Магда Ю.С.

RASPBERRY PI

Руководство по настройке и применению

ДМК Пресс
издательство

УДК 004.42:004.3'144:621.3.049.774ARM
ББК 32.973.26-018.2
М12

Магда Ю. С.

М12 Raspberry Pi. Руководство по настройке и применению – М.: ДМК Пресс, 2014. – 188 с.

ISBN 978-5-94074-964-6

Быстрый прогресс современной электроники в последние годы существенно повлиял на все сферы человеческой деятельности, включая применение компьютерных технологий. Существенным прорывом стало создание полнофункциональных компьютерных систем на одном кристалле, так называемом System-On-Chip (SoC). В SoC интегрируются все основные функциональные блоки, присущие компьютерам (процессор, память, графический процессор и др.). На одном из таких SoC-кристаллов реализован один из наиболее популярных современных миниатюрных компьютеров, известный под названием Raspberry Pi.

Эта книга посвящена практическим аспектам применения Raspberry Pi, начиная от программирования простых систем управления и измерения на языке Python и заканчивая разработкой мультимедийных систем и созданием игровых приложений на языке Scratch. Хотя Raspberry Pi помещается на ладони, он способен выполнять многие функции, доступные мощным настольным системам. Многие популярные приложения, работающие на настольных компьютерах, могут выполняться и на Raspberry Pi. Вдобавок Raspberry Pi обладает мощными мультимедийными и графическими возможностями, в частности, при работе с 3D графикой, поэтому этот миниатюрный компьютер можно использовать как платформу для разработки игровых приложений, что может заинтересовать многих будущих программистов. Raspberry Pi можно использовать и для создания своих собственных измерительных и робототехнических систем с различными датчиками и исполнительными устройствами. Создание таких систем возможно благодаря наличию цифрового порта ввода/вывода (GPIO) – подобная возможность отсутствует в обычных настольных ПК.

Материал книги будет полезен самой широкой аудитории, начиная от школьников и студентов и заканчивая разработчиками приложений для мультимедиа, Интернета и систем управления.

УДК 004.42:004.3'144:621.3.049.774ARM
ББК 32.973.26-018.2

ISBN 978-5-94074-964-6

СОДЕРЖАНИЕ

Функционирование программного обеспечения в Raspberry Pi выполняется в среде Raspbian OS, которая является адаптированной версией Debian Linux. В этой главе дается обзор основных возможностей Raspbian OS, а также наиболее популярных и полезных программ, выполняющихся в данной среде.

Операционная система Linux базируется на концепции открытого кода (open source), что позволяет пользователям свободно использовать и/или модифицировать ядро системы. Ядро Linux является сердцем операционной системы, осуществляя взаимодействие пользователя с аппаратными и программными ресурсами системы. Изначально под термином Linux подразумевалось именно ядро, хотя в настоящее время это определение относится ко всем приложениям (коллекциям или наборам) с открытым кодом, включенным в состав операционной системы.

Каждая такая коллекция программ может представлять собой отдельную реализацию (дистрибутив) Linux. В Raspberry Pi используется модифицированная версия дистрибутива Debian Linux, Raspbian OS.

Подобно другим операционным системам, таким как Windows или MacOS в Linux, реализован доступ нескольких к ресурсам системы, то есть Linux является многопользовательской системой с возможностью доступа к аппаратно-программным ресурсам для нескольких пользователей. При этом каждый пользователь системы имеет собственную учетную запись (account), для которой определены права доступа (привилегии) к тем или иным ресурсам Linux. Такой подход позволяет обеспечить эффективную защиту операционной системы от случайных или умышленных действий пользователя.

Наивысшие привилегии доступа к ресурсам операционной системы имеет суперпользователь (root), который может выполнять практически все операции по управлению аппаратными и программными ресурсами системы без каких-либо ограничений. Суперпользователь имеет практически неограниченный доступ к файловой системе, физическим и логическим (виртуальным) устройствам системы, поэтому работа в режиме суперпользователя *root* требует от пользователя высокой квалификации и осторожности, поскольку легко можно разрушить саму операционную систему. Режим суперпользователя *root* обычно применяется опытными пользователями, как правило, системными администраторами, для настройки и конфигурирования аппаратных и программных ресурсов Linux.

Режим суперпользователя не следует использовать в повседневной работе, поскольку всегда можно где-то ошибиться и нарушить работу операционной системы. Для выполнения обычных программ вполне достаточно привилегий обычного пользователя. В Raspbian OS таким пользователем является *pi*, имеющий по умолчанию пароль для входа *raspbian*. Этот пароль можно в любой момент изменить, если требуется большая безопасность системы. Пользователь *pi* имеет достаточно широкие полномочия по управлению системой, инсталляции и выполнению приложений, поэтому в большинстве случаев привилегий суперпользователя *root* не требуется.

В дальнейшем мы рассмотрим возможные случаи, когда без использования доступа посредством *root* нельзя обойтись.

Ниже (табл. 3.1) приводятся наиболее часто употребляемые термины при работе в операционной системе Linux, которые будут нередко использоваться далее в этой книге.

Таблица 3.1. Краткий перечень терминов и определений операционной системы Linux

Термин	Определение
Bash	Наиболее популярная командная оболочка, используемая в большинстве дистрибутивов Linux
Bootloader	Программное обеспечение, выполняющее загрузку ядра операционной системы Linux. В настоящее время наиболее популярным загрузчиком является GRUB
Console	Приложение командной строки для выполнения команд и приложений Linux
Desktop environment	Программное обеспечение для реализации графического интерфейса пользователя (GUI). К числу наиболее популярных графических оболочек относятся GNOME и KDE
Directory	Каталог (или, по-другому, папка) – общее название места, где хранятся файлы
Distribution	Этим термином обозначают конкретную версию (дистрибутив) операционной системы Linux, например Fedora Remix, Arch и Debian являются дистрибутивами Linux
Executable	Файл, который может быть выполнен как программа. Для запуска программы файл должен иметь атрибут «executable»
EXT2/3/4	Наиболее часто используемые в Linux форматы файловых систем
File system	Способ организации файлов и каталогов в операционной системе Linux
GNOME	Одна из наиболее популярных графических оболочек Linux
GNU	Проект для реализации свободно распространяемого программного обеспечения. Многие программы и утилиты, разработанные под эгидой этого проекта, используются в различных дистрибутивах Linux

Таблица 3.1 (окончание)

Термин	Определение
GRUB	Загрузчик ядра операционной системы Linux (GRand Unified Bootloader), разработанный в рамках проекта GNU
GUI	Графический интерфейс пользователя, в котором пользователь может управлять приложениями с помощью мыши
KDE	Еще одна популярная графическая оболочка
Linux	Ядро Linux, разработанное в рамках проекта GNU, или, по-другому, операционная система с открытым кодом
Live CD	Дистрибутив Linux, записанный на CD или DVD и не требующий инсталляции на жесткий диск
Package	Пакет (группа) взаимосвязанных файлов, требуемых для выполнения приложения. Пакеты программ обрабатываются менеджером пакетов
Package manager	Приложение для установки и настройки программного обеспечения, реализуемого в форме пакетов прикладных программ
Partition	Область жесткого диска, подготовленная для размещения файловой системы
root	Наиболее привилегированный пользователь Linux
Superuser	То же, что и root
Shell	Командная оболочка терминального приложения для выполнения команд Linux в текстовом режиме
Terminal	Приложение текстовой строки, позволяющее выполнять команды Linux, используя командную оболочку
X11	Графическая система X Window, позволяющая реализовать графический интерфейс пользователя

Для эффективного использования Raspberry Pi в качестве настольного ПК пользователь обязательно должен располагать базовыми знаниями операционной системы Linux, в частности версии Debian Linux, используемой в качестве базовой в Raspberry Pi. Все без исключения версии Linux обладают базовым набором свойств и характеристик, поэтому следующие разделы главы посвящены основам работы в операционной системе Linux, после чего мы ознакомимся с особенностями функционирования Debian Linux.

Основы функционирования операционной системы Linux

Операционная система Linux по своей сути является потомком классической и популярной, начиная с 70-х годов прошлого века, операционной системы UNIX и наследует ее лучшие характеристики.

С момента своего появления операционная система UNIX 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 того, 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, некоторые из них перечислены ниже:

- 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.

Linux является одной из наиболее популярных в настоящее время реализаций операционной системы UNIX. 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 многих ведущих производителей программного обеспечения, таких как Oracle, IBM и др.

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 gcc (компилятор GNU C) и bash (командная оболочка).

Большинство из свободно распространяемого через Интернет программного обеспечения для Linux может быть откомпилировано для работы с любым дистрибутивом 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 файловые системы ext2/3/4fs, 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).

Архитектура Linux

В этом разделе мы рассмотрим базовые концепции построения операционной системы Linux и взаимодействие ее различных функциональных частей. Знание основных принципов функционирования системы является основой для успешной работы в Linux и помогает эффективно использовать возможности этой мощной операционной системы.

В основу построения операционных систем Linux было положено несколько основных 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 являются 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 Linux. 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 (вероятно, 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. Linux 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 Linux. В операционной системе Linux 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 Linux 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.

Вышеперечисленные возможности операционной системы Linux можно представить пятиуровневой моделью (рис. 3.1).

Как видно по рис. 3.1, операционную систему Linux 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 си-

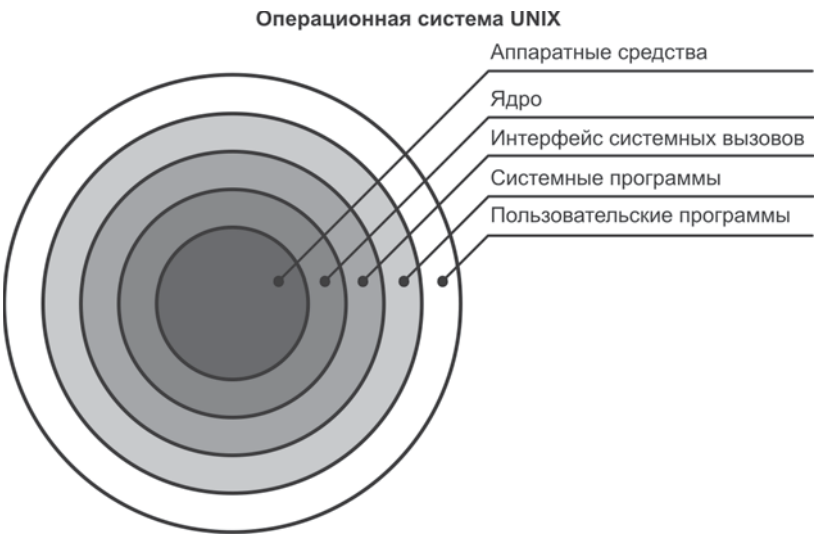

Рис. 3.1

стемы (процессор, оперативная память, жесткий диск, устройства ввода/вывода), непосредственный доступ к которым может осуществлять только ядро операционной системы. Прикладные пользовательские программы не могут получить прямого доступа 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 Linux 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 Linux, 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 Linux 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, то есть 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 Linux реализованы как комбинированные, то есть, являясь в принципе монолитными, допускают загрузку дополнительных модулей во время работы операционной системы. Современные версии ядра Linux-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 Linux 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 Linux выполняется в режиме невытесняющей многозадачности (non-preemptive multitasking). Это означает, что операционная система не может прерывать выполнение процесса, происходящегося в режиме ядра. Ядро, работающее в режиме вытесняющей многозадачности (preemptive multitasking), используется, как правило, в UNIX-подобных операционных системах, функционирующих в режиме реального времени.

Независимо от архитектуры ядро Linux-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, одно-

временно 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 Linux 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 Linux.

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; управляют доступом процессов к периферийным устройствам, таким как терминалы, накопители на магнитных лентах и сетевое оборудование.

С точки зрения пользователя функции ядра можно представить схемой, показанной на рис. 3.2.

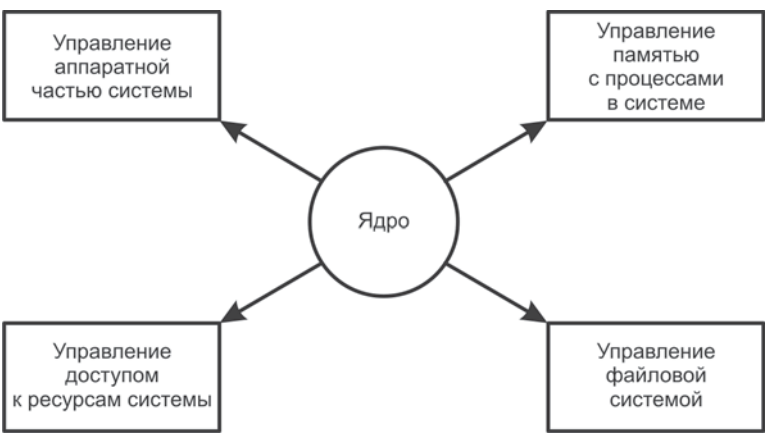

Рис. 3.2

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, реализуются через системные вызовы ядра.

Здесь нужно сделать небольшое отступление и чуть более подробно остановиться на некоторых терминах и понятиях, нередко используемых при анализе функционирования операционной системы Linux. 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 Linux 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. 3.3.

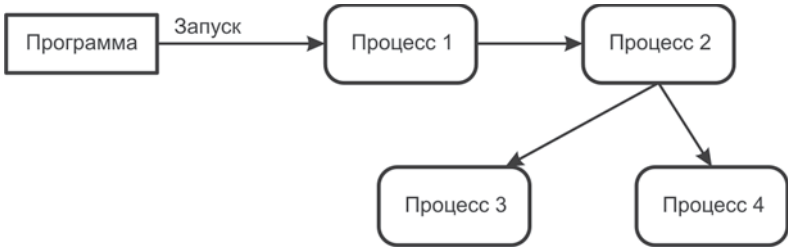

Рис. 3.3

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 Linux, 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 Linux предоставляет процессам, работающим в режиме пользователя (User Mode), набор интерфейсов для взаимодействия с аппаратными устройствами, такими как процес-