MyOffice Plus

Руководство по администрированию

ОПЕРАЦИОННАЯ СИСТЕМА АЛЬТ СЕРВЕР 10.0

ОПЕРАЦИОННАЯ СИСТЕМА

АЛЬТ СЕРВЕР 10.0

РУКОВОДСТВО ПО АДМИНИСТРИРОВАНИЮ

1.0

На 49 листах

Москва
2022

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем. Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Сокращения, которые используются в настоящем документе, приведены в Таблице 1.

Таблица 1 – Сокращения и расшифровки

Сокращения	Расшифровка
ОС Альт Сервер	Операционная система «Альт Сервер»
ПЭВМ	Персональная электронно-вычислительная машина

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение MyOffice Plus

MyOffice Plus – комплексный продукт для организации рабочей среды в крупных государственных организациях и коммерческих предприятиях. Единая лицензия МойОфис Plus включает операционную систему, средства безопасного хранения файлов, работы с документами, ведения переписки по электронной почте и планирования рабочего времени.

1.2 Описание ОС Альт Сервер

Операционная система «Альт Сервер» – многофункциональный дистрибутив для серверов с возможностью использования в качестве рабочей станции разработчика комплексных систем, прежде всего, предназначен для использования в корпоративных сетях.

ОС Альт Сервер обладает следующими функциональными характеристиками:

– обеспечивает возможность обработки, хранения и передачи информации;
– обеспечивает возможность функционирования в многозадачном режиме (одновременное выполнение множества процессов);
– обеспечивает возможность масштабирования системы: возможна эксплуатация ОС как на одной ПЭВМ, так и в информационных системах различной архитектуры;
– обеспечивает многопользовательский режим эксплуатации;
– обеспечивает поддержку мультипроцессорных систем;
– обеспечивает поддержку виртуальной памяти;
– обеспечивает поддержку запуска виртуальных машин;
– обеспечивает сетевую обработку данных, в том числе разграничение доступа к сетевым пакетам.

ОС Альт Сервер состоит из набора компонентов предназначенных для реализации функциональных задач необходимых пользователям (должностным лицам для выполнения определённых должностными инструкциями, повседневных действий) и поставляется в виде дистрибутива и комплекта эксплуатационной документации.

В структуре ОС Альт Сервер можно выделить следующие функциональные элементы:

– ядро ОС;

– системные библиотеки;
– утилиты и драйверы;
– средства обеспечения информационной безопасности;
– системные приложения;
– средства обеспечения облачных и распределенных вычислений, средства виртуализации и системы хранения данных;
– системы мониторинга и управления;
– средства подготовки исполнимого кода;
– средства версионного контроля исходного кода;
– библиотеки подпрограмм (SDK);
– среды разработки, тестирования и отладки;
– интерактивные рабочие среды;
– программные серверы;
– веб-серверы;
– системы управления базами данных;
– графическая оболочка MATE;
– командные интерпретаторы;
– прикладное программное обеспечение общего назначения;
– офисные приложения.

Ядро ОС Альт Сервер управляет доступом к оперативной памяти, сети, дисковым и прочим внешним устройствам. Оно запускает и регистрирует процессы, управляет разделением времени между ними, реализует разграничение прав и определяет политику безопасности, обойти которую, не обращаясь к нему, нельзя.

Ядро работает в режиме «супервизора», позволяющем ему иметь доступ сразу ко всей оперативной памяти и аппаратной таблице задач. Процессы запускаются в «режиме пользователя»: каждый жестко привязан ядром к одной записи таблицы задач, в которой, в числе прочих данных, указано, к какой именно части оперативной памяти этот процесс имеет доступ. Ядро постоянно находится в памяти, выполняя системные вызовы – запросы от процессов на выполнение этих подпрограмм.

Системные библиотеки – наборы программ (пакетов программ), выполняющие различные функциональные задачи и предназначенные для динамического подключения к работающим программам, которым необходимо выполнение этих задач.

Серверные программы и приложения предоставляют пользователю специализированные услуги (почтовые службы, хранилище файлов, веб-сервер, система управления базой данных, обеспечение документооборота, хранилище данных пользователей и так далее) в локальной или глобальной сети и обеспечивают их выполнение.

В состав ОС Альт Сервер включены следующие серверные программы и приложения:

– приложения, обеспечивающие поддержку сетевого протокола DHCP (Dynamic Host Configuration Protocol);
– приложения, обеспечивающие поддержку протокола аутентификации LDAP (Lightweight Directory Access Protocol);
– приложения, обеспечивающие поддержку протоколов FTP, SFTP, SSHD;
– программы, обеспечивающие работу сервера виртуализации;
– программы, обеспечивающие работу SMB-сервера (Сервер файлового обмена);
– программы почтового сервера Postfix;
– программы прокси-сервера Squid;
– программы, обеспечивающие работу сервера совместной работы Sogo;
– программы, обеспечивающие работу сервера домена FreeIPA;
– программы менеджера виртуальных машин libvirt;
– программы веб-сервера Apache2;
– программы DNS-сервера.

В состав ОС «Альт Сервер» включены следующие дополнительные системные приложения:

– архиваторы;
– приложения для управления RPM-пакетами;
– приложения резервного копирования;
– приложения мониторинга системы;
– приложения для работы с файлами;
– приложения для настройки системы;
– настройка параметров загрузки;
– настройка оборудования;
– настройка сети.

Основные преимущества ОС Альт Сервер:

– установка серверных решений и решений конечных пользователей с одного диска;
– графическая рабочая среда MATE;
– возможность как развернуть, так и использовать только определённые службы без Alterator;
– возможность обеспечить единую аутентификацию, общие ресурсы и совместную работу через сервер каталогов.

2 ОБЩИЕ ПРИНЦИПЫ РАБОТЫ ОС

2.1 Процессы и файлы

ОС Альт Сервер является многопользовательской интегрированной системой. Это значит, что она разработана в расчете на одновременную работу нескольких пользователей.

Пользователь может либо сам работать в системе, выполняя некоторую последовательность команд, либо от его имени могут выполняться прикладные процессы.

Пользователь взаимодействует с системой через командный интерпретатор. Командный интерпретатор представляет собой прикладную программу, которая принимает от пользователя команды или набор команд и транслирует их в системные вызовы к ядру системы. Интерпретатор позволяет пользователю просматривать файлы, передвигаться по дереву файловой системы, запускать прикладные процессы. Все командные интерпретаторы UNIX имеют развитый командный язык и позволяют писать достаточно сложные программы, упрощающие процесс администрирования системы и работы с ней.

2.1.1 Процессы функционирования ОС

Все программы, которые выполняются в текущий момент времени, называются процессами. Процессы можно разделить на два основных класса: системные процессы и пользовательские процессы.

Системные процессы – программы, решающие внутренние задачи ОС, например, организацию виртуальной памяти на диске или предоставляющие пользователям те или иные сервисы (процессы-службы).

Пользовательские процессы – процессы, запускаемые пользователем из командного интерпретатора для решения задач пользователя или управления системными процессами. Linux изначально разрабатывался как многозадачная система. Он использует технологии, опробованные и отработанные другими реализациями UNIX, которые существовали ранее.

Фоновый режим работы процесса – режим, когда программа может работать без взаимодействия с пользователем. В случае необходимости интерактивной работы с пользователем (в общем случае) процесс будет «остановлен» ядром, и работа его продолжается только после переведения его в «нормальный» режим работы.

2.1.2 Файловая система ОС

ОС использована файловая система Linux, которая, в отличие от файловых систем DOS и Windows(™), является единым деревом. Корень этого дерева – каталог, называемый root (рут) и обозначаемый «/».

Части дерева файловой системы могут физически располагаться в разных разделах разных дисков или вообще на других компьютерах – для пользователя это прозрачно. Процесс присоединения файловой системы раздела к дереву называется монтированием, удаление – размонтированием. Например, файловая система CD-ROM в дистрибутиве монтируется по умолчанию в каталог /**media/cdrom** (путь в дистрибутиве обозначается с использованием «/», а не «\», как в DOS/Windows).

Текущий каталог обозначается «./».

2.1.3 Структура каталогов

Корневой каталог /:

– **/bin** – командные оболочки (shell), основные утилиты;
– **/boot** – содержит ядро системы;
– **/dev** – псевдофайлы устройств, позволяющие работать с устройствами напрямую. Файлы в /dev создаются сервисом udev;
– **/etc** – общесистемные конфигурационные файлы для большинства программ в системе;
– **/etc/rc?.d**, **/etc/init.d**, **/etc/rc.boot**, **/etc/rc.d** – каталоги, где расположены командные файлы, выполняемые при запуске системы или при смене её режима работы;
– **/etc/passwd** – база данных пользователей, в которой содержится информация об имени пользователя, его настоящем имени, личном каталоге, его зашифрованный пароль и другие данные;
– **/etc/shadow** – теневая база данных пользователей. При этом информация из файла /**etc/passwd** перемещается в **/etc/shadow**, который недоступен для чтения всем, кроме пользователя root. В случае использования альтернативной схемы управления теневыми паролями (TCB), все теневые пароли для каждого пользователя располагаются в каталоге /etc/tcb/имя пользователя/shadow;
– **/home** – домашние каталоги пользователей;

– **/lib** – содержит файлы динамических библиотек, необходимых для работы большей части приложений, и подгружаемые модули ядра;
– **/lost+found** – восстановленные файлы;
– **/media** – подключаемые носители (каталоги для монтирования файловых систем сменных устройств);
– **/mnt** – точки временного монтирования;
– **/opt** – вспомогательные пакеты;
– **/proc** – виртуальная файловая система, хранящаяся в памяти компьютера при загруженной ОС. В данном каталоге расположены самые свежие сведения обо всех процессах, запущенных на компьютере;
– **/root** – домашний каталог администратора системы; /run –□файлы состояния приложений;
– **/sbin** – набор программ для административной работы с системой (системные утилиты);
– **/selinux** – виртуальная файловая система SELinux;
– **/srv** – виртуальные данные сервисных служб;
– **/sys** – файловая система, содержащая информацию о текущем состоянии системы;
– **/tmp** – временные файлы;
– **/usr** – пользовательские двоичные файлы и данные, используемые только для чтения (программы и библиотеки);
– **/var** – файлы для хранения изменяющихся данных (рабочие файлы программ, очереди, журналы).

Каталог **/usr**:
– **/usr/bin** – дополнительные программы для всех учетных записей;
– **/usr/sbin** – команды, используемые при администрировании системы и не предназначенные для размещения в файловой системе root;
– **/usr/local** – место, где рекомендуется размещать файлы, установленные без использования пакетных менеджеров, внутренняя организация каталогов практически такая же, как и корневого каталога;
– **/usr/man** – каталог, где хранятся файлы справочного руководства man;
– **/usr/share** – каталог для размещения общедоступных файлов большей части приложений.

Каталог /var:

– **/var/log** – место, где хранятся файлы аудита работы системы и приложений;
– **/var/spool** – каталог для хранения файлов, находящихся в очереди на обработку для того или иного процесса (очереди печати, непрочитанные или не отправленные письма, задачи cron т.д.).

2.1.4 Организация файловой структуры

Система домашних каталогов пользователей помогает организовывать безопасную работу пользователей в многопользовательской системе. Вне своего домашнего каталога пользователь обладает минимальными правами (обычно чтение и выполнение файлов) и не может нанести ущерб системе, например, удалив или изменив файл.

Кроме файлов, созданных пользователем, в его домашнем каталоге обычно содержатся персональные конфигурационные файлы некоторых программ.

Маршрут (путь) – это последовательность имён каталогов, представляющая собой путь в файловой системе к данному файлу, где каждое следующее имя отделяется от предыдущего наклонной чертой (слешем). Если название маршрута начинается со слеша, то путь в искомый файл начинается от корневого каталога всего дерева системы. В обратном случае, если название маршрута начинается непосредственно с имени файла, то путь к искомому файлу должен начаться от текущего каталога (рабочего каталога).

Имя файла может содержать любые символы за исключением косой черты (/). Однако следует избегать применения в именах файлов большинства знаков препинания и непечатаемых символов. При выборе имен файлов рекомендуется ограничиться следующими символами:

– **строчные** и **ПРОПИСНЫЕ** буквы. Следует обратить внимание на то, что регистр всегда имеет значение;
– символ подчеркивания (_);
– точка (.).

Для удобства работы точку можно использовать для отделения имени файла от расширения файла. Данная возможность может быть необходима пользователям или некоторым программам, но не имеет значение для shell.

2.1.5 Имена дисков и разделов

Все физические устройства вашего компьютера отображаются в каталог **/dev** файловой системы дистрибутива (об этом – ниже). Диски (в том числе IDE/SATA/SCSI/SAS жёсткие диски, USBдиски) имеют имена:

– **/dev/sda** – первый диск;

– **/dev/sdb** – второй диск;

– и т.д.

Диски обозначаются **/dev/sdX**, где X – a, b, c, d, e, … в зависимости от порядкового номера диска на шине.

Раздел диска обозначается числом после его имени. Например, **/dev/sdb4** – четвертый раздел второго диска.

2.1.6 Разделы, необходимые для работы ОС

Для работы ОС на жестком диске (дисках) должны быть созданы, по крайней мере, два раздела: корневой (то есть тот, который будет содержать каталог /) и раздел подкачки (swap). Размер последнего, как правило, составляет от однократной до двукратной величины оперативной памяти компьютера. Если на диске много свободного места, то можно создать отдельные разделы для каталогов **/usr**, **/home**, **/var**.

2.2 Работа с наиболее часто используемыми компонентами

2.2.1 Виртуальная консоль

Система Альт Сервер предоставляет доступ к виртуальным консолям, с которых можно осуществлять одновременно несколько сеансов работы в системе (login session).

Только что установленная система Альт Сервер, возможно, предоставляет доступ только к первым шести виртуальным консолям, к которым можно обращаться, нажимая комбинации клавиш **Alt** + **F1** – **Alt** + **F6** (**Ctrl** + **Alt** + **F1** – **Ctrl** + **Alt** + **F6**).

2.2.2 Командные оболочки (интерпретаторы)

Для управления ОС используются командные интерпретаторы (shell).

Зайдя в систему, Вы увидите приглашение – строку, содержащую символ «$» (далее этот символ будет обозначать командную строку). Программа ожидает ваших команд. Роль командного интерпретатора – передавать ваши команды операционной системе. По своим функциям он соответствует **command.com** в DOS, но несравненно мощнее. При помощи командных интерпретаторов можно писать небольшие программы – сценарии (скрипты). В Linux доступны следующие командные оболочки:

– **bash** – самая распространенная оболочка под linux. Она ведет историю команд и предоставляет возможность их редактирования;

– **pdksh** – клон korn shell, хорошо известной оболочки в UNIX™ системах.

Проверить, какая оболочка используется в данный момент можно, выполнив команду:

```
$ echo $SHELL
```

Оболочкой по умолчанию является Bash (Bourne Again Shell) – самая распространённая оболочка под Linux, которая ведет историю команд и предоставляет возможность их редактирования. В дальнейшем описании работы с Альт Сервер будут использоваться примеры с использованием этой оболочки.

2.2.3 Командная оболочка Bash

В Bash имеется несколько приемов для работы со строкой команд. Например, можно использовать следующие сочетания:

– **Ctrl** + **A** – перейти на начало строки;

– **Ctrl** + **U** – удалить текущую строку;

– **Ctrl** + **C** – остановить текущую задачу.

Для ввода нескольких команд одной строкой можно использовать разделитель «;». По истории команд можно перемещаться с помощью клавиш ↑ («вверх») и ↓ («вниз»).

Чтобы найти конкретную команду в списке набранных, не пролистывая всю историю, можно нажать Ctrl+R и начать вводить символы ранее введенной команды.

Для просмотра истории команд можно воспользоваться командой history. Команды, присутствующие в истории, отображаются в списке пронумерованными. Чтобы запустить конкретную команду необходимо набрать:

```
!номер команды
```

Если ввести:

```
!!
```

запустится последняя из набранных команд.

В Bash имеется возможность самостоятельного завершения имен команд из общего списка команд, что облегчает работу при вводе команд, в случае, если имена программ и команд слишком длинны. При нажатии клавиши Tab Bash завершает имя команды, программы или каталога, если не существует нескольких альтернативных вариантов. Например, чтобы использовать программу декомпрессии **gunzip**, можно набрать следующую команду:

```
gu
```

Затем нажать клавишу **Tab**. Так как в данном случае существует несколько возможных вариантов завершения команды, то необходимо повторно нажать клавишу **Tab**, чтобы получить список имен, начинающихся с **gu**.

В предложенном примере можно получить следующий список:

```
$ gu
guile gunzip gupnp-binding-tool
```

Если набрать: **n** (**gunzip** – это единственное имя, третьей буквой которого является «n»), а затем нажать клавишу **Tab**, то оболочка самостоятельно дополнит имя. Чтобы запустить команду нужно нажать **Enter**.

Программы, вызываемые из командной строки, Bash ищет в каталогах, определяемых в системной переменной **$PATH**. По умолчанию в этот перечень каталогов не входит текущий каталог, обозначаемый ./ (точка слеш) (если только не выбран один из двух самых слабых уровней защиты). Поэтому, для запуска программы из текущего каталога, необходимо использовать команду (в примере запускается команда **prog**):

```
./prog
```

2.2.4 Команда

Простейшая команда состоит из одного «слова», например, команда cal, выводящая календарь на текущий месяц.

```
 $ cal
     Май 2021
Пн Вт Ср Чт Пт Сб Вс
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

2.2.5 Команда и параметры

```
$ cal 1 2022
 Январь 2022
Пн Вт Ср Чт Пт Сб Вс
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

Команда **cal 1 2022** состоит из двух частей – собственно команды **cal** и «остального». То, что следует за командой называется параметрами (или аргументами) и они вводятся для изменения поведения команды. В большинстве случаев, первое слово считается именем команды, а остальные – её параметрами.

2.2.6 Команда и ключи

Для решения разных задач одни и те же действия необходимо выполнять по-разному. Например, для синхронизации работ в разных точках земного шара лучше использовать единое для всех время (по Гринвичу), а для организации собственного рабочего дня – местное время (с учётом сдвига по часовому поясу и разницы зимнего и летнего времени). И то, и другое время показывает команда **date**, только для работы по Гринвичу ей нужен дополнительный параметр **- u** (он же **--universal**).

```
$ date
Ср мая 5 11:57:57 EET 2021
$ date -u
Ср мая 5 09:58:04 UTC 2021
```

Такого рода параметры называются ключами или модификаторами выполнения. Ключ принадлежит данной конкретной команде и сам по себе смысла не имеет. Этим он отличается от других параметров (например, имён файлов, чисел), имеющих собственный смысл, не зависящий ни от какой команды. Каждая команда может распознавать некоторый набор ключей и соответственно изменять своё поведение. Один и тот же ключ может определять для разных команд совершенно разные значения.

Для формата ключей нет жёсткого стандарта, однако существуют договорённости:

– Если ключ начинается на «-», то это *однобуквенный ключ*. За «-», как правило, следует один символ, чаще всего буква, обозначающая действие или свойство, которое этот ключ придаёт команде. Так проще отличать ключи от других параметров.
– Если ключ начинается на «--», то он называется полнословным ключом. Полнословный формат ключа начинается на два знака «--», за которыми следует полное имя обозначаемого этим ключом содержания.

Некоторые ключи имеют и однобуквенный, и полнословный формат, а некоторые – только полнословный.

Информацию о ресурсах каждой команды можно получить, используя ключ **--help**. К примеру, получить подсказку о том, что делает команда rm, можно, набрав в терминале **rm --help**.

2.2.7 Обзор основных команд системы

Все команды, приведенные ниже, могут быть запущены в режиме консоли. Для получения более подробной информации используйте команду man. Пример:

```
$ man ls
```

Параметры команд обычно начинаются с символа «-», и обычно после одного символа «-» можно указать сразу несколько опций. Например, вместо команды **ls -l -F** можно ввести команду **ls -lF**.

2.2.7.1 Учетные записи пользователей

Команда su

Команда su позволяет изменить «владельца» текущего сеанса (сессии) без необходимости завершать сеанс и открывать новый.

Синтаксис:

```
su [ОПЦИИ...] [ПОЛЬЗОВАТЕЛЬ]
```

Команду можно применять для замены текущего пользователя на любого другого, но чаще всего она используется для получения пользователем прав суперпользователя (root).

При вводе команды **su -**, будет запрошен пароль суперпользователя (root), и, в случае ввода корректного пароля, пользователь получит права администратора. Чтобы вернуться к правам пользователя, необходимо ввести команду:

```
exit
```

Более подробную информацию о режиме суперпользователя вы можете прочитать в главе Режим суперпользователя.

Команда id

Команда **id** выводит информацию о пользователе и группах, в которых он состоит для заданного пользователя или о текущем пользователе (если ничего не указано). Синтаксис:

```
id [ОПЦИИ...] [ПОЛЬЗОВАТЕЛЬ]
```

Команда passwd

Команда **passwd** меняет (или устанавливает) пароль, связанный с входным_именем пользователя.

Обычный пользователь может менять только пароль, связанный с его собственным входным_именем.

Команда запрашивает у обычных пользователей старый пароль (если он был), а затем дважды запрашивает новый. Новый пароль должен соответствовать техническим требованиям к паролям, заданным администратором системы.

2.2.7.2 Основные операции с файлами и каталогами

Команда ls

Команда ls (list) печатает в стандартный вывод содержимое каталогов. Синтаксис:

```
ls [ОПЦИИ...] [ФАЙЛ...]
```

Основные опции:

- **-a** – просмотр всех файлов, включая скрытые;
- **-l** – отображение более подробной информации;
- **-R** – выводить рекурсивно информацию о подкаталогах.

Команда cd

Команда **cd** предназначена для смены каталога. Команда работает как с абсолютными, так и с относительными путями. Если каталог не указан, используется значение переменной окружения **$HOME** (домашний каталог пользователя). Если каталог задан полным маршрутным именем, он становится текущим. По отношению к новому каталогу нужно иметь право на выполнение, которое в данном случае трактуется как разрешение на поиск.

Синтаксис:

```
cd [-L|-P] [КАТАЛОГ]
```

Если в качестве аргумента задано «-», то это эквивалентно **$OLDPWD**. Если переход был осуществлен по переменной окружения **$CDPATH** или в качестве аргумента был задан «-» и смена каталога была успешной, то абсолютный путь нового рабочего каталога будет выведен на стандартный вывод.

Пример. Находясь в домашнем каталоге перейти в его подкаталог **docs/** (относительный путь):

```
cd docs/
```

Сделать текущим каталог **/usr/bin** (абсолютный путь):

```
cd /usr/bin/
```

Сделать текущим родительский каталог:

```
cd ..
```

Вернуться в предыдущий каталог:

```
cd -
```

Сделать текущим домашний каталог:

```
cd
```

Команда pwd

Команда pwd выводит абсолютный путь текущего (рабочего) каталога.

Синтаксис:

```
pwd [-L|-P]
```

Опции:

- **-P** – не выводить символические ссылки;
- **-L** – выводить символические ссылки.

Команда rm

Команда rm служит для удаления записей о файлах. Если заданное имя было последней ссылкой на файл, то файл уничтожается.

Удалив файл, вы не сможете его восстановить!

Синтаксис:

```
rm [ОПЦИИ...] <ФАЙЛ>
```

Основные опции:

- **-f** – никогда не запрашивать подтверждения;
- **-i** – всегда запрашивать подтверждение;
- **-r, -R** – рекурсивно удалять содержимое указанных каталогов.

Пример. Удалить все файлы html в каталоге ~/html:

```
rm –i ~/html/*.html
```

Команда mkdir

mkdir –команда для создания новых каталогов. Синтаксис:

```
mkdir [-p] [-m права] <КАТАЛОГ...>
```

Команда rmdir

Команда rmdir удаляет каталоги из файловой системы. Каталог должен быть пуст перед удалением. Синтаксис:

```
rmdir [ОПЦИИ] <КАТАЛОГ...>
```

Основные опции:

- **-p** – удалить каталог и его потомки.

Команда **rmdir** часто заменяется командой **rm -rf**, которая позволяет удалять каталоги, даже если они не пусты.

Команда cp

Команда **cp** предназначена для копирования файлов из одного в другие каталоги. Синтаксис:

```
cp [-fip] [ИСХ_ФАЙЛ...] [ЦЕЛ_ФАЙЛ...]
```

```
cp [-fip] [ИСХ_ФАЙЛ...] [КАТАЛОГ]
```

```
cp [-R] [[-H] | [-L] | [-P]] [-fip] [ИСХ_ФАЙЛ...] [КАТАЛОГ]
```

Основные опции:

- **-p** – сохранять по возможности времена изменения и доступа к файлу, владельца и группу, права доступа;
- **-i** – запрашивать подтверждение перед копированием в существующие файлы;
- **-r, -R** – рекурсивно копировать содержимое каталогов.

Команда mv

Команда **mv** предназначена для перемещения файлов. Синтаксис:

```
mv [-fi] [ИСХ_ФАЙЛ...] [ЦЕЛ_ФАЙЛ...]
```

```
mv [-fi] [ИСХ_ФАЙЛ...] [КАТАЛОГ]
```

В первой синтаксической форме, характеризующейся тем, что последний операнд не является ни каталогом, ни символической ссылкой на каталог, **mv** перемещает исх_файл в цел_файл (происходит переименование файла).

Во второй синтаксической форме mv перемещает исходные файлы в указанный каталог под именами, совпадающими с краткими именами исходных файлов.

Основные опции:

- **-f** – не запрашивать подтверждения перезаписи существующих файлов;
- **-i** – запрашивать подтверждение перезаписи существующих файлов.

Команда cat

Команда cat последовательно выводит содержимое файлов. Синтаксис:

```
cat [ОПЦИИ] [ФАЙЛ...]
```

Основные опции:

- **-n, --number** – нумеровать все строки при выводе;

- **-E, --show-ends** – показывать $ в конце каждой строки.

Если файл не указан, читается стандартный ввод. Если в списке файлов присутствует имя «-», вместо этого файла читается стандартный ввод.

Команда head

Команда head выводит первые 10 строк каждого файла на стандартный вывод. Синтаксис:

```
head [ОПЦИИ] [ФАЙЛ...]
```

Основные опции:

- **-n, --lines=[-]K** – вывести первые **K** строк каждого файла, а не первые 10;
- **-q, --quiet** – не печатать заголовки с именами файлов.

Команда less

Команда **less** позволяет постранично просматривать текст (для выхода необходимо нажать q). Синтаксис:

```
less ФАЙЛ
```

Команда grep

Команда grep имеет много опций и предоставляет возможности поиска символьной строки в файле. Синтаксис:

```
grep [шаблон_поиска] ФАЙЛ
```

Команда chmod

Команда **chmod** предназначена для изменения прав доступа файлов и каталогов. Синтаксис:

```
chmod [ОПЦИИ] РЕЖИМ[,РЕЖИМ]... <ФАЙЛ>
```

```
chmod [ОПЦИИ] --reference=ИФАЙЛ <ФАЙЛ>
```

Основные опции:

- **-R** – рекурсивно изменять режим доступа к файлам, расположенным в указанных каталогах;
- **--reference=ИФАЙЛ** – использовать режим файла ИФАЙЛ.

chmod изменяет права доступа каждого указанного файла в соответствии с правами доступа, указанными в параметре режим, который может быть представлен как в символьном виде, так и в виде восьмеричного, представляющего битовую маску новых прав доступа.

Формат символьного режима следующий:

```
[ugoa...][[+-=][разрешения...]...]
```

Здесь разрешения – это ноль или более букв из набора «rwxXst» или одна из букв из набора «ugo».

Каждый аргумент – это список символьных команд изменения прав доступа, разделены запятыми. Каждая такая команда начинается с нуля или более букв «ugoa», комбинация которых указывает, чьи права доступа к файлу будут изменены: пользователя, владеющего файлом (u), пользователей, входящих в группу, к которой принадлежит файл (g), остальных пользователей (o) или всех пользователей (a). Если не задана ни одна буква, то автоматически будет использована буква «a», но биты, установленные в umask, не будут затронуты.

Оператор «+» добавляет выбранные права доступа к уже имеющимся у каждого файла, «-» удаляет эти права. «=» присваивает только эти права каждому указанному файлу.

Буквы «rwxXst» задают биты доступа для пользователей: «r» – чтение, «w» – запись, «x» – выполнение (или поиск для каталогов), «X» – выполнение/поиск только если это каталог или же файл с уже установленным битом выполнения, «s» – задать ID пользователя и группы при выполнении, «t» – запрет удаления.

Примеры. Позволить всем выполнять файл f2:

```
chmod +x f2
```

Запретить удаление файла f3:

```
chmod +t f3
```

Команда chown

Команда **chown** изменяет владельца и/или группу для каждого заданного файла. Синтаксис:

```
chown [КЛЮЧ]…[ВЛАДЕЛЕЦ][:[ГРУППА]] <ФАЙЛ>
```

Изменить владельца может только владелец файла или суперпользователь. Владелец не изменяется, если он не задан в аргументе. Группа также не изменяется, если не задана, но если после символьного ВЛАДЕЛЬЦА стоит символ «:», подразумевается изменение группы на основную группу текущего пользователя. Поля ВЛАДЕЛЕЦ и ГРУППА могут быть как числовыми, так и символьными.

Примеры. Поменять владельца каталога /u на пользователя test:

```
chown test /u
```

Поменять владельца и группу каталога /u:

```
chown test:staff /u
```

Поменять владельца каталога /u и вложенных файлов на test:

```
chown –hR test /u
```

2.2.7.3 Поиск файлов

Команда find

Команда **find** предназначена для поиска всех файлов, начиная с корневого каталога. Поиск может осуществляться по имени, типу или владельцу файла.

Синтаксис:

```
find [-H] [-L] [-P] [-Оуровень] [-D help|tree|search|stat|rates|opt| exec]
[ПУТЬ…] [ВЫРАЖЕНИЕ]
```

Ключи для поиска:

- **-name** – поиск по имени файла;
- **-type** – поиск по типу f=файл, d=каталог, l=ссылка(lnk);
- **-user** – поиск по владельцу (имя или UID).

Когда выполняется команда **find**, можно выполнять различные действия над найденными файлами. Основные действия:

- -exec команда \; – выполнить команду. Запись команды должна заканчиваться экранированной точкой с запятой. Строка «{}» заменяется текущим маршрутным именем файла;
- **execdir команда \;** – то же самое что и **-exec**, но команда вызывается из подкаталога, содержащего текущий файл;
- **-ok команда** – эквивалентно **-exec** за исключением того, что перед выполнением команды запрашивается подтверждение (в виде сгенерированной командной строки со знаком вопроса в конце) и она выполняется только при ответе: y;
- **-print** – вывод имени файла на экран.

Путем по умолчанию является текущий подкаталог. Выражение по умолчанию **-print**.

Примеры. Найти в текущем каталоге обычные файлы (не каталоги), имя которых начинается с символа «~»:

```
find . -type f -name "~*" -print
```

Найти в текущем каталоге файлы, измененные позже, чем файл **file.bak**:

```
find . -newer file.bak -type f -print
```

Удалить все файлы с именами **a.out** или ***.o**, доступ к которым не производился в течение недели:

```
find / \( -name a.out -o -name '*.o' \) \ -atime +7 -exec rm {} \;
```

Удалить из текущего каталога и его подкаталогов все файлы нулевого размера, запрашивая подтверждение:

```
find . -size 0c -ok rm {} \;
```

Команда whereis

whereis сообщает путь к исполняемому файлу программы, ее исходным файлам (если есть) и соответствующим страницам справочного руководства.

Синтаксис:

```
whereis [ОПЦИИ] <ИМЯ>
```

Опции:

- **-b** – вывод информации только об исполняемых файлах;
- **-m** – вывод информации только о страницах справочного руководства;
- **-s** – вывод информации только об исходных файлах.

2.2.7.4 Мониторинг и управление процессами

Команда ps

Команда ps отображает список текущих процессов.

Синтаксис:

```
ps [ОПЦИИ]
```

По умолчанию выводится информация о процессах с теми же действующим UID и управляющим терминалом, что и у подающего команду пользователя.

Основные опции:

- **-a** – вывести информацию о процессах, ассоциированных с терминалами;
- **-f** – вывести «полный» список;
- **-l** – вывести «длинный» список;
- **-p список** – вывести информацию о процессах с перечисленными в списке PID;
- **-u список** – вывести информацию о процессах с перечисленными идентификаторами или именами пользователей.

Команда kill

Команда **kill** позволяет прекратить исполнение процесса или передать ему сигнал.

Синтаксис:

```
kill [-s] [сигнал] [идентификатор] [...]
```

```
kill [-l] [статус_завершения]
```

```
kill [-номер_сигнала] [идентификатор] [...]
```

Идентификатор – PID ведущего процесса задания или номер задания, предварённый знаком «%».

Основные опции:

- **-l** – вывести список поддерживаемых сигналов;
- **-s сигнал**, **-сигнал** – послать сигнал с указанным именем.

Если обычная команда **kill** не дает желательного эффекта, необходимо использовать команду **kill** с параметром **-9** (**kill -9** PID_номер).

Команда df

Команда **df** показывает количество доступного дискового пространства в файловой системе, в которой содержится файл, переданный как аргумент. Если ни один файл не указан, показывается доступное место на всех смонтированных файловых системах.

Размеры по умолчанию указаны в блоках по 1КБ.

Синтаксис:

```
df [ОПЦИИ] [ФАЙЛ...]
```

Основные опции:

- **--total** – подсчитать общий объем в конце;
- **-h, --human-readable** – печатать размеры в удобочитаемом формате (например, 1K, 234M, 2G).

Команда du

Команда du подсчитывает использование диска каждым файлом, для каталогов подсчет происходит рекурсивно.

Синтаксис:

```
du [ОПЦИИ] [ФАЙЛ...]
```

Основные опции:

- **-a, --all** – выводить общую сумму для каждого заданного файла, а не только для каталогов;
- **-c, --total** – подсчитать общий объем в конце. Может быть использовано для выяснения суммарного использования дискового пространства для всего списка заданных файлов;

- **-d, --max-depth=N** – выводить объем для каталога (или файлов, если указано **--all**) только если она на N или менее уровней ниже аргументов командной строки;
- **-S, --separate-dirs** – выдавать отдельно размер каждого каталога, не включая размеры подкаталогов;
- **-s, --summarize** – отобразить только сумму для каждого аргумента.

Команда which

Команда **which** отображает полный путь к указанным командам или сценариям.

Синтаксис:

```
which [ОПЦИИ] <ФАЙЛ...>
```

Основные опции:

- **-a, --all** – выводит все совпавшие исполняемые файлы по содержимому в переменной окружения $PATH, а не только первый из них;
- **-c, --total** – подсчитать общий объем в конце. Может быть использовано для выяснения суммарного использования дискового пространства для всего списка заданных файлов;
- **-d, --max-depth=N** – выводить объем для каталога (или файлов, если указано -- all) только если она на N или менее уровней ниже аргументов командной строки;
- **-S, --separate-dirs** – выдавать отдельно размер каждого каталога, не включая размеры подкаталогов;
- **--skip-dot** – пропускает все каталоги из переменной окружения $PATH, которые начинаются с точки.

2.2.7.5 Использование многозадачности

Альт Сервер – это многозадачная система.

Для того, чтобы запустить программу в фоновом режиме, необходимо набрать «&» после имени программы. После этого оболочка даст возможность запускать другие приложения.

Так как некоторые программы интерактивны – их запуск в фоновом режиме бессмысленен. Подобные программы просто остановятся, если их запустить в фоновом режиме.

Можно также запускать нескольких независимых сеансов. Для этого в консоли необходимо набрать **Alt** и одну из клавиш, находящихся в интервале от **F1** до **F6**. На экране появится новое приглашение системы, и можно открыть новый сеанс. Этот метод также позволяет вам работать на другой консоли, если консоль, которую вы использовали до этого, не отвечает или вам необходимо остановить зависшую программу.

Команда bg

Команда **bg** позволяет перевести задание на задний план.

Синтаксис:

```
bg [ИДЕНТИФИКАТОР ...]
```

Идентификатор – PID ведущего процесса задания или номер задания, предварённый знаком «%».

Команда fg

Команда **fg** позволяет перевести задание на передний план.

Синтаксис:

```
fg [ИДЕНТИФИКАТОР ...]
```

Идентификатор – PID ведущего процесса задания или номер задания, предварённый знаком «%».

2.2.7.6 Сжатие и упаковка файлов

Команда tar

Сжатие и упаковка файлов выполняется с помощью команды **tar**, которая преобразует файл или группу файлов в архив без сжатия (tarfile).

Упаковка файлов в архив чаще всего выполняется следующей командой:

```
tar -cf [имя создаваемого файла архива] [упаковываемые файлы и/или
каталоги]
```

Пример использования команды упаковки архива:

```
tar -cf moi_dokumenti.tar Docs project.tex
```

Распаковка содержимого архива в текущий каталог выполняется командой:

```
tar -xf [имя файла архива]
```

Для сжатия файлов используются специальные программы сжатия: **gzip**, **bzip2** и **7z**.

2.3 Стыкование команд в системе Linux

2.3.1 Стандартный ввод и стандартный вывод

Многие команды системы имеют так называемые стандартный ввод (standard input) и стандартный вывод (standard output), часто сокращаемые до stdin и stdout. Ввод и вывод здесь – это входная и выходная информация для данной команды. Программная оболочка делает так, что стандартным вводом является клавиатура, а стандартным выводом – экран монитора.

Пример с использованием команды **cat**. По умолчанию команда cat читает данные из всех файлов, которые указаны в командной строке, и посылает эту информацию непосредственно в стандартный вывод (stdout). Следовательно, команда:

```
cat history-final masters-thesis
```

выведет на экран сначала содержимое файла **history-final**, а затем – файла **masters-thesis**.

Если имя файла не указано, программа **cat** читает входные данные из stdin и возвращает их в stdout. Пример:

```
cat
Hello there.
Hello there.
Bye.
Bye.
Ctrl-D
```

Каждую строку, вводимую с клавиатуры, программа **cat** немедленно возвращает на экран. При вводе информации со стандартного ввода конец текста сигнализируется вводом специальной комбинации клавиш, как правило, **Ctrl** + **D**. Сокращённое название сигнала конца текста – EOT (end of text).

2.3.2 Перенаправление ввода и вывода

При необходимости можно перенаправить стандартный вывод, используя символ >, и стандартный ввод, используя символ <.

Фильтр (filter) – программа, которая читает данные из стандартного ввода, некоторым образом их обрабатывает и результат направляет на стандартный вывод. Когда применяется перенаправление, в качестве стандартного ввода и вывода могут выступать файлы. Как указывалось выше, по умолчанию, stdin и stdout относятся к клавиатуре и к экрану соответственно. Программа sort является простым фильтром – она сортирует входные данные и посылает результат на стандартный вывод. Совсем простым фильтром является программа с **at** – она ничего не делает с входными данными, а просто пересылает их на выход.

2.3.3 Использование состыкованных команд

Стыковку команд (pipelines) осуществляет командная оболочка, которая stdout первой команды направляет на stdin второй команды. Для стыковки используется символ |. Направить stdout команды **ls** на stdin команды **sort**:

```
ls | sort -r
notes
masters-thesis
history-final
english-list
```

Вывод списка файлов частями:

```
ls /usr/bin | more
```

Если необходимо вывести на экран последнее по алфавиту имя файла в текущем каталоге, можно использовать следующую команду:

```
ls | sort -r | head -1 notes
```

где команда **head -1** выводит на экран первую строку получаемого ей входного потока строк (в примере поток состоит из данных от команды **ls**), отсортированных в обратном алфавитном порядке.

2.3.4 Недеструктивное перенаправление вывода

Эффект от использования символа > для перенаправления вывода файла является деструктивным; т.е, команда

```
ls > file-list
```

уничтожит содержимое файла **file-list**, если этот файл ранее существовал, и создаст на его месте новый файл. Если вместо этого перенаправление будет сделано с помощью символов >>, то вывод будет приписан в конец указанного файла, при этом исходное содержимое файла не будет уничтожено.

Перенаправление ввода и вывода и стыкование команд осуществляется командными оболочками, которые поддерживают использование символов >, >> и |. Сами команды не способны воспринимать и интерпретировать эти символы.

3 РЕЖИМ СУПЕРПОЛЬЗОВАТЕЛЯ

3.1 Типы пользователей

Linux – система многопользовательская, а потому пользователь – ключевое понятие для организации всей системы доступа в Linux. Файлы всех пользователей в Linux хранятся раздельно, у каждого пользователя есть собственный домашний каталог, в котором он может хранить свои данные. Доступ других пользователей к домашнему каталогу пользователя может быть ограничен.

Суперпользователь в Linux – это выделенный пользователь системы, на которого не распространяются ограничения прав доступа. Именно суперпользователь имеет возможность произвольно изменять владельца и группу файла. Ему открыт доступ на чтение и запись к любому файлу или каталогу системы.

Среди учётных записей Linux всегда есть учётная запись суперпользователя – root. Поэтому вместо «суперпользователь» часто говорят «root». Множество системных файлов принадлежат root, множество файлов только ему доступны для чтения или записи. Пароль этой учётной записи – одна из самых больших драгоценностей системы. Именно с её помощью системные администраторы выполняют самую ответственную работу.

3.2 Назначение режима суперпользователя

Системные утилиты, например, такие, как **Центр управления системой** или **Программа управления пакетами Synaptic** требуют для своей работы привилегий суперпользователя, потому что они вносят изменения в системные файлы. При их запуске выводится диалоговое окно с запросом пароля системного администратора.

3.3 Получение прав суперпользователя

Для опытных пользователей, умеющих работать с командной строкой, существует два различных способа получить права суперпользователя.

Первый – это зарегистрироваться в системе под именем root.

Второй способ – воспользоваться специальной утилитой su (shell of user), которая позволяет выполнить одну или несколько команд от лица другого пользователя. По умолчанию эта утилита выполняет команду sh от пользователя root, то есть запускает командный интерпретатор. Отличие от предыдущего способа в том, что всегда известно, кто именно запускал su, а значит, ясно, кто выполнил определённое административное действие.

В некоторых случаях удобнее использовать не su, а утилиту sudo, которая позволяет выполнять только заранее заданные команды.

Для того чтобы воспользоваться командами **su** и **sudo**, необходимо быть членом группы **wheel**. Пользователь, созданный при установке системы, по умолчанию уже включён в эту группу.

В дистрибутивах Альт для управления доступом к важным службам используется подсистема control. control – механизм переключения между неким набором фиксированных состояний для задач, допускающих такой набор.

Команда **control** доступна только для суперпользователя (root). Для того, чтобы посмотреть, что означает та или иная политика **control** (разрешения выполнения конкретной команды, управляемой **control**), надо запустить команду с ключом help:

```
# control su help
```

Запустив **control** без параметров, можно увидеть полный список команд, управляемых командой (facilities) вместе с их текущим состоянием и набором допустимых состояний.

3.4 Переход в режим суперпользователя

Для перехода в режим суперпользователя наберите в терминале команду **su -**.

Если воспользоваться командой **su** без ключа, то происходит вызов командного интерпретатора с правами **root**. При этом значение переменных окружения, в частности **$PATH**, остаётся таким же, как у пользователя: в переменной **$PATH** не окажется каталогов **/sbin**, **/usr/sbin**, без указания полного имени будут недоступны команды **route**, **shutdown**, **mkswap** и другие. Более того, переменная **$HOME** будет указывать на каталог пользователя, все программы, запущенные в режиме суперпользователя, сохранят свои настройки с правами **root** в каталоге пользователя, что в дальнейшем может вызвать проблемы.

Чтобы избежать этого, следует использовать **su -**. В этом режиме **su** запустит командный интерпретатор в качестве login shell, и он будет вести себя в точности так, как если бы в системе зарегистрировался **root**.

4 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

Пользователи и группы внутри системы обозначаются цифровыми идентификаторами – UID и GID, соответственно.

Пользователь может входить в одну или несколько групп. По умолчанию он входит в группу, совпадающую с его именем. Чтобы узнать, в какие еще группы входит пользователь, введите команду **id**, вывод её может быть примерно следующим:

```
uid=500(test) gid=500(test) группы=500(test),16(rpm)
```

Такая запись означает, что пользователь test (цифровой идентификатор 500) входит в группы test и rpm. Разные группы могут иметь разный уровень доступа к тем или иным каталогам; чем в большее количество групп входит пользователь, тем больше прав он имеет в системе.

В связи с тем, что большинство привилегированных системных утилит в дистрибутивах Альт имеют не SUID-, а SGID-бит, будьте предельно внимательны и осторожны в переназначении групповых прав на системные каталоги.

4.1 Команда passwd

Команда passwd поддерживает традиционные опции **passwd** и утилит **shadow**.

Синтаксис:

```
passwd [ОПЦИИ...] [ИМЯ ПОЛЬЗОВАТЕЛЯ]
```

Возможные опции:

- **-d --delete** – удалить пароль для указанной записи;
- **-f, --force** – форсировать операцию;
- **-k, --keep-tokens** – сохранить не устаревшие пароли;
- **-l, --lock** – блокировать указанную запись;
- **--stdin** – прочитать новые пароли из стандартного ввода;
- **-S, --status** – дать отчет о статусе пароля в указанной записи;
- **-u, --unlock** – разблокировать указанную запись;
- **-?, --help** – показать справку и выйти;
- **--usage** – дать короткую справку по использованию;

- **-V, --version** – показать версию программы и выйти.

Код выхода: при успешном завершении **passwd** заканчивает работу с кодом выхода *0*. Код выхода *1* означает, что произошла ошибка. Текстовое описание ошибки выводится на стандартный поток ошибок.

Пользователь может в любой момент поменять свой пароль. Единственное,что требуется для смены пароля знать текущий пароль.

Только суперпользователь может обновить пароль другого пользователя.

4.2 Добавления нового пользователя

Для добавления нового пользователя используйте команды useradd и passwd:

```
# useradd test1
# passwd test1
passwd: updating all authentication tokens for user test1.
You can now choose the new password or passphrase.
A valid password should be a mix of upper and lower case letters,
digits, and other characters. You can use an 8 character long
password with characters from at least 3 of these 4 classes, or
a 7 character long password containing characters from all the
classes. An upper case letter that begins the password and a
digit that ends it do not count towards the number of character
classes used.
A passphrase should be of at least 3 words, 11 to 40 characters
long, and contain enough different characters.
Alternatively, if no one else can see your terminal now, you can
pick this as your password: "holder5dinghy-Arm".
Enter new password:
```

В результате описанных действий в системе появился пользователь test1 с некоторым паролем. Если пароль оказался слишком слабым с точки зрения системы, она об этом предупредит (как в примере выше). Пользователь в дальнейшем может поменять свой пароль при помощи команды passwd – но если он попытается поставить слабый пароль, система откажет ему (в отличие от root) в изменении.

В ОС Альт Сервер для проверки паролей на слабость используется модуль PAM passwdqc.

Программа **useradd** имеет множество параметров, которые позволяют менять её поведение по умолчанию. Например, можно принудительно указать, какой будет UID или какой группе будет принадлежать пользователь.

4.3 Модификация пользовательских записей

Для модификации пользовательских записей применяется утилита usermod:

```
# usermod -G audio,rpm,test1 test1
```

Такая команда изменит список групп, в которые входит пользователь test1 – теперь это audio, rpm, test1.

```
# usermod -l test2 test1
```

Будет произведена смена имени пользователя с test1 на test2.

Команды **usermod -L test2** и **usermod -U test2** соответственно временно блокируют возможность входа в систему пользователю test2 и возвращают всё на свои места.

Изменения вступят в силу только при следующем входе пользователя в систему.

При неинтерактивной смене или задании паролей для целой группы пользователей используйте утилиту **chpasswd**. На стандартный вход ей следует подавать список, каждая строка которого будет выглядеть как имя:пароль.

4.4 Удаление пользователей

Для удаления пользователей используйте **userdel**.

Команда **userdel test2** удалит пользователя test2 из системы. Если будет дополнительно задан параметр -r, то будет уничтожен и домашний каталог пользователя. Нельзя удалить пользователя, если в данный момент он еще работает в системе.

5 СИСТЕМА ИНИЦИАЛИЗАЦИИ SYSTEMD И SYSVINIT

5.1 Запуск операционной системы

5.1.1 Запуск системы

Алгоритм запуска компьютера приблизительно такой:

1. BIOS компьютера.
2. Загрузчик системы (например, LILO, GRUB или другой). В загрузчике вы можете задать параметры запуска системы или выбрать систему для запуска.
3. Загружается ядро Linux.
4. Запускается на выполнение первый процесс в системе – init.

Ядром запускается самая первая программа в системе **init**. Её задачей является запуск новых процессов и повторный запуск завершившихся. Вы можете посмотреть, где расположился init в иерархии процессов вашей системы, введя команду **pstree**.

От конфигурации init зависит, какая система инициализации будет использована.

5.1.2 Система инициализации

Система инициализации – это набор скриптов, которые будут выполнены при старте системы.

Существуют разные системы инициализации, наиболее популярной системой являются **sysvinit** и ее модификации. **systemd** разрабатывается как замена для **sysVinit**.

В ОС Альт Сервер используется **sysvinit** (от System V init).

5.2 Системы инициализации systemd и sysvinit

5.2.1 Система инициализации sysvinit

System V – классическая схема инициализации, на которой базируются многие дистрибутивы. Привычна и довольно проста для понимания: **init** описывает весь процесс загрузки в своем конфигурационном файле **/etc/inittab**, откуда вызываются другие программы и скрипты на определенном этапе запуска.

5.2.2 Система инициализации systemd

systemd является альтернативной системой инициализации Linux, вобравшей в себя достоинства классического **System V init** и более современных **launchd** (OS X), **SMF** (Solaris) и **Upstart** (Ubuntu, Fedora), но при этом лишенной многих их недостатков. Он разрабатывался для обеспечения лучшего выражения зависимостей между службами, что позволяет делать одновременно больше работы при загрузке системы, и уменьшить время загрузки системы.

systemd (system daemon) реализует принципиально новый подход к инициализации и контролю работы системы. Одним из ключевых новшеств этого подхода является высокая степень параллелизации запуска служб при инициализации системы, что в перспективе позволяет добиться гораздо более высокой скорости, чем традиционный подход с последовательным запуском взаимозависимых служб. Другим важным моментом является контроль над точками монтирования (не-жизненно-важные файловые системы можно монтировать только при первом обращении к ним, не тратя на это время при инициализации системы) и устройствами (можно запускать и останавливать определенные службы и при появлении или удалении заданных устройств). Для отслеживания групп процессов используется механизм cgroups, который также может быть использован для ограничения потребляемых ими системных ресурсов.

Удобство **systemd** особенно заметно на компьютерах для домашнего пользования – когда пользователи включают и перезагружают компьютер ежедневно. В отличие от sysvinit, подвисание при запуске одного сервиса не приведет к остановке всего процесса загрузки.

5.3 Примеры команд управления службами, журнал в systemd

Обратите внимание, что команды **service** и **chkconfig** продолжат работать в мире **systemd** практически без изменений. Тем не менее, в этой таблице показано как выполнить те же действия с помощью встроенных утилит **systemctl**.

В Таблице 2 представлены команды управления службами.

Таблица 2 – Команды управления службами

Команды Sysvinit	Команды Systemd	Примечания
service frobozz start	systemctl start frobozz.service	Используется для запуска службы (не перезагружает постоянные)

Команды Sysvinit	Команды Systemd	Примечания
service frobozz stop	systemctl stop frobozz.service	Используется для остановки службы (не перезагружает постоянные) service frobozz restart systemctl restart frobozz.service Используется д
service frobozz reload	systemctl reload frobozz.service	Если поддерживается, перезагружает файлы конфигурации без прерывания незаконченных операций
service frobozz condrestart	systemctl condrestart frobozz.service	Перезапускает службу, если она уже работает
service frobozz status	systemctl status frobozz.service	Сообщает, запущена ли уже служба
s /etc/rc.d/init.d/	systemctl list-unit-files --type=service (preferred) ls /lib/systemd/system/*.service /etc/systemd/system/*.service	Используется для отображения списка служб, которые можно запустить или остановить. Используется для отображения списка всех служб.
chkconfig frobozz on	systemctl enable frobozz.service	Включает службу во время следующей перезагрузки, или любой другой триггер
chkconfig frobozz off	systemctl disable frobozz.service	Выключает службу во время следующей перезагрузки, или любой другой триггер
chkconfig frobozz	systemctl is-enabled frobozz.service	Используется для проверки, сконфигурирована ли служба для запуска в текущем окружении
chkconfig --list	systemctl list-unit-files --type=service(preferred) ls /etc/systemd/system/*.wants/	Выводит таблицу служб. В ней видно, на каких уровнях загрузки они (не) запускаются

Команды Sysvinit	Команды Systemd	Примечания
chkconfig frobozz --list	ls /etc/systemd/system/*.wants/ frobozz.service	Используется, для отображения на каких уровнях служба (не) запускается
chkconfig frobozz --add	systemctl daemon-reload	Используется, когда вы создаете новую службу или модифицируете любую конфигурацию

5.4 Журнал в systemd

В **systemd** включена возможность ведения системного журнала. Для чтения журнала следует использовать команду **journalctl**. По умолчанию, больше не требуется запуск службы **syslog**.

Вы можете запускать **journalctl** с разными ключами:

– **journalctl -b** – покажет сообщения только с текущей загрузки;

– **journalctl -f** – покажет только последние сообщения.

Так же вы можете посмотреть сообщения определенного процесса:

– **journalctl _PID=1** – покажет сообщения первого процесса (init).

Для ознакомления с прочими возможностями, читайте руководство по **journalctl**. Для этого используйте команду **man journalctl**.

6 ДОКУМЕНТАЦИЯ

Каждый объект системы Linux обязательно сопровождается документацией, описывающей их назначение и способы использования. От пользователя системы не требуется заучивать все возможные варианты взаимодействия с ней. Достаточно понимать основные принципы её устройства и уметь находить справочную информацию.

Не пренебрегайте чтением документации: она поможет вам избежать многих сложностей, сэкономить массу времени и усилий при установке, настройке и администрировании системы, поможет найти нужное для работы приложение и быстро разобраться в нём.

6.1 Экранная документация

Почти все системы семейства UNIX, включая систему Linux, имеют экранную документацию. Её тексты содержат документацию по системным командам, ресурсам, конфигурационным файлам и т. д., а также могут быть выведены на экран в процессе работы.

6.1.1 Команда man

Для доступа к экранной документации используется команда man (сокращение от manual). Каждая страница руководства посвящена одному объекту системы. Для того чтобы прочесть страницу руководства по программе, необходимо набрать man название_программы. К примеру, если вы хотите узнать, какие опции есть у команды date, вы можете ввести команду:

```
$ man date
```

Большинство экранной документации написано для пользователей, имеющих некоторое представление о том, что делает данная команда. Поэтому большинство текстов экранной документации содержит исключительно технические детали команды без особых пояснений. Тем не менее, экранная документация оказывается очень ценной в том случае, если вы помните название команды, но её синтаксис просто выпал у вас из памяти.

Поиск по описаниям **man** осуществляется командой **apropos**. Если вы точно не знаете, как называется необходимая вам программа, то поиск осуществляется по ключевому слову, к примеру, **apropos** *date* или при помощи ввода слова, обозначающего нужное действие, после команды **man -k** (например, **man -k** *copy*). Слово, характеризующее желаемое для вас действие, можно вводить и на русском языке. При наличии русского перевода страниц руководства man результаты поиска будут выведены на запрашиваемом языке.

«Страница руководства» занимает, как правило, больше одной страницы экрана. Для того чтобы читать было удобнее, **man** запускает программу постраничного просмотра текстов. Страницы перелистывают пробелом, для выхода из режима чтения описания команд **man** необходимо нажать на клавиатуре **q**. Команда man man выдаёт справку по пользованию самой командой **man**.

Документация в подавляющем большинстве случаев пишется на простом английском языке. Необходимость писать на языке, который будет более или менее понятен большинству пользователей, объясняется постоянным развитием Linux. Дело не в том, что страницу руководства нельзя перевести, а в том, что её придётся переводить всякий раз, когда изменится описываемый ею объект! Например, выход новой версии программного продукта сопровождается изменением его возможностей и особенностей работы, а следовательно, и новой версией документации.

Тем не менее, некоторые наиболее актуальные руководства существуют в переводе на русский язык. Свежие версии таких переводов на русский язык собраны в пакете man-pages-ru. Установив этот пакет, вы добавите в систему руководства, для которых есть перевод, и man по умолчанию будет отображать их на русском языке.

6.1.2 Команда info

Другой источник информации о Linux и составляющих его программах – справочная подсистема *info*. Страница руководства, несмотря на обилие ссылок различного типа, остаётся «линейным» текстом, структурированным только логически. Документ info –□это настоящий гипертекст, в котором множество небольших страниц объединены в дерево. В каждом разделе документа info всегда есть оглавление, из которого можно перейти к нужному подразделу, а затем вернуться обратно (ссылки для перемещения по разделам текста помечены *). Для получения вспомогательной информации о перемещении по тексту используйте клавишу **h**. Полное руководство info вызывается командой **info info**. Команда **info**, введённая без параметров, предлагает пользователю список всех документов info, установленных в системе.

6.2 Документация по пакетам

Дополнительным источником информации об интересующей вас программе, в основном на английском языке, является каталог **/usr/share/doc** – место хранения разнообразной документации.

Каждый пакет также содержит поставляемую вместе с включённым в него ПО документацию, располагающуюся обычно в каталоге **/usr/share/doc/имя_пакета**. Например, документация к пакету foo-1.0-alt1 находится в **/usr/share/doc/foo-1.0-alt1**. Для получения полного списка файлов документации, относящихся к пакету, воспользуйтесь командой **rpm -qd** *имя_установленного_пакета*.

В документации к каждому пакету вы можете найти такие файлы как **README**, **FAQ**, **TODO**, **ChangeLog** и другие. В файле **README** содержится основная информация о программе – имя и контактные данные авторов, назначение, полезные советы и пр. **FAQ** содержит ответы на часто задаваемые вопросы; этот файл стоит прочитать в первую очередь, если у вас возникли проблемы или вопросы по использованию программы, поскольку большинство проблем и сложностей типичны, вполне вероятно, что в **FAQ** вы тут же найдёте готовое решение. В файле **TODO** записаны планы разработчиков на реализацию той или иной функциональности. В файле **ChangeLog** записана история изменений в программе от версии к версии.

Для поиска внешней информации о программе, например, адреса сайта программы в сети Интернет можно использовать команду **rpm -qi** *имя_установленного_пакета*. В информационном заголовке соответствующего пакета, среди прочей информации, будет выведена искомая ссылка.

Возможно, будет полезно знать расположение собрания практических рекомендаций по самым различным вопросам, связанным с использованием Linux. Файлы **HOWTO** в формате HTML (от англ. how to – «как сделать») каталога **/usr/share/doc/HOWTO/** (при условии их наличия в системе) содержат многообразную информацию о работе Linux-систем.

6.3 Документация к программам, имеющим графический интерфейс

Каждая программа, имеющая графический интерфейс, как правило, сопровождается справочной информацией, вызываемой из меню программы. Обычно, это разделы меню **Справка**.

По обыкновению, это меню предоставляет информацию о программе, её версии, лицензии и авторах. В большинстве случаев, справка содержит встроенное руководство, ссылки на локальные сведения и интернет-страницы документации на официальных сайтах программ (традиционная кнопка **F1**), информацию о сочетании клавиш, а также сообщения о процедурах и отладке в программе.