каждый процесс получает в распоряжение собственное адресное пространство, отображаемое аппаратурой на определенную область памяти вычислительной системы. Процесс не видит других процессов которые присутствуют в системе если отображение их адресных пространств на память не пересекаются.

Процессоры intel поддерживают 4 режима работы процессора которые называются кольцами защиты, они отличаются уровнем предоставляемых программам привилегий от максимальных (кольцо 0) до минимальных (кольцо 3). Некоторые операционные (real time system) используют кольцо 0 для ядра системы, кольца 1 и 2 для других системных программ и кольцо 3 для пользователей. Достигается таким образом повышение надежности систем. Большинство операционных систем (винда) не используют кольца 1 и 2.

ВАРИАНТЫ АРХИТЕКТУРЫ ЯДРА:

В практике разработки операционных систем конкурируют разные подходы к определению объема функции которые выполняются ядром. А так же внутренние структуры ядра.

1. Монолитное ядро – это традиционный тип архитектуры ядра. Представляет собой единую и неделимую программу выполняющую все основные функции операционной системы. Отдельные подсистемы составляющие ядро работают в общем адресном пространстве и поэтому могут использовать общие структуры данных и передавать управление друг другу. Объединением функций в монолитные программные структуры приводит к разделению функций системы на отдельные группы достаточно условно.

КРАТКИЙ СПИСОК ДЕЙСТВИЙ, КТОТОРЫЕ ВЫПОЛНЯЮТСЯ СИСТЕМОЙ ПРИ ЗАПУСКЕ ПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ ПРИ ВЫПОЛНЕНИИ:  
Лекция 3

КРАТКИЙ СПИСОК ДЕЙСТВИЙ, КТОТОРЫЕ ВЫПОЛНЯЮТСЯ СИСТЕМОЙ ПРИ ЗАПУСКЕ ПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ ПРИ ВЫПОЛНЕНИИ:  
1. Ввод и интерпретация

1. Создание нового процесса
2. Выделение памяти для процесса
3. Поиск и открытие для чтения исполняемого файла программы
4. Обращение к дисковому устройству для чтения кода программы
5. Фиксация всех предпринятых действий в системном журнале
6. Запуск загруженных программ в процессор на выполнение

Все основные функции операционной систем:

Управление процессами, управление памятью, периферийными устройствами, ввод и интерпретация команды, сбор статистики.

Эффективность выполнения системных функций тем выше чем легче переход от функций одной группы к другой. Чем теснее руппы связанны между собой, поэтому монолитное ядро выглядит наилучшим решением.

Но монолитная архитектура имеет недостатки:

1. Проектирование, отладка, сопровождение большой по объему и сложности программы будет затруднено.
2. Сложность отладки большой программы примерно пропорциональна квадрату ее размера, с учетом того что монолит может достигать нескольких миллионов команд
3. Надежность монолитного ядра не высокая. Считается что любая сложная программа даже отлаженная содержит некоторое число не выявленных ошибок. Примерно 1 ошибка на 1000 команд и чем теснее связаны друг с другом отдельные части ядра тем необратимее последствия ошибки. Особенно опасны ошибки это драйверы периферийных устройств которые при загрузки становятся частью ядра. Ошибка в работе драйвера может привести к краху всей системы.

МОДУЛЬНОЕ И МНОГОУРОВНЕВОЕ ЯДРО:

Проблема разработки монолитного ядра смягчается использованием принципа модульности. Разбиение программы на модули которая минимально зависит друг от друга а так же четкое формализованное описание связей сохраняющееся между модулями это условие успешной работы и успешной разработки сложных программных систем, в том числе и ядер ОС. Изменение в одном из модулей системы не должны приводить к существенным переработкам других модулей. При проектировании ядер ОС как многоуровневая структура ядра.

Верхний уровень – библиотека системных вызовов и их выполнение требует обращения к нижележащим уровням и так вплоть до самого нижнего уровня который и взаимодействует с аппоратурой.

Модульная и многоуровневая структура ядра облегчает разработку больших систем но ценой незначительной снижения производительности. Понятная и четкая структура ядра положительно сказывается на надежности. Уменьшается кол-во трудно обнаруживаемых ошибок. Не снимается проблема влияния ошибок в любом модуле ядра на работу системы в целом.

МИКРОЯДРО:

Альтернатива монолиту. В ОС на микроядре большая часть функциональных возможностей системы выносится за пределы ядра и передается вспомогательным системным процессам которые выполняют свои задачи и работают в пользовательском режиме. Такие процессы называют сервисами. К ним относятся драйверы периферийных устройств а так же модули выполняющие управлением процессами, распределением памяти и управлением файлами. Сбой в работе одного из сервисов не приводит к краху всей системы в целом. Система может перезапустить драйвер и восстановить работоспособность. Любое программное или аппаратное прерывание приводит к вызову микроядра, обычно микроядро анализирует причину ядра вызывает для его обработки соответствующий сервис. И самому микроядру остаются функции которые могут быть выполнены только в режиме ядра:

1. Запуск и завершение процессов
2. Переключение работающих процессов
3. Управление отображением адресных пространств процессов
4. Управление привилегиями сервисных процессов. (разрешения драйверам
5. Поддержка обмена сообщениями между процессами.

Операционные системы построенные на основе микроядра имеют многоуровневую модульную структуру, системы более надежные и гибкие чем системы с монолитным ядром.

Узкое место микро ядерных систем это производительность, работа ОС требует тесного взаимодействия различных подсистем и если модули монолитного ядра передают друг другу управление с помощью обычных команд перехода или вызова подпрограмм то в системе с микро ядром отдельные сервисы взаимодействует друг с другом только посредством микроядра. Всякий раз нужно вызывать программные прерывания для переключения. Обрабатывая прерывание микро ядро посылает сообщение сервису, адресату который выполнив запрос должен вызвать микроядро для доставки результатов сервису отправителю. За счёт приемов программирования достигают высокой производительности, она уступает монолиту на 15-20%, для большей части распространенных ОС характерно либо модульно-монолитное ядро(Линуха) либо гибридное ядро(это компромисс между монолитным и микро ядерным подходом) (Windows NT и дальше).

Системные функции:

В качестве программного интерфейса системы то есть средств обращение ПП к службам ОС используется документированный набор системных вызовов или функций API.

Системный вызов - это функции реализуемые непосредственно программами ядра системы. При микро ядерной архитектуры с участием ядра. При их выполнении происходит смена режима на режим ядра и обратно.

Api функции – это функция описанная в документации ОС независимо от того выполняются они ядром или системными библиотеками работающими в режиме пользователя. В windows несколько api обращаются к одному и тому же недокументированному системному вызову но имеют различные обрамляющие части работающими в режиме пользователя.

И там где различия несущественно между понятиями используют нейтральный термин системная функция.

История ОС:

1. Ms-DOS – Однозадачная система
2. Windows – сложная система на базе ms-dos
3. Unix – система по возможности сопоставимая с windows но отличающиеся по основным концепций и методом реализации

MS DOS разработанв 1981 для первого 16 разрядного ПК. Однозадачная система, ведет диалог с пользователем в текстовом режиме рассчитана на обслуживание прикладных программ в текстовом режиме но допускает экран. Работа с мышью обеспечивается самими программами, при нимиальной поддержке ПО. Для размещения ПО пользователя мс дос может 640 КБ памяти, позднее были добавлены средства позволяющие использовать до 4МБ памяти. Интерфейс ms dos с прикладными программами основан на вызовах программных прерываний которые обрабатываются системой, большую часть этих прерываний называют функциями dos. Это стартовая площадка для винды. Все версии Виндовс, обеспечивают выполнение большей части программ на дос.

Винда:

Первоначально она была разработана как графическая оболочка запускаемая под ms dos. Версии windows 1.0 (1985 год) работала на 1 МБ памяти с неперекрывающимися окнами. Версии 3.0 и 3.1 (90,91 год) Использовали 16 МБ памяти. Эти версии оставались надстройками и добавляли собственное управлениями процессами. Комбинация Dos + Windows это многозадачная система с графическим интерфейсом.

1993 Появляется NT это уже полноценная многозадачная и многопользовательская ОС без Ms Dos. Система требовала крутые процессора

1995 Появилась Windows95 и повышение скорости работы путем отказа от многопользовательской защиты и ослаблением надежности системы. (Неаккуратно написанные ПП могла привести к краху, но правильная программа переносились на WindowsXP). Винда дает несколько тысяч документированных API функций и распространение ОС способствует простота установки и обилие приложений.

Unix и PostX. В 1973 году система переписана на Си и оставил минимум текста на ссылка. Первая десятилетие система распространялась свободно, поэтому популярность в науке и у студентов. Пробелма, не полные совместимые разные версии.

В 1983 году Unix v5. Тогда решили сделать стандарт. 1988-1990 был разработан набор стандартов ктороые названы Posigs, которые развиваются. Эти стандарты касаются поведения системы а не ее устройства. Они определяют требования к набору системных команд и функций.

В 1990-ие возврат коммерции. Появляется linux, Отличие от unix состоит в большей надежности и эффективности. Исходные тексты свободно распространяются. Таком образом Unix многозадачная многопользовательская система разделения времени.

Достоинства:

Мобильность, крутой интерфейс, хорошее архитектурное решение

Недостатки:

Более низкая эффективность и надежность работы(плата за мобильность системы). Настроить сложнее, linux система для программистов.

Архитектура Linux предназначена для системы разделения времени с одним процессором, стала подходящей для поддержки сетевых систем.

У винды главные окна а у unix главное текст (окна дополнительное, не обязательное). Особенность linux: развитый язык, shell (он нужен для диалога системы, скрипты без обычных языков). Системы Linux лидируют для серверов и суперпк, мобилок. Винда удобна дома