**Наблюдение за ресурсами, пользователями ОС UNIX. Передача сообщений. Управление ресурсами.**

Лабораторная работа №3

Выполнили студенты 1 курса группы ПИЭ-02-1 Ехлаков Олег и Черепнин Василий

Липецкий государственный педагогический университет

Липецк 2003

**Пользователь**

С самого начала ОС UNIX замышлялась как интерактивная система. Другими словами, UNIX предназначен для терминальной работы. Чтобы начать работать, человек должен "войти" в систему, введя со свободного терминала свое учетное имя (account name) и, возможно, пароль (password). Человек, зарегистрированный в учетных файлах системы, и, следовательно, имеющий учетное имя, называется зарегистрированным пользователем системы. Регистрацию новых пользователей обычно выполняет администратор системы. Пользователь не может изменить свое учетное имя, но может установить и/или изменить свой пароль. Пароли хранятся в отдельном файле в закодированном виде. Не забывайте свой пароль, снова узнать его не поможет даже администратор!

Все пользователи ОС UNIX явно или неявно работают с файлами. Файловая система ОС UNIX имеет древовидную структуру. Промежуточными узлами дерева являются каталоги со ссылками на другие каталоги или файлы, а листья дерева соответствуют файлам или пустым каталогам. Каждому зарегистрированному пользователю соответствует некоторый каталог файловой системы, который называется "домашним" (home) каталогом пользователя. При входе в систему пользователь получает неограниченный доступ к своему домашнему каталогу и всем каталогам и файлам, содержащимся в нем. Пользователь может создавать, удалять и модифицировать каталоги и файлы, содержащиеся в домашнем каталоге. Потенциально возможен доступ и ко всем другим файлам, однако он может быть ограничен, если пользователь не имеет достаточных привилегий.

**Интерфейс пользователя**

Традиционный способ взаимодействия пользователя с системой UNIX основывается на использовании командных языков (правда, в настоящее время все большее распространение получают графические интерфейсы). После входа пользователя в систему для него запускается один из командных интерпретаторов (в зависимости от параметров, сохраняемых в файле /etc/passwd). Обычно в системе поддерживается несколько командных интерпретаторов с похожими, но различающимися своими возможностями командными языками. Общее название для любого командного интерпретатора ОС UNIX - shell (оболочка), поскольку любой интерпретатор представляет внешнее окружение ядра системы.

Вызванный командный интерпретатор выдает приглашение на ввод пользователем командной строки, которая может содержать простую команду, конвейер команд или последовательность команд. После выполнения очередной командной строки и выдачи на экран терминала или в файл соответствующих результатов, shell снова выдает приглашение на ввод командной строки, и так до тех пор, пока пользователь не завершит свой сеанс работы путем ввода команды logout или нажатием комбинации клавиш Ctrl-d.

Командные языки, используемые в ОС UNIX, достаточно просты, чтобы новые пользователи могли быстро начать работать, и достаточно мощны, чтобы можно было использовать их для написания сложных программ. Последняя возможность опирается на механизм командных файлов (shell scripts), которые могут содержать произвольные последовательности командных строк. При указании имени командного файла вместо очередной команды интерпретатор читает файл строка за строкой и последовательно интерпретирует команды.

**Привилегированный пользователь**

Ядро ОС UNIX идентифицирует каждого пользователя по его идентификатору (UID - User Identifier), уникальному целому значению, присваиваемому пользователю при регистрации в системе. Кроме того, каждый пользователь относится к некоторой группе пользователей, которая также идентифицируется некоторым целым значением (GID - Group IDentifier). Значения UID и GID для каждого зарегистрированного пользователя сохраняются в учетных файлах системы и приписываются процессу, в котором выполняется командный интерпретатор, запущенный при входе пользователя в систему. Эти значения наследуются каждым новым процессом, запущенным от имени данного пользователя, и используются ядром системы для контроля правомощности доступа к файлам, выполнения программ и т.д.

Понятно, что администратор системы, который, естественно, тоже является зарегистрированным пользователем, должен обладать большими возможностями, чем обычные пользователи. В ОС UNIX эта задача решается путем выделения одного значения UID (нулевого). Пользователь с таким UID называется суперпользователем (superuser) или root. Он имеет неограниченные права на доступ к любому файлу и на выполнение любой программы. Кроме того, такой пользователь имеет возможность полного контроля над системой.

Еще одним отличием суперпользователя от обычного пользователя ОС UNIX является то, что на суперпользователя не распространяются ограничения на используемые ресурсы. Для обычных пользователей устанавливаются такие ограничения как максимальный размер файла, максимальное число сегментов разделяемой памяти, максимально допустимое пространство на диске и т.д. Суперпользователь может изменять эти ограничения для других пользователей, но на него они не действуют.

**Программы**

ОС UNIX одновременно является операционной средой использования существующих прикладных программ и средой разработки новых приложений. Новые программы могут писаться на разных языках (Фортран, Паскаль, Модула, Ада и др.). Однако стандартным языком программирования в среде ОС UNIX является язык Си (который в последнее время все больше заменяется на Си++).

Выполняемая программа может быть запущена в интерактивном режиме как команда shell или выполнена в отдельном процессе, образуемом уже запущенной программой.

Процессы

Процесс в ОС UNIX - это программа, выполняемая в собственном виртуальном адресном пространстве. Когда пользователь входит в систему, автоматически создается процесс, в котором выполняется программа командного интерпретатора. Если командному интерпретатору встречается команда, соответствующая выполняемому файлу, то он создает новый процесс и запускает в нем соответствующую программу, начиная с функции main. Эта запущенная программа, в свою очередь, может создать процесс и запустить в нем другую программу (она тоже должна содержать функцию main) и т.д.

Для образования нового процесса и запуска в нем программы используются два системных вызова (примитива ядра ОС UNIX) - fork() и exec (имя-выполняемого-файла). Системный вызов fork приводит к созданию нового адресного пространства, состояние которого абсолютно идентично состоянию адресного пространства основного процесса (т.е. в нем содержатся те же программы и данные).

**Управление устройствами**

Управление внешними устройствами - это одна из важнейших функций любой операционной системы. Система должна обеспечивать эффективный и удобный доступ к периферийным устройствам, а также обеспечивать возможность унифицированной разработки программного обеспечения для вновь подключаемых внешних устройств. Рассмотрим, как эта проблема решается в ОС UNIX.

Устройство как специальный файл

Для доступа к внешним устройствам в ОС UNIX используется универсальная абстракция файла. Помимо настоящих файлов (обычных файлов или каталогов), которые реально занимают память на магнитных дисках, файловая система содержит так называемые специальные файлы, для которых, как и для настоящих файлов, отводятся отдельные i-узлы, но которым на самом деле соответствуют внешние устройства. Это решение позволяет естественным образом работать в одном и том же интерфейсе с любым файлом или внешним устройством. (На самом деле, в некоторых случаях нестандартных внешних устройств, приходится выходить за пределы стандартного интерфейса).

Драйверы устройств

Любому программисту должно быть ясно, что простое объявление внешнего устройства специальным файлом не даст возможности работать с этим устройством, если не создан и соответствующим образом не подключен к системе специальный программный код, соответствующий специфике данного устройства. Как и в большинстве современных операционных систем, такого рода программный код в ОС UNIX называется драйвером устройства (в этом контексте слово драйвер лучше всего понимать в значении "управляющий").

Для профессионалов в области операционных систем драйверы ОС UNIX, в сущности, не представляют ничего нового. По-простому говоря, в любой системе драйвер устройства - это многовходовой программный модуль со своими статическими данными, который умеет инициировать работу с устройством, выполнять заказываемые пользователем обмены (на ввод или вывод данных), терминировать работу с устройством и обрабатывать прерывания от устройства. Однако, в любой операционной системе имеется своя технология разработки драйверов. В частности, в ОС UNIX различаются символьные, блочные и потоковые драйверы.

Последний момент, на который мы хотим обратить внимание в этом пункте, состоит в том, что (опять же, как и в большинстве развитых операционных систем) в ОС UNIX возможны два способа включения драйвера в состав ядра ОС. Первый способ состоит в полном включении драйвера в состав ядра на стадии генерации системы (т.е. драйвер статически объявляется частью ядра системы). Второй способ позволяет обойтись минимальным количеством статических объявлений на стадии генерации ядра (фактически, обеспечиваются лишь необходимые элементы статических таблиц). В любой момент работы системы такой драйвер может быть динамически загружен в ядро системы. После появления (статического или динамического) в ядре ОС UNIX драйверы всех разновидностей функционируют единообразно.

Внешний и внутренний интерфейсы устройств

Независимо от типа файла (обычный файл, каталог, связь или специальный файл) пользовательский процесс может работать с файлом через стандартный интерфейс, включающий системные вызовы open, close, read и write. Ядро само распознает, нужно ли обратиться к его стандартным функциям или вызвать подпрограмму драйвера устройства. Другими словами, если процесс пользователя открывает для чтения обычный файл, то системные вызовы open и read обрабатываются встроенными в ядро подпрограммами open и read соответственно. Однако, если файл является специальным, то будут вызваны подпрограммы open и read, определенные в соответствующем драйвере устройства.

Распределенные файловые системы

Основная идея распределенной файловой системы состоит в том, чтобы обеспечить совместный доступ к файлам локальной файловой системы для процессов, которые, вообще говоря, выполняются на других компьютерах. Эта идея может быть реализована многими разными способами, однако в среде ОС UNIX все известные подходы основываются на монтировании удаленной файловой системы к одному из каталогов локальной файловой системы. После выполнения этой процедуры файлы, хранимые в удаленной файловой системе, доступны процессам локального компьютера точно таким же образом, как если бы они хранились на локальном дисковом устройстве.

На рисунке 1 приведен пример, в котором два подкаталога удаленной файловой системы-сервера (share и X11) монтируются к двум (пустым) каталогам файловой системы-клиента.

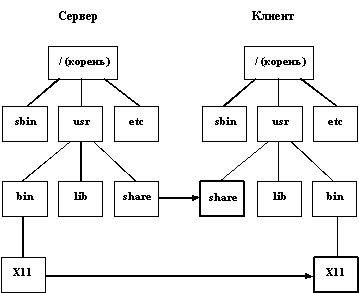


Рис. 1. Схема монтирования удаленной файловой системы

В принципе, такая схема обладает и достоинствами, и недостатками. К достоинствам, конечно, относится то, что при работе в сети можно экономить дисковое пространство, поддерживая совместно используемые информационные ресурсы только в одном экземпляре. Но, с другой стороны, пользователи удаленной файловой системы неизбежно будут работать с удаленными файлами существенно более медленно, чем с локальными. Кроме того, реальная возможность доступа к удаленному файлу критически зависит от работоспособности сервера и сети. Заметим, что распространенные в мире UNIX сетевые файловые системы NFS (Network File System - сетевая файловая система) и RFS (Remote File Sharing - совместное использование удаленных файлов) являются достаточно тщательно спроектированными и разработанными продуктами, во многом сглаживающими отмеченные недостатки.

Некоторые команды ОС Unix.

WHOAMI - идентификация пользователя ( информация о текущем сеансе ).

ОС Unix позволяет манипулировать файлами и директориями различными способами. Можно копировать, уничтожать, переименовывать, записывать, распечатывать и разделять файлы в ЛВС. Есть также определённая система прав доступа к файлам и директориям.

Как файлы так и директории на сервере в ЛВС под управлением ОС Unix имеют атрибуты. Эти атрибуты могут отменять права, предоставленные пользователям в ЛВС.

CHMOD – изменение прав доступа пользователям, группам, всем остальным. 1 – r, 2 – w, 4 – x. Для разрешения различных действий, записывается сумма присвоенных атрибутам значений. Например, команда chmod 777 Virt устанавливает для всех полный доступ к директории Virt.

MESG Y – включает message.

WRITE USER – отправляет сообщение на указанный терминал.