Лекция 8. Принципы системной буферизации ввода/вывода и прерывания и особые ситуации в ОС UNIX

План :

1 Принципы системной буферизации ввода/вывода

Традиционно в ОС UNIX выделяются три типа организации ввода/вывода и, соответственно, три типа драйверов.

Блочный ввод/вывод главным образом предназначен для работы с каталогами и обычными файлами файловой системы, которые на базовом уровне имеют блочную структуру. Блочный ввод/вывод, кроме того, поддерживается системной.

Символьный ввод/вывод служит для прямого (без буферизации) выполнения обменов между адресным пространством пользователя и соответствующим устройством. Общей для всех символьных драйверов поддержкой ядра является обеспечение функций пересылки данных между пользовательскими и ядерным адресными пространствами.

Потоковый ввод/вывод похож на символьный ввод/вывод, но по причине наличия возможности включения в поток промежуточных обрабатывающих модулей обладает существенно большей гибкостью.

Традиционным способом снижения накладных расходов при выполнении обменов с устройствами внешней памяти, имеющими блочную структуру, является буферизация блочного ввода/вывода. Это означает, что любой блок устройства внешней памяти считывается прежде всего в некоторый буфер области основной памяти, называемой в ОС UNIX системным кэшем, и уже оттуда полностью или частично (в зависимости от вида обмена) копируется в соответствующее пользовательское пространство.

Принципами организации традиционного механизма буферизации является, во-первых, то, что копия содержимого блока удерживается в системном буфере до тех пор, пока не возникнет необходимость ее замещения по причине нехватки буферов. Во-вторых, при выполнении записи любого блока устройства внешней памяти реально выполняется лишь обновление (или образование и наполнение) буфера кэша. Действительный обмен с устройством выполняется либо при выталкивании буфера вследствие замещения его содержимого, либо при выполнении специального системного вызова sync (или fsync), поддерживаемого специально для насильственного выталкивания во внешнюю память обновленных буферов кэша.

Эта традиционная схема буферизации вошла в противоречие с развитыми в современных вариантах ОС UNIX средствами управления виртуальной памятью и в особенности с механизмом отображения файлов в сегменты виртуальной памяти Поэтому в System V Release 4 появилась новая схема буферизации, пока используемая параллельно со старой схемой. Суть новой схемы состоит в том, что на уровне ядра фактически воспроизводится механизм отображения файлов в сегменты виртуальной памяти. Новая схема буферизации в ядре ОС UNIX главным образом основывается на том, что для организации буферизации можно не делать почти ничего специального. Когда один из пользовательских процессов открывает не открытый до этого времени файл, ядро образует новый сегмент и подключает к этому сегменту открываемый файл. После этого (независимо от того, будет ли пользовательский процесс работать с файлом в традиционном режиме с использованием системных вызовов read и write или подключит файл к

сегменту своей виртуальной памяти) на уровне ядра работа будет производиться с тем ядерным сегментом, к которому подключен файл на уровне ядра. Основная идея нового подхода состоит в том, что устраняется разрыв между управлением виртуальной памятью и общесистемной буферизацией.

Семафоры.

Семафор может рассматриваться как целочисленная переменная, которая принимает только неотрицательные значения.

Так, процесс, требующий защищенного семафором ресурса, вынужден ожидать до тех пор, пока семафор не станет доступным, что свидетельствует об освобождении ожидаемого ресурса, и, захватив ресурс, установить семафор. В свою очередь, другие процессы также будут ожидать доступа к ресурсу вплоть до того момента, когда семафор возвратит соответствующий ресурс системе распределения ресурсов.

Семафор в ОС UNIX состоит из следующих элементов:

- значение семафора;
- идентификатор процесса, который хронологически последним работал с семафором;
- число процессов, ожидающих увеличения значения семафора;
- число процессов, ожидающих нулевого значения семафора.

Для работы с семафорами поддерживаются три системных вызова:

- **semget** для создания и получения доступа к набору семафоров;
- **semop** для манипулирования значениями семафоров (это именно тот системный вызов, который позволяет процессам синхронизоваться на основе использования семафоров);
- **semctl** для выполнения разнообразных управляющих операций над набором семафоров.

Основным поводом для введения массовых операций над семафорами было стремление дать программистам возможность избегать тупиковых ситуаций в связи с семафорной синхронизацией. Это обеспечивается тем, что системный вызов semop, каким бы длинным он не был, выполняется как атомарная операция, т.е. во время выполнения semop ни один другой процесс не может изменить значение какого-либо семафора.

Очереди сообщений.

Для обеспечения возможности обмена сообщениями между процессами этот механизм поддерживается следующими системными вызовами:

- **msgget** для образования новой очереди сообщений или получения дескриптора существующей очереди;
- **msgsnd** для посылки сообщения (вернее, для его постановки в указанную очередь сообщений);
- **msgrcv** для приема сообщения (вернее, для выборки сообщения из очереди сообщений);
- **msgctl** для выполнения ряда управляющих действий.

Как обычно, при выполнении системного вызова msgget ядро ОС UNIX либо создает новую очередь сообщений, помещая ее заголовок в таблицу очередей сообщений и возвращая пользователю дескриптор вновь созданной очереди, либо находит элемент таблицы очередей сообщений, содержащий указанный ключ, и возвращает соответствующий дескриптор очереди.

Для того чтобы ядро успешно поставило указанное сообщение в указанную очередь сообщений, должны быть выполнены следующие условия:

- обращающийся процесс должен иметь соответствующие права по записи в данную очередь сообщений;
- длина сообщения не должна превосходить установленный в системе верхний предел;
- общая длина сообщений (включая вновь посылаемое) не должна превосходить установленный предел;
- указанный в сообщении тип сообщения должен быть положительным целым числом.

В этом случае обратившийся процесс успешно продолжает свое выполнение, оставив отправленное сообщение в буфере очереди сообщений. Тогда ядро активизирует (пробуждает) все процессы, ожидающие поступления сообщений из данной очереди.

Если же оказывается, что новое сообщение невозможно буферизовать в ядре по причине превышения верхнего предела суммарной длины сообщений, находящихся в одной очереди сообщений, то обратившийся процесс откладывается (усыпляется) до тех пор, пока очередь сообщений не разгрузится процессами, ожидающими получения сообщений.

2 Прерывания и особые ситуации

Система UNIX позволяет таким устройства, как внешние устройства ввода-вывода и системные часы, асинхронно прерывать работу центрального процессора. По получении сигнала прерывания ядро операционной системы сохраняет свой текущий контекст (застывший образ выполняемого процесса), устанавливает причину прерывания и обрабатывает прерывание. После того, как прерывание будет обработано ядром, прерванный контекст восстановится и работа продолжится так, как будто ничего не случилось. Устройствам обычно приписываются приоритеты в соответствии с очередностью обработки прерываний. В процессе обработки прерываний ядро учитывает их приоритеты и блокирует обслуживание прерывания с низким приоритетом на время обработки прерывания с более высоким приоритетом.

Особые ситуации связаны с возникновением незапланированных событий, вызванных процессом, таких как недопустимая адресация, задание привилегированных команд, деление на ноль и т.д. Они отличаются от прерываний, которые вызываются событиями, внешними по отношению к процессу. Особые ситуации возникают прямо "посредине" выполнения команды, и система, обработав особую ситуацию, пытается перезапустить команду; считается, что прерывания возникают между выполнением двух команд, при этом система после обработки прерывания продолжает выполнение процесса уже начиная со следующей команды. Для обработки прерываний и особых ситуаций в системе UNIX используется один и тот же механизм.

Уровни прерывания процессора

Ядро иногда обязано предупреждать возникновение прерываний во время критических действий, могущих в случае прерывания запортить информацию. Например, во время обработки списка с указателями возникновение прерывания от диска для ядра нежелательно, т.к. при обработке прерывания можно запортить указатели. Обычно имеется ряд привилегированных команд, устанавливающих уровень прерывания процессора в слове состояния процессора. Установка уровня прерывания на определенное значение отсекает прерывания этого и более низких уровней, разрешая обработку только прерываний с более высоким приоритетом. На Рисунке 4 показана последовательность уровней прерывания. Если ядро игнорирует прерывания от диска, в этом случае игнорируются и все остальные прерывания, кроме прерываний от часов и машинных сбоев.

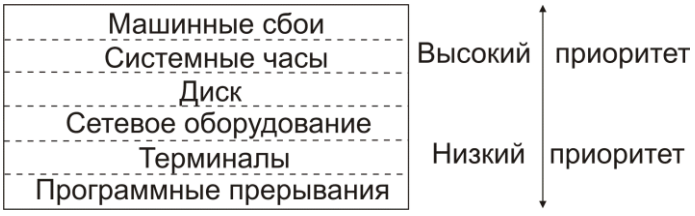

Рисунок 1. Стандартные уровни прерываний

3 Перспективные ОС, поддерживающие среду ОС UNIX

Микроядро - это минимальная стержневая часть операционной системы, служащая основой модульных и переносимых расширений. По-видимому, большинство операционных систем следующего поколения будут обладать микроядрами.

В широкий обиход понятие микроядра ввела компания Next, в операционной системе которой использовалось микроядро Mach. Небольшое привилегированное ядро этой ОС, вокруг которого располагались подсистемы, выполняемые в режиме пользователя, теоретически должно было обеспечить небывалую гибкость и модульность системы. Но на практике это преимущество было несколько обесценено наличием монолитного сервера, реализующего операционную систему UNIX BSD 4.3, которую компания Next выбрала в качестве оболочки микроядра Mach.

Следующей микроядерной операционной системой была Windows NT компании Microsoft, в которой ключевым преимуществом использования микроядра должна была стать не только модульность, но и переносимость. ОС NT была построена таким образом, чтобы ее можно было применять в одно- и мультипроцессорных системах, основанных на процессорах Intel, Mips и Alpha. Поскольку в среде NT должны были выполняться программы, написанные для DOS, Windows, OS/2 и систем, совместимых со стандартами Posix, компания Microsoft использовала присущую микроядерному подходу модульность для создания общей структуры NT, не повторяющей ни одну из существующих операционных систем. Каждая операционная система эмулируется в виде отдельного модуля или подсистемы.

Позднее микроядерные архитектуры операционных систем были объявлены компаниями Novell/USL, Open Software Foundation (OSF), IBM, Apple и другими. Одним из основных конкурентов NT в области микроядерных ОС является Mach 3.0, система, созданная в университете Карнеги-Меллон, которую как IBM, так и OSF взялись довести до коммерческого вида

Очевидна тенденция к переходу от монолитных к микроядерным системам. Это совсем не новость для компаний QNX Software Systems и Unisys, которые уже в течение нескольких лет выпускают пользующиеся успехом микроядерные операционные системы. ОС QNX пользуется спросом на рынке систем реального времени, а CTOS фирмы Unisys популярна в области банковского дела. В обеих системах успешно использована модульность, присущая микроядерным ОС.

Средства графического интерфейса пользователей.

Во всех современных вариантах ОС UNIX поддерживаются графические интерфейсы пользователя с системой, а пользователям предоставляются инструментальные средства для разработки графических интерфейсов с разрабатываемыми ими программами. С точки зрения конечного пользователя средства графического интерфейса, поддерживаемого в разных вариантах ОС UNIX, да и в других системах (например, MS Windows или Windows NT), примерно одинаковы по своему стилю.

Во-первых, во всех случаях поддерживается многооконный режим работы с экраном терминала. В любой момент времени пользователь может образовать новое окно и связать его с нужной программой, которая работает с этим окном как с отдельным терминалом. Окна можно перемещать, изменять их размер, временно закрывать и т.д.

Во-вторых, во всех современных разновидностях графического интерфейса поддерживается управление мышью.

В-третьих, такое распространение "мышиного" стиля работы оказывается возможным за счет использования интерфейсных средств, основанных на пиктограммах (icons) и меню. В большинстве случаев, программа, работающая в некотором окне, предлагает пользователю выбрать какую-либо выполняемую ей функцию либо путем отображения в окне набора символических образов возможных функций (пиктограмм), либо посредством предложения многоуровневого меню. В любом случае для дальнейшего выбора оказывается достаточным управления курсором соответствующего окна с помощью мыши.

Наконец, современные графические интерфейсы обладают "дружественностью по отношению к пользователю", обеспечивая возможность немедленного получения интерактивной подсказки по любому поводу.

Оконная система X как базовое средство графических интерфейсов в среде ОС UNIX.

Для нормальной организации работы пользовательских программ с графическими терминалами требуется наличие некоторого базового слоя программного обеспечения, скрывающего аппаратные особенности терминала; обеспечивающего создание окон на экране терминала, управление этими окнами и работу с ними со стороны пользовательской программы; дающего возможность пользовательской программе реагировать на события, происходящие в соответствующем окне (ввод с клавиатуры, движение курсора, нажатие клавиш мыши и т.д.). Такой базовый слой графического программного обеспечения принято называть оконной системой.

На этих идеях построена и оконная система X. На стороне пользовательского терминала находится сервер системы X, обеспечивающий единообразное управление графическим терминалом вне зависимости от его специфических аппаратных характеристик. В других компьютерах сети (которые, фактически, являются серверами с точки зрения организации вычислительного процесса) установлены клиентские части системы X, создающие впечатление у выполняемой программы, что она взаимодействует с локальным терминалом, а на самом деле поддерживающие точно специфицированный протокол взаимодействий с сервером системы X.

Клиентская и серверная части оконной системы X, хотя в целом соответствуют идеологии архитектуры "клиент-сервер", обладают тем своеобразием, что серверная часть системы находится вблизи пользователя (т.е. основного клиента вычислительной сети), а клиентская часть системы базируется на мощных серверах сети. В зависимости от конфигурации системы X-сервер может обслуживать один или несколько графических экранов, клавиатуру и мышь, реально представляя собой процесс, группу процессов или выделенное компьютерное устройство (X-терминал).

Для обеспечения требуемой гибкости, взаимодействия клиентской и пользовательской частей системы X по мере возможностей не должны были зависеть от используемых сетевой среды передачи данных и сетевых протоколов.

Одним из клиентов оконной системы обычно является так называемый "оконный менеджер" (window manager). Это специально выделенный клиент оконной системы, обладающий полномочиями на управление расположением окон на экране терминала. Некоторые из возможностей X-протокола (связанные, например, с перемещением окон) доступны только клиентам с полномочиями оконного менеджера. Во всем остальном оконный менеджер является обычным клиентом.

Системы, основанные на System V Release 4.

На базе System V возникло много коммерческих компаний. Их продукты мы кратко рассмотрим.

Solaris компании Sun Microsystems.

В течение многих лет основой операционных систем (SunOS) компании Sun являлся UNIX BSD. Однако, начиная с SunOS 4.0, произошел полный переход на System V 4.0. Sun Microsystems внесла ряд существенных расширений в SVR 4.0. Прежде всего это касается обеспечения распараллеливания программ при использовании симметричных мультипроцессорных компьютеров (механизм потоков управления - threads).

Solaris является внешней оболочкой SunOS и дополнительно включает средства графического пользовательского интерфейса и высокоуровневые средства сетевого взаимодействия (в частности, средства вызова удаленных процедур - RPC).

HP/UX компании Hewlett-Packard, DG/UX компании Data General, AIX компании IBM.

HP/UX, DG/UX и AIX обладают многими отличиями. В частности, в этих версиях ОС UNIX поддерживаются разные средства генерации графических пользовательских интерфейсов

(хотя все они основаны на использовании оконной системы X), по-разному реализованы threads и т.д. Однако все эти системы объединяет тот факт, что в основе каждой из них находится SVR 4.x. Поэтому основной набор системных и библиотечных вызовов в этих реализациях совпадает.

Santa Cruz Operation и SCO UNIX.

Варианты ОС UNIX, производимые компанией SCO и предназначенные исключительно для использования на Intel-платформах, до сих пор базируются на лицензированных исходных текстах System V 3.2. Однако SCO довела свои продукты до уровня полной совместимости со всеми основными стандартами (в тех позициях, для которых существуют стандарты).

Консерватизм компании объясняется прежде всего тем, что ее реализация ОС UNIX включает наибольшее количество драйверов внешних устройств и поэтому может быть установлена практически на любой Intel-платформе.

Open Software Foundation и OSF-1.

OSF была первой коммерческой компанией, решившейся на полную реализацию ОС UNIX на базе микроядра Mach. Результатом этой работы явилось создание ОС OSF-1. На сегодняшний день наиболее серьезным потребителем OSF-1 является компания Digital Equipment на своих платформах, основанных на микропроцессорах Alpha. В OSF-1 поддерживаются все основные стандарты ОС UNIX, хотя многие утверждают, что пока система работает не очень устойчиво.

Свободно распространяемые и коммерческие варианты ОС UNIX семейства BSD.

Многие годы варианты ОС UNIX, разработанные в Калифорнийском университете г. Беркли, являлись реальной альтернативой AT&T UNIX. Например, ОС UNIX BSD 4.2 была бесплатно доступна в исходных текстах и достаточно широко использовалась даже в нашей стране на оригинальных и воспроизведенных машинах линии DEC.

Группа BSD оказала огромное влияние на общее развитие ОС UNIX. До появления SVR 4.0 проблемой для пользователей являлась несовместимость наборов системных вызовов BSD и System V.

Несколько лет назад группа BSD разделилась на коммерческую и некоммерческую части. Новая коммерческая компания получила название BSDI. Обе подгруппы выпустили варианты ОС UNIX для Intel-платформ под названиями 386BSD и BSD386, причем коммерческий вариант был гораздо более полным.

Сегодня популярен новый свободно распространяемый вариант ОС UNIX, называемый FreeBSD. Ведутся работы над более развитыми версиями BSDNet.

Системы семейства BSD на сегодняшний день не являются единственными свободно доступными вариантами ОС UNIX.

Linux университета Хельсинки.

LINUX - это оригинальная реализация ОС UNIX для Intel-платформ, выполненная молодым сотрудником университета Хельсинки Линусом Торвальдом.

Ядро системы написано в традиционной технологии (т.е. без использования микроядра). Однако по отзывам любителей ядро LINUX отличается высоким качеством кода и хорошей модульностью. Кроме того, утверждается, что при аккуратном программировании прикладные программы, созданные в среде LINUX, без особых проблем переносятся в среду коммерческих систем, базирующихся на System V.

Hurd Free Software Foundation.

Проект системы Hurd явился попыткой довести до логического завершения знаменитый проект GNU Ричарда Столлмана, основателя и президента Фонда свободного программного обеспечения (Free Software Foundation - FSF). Основной идеей проекта Hurd было использование в качестве основы системы готового варианта микроядра Mach, бесплатно распространяемого университетом Карнеги-Меллон. Сам Ричард Столлман рекомендует пока использовать LINUX совместно с продуктами линии GNU.

Упражнения к лекции.

1. Какие основные типы организации ввода/вывода существуют в ОС UNIX?
2. Семафоры в ОС UNIX.
3. Очереди сообщений в ОС UNIX.
4. Прерывания и особые ситуации в ОС UNIX.
5. Перспективные ОС, поддерживающие среду Ос Unix.

Литература к лекции.

5.1 А.В.Гордеев, А.Ю.Молчанов. Системное программное обеспечение. — "Питер", 2002. — 736с.

5.2 Кристиан К. Введение в операционную систему Unix: пер. с англ. — М. Финансы и статистика, 1985. – 360с.

5.3 Робачевский А.М., Немнюгин С.А., Стесик О.Л. Операционная система Unix. 2-е изд.– СПб.: БХВ – Петербург, 2005. – 635с.