Юрий Магда

bhv

UNIX

- Базовые концепции UNIX
- Работа в сетях TCP/IP
- Электронная почта и Интернет
- Графические оболочки UNIX
- Редакторы текста
- Командный интерпретатор Shell
- Разработка приложений на C и Perl

Наиболее полное руководство

В ПОДЛИННИКЕ®

Юрий Магда

UNIX

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.066
ББК 32.973.26-018.2
М12

Магда Ю. С.

М12 UNIX. — СПб.: БХВ-Петербург, 2006. — 528 с.: ил.

ISBN 5-94157-824-5

Рассматривается широкий круг вопросов функционирования операционной системы UNIX. Анализируются принципы взаимодействия процессов, управления учетными записями пользователей и построения файловой системы. Изложены базовые концепции функционирования и настройки сетей на основе протокола TCP/IP и их реализация в операционных системах UNIX. С позиции пользователя описаны современные методы обработки текстовой документации и работа с графическими оболочками операционной системы. Значительная часть материала книги посвящена основам разработки приложений на языках C и Perl в среде UNIX, созданию командных файлов в интерпретаторе shell.

Теоретические аспекты функционирования UNIX иллюстрируются многочисленными примерами программ, разработанных на языке C.

Для пользователей UNIX

УДК 681.3.066
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 15.06.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-824-5

Оглавление

Благодарности

Автор выражает огромную благодарность сотрудникам издательства "БХВ-Петербург" за подготовку материалов книги к изданию. Особая признательность жене Юлии за неоценимую помощь и поддержку при подготовке рукописи.

Введение

В данной книге рассматриваются основные принципы функционирования операционной системы UNIX, а также практические аспекты настройки рабочей среды пользователя и основы разработки программного обеспечения. По этой теме издано достаточно много книг, предназначенных для очень широкой аудитории, начиная новичками и заканчивая опытными пользователями и системными администраторами.

Отличие этой книги от других подобных изданий состоит в том, что в ней рассматривается очень широкий круг вопросов, касающихся как функционирования самой операционной системы UNIX, так и работы пользователя в ней. Большинство книг по UNIX имеет, как правило, конкретную направленность и предназначено для определенного круга читателей. Так, в литературе для новичков основной акцент делается, как правило, на изучении пользовательского интерфейса. Системных администраторов и разработчиков программного обеспечения больше интересуют принципы построения и функционирования операционной системы, которые описаны в книгах для профессиональных пользователей.

В этой книге делается попытка объединить различные подходы к анализу операционной системы, рассматривая программную архитектуру UNIX, программный интерфейс разработчика и интерфейс пользователя как части единого целого, а именно операционной системы UNIX. По этой причине многие аспекты взаимодействия пользователя и операционной системы иллюстрируются примерами программного кода на языке C, что, по мнению автора, способствует глубокому пониманию функционирования различных подсистем UNIX и осмысленным действиям по конфигурированию и настройке операционной системы пользователем. Тем не менее, от читателя не требуется глубоких знаний по программированию на C, достаточно знать основы этого языка.

Материал книги охватывает следующий круг вопросов:

- анализ теоретических основ функционирования операционных систем UNIX, позволяющий понять те или иные практические приемы работы, а также смысл параметров, используемых для настройки;
- настройка и использование графического интерфейса пользователя и программ обработки текстовой документации;
- конфигурирование и настройка сети в операционной системе UNIX, а также теоретические и практические аспекты функционирования Интернета и электронной почты;
- принципы разработки программного обеспечения в UNIX с использованием языков C и Perl.

Книга состоит из двенадцати глав, краткое описание каждой из них приведено далее.

- *Глава 1. "Обзор операционных систем UNIX".*

  В главе рассматриваются стандарты операционных систем UNIX и проводится сравнительный анализ наиболее популярных программных продуктов, таких как Linux, FreeBSD и Solaris. Рассмотрены также и наиболее распространенные аппаратные платформы, на которых работает UNIX.

- *Глава 2. "Архитектура UNIX".*

  Материал главы посвящен обзору архитектуры операционных систем UNIX. Рассматриваются принципы построения UNIX, взаимодействие ядра с другими подсистемами, а также механизмы запуска и выполнения программ пользователя. Здесь же анализируются механизмы реализации многозадачности и многопользовательского режима. Подробно рассматриваются механизмы взаимодействия процессов, выполняющихся в операционной системе, с ядром.

- *Глава 3. "Учетные записи пользователей".*

  Здесь рассмотрены вопросы управления учетными записями пользователей: регистрация, изменение и удаление учетных записей пользователей, а также получение информации о пользователях.

- *Глава 4. "Командный интерпретатор shell".*

  В этой главе рассматривается использование встроенного командного интерпретатора shell. Приводится описание синтаксиса shell, включая описание переменных и логических программных структур интерпретатора. Представлены многочисленные примеры применения командного интерпретатора shell для решения различных задач.

- *Глава 5. "Файловая система UNIX".*

  Эта глава посвящена администрированию файловой системы. Рассмотрены организация и структура файловых систем, операции с файлами и каталогами, а также назначение и изменение прав доступа к объектам файловой системы. Анализируются практические аспекты создания и управления файловыми системами, монтирование и демонтирование файловых систем.

- *Глава 6. "Обработка текста в UNIX".*

  Материал главы знакомит читателя с популярными программами для обработки текстовой документации, раскрывает широкий диапазон возможностей современных текстовых редакторов. Подробное описание различных режимов работы этих приложений значительно облегчает усвоение принципов функционирования и настройки текстовых редакторов.

- *Глава 7. "Процессы в UNIX".*

  В главе рассматривается один из ключевых аспектов функционирования операционной системы — взаимодействие процессов UNIX и управление ими. Здесь рассмотрены концепции создания, жизненного цикла и уничтожения процессов, а также программные средства для мониторинга процессов и получения статистической информации.

- *Глава 8. "Поддержка сетей в UNIX".*

  Глава знакомит читателя с принципами функционирования операционной системы UNIX в сетях TCP/IP. Детально рассматриваются основы построения сетей и основные протоколы взаимодействия сетевых приложений, а также вопросы настройки и конфигурирования протокола TCP/IP в операционной системе UNIX. Проанализированы принципы функционирования сетевых сервисов операционной системы UNIX, таких как DNS, DHCP и NFS. Значительная часть материала посвящена тестированию сетей и поиску неисправностей.

- *Глава 9. "Электронная почта".*

  Материал главы посвящен анализу работы и настройке систем электронной почты в UNIX. Здесь рассматриваются теоретические основы построения таких систем, а также анализируются основные аспекты использования наиболее популярных программ электронной почты. В частности, даются рекомендации по настройке базовой программы для пересылки электронной почты `sendmail`. В главе описывается функционирование основных протоколов электронной почты — SMTP, POP3 и IMAP4 и тестирование их работоспособности в UNIX-системах.

- *Глава 10. "UNIX и Интернет".*

  Эта глава посвящена вопросам работы систем UNIX в Интернете. Здесь описан протокол HTTP, являющийся базисным при обмене информацией между клиентами и Web-узлами, а также приводится пример настройки и тестирования популярного Web-сервера Apache. Часть материала главы посвящена примерам программирования простейших интернет-приложений.

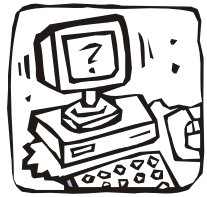
- *Глава 11. "Графические оболочки UNIX".*

  Здесь рассматриваются основы построения графических интерфейсов UNIX-систем. Значительная часть материала главы посвящена конфигурированию и настройке графической системы X Window, являющейся основой создания графических приложений для операционных систем UNIX. Кроме этого, рассматриваются основные принципы функционирования и настройки популярных графических оболочек GNOME и KDE.

- *Глава 12. "Разработка приложений в среде UNIX".*

  В главе обсуждаются наиболее важные аспекты разработки программного обеспечения для систем UNIX с использованием наиболее популярных языков программирования C и Perl. Здесь рассмотрены принципы компиляции и сборки приложений с использованием популярного компилятора g++. Ввиду большой популярности языка Perl значительная часть материала главы посвящена описанию синтаксиса языка с объяснением примеров программ.

Материал книги окажет существенную помощь читателям, имеющим определенный опыт работы с операционными системами UNIX, которые хотели бы существенно пополнить свои знания в этой области. Книга может быть полезна также системным администраторам и разработчикам программного обеспечения.



Глава 1

Обзор операционных систем UNIX

С момента своего появления операционная система UNIX стала довольно популярной и получила широкое распространение в вычислительных системах различной производительности, начиная от персональных компьютеров и заканчивая большими вычислительными комплексами. Такая популярность этой операционной системы обусловлена несколькими факторами.

Во-первых, это более чем трехдесятилетний цикл развития. За этот период операционная система UNIX выдержала проверку временем и проявила себя как очень эффективная программная среда. Во-вторых, программный код системы полностью написан на языке высокого уровня C, что делает ее понятной для пользователей и разработчиков программного обеспечения. Это позволяет относительно легко вносить изменения как в существующие версии UNIX, так и переносить операционную систему на другие аппаратные платформы. Кроме этого, во многих случаях версии этой операционной системы поставляются вместе с исходными текстами, что позволяет легко адаптировать UNIX под специфические требования, после чего перекомпилировать систему.

Изначально операционная система UNIX создавалась как многопользовательская и многозадачная система, ориентированная в первую очередь на выполнение серверных или управляющих функций для клиентских компьютеров. Такие подходы, а также мощные встроенные средства удаленного администрирования, способствовали тому, что UNIX заняла лидирующие позиции на рынке интернет-серверов и серверов баз данных. Немаловажную роль в популяризации операционной системы сыграла и структура ее файловой системы, представляющая единую иерархическую систему с унифицированным доступом как к файлам данных, так и к аппаратным ресурсам, таким как диски, терминалы, принтеры, сеть, память и т. п.

В практическом плане операционная система UNIX предоставляет пользователю целый ряд преимуществ. Перечислим только некоторые из них:

- все популярные приложения (пакеты офисных программ, базы данных и др.) известных производителей программного обеспечения, как правило, реализованы для работы в операционной системе UNIX;
- UNIX поддерживает широкий спектр средств передачи информации, включая многочисленные сетевые и коммуникационные протоколы, что обеспечивает эффективную работу операционной системы в сетях;
- операционная система UNIX имеет весьма развитые средства системного и сетевого управления, в том числе эффективные инструменты для удаленного администрирования и настройки;
- UNIX является мощной платформой для разработки приложений, в нее включены наиболее популярные языки разработки приложений, средства работы с базами данных, а также другое программное обеспечение;
- операционная система поддерживает все основные аппаратные процессорные архитектуры, включая надежную поддержку SMP, MMP и кластерных систем. В других серверных средах такая поддержка отсутствует;
- практически все важнейшие промышленные, международные, официально утвержденные и неофициальные стандарты были впервые разработаны для UNIX и только впоследствии распространились и на другие операционные системы. В настоящее время основным стандартом является разработанная консорциумом X/Open Единая спецификация UNIX, содержащая более тысячи интерфейсов прикладных программ и поддерживаемая всеми основными производителями операционной системы UNIX. Несмотря на то, что различные версии UNIX обладают уникальными возможностями, все они отвечают требованиям стандарта POSIX (Portable Operating System Interface, переносимый интерфейс операционной системы), а также стандарту X/Open Portability Guide, Edition 4 (XPG 4) и сертифицированы X/Open на соответствие стандартам UNIX 93;
- операционная система UNIX к настоящему времени утвердилась как платформа для персональных приложений, приложений для рабочих групп и приложений корпоративного класса.

В настоящее время единого стандарта для UNIX не существует, и на рынке присутствует множество версий этой операционной системы, каждая из которых имеет свои названия и особенности. Тем не менее, все без исключения UNIX-системы имеют однотипную архитектуру, интерфейсы и среду программирования, что дает основание считать все UNIX-системы в той или иной степени родственными.

Простота и способность операционных систем UNIX к расширению и модификациям являются весьма серьезными преимуществами по сравнению с другими системами, поэтому UNIX стали переносить на множество платформ. Тем не менее, несмотря на множество реализаций базовой системы, среди них выделяются две основные ветви, берущие начало от System V UNIX и BSD UNIX.

Одна ветвь происходит от версий 4.2, 4.3 или 4.4BSD, другая базируется на системах SVR3 (System V Release 3) или SVR4 (System V Release 4). На протяжении ряда лет версия BSD пользовалась бóльшей популярностью в академических и научных кругах, в то время как версии System V, разработанные компанией AT&T, занимали лидирующие позиции в коммерческих организациях и в промышленности. Несмотря на существующие различия между этими основными типами UNIX-систем, подавляющее большинство пользовательских команд работает одинаково и имеет один и тот же синтаксис во всех версиях операционных систем, независимо от того, используется ли AIX, BSD, HP/UX, Linux или Solaris.

Существующие в настоящее время отличия между операционными системами не имеют принципиального значения, и определить, к какой из ветвей принадлежит та или иная реализация операционной системы, иногда бывает довольно сложно. Вот основные различия между операционными системами System V и BSD:

- разные способы установки и настройки терминальных устройств;
- разные способы инициализации, именования конфигурационных файлов и файлов инициализации системы;
- различная настройка параметров файловой системы;
- различные методы получения диагностической информации и ее отображения на консоли

и т. п.

С точки зрения пользователя принципиальных различий между разными ветками операционной системы UNIX не существует. Справедливости ради следует отметить, что BSD и System V — далеко не единственные реализации операционной системы UNIX. Рассмотрим особенности реализации некоторых, наиболее популярных версий операционной системы UNIX и начнем с Solaris.

Исходный код UNIX System V Release 4 был доработан компанией Sun, в результате чего появилась реализация операционной системы UNIX, названная Solaris. Эта операционная система имеет несколько основных отличий от ба-

зовой операционной системы. В частности, в Solaris были добавлены следующие возможности:

- многонитевость;
- симметричная многопроцессорная обработка;
- режим реального времени.

В настоящее время Solaris является одной из самых распространенных версий операционной системы UNIX и работает на платформах SPARC и Intel86. Для развития продукта фирма Sun Microsystems предоставила более открытый доступ к кодам операционной среды Solaris.

Другая ветвь UNIX — FreeBSD — берет свое название от "Berkeley Software Distribution". Эта операционная система основана на версии 4.4BSD-Lite и сохраняет специфичные черты модели развития BSD-систем.

На базе версии 4.4BSD-Lite было создано несколько операционных систем с открытыми исходными кодами, среди которых особо следует выделить проект GNU. Операционная система FreeBSD позволяет:

- выполнять одновременно несколько задач с динамическим регулированием приоритетов, что распределяет ресурсы компьютера между приложениями и пользователями оптимальным образом;
- одновременно работать многим пользователям и использовать систему совместно для решения ряда задач. Это означает, что системные ресурсы, такие, например, как принтеры и накопители на магнитных лентах, могут распределяться между пользователями в системе или сети, при этом для каждого пользователя или группы пользователей могут быть установлены определенные ограничения на использование того или иного ресурса. Это позволяет избежать перегрузок в работе операционной системы;
- работать с распространенными сетевыми протоколами и стандартами, такими как SLIP, PPP, NFS, DHCP и NIS. Это позволяет операционной системе FreeBSD эффективно функционировать совместно с другими операционными системами, например, Windows. Кроме того, FreeBSD может использоваться в качестве интернет-сервера, предоставляя полный спектр сервисов (WWW, FTP, маршрутизация);
- использовать стандарт X Window System (X11R6), предоставляющий графический интерфейс пользователю.

Операционная система FreeBSD обеспечивает совместимость на уровне программного кода с большинством программ, разработанных для Linux и System V Release 4, обладая полным комплектом инструментальных средств для разработки программ (языки C, C++, Fortran и Perl). Исходные

тексты FreeBSD свободно распространяются через Интернет, так что систему можно оптимизировать для специальных приложений или проектов. Для коммерческих операционных систем такая возможность отсутствует.

FreeBSD очень часто используется как платформа для высокопроизводительных рабочих станций, при этом она оказывается более эффективной по сравнению с другой, не менее популярной операционной системой Linux. Системы на основе BSD могут демонстрировать бóльшую по сравнению с Linux производительность, обеспечивая при этом более высокую надежность. Наконец, операционная система FreeBSD может выполнять код, разработанный для Linux, но не наоборот.

Еще одной, очень популярной реализацией UNIX является Linux. Эта версия UNIX обладает большинством свойств, присущих другим реализациям, и, кроме того, включает некоторые дополнительные возможности. Linux — это полная многозадачная многопользовательская операционная система, допускающая одновременную работу многих пользователей.

Операционная система Linux очень популярна среди миллионов пользователей. GNU/Linux вместе с набором инструментальных средств по оценкам экспертов охватывает около 40% рынка UNIX. Многие компании выпускают дистрибутивы Linux — пакеты, включающие ядро, множество утилит, приложений и программное обеспечение для установки ОС. GNU/Linux получила поддержку у таких компаний, как Sun и IBM.

Для разработки программного обеспечения, работающего в Linux, был создан специальный фонд под названием Free Software Foundation (FSF), цель которого заключается в поиске источников финансирования разработки программного обеспечения GNU. Несмотря на относительно короткую историю существования, под эгидой проекта GNU было создано и адаптировано огромное количество программ, среди которых наиболее известными являются утилиты Emacs, gcc (компилятор GNU C) и bash (командная оболочка).

Необходимо отметить, что эта операционная система хорошо совместима с рядом стандартов для UNIX на уровне исходных текстов, включая IEEE POSIX.1, System V и BSD, поскольку создавалась с расчетом на такую совместимость.

Большинство из свободно распространяемого через Интернет программного обеспечения для UNIX может быть откомпилировано для работы в Linux с минимальными изменениями. Более того, все исходные тексты для Linux, включая ядро, драйверы устройств, библиотеки, пользовательские программы и инструментальные средства, также распространяются свободно.

К специфическим особенностям Linux следует отнести контроль работ по стандарту POSIX (используемый оболочками, такими как csh и bash), работу

с псевдотерминалами, поддержку национальных и стандартных клавиатур с динамически загружаемыми драйверами клавиатур.

Операционная система Linux поддерживает различные типы файловых систем. Некоторые из них, такие, например, как файловая система ext2fs, были созданы специально для Linux. Кроме того, поддерживаются и другие типы файловых систем, такие, например, как Minix и Xenix. Система включает реализацию файловой системы MS-DOS, позволяющую прямо обращаться к файлам MS-DOS на жестком диске. Наконец, для работы с CD-ROM поддерживается стандарт файловой системы ISO 9660.

Как и другие операционные системы, Linux обеспечивает полный набор протоколов TCP/IP для сетевой работы. Сюда входят драйверы устройств для многих сетевых карт Ethernet, SLIP (Serial Line Internet Protocol, обеспечивающий пользователям доступ по TCP/IP при последовательном соединении), PLIP (Parallel Line Internet Protocol), PPP (Point-to-Point Protocol), NFS и т. д. В систему включена поддержка всего спектра сервисов TCP/IP (FTP, telnet, NNTP и SMTP).

Глава 2

Архитектура UNIX

В этой главе мы рассмотрим базовые концепции построения операционных систем UNIX и взаимодействие различных функциональных частей. Знание основных принципов функционирования системы является основой для успешной работы в UNIX и помогает эффективно использовать возможности этой мощной операционной системы. Кроме того, многие принципы построения UNIX используются в других популярных операционных системах, например, Windows, что может помочь и при изучении таких систем.

В основу построения операционных систем UNIX было положено несколько важных принципов.

- **Надежность.** Операционная система должна быть по возможности максимально надежной. Сбои в работе системы в большинстве случаев могут вызываться неполадками в аппаратной части, неправильными действиями пользователя либо некорректной работой программного обеспечения. Во всех этих случаях для повышения надежности требуется каким-то образом сохранять работоспособность системы, обеспечивая определенный минимальный уровень функционирования и возможности восстановления. Самым неприятным следствием сбоя в работе может быть потеря важных пользовательских данных, поэтому сохранение и восстановление данных является основным критерием надежности системы. Одним из эффективных методов повышения надежности системы является разделение программного кода операционной системы и кода пользовательской программы таким образом, чтобы программа пользователя не могла разрушить выполняющийся программный код UNIX. С другой стороны, пользовательская программа должна иметь доступ к различным ресурсам операционной системы: памяти, процессору, жестким дискам, периферийным устройствам (принтерам, плоттерам, накопителям на магнитных лентах и т. д.), что является потенциально опасным для надежного функционирования системы.

Компромиссным решением стала концепция построения операционной системы на основе базового программного модуля ("ядра"), выполняющего обслуживание запросов программ пользователя, одновременно изолируя их от прямого доступа к ресурсам системы.

- **Возможность одновременной работы нескольких пользователей с одной системой.** В этом случае необходимо предоставлять ресурсы операционной системы (возможно, одни и те же) нескольким пользователям одновременно, причем возможности доступа к одним и тем же ресурсам для разных пользователей могут отличаться. Операционные системы, допускающие работу в таком режиме, называются многопользовательскими. Очень часто разным пользователям требуется доступ к одним и тем же ресурсам, например, к файлу на диске. При этом одни пользователи могут читать и записывать данные в файл, в то время как другие могут только читать данные из файла или вообще не иметь доступа к данным. Многопользовательская операционная система должна обладать механизмами разделения доступа к ресурсам и, кроме того, иметь возможности для защиты общих ресурсов от несанкционированного доступа. UNIX относится именно к такому классу операционных систем.
- **Возможность одновременного выполнения множества процессов.** В операционной системе должна быть возможность выполнения одновременно множества процессов (работающих программ), как пользовательских, так и системных. При этом должен обеспечиваться механизм синхронизации процессов. Это означает, что операционная система должна контролировать запуск, выполнение и уничтожение процессов, обеспечивая работающим процессам доступ к требуемым ресурсам наиболее эффективным способом (мультизадачность).
- **Унифицированный (единообразный) способ доступа к ресурсам операционной системы.** Эта концепция положена в основу построения файловой системы UNIX. В операционной системе UNIX объектами файловой системы являются как файлы программ или данных, так и устройства, например, принтеры, жесткие диски, терминальные линии. Подобный подход очень удобен, поскольку позволяет работать с единым интерфейсом и использовать одни и те же функции как для работы с файлами данных, так и с файлами устройств. Существенным преимуществом такого подхода является и то, что можно использовать единые принципы для разделения ресурсов системы в многопользовательской среде и установки защиты объектов файловой системы. Кроме того, разработчики программного обеспечения могут использовать единый интерфейс для разработки программ, что значительно снижает трудоемкость.

Реализованная на основе этих концепций операционная система UNIX обладает следующими характеристиками:

- ❒ она легко переносима на другие аппаратные платформы;
- ❒ допускает работу в режиме вытесняющей многозадачности, обеспечивая работу процессов в изолированных адресных пространствах в виртуальной памяти;
- ❒ обеспечивает поддержку одновременной работы многих пользователей;
- ❒ поддерживает работу асинхронных процессов;
- ❒ имеет иерархическую файловую систему;
- ❒ обеспечивает поддержку независимых от устройств операций ввода/вывода путем использования специальных файлов устройств;
- ❒ предоставляет стандартный интерфейс для программ (программные каналы) и пользователей (командный интерпретатор, не входящий в ядро операционной системы);
- ❒ имеет встроенные средства мониторинга операционной системы.

Посмотрим, как отображены вышеперечисленные возможности операционной системы UNIX в ее программной архитектуре. В наиболее общем виде операционную систему UNIX можно представить пятиуровневой моделью (рис. 2.1).

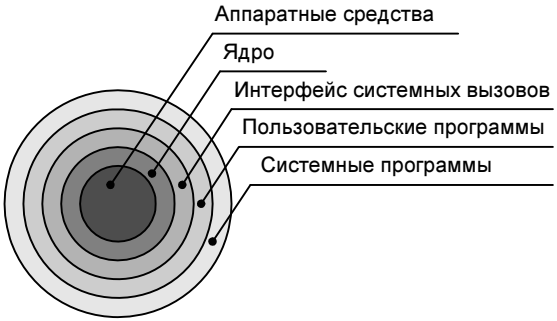


Рис. 2.1. Архитектура UNIX

Как видно из рис. 2.1, операционную систему UNIX можно представить как совокупность пяти взаимосвязанных функциональных частей:

- ❒ аппаратной части (hardware);
- ❒ ядра;
- ❒ интерфейса системных вызовов;

- системных служебных программ (утилит) и командных оболочек (командных интерпретаторов);
- пользовательских программ.

Аппаратная часть представляет собой физические ресурсы системы (процессор, оперативная память, жесткий диск, устройства ввода/вывода), непосредственный доступ к которым может осуществлять только ядро операционной системы. Прикладные пользовательские программы не могут получить прямой доступ к оборудованию системы, а взаимодействуют с ним посредством ядра, вернее через системные вызовы функций ядра.

Такое построение операционной системы обусловлено несколькими причинами. Во-первых, это гарантирует надежность работы системы, поскольку невозможно произвольным образом, например, из программы пользователя, изменить конфигурацию системы, что чревато крахом UNIX. Во-вторых, ядро операционной системы балансирует работу всех подсистем без вмешательства пользователя, обеспечивая тем самым оптимальную производительность работы. В-третьих, ограничение доступа к аппаратным средствам пользовательских программ обеспечивает надежную работу аппаратуры, поскольку ядро управляет функционированием физических устройств посредством специальных программ-драйверов устройств.

Операционная система UNIX является многопользовательской и многозадачной системой. Это означает, что в ней могут работать одновременно несколько пользователей, каждый из которых может выполнять несколько задач одновременно.

В основе функционирования операционной системы UNIX, как было упомянуто, лежит взаимодействие между пользовательскими программами, ядром и аппаратными ресурсами. Фактически функционирование операционной системы определяется особенностями работы ядра, поэтому остановимся на взаимодействии ядра и остальных функциональных частей UNIX более подробно.

Компиляция и сборка ядра операционной системы UNIX обычно выполняются статически. Это означает, что ядро загружается как один большой исполняемый программный модуль при инициализации системы. Такой тип ядра называют монолитным. Некоторые версии операционных систем допускают работу с другим типом ядра, который называют модульным или, иногда, микроядром. При использовании модульного ядра дополнительные модули программного кода (обычно это драйверы устройств) подгружаются в оперативную память динамически, т. е. по мере необходимости, например, при включении аппаратного устройства. Такие дополнительные модули реализованы в виде загружаемых модулей ядра.

Преимущество модульного ядра состоит в том, что базовый модуль ядра имеет небольшой размер, быстрее загружается и требует меньше ресурсов операционной системы. В то же время монолитное ядро работает чуть быстрее, поскольку не требуется переключений контекста выполняемых процессов (что имеет место в случае загрузки/выгрузки дополнительных модулей) и дополнительной синхронизации, как при использовании отдельных модулей. Кроме того, в монолитном ядре реализовано намного больше функций управления аппаратурой, поскольку такое ядро содержит драйверы устройств.

В настоящее время большинство ядер UNIX реализованы как комбинированные, т. е., являясь в принципе монолитными, допускают загрузку дополнительных модулей во время работы операционной системы.

Современные версии ядра UNIX-систем, в большинстве своем, обладают еще одной особенностью. Эта особенность — возможность функционирования ядра как совокупности отдельно выполняющихся потоков (kernel threads). Подобная особенность называется многопоточностью ядра и позволяет повысить эффективность функционирования как ядра, так и всей операционной системы. В первом приближении поток можно представить себе как отдельный выполняющийся фрагмент программного кода в рамках одного процесса. При этом ядро управляет выполнением потоков и их синхронизацией.

Более высокая эффективность выполнения многопоточных функций обусловлена тем, что переключение контекста отдельных потоков требует меньше времени, чем переключение контекста отдельных процессов. В значительной степени подобный выигрыш получается за счет того, что потоки выполняются в общем адресном пространстве, в то время как каждый процесс требует отдельного адресного пространства, а это занимает определенное время.

Особенностью современных операционных систем UNIX является еще и то, что ядро таких систем, кроме обработки отдельных потоков, поддерживает также работу многопоточных пользовательских программ, что повышает их производительность. В этом случае многопоточные приложения выполняются как совокупность элементарных (lightweight) процессов, которые используют общее адресное пространство, общие страницы памяти и открытые файлы. Естественно, что для получения выигрыша в производительности выполняющаяся программа должна поддерживать реализацию многопоточности.

Большинство реализаций ядра операционной системы UNIX выполняется в режиме невытесняющей многозадачности (non-preemptive multitasking). Это означает, что операционная система не может прерывать выполнение процесса, выполняющегося в режиме ядра. Ядро, работающее в режиме вытес-

няющей многозадачности (preemptive multitasking), используется, как правило, в UNIX-подобных операционных системах, функционирующих в режиме реального времени. Некоторые популярные операционные системы, например, Sun 2.*x*, используют такой тип ядра.

Независимо от архитектуры ядро UNIX-системы обеспечивает поддержку многопользовательского и многозадачного режима работы, выполняя следующие функции:

- создание, выполнение, остановку и завершение процессов, а также синхронизацию их взаимодействия;
- планирование приоритетов выполнения процессов путем выделения им времени центрального процессора. В этом случае центральный процессор выполняет процесс в течение определенного ядром интервала времени, после чего процесс приостанавливается, и ядро начинает выполнение другого процесса. Через определенный интервал времени приостановленный процесс возобновляет выполнение и т. д.;
- выделение исполняемому процессу определенного объема оперативной памяти. В этом случае ядро защищает адресное пространство процесса от доступа из других процессов, одновременно позволяя разным процессам совместно использовать участки адресного пространства на определенных условиях. Если системе требуется некоторый объем свободной памяти, ядро освобождает память за счет процесса. При этом контекст (данные и ссылки) процесса сохраняется на жестких дисках или иных внешних устройствах. Такая реализация системы UNIX называется системой со свопингом (подкачкой). Если же на жестком диске сохраняются страницы памяти, то такая система называется системой с замещением страниц;
- выделение памяти на устройствах постоянного хранения информации (жесткие диски и магнитные ленты) для обеспечения эффективного хранения и выборки данных пользователя. Эта функция ядра реализуется через обращение к файловой системе UNIX. Файловая система управляется посредством функций ядра, которые выделяют внешнюю память для файлов, выполняют отображение физической структуры файловой системы в логическую форму, доступную для работы пользователей, а также устанавливают атрибуты доступа к объектам файловой системы, защищая пользовательские файлы от несанкционированного доступа;
- управление доступом процессов к периферийным устройствам, таким как терминалы, накопители на магнитных лентах и сетевое оборудование.

С точки зрения пользователя функции ядра можно представить так, как показано на рис. 2.2.

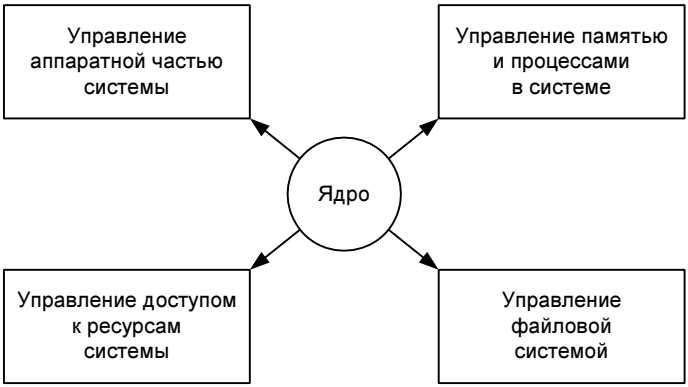


Рис. 2.2. Функции ядра UNIX

Ядро операционной системы является прозрачным для пользовательской программы. Это означает, что детали взаимодействия программы пользователя и операционной системы скрыты от пользователя. К примеру, если программа пользователя обращается к какому-либо файлу и записывает в него данные, то ядро системы выполняет последовательность довольно сложных действий: определяет местоположение файла на носителе, получает информацию о расположении требуемых данных в физических секторах накопителя, определяет место записи блока данных в физическую область дискового пространства и т. д. Наконец, ядро вызывает драйвер устройства и передает ему параметры и код операции (записи данных), после чего и выполняется запись данных.

Кроме вышеперечисленных ядро реализует ряд необходимых функций по обеспечению выполнения процессов пользовательского уровня, за исключением функций, которые реализуются на самом пользовательском уровне.

Например, ядро выполняет определенные действия, необходимые для работы командного интерпретатора shell. Такие функции командного интерпретатора, как чтение вводимых с терминала данных, динамическое создание процессов, синхронизация выполнения процессов, открытие программных конвейеров и переадресация ввода/вывода — все они реализуются через системные вызовы ядра.

Здесь я хочу сделать небольшое отступление и чуть более подробно остановиться на некоторых терминах и понятиях, часто используемых при анализе функционирования операционной системы UNIX. Эти термины будут встречаться очень часто и являются фундаментальными при анализе операционной системы. К таким терминам относятся "программа" и "процесс".

Под термином "программа" мы будем понимать записанный на носителе исполняемый файл, в то время как термин "процесс" в первом приближении будет означать программу, находящуюся в стадии выполнения. Ввиду важности понятия "процесс" остановимся на нем подробнее.

Процесс можно представить как исполняемый модуль программного кода, которому предоставлены определенные ресурсы системы (память, процессорное время и т. д.). В операционной системе UNIX может одновременно выполняться множество процессов (многозадачность), причем их число логически не ограничивается, и одна программа может создавать множество процессов. При этом существующие в системе процессы могут создавать новые или завершать другие процессы. Ядро операционной системы синхронизирует выполнение этапов процесса и управляет реакцией на наступление различных событий. Благодаря наличию системы защиты процессы выполняются независимо и не влияют друг на друга.

Создание и выполнение процессов иллюстрирует рис. 2.3.

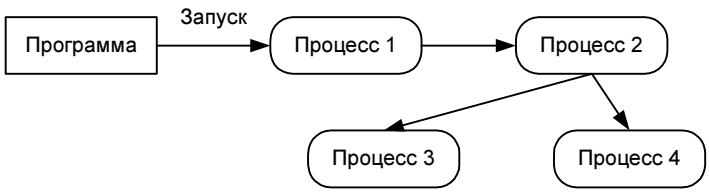


Рис. 2.3. Функционирование процессов в операционной системе UNIX

В простейшем случае программа порождает один процесс, в других случаях таких процессов может быть множество. Термин "процесс" применим не только к пользовательским или системным программам, но и к ядру, поскольку оно само функционирует как совокупность взаимосвязанных процессов.

Все процессы, выполняющиеся в среде операционной системы UNIX, могут выполняться либо в пользовательском режиме (User Mode), либо в режиме ядра (Kernel Mode). Подобное разделение обусловлено архитектурой системы и возможностью доступа процессов к ресурсам системы *(об этом упоминалось ранее в этой главе)*. Пользовательский режим не позволяет напрямую обращаться к аппаратным ресурсам системы и системным структурам данных, в то время как процесс, выполняющийся в режиме ядра, имеет такую возможность.

В пользовательском режиме могут работать не только программы пользователя, но и значительная часть системных программ, входящих в состав операционной системы.

Возникает вопрос: каким образом пользовательский процесс в случае необходимости может получить доступ к ресурсам системы?

Операционная система UNIX предоставляет процессам, работающим в режиме пользователя (User Mode), набор интерфейсов для взаимодействия с аппаратными устройствами, такими как процессор, жесткие диски, принтеры и т. д. UNIX реализует такие интерфейсы между режимом пользователя и аппаратурой посредством так называемых системных вызовов (system calls), которые взаимодействуют с функциями ядра. Совокупность системных вызовов образует "интерфейс системных вызовов" (см. рис. 2.1).

Системные вызовы инициируют смену контекста выполнения процесса: процесс, работающий в режиме пользователя, переключается на выполнение в защищенном режиме (режим ядра). Такое переключение позволяет процессу вызывать защищенные процедуры ядра для выполнения системных функций. Таким образом, системные вызовы обеспечивают программный интерфейс для доступа к управлению системными ресурсами, такими как память, дисковое пространство и периферийные устройства. Системные вызовы реализованы в виде библиотеки времени выполнения (run-time library), а многие из них используются командными интерпретаторами shell. Следует заметить, что системные функции ядра для корректной работы требуют упаковки аргументов специальным образом.

В качестве примеров системных вызовов можно привести низкоуровневые функции ввода/вывода, такие как `open()`, `read()`, `write()` и `close()`.

Системные вызовы обеспечивают выполнение целого ряда операций:

- трансляции операций пользователя в запросы к драйверам устройств;
- создания, запуска и уничтожения процессов;
- ввода/вывода;
- доступа к файлам и дисковым устройствам;
- поддержки терминальных устройств.

Наличие интерфейсов в форме системных вызовов предоставляет определенные преимущества разработчикам программ. Во-первых, процесс разработки программ становится намного легче, поскольку программисту нет необходимости изучать особенности программных интерфейсов низкого уровня для каждого из устройств.

Во-вторых, повышается надежность системы в целом, поскольку ядро UNIX может проверить корректность запроса программы пользователя на уровне интерфейса, прежде чем ответить на такой запрос.

Наконец, наличие таких интерфейсов позволяет легко переносить программы на другие реализации UNIX.

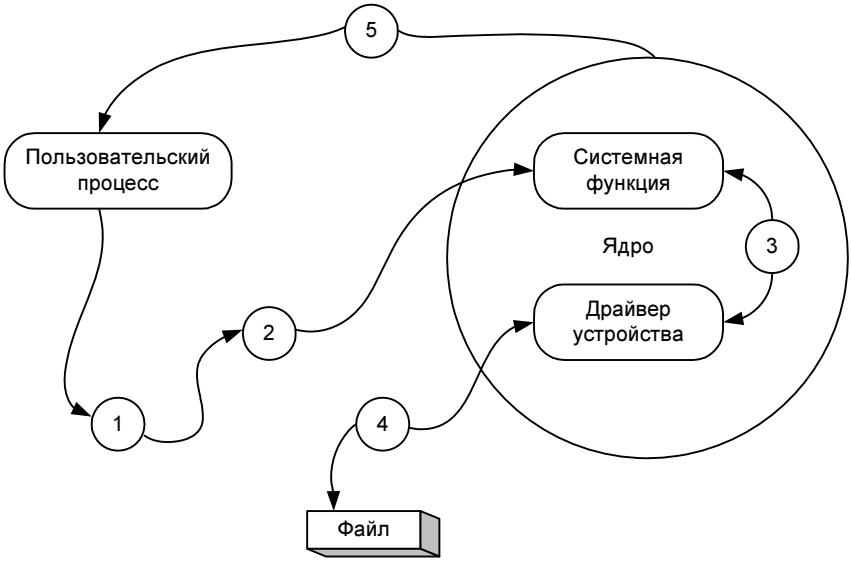


Рис. 2.4. Взаимодействие программы пользователя и ядра

Рассмотрим пример взаимодействия пользовательского процесса и ядра операционной системы UNIX. Предположим, что процессу пользователя необходимо открыть файл на диске для записи или чтения. В упрощенном виде последовательность шагов при открытии файла показана на рис. 2.4.

На шаге 1 процесс пользователя инициирует запрос на доступ к файлу (открытие файла). При этом программный код пользовательского процесса выполняет системный вызов при помощи функции `open()` (шаг 2). Далее (шаг 3) выполняется переключение контекста процесса из пользовательского режима в режим ядра и происходит обращение к системной функции ядра, соответствующей системному вызову `open()`.

Системная функция выполняет ряд операций, необходимых для открытия файла, находящегося на диске, например, формирует запросы к драйверу устройства (шаг 3) и анализирует статус (состояние) требуемого ресурса, полученный от драйвера. Драйвер устройства обращается к требуемому ресурсу и формирует статусную информацию и данные для системной функции (шаг 4).

На шаге 5 процесс пользователя получает от ядра результат выполнения запроса либо в виде определенных структур данных (для системного вызова `open()` это дескриптор открытого файла), либо сообщение об ошибке.

Системные вызовы в той или иной степени используются всеми без исключения пользовательскими и системными программами, и мы более подробно познакомимся с ними в *главе 12*.

Следующий уровень функциональности, который мы рассмотрим, — системные программы (см. рис. 2.1). К системным программам или, по-другому, к системному программному обеспечению относят командные оболочки shell (интерпретаторы), команды и утилиты системного администрирования, драйверы и протоколы коммуникаций. Как известно, операционная система UNIX включает ряд стандартных системных программ для выполнения задач администрирования, конфигурирования и поддержки файловой системы. Кроме того, к этой группе программ следует отнести утилиты:

- ❒ настройки параметров конфигурации системы;
- ❒ перекомпоновки ядра (если она необходима) и добавления новых драйверов устройств;
- ❒ создания и удаления учетных записей пользователей;
- ❒ создания и подключения физических файловых систем;
- ❒ установки параметров контроля доступа к файлам.

В качестве пользовательских программ могут выступать командные файлы, написанные с помощью командного интерпретатора shell, или разработанные на одном из языков высокого уровня (C, Pascal, Fortran) приложения. К пользовательским программам относятся многочисленные текстовые и графические редакторы, программы отправки и получения электронной почты и т. д. Следует отметить, что в некоторых случаях пользовательским программам не требуется обращение к функциям ядра. Процессы, порожденные пользовательскими программами, защищены от других пользовательских процессов, не имеют доступа к функциям ядра, кроме как через системные вызовы и, кроме того, не могут непосредственно обращаться к пространству памяти ядра.

Рассмотрим более подробно взаимодействие пользовательского процесса и ядра операционной системы.

Пространство памяти ядра представляет собой область памяти, в которой процессы ядра или, по-другому, процессы, работающие в контексте ядра, реализуют функции ядра. Пространство ядра — защищенная область, и пользователь получает к ней доступ только через интерфейс системных вызовов. Пользовательский процесс не имеет прямого доступа ко всем инструкциям и физическим устройствам — их имеет процесс ядра. Процесс ядра также может менять карту памяти, что необходимо для переключения процессов (смены контекста). Пользовательский процесс начинает работать в режиме ядра в тот момент, когда выполняется программный код ядра посредством системного вызова.

Поскольку пользовательские процессы и ядро не имеют общего адресного пространства памяти, необходим механизм передачи данных между ними.