УТВЕРЖДЕН

ЛКНВ.11100-01 91 01-ЛУ

ОПЕРАЦИОННАЯ СИСТЕМА АЛЬТ 8 СП

(ОС Альт 8 СП)

Руководство пользователя

ЛКНВ.11100-01 91 01

Листов 76

2018

Литера О

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

Настоящий документ является руководством пользователя программного изделия «Операционная система Альт 8 СП» (ОС Альт 8 СП) на архитектуре **Intel**.

Версия документа **1.1**.

В документе представлены общие сведения о назначении, условиях выполнения ОС Альт 8 СП, о ее функциональных возможностях, а также приведен порядок выполнения и список действий пользователя в ходе работы с ОС Альт 8 СП.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

ОС Альт 8 СП предназначена для группового и корпоративного использования в качестве средства автоматизации информационных, конструкторских и производственных процессов предприятий (организаций, учреждений). ОС Альт 8 СП обеспечивает выполнение программ в защищенной среде, обработку, хранение и передачу информации.

ОС Альт 8 СП поддерживает клиент-серверную архитектуру и может обслуживать процессы как в пределах одной компьютерной системы, так и процессы на других персональных электронных вычислительных машинах (далее – ПЭВМ) через каналы передачи данных или сетевые соединения.

ОС Альт 8 СП обладает следующими функциональными характеристиками:

- обеспечивает возможность обработки, хранения и передачи информации в защищенной программной среде;
- обеспечивает возможность запуска пользовательского программного обеспечения в сертифицированном окружении;
- 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 сетевую обработку данных, в том числе разграничение доступа к сетевым пакетам.

Для поддержки выполнения описанных функций в ОС Альт 8 СП реализованны следующие возможности:

- управление процессами и информационными ресурсами;
- управление системными ресурсами;
- управление памятью;
- управление файлами и внешними устройствами;
- управление доступом к обрабатываемой информации;
- защита хранимых, обрабатываемых и передаваемых информационных ресурсов комплексом средств защиты (далее – КСЗ) операционной системы (далее – ОС);
- администрирование;
- поддержка интерфейса прикладного программирования;
- поддержка пользовательского интерфейса.

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1. Минимальный состав аппаратных средств

Для функционирования ОС Альт 8 СП необходима ПЭВМ, обладающая следующими необходимыми характеристиками:

- аппаратная платформа – процессор архитектуры Intel x86, x86-64;
- оперативная память – не менее 512 Мбайт (рекомендуется 1 Гбайт и более);
- объем доступного пространства накопителя на жестких магнитных дисках – не менее 30 Гбайт;
- периферийные устройства ввода/вывода – устройство чтения и записи компакт-дисков.

2.2. Требования и условия организационного и технологического характера

К пользователю ОС Альт 8 СП предъявляется следующее требование: базовые навыки работы с ОС семейства Linux.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Загрузка ОС Альт 8 СП, установленной на жесткий диск

Вызов ОС Альт 8 СП, установленной на жесткий диск, происходит автоматически и выполняется после запуска ПЭВМ и отработки набора программ BIOS. ОС Альт 8 СП вызывает специальный загрузчик.

Загрузчик настраивается автоматически и включает в свое меню все системы, установку которых на ПЭВМ он определил. Поэтому загрузчик также может использоваться для вызова других ОС, если они установлены на компьютере.

Примечание. При наличии на компьютере нескольких ОС (или при наличии нескольких вариантов загрузки), пользователь будет иметь возможность выбрать необходимую ОС (вариант загрузки). В случае если пользователем ни один вариант не был выбран, то по истечении заданного времени будет загружена ОС (вариант загрузки), заданные по умолчанию.

Для перенастройки или перегенерации загрузочного меню и работы загрузчика необходимо воспользоваться следующей командой:

```
grub-mkconfig -o /boot/grub/grub.cfg
```

Команда генерирует конфигурационный файл на основе содержимого каталога `/etc/grub.d/`.

При стандартной установке ОС Альт 8 СП в начальном меню загрузчика доступны несколько вариантов загрузки (рис. 1): обычная загрузка, загрузка с дополнительными параметрами (например, «recovery mode» – загрузка с минимальным количеством драйверов), загрузка в программу проверки оперативной памяти (memtest).

По умолчанию, если не были нажаты управляющие клавиши на клавиатуре, загрузка ОС Альт 8 СП продолжится автоматически.

Рис. 1 – Варианты загрузки

В процессе загрузки ОС Альт 8 СП пользователь может следить за информацией процесса загрузки, которая отображает этапы запуска различных служб и программных серверов в виде отдельных строк (рис. 2), на экране монитора.

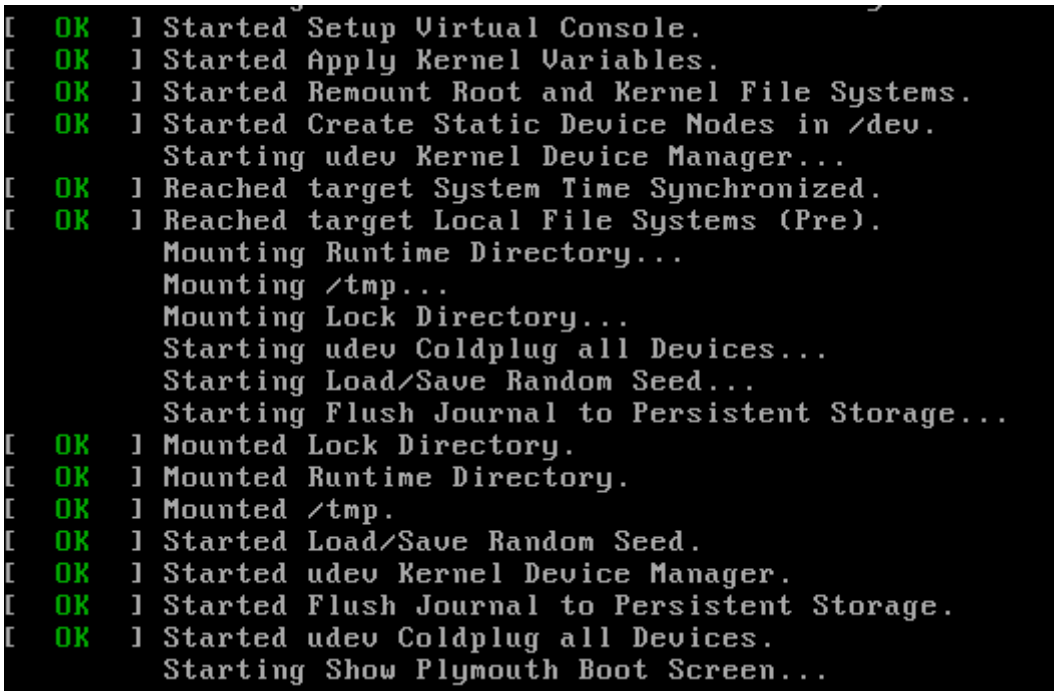

Рис. 2 – Загрузка ОС

При этом каждая строка начинается словом вида [XXXXXXX] (FAILED или OK), являющегося признаком нормального или ненормального завершения этапа загрузки. Слово XXXXXXX=FAILED (авария) свидетельствует о неуспешном завершении этапа загрузки, что требует вмешательства и специальных действий администратора системы.

После окончания работы загрузчика на экране появляется запрос на ввод имени учетной записи (login). Для прохождения процедуры идентификации и аутентификации необходимо ввести корректное имя учетной записи (login), а затем корректный пароль (Password), соответствующий учетной записи.

3.1.1. Вход и работа в системе в консольном режиме

Стандартная установка ОС Альт 8 СП включает базовую систему, работающую в консольном режиме.

При загрузке в консольном режиме работа загрузчика ОС Альт 8 СП завершается запросом на ввод логина и пароля учетной записи (рис. 3). В случае необходимости на другую консоль можно перейти, нажав <Ctrl>+<Alt>+<F2>.

Рис. 3 – Запрос на ввод логина

Для дальнейшего входа в систему необходимо ввести логин и пароль учетной записи пользователя.

В случае успешного прохождения процедуры аутентификации и идентификации будет выполнен вход в систему. ОС Альт 8 СП перейдет к штатному режиму работы и предоставит дальнейший доступ к консоли (рис. 4).

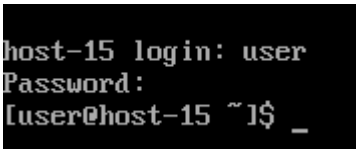

Рис. 4 – Приглашение для ввода команд

3.1.2. Вход и работа в системе в графическом режиме

В состав ОС Альт 8 СП также может входить графическая оболочка MATE. Графическая оболочка состоит из набора различных программ и технологий, используемых для управления ОС и предоставляющих пользователю удобный графический интерфейс для работы в виде графических оболочек и оконных менеджеров.

При загрузке в графическом режиме работа загрузчика ОС заканчивается переходом к окну входа в систему (рис. 5), в котором необходимо выбрать одну из учетных записей, предлагаемых в окне аутентификации, ввести пароль, соответствующий этой учетной записи и нажать кнопку «Войти».

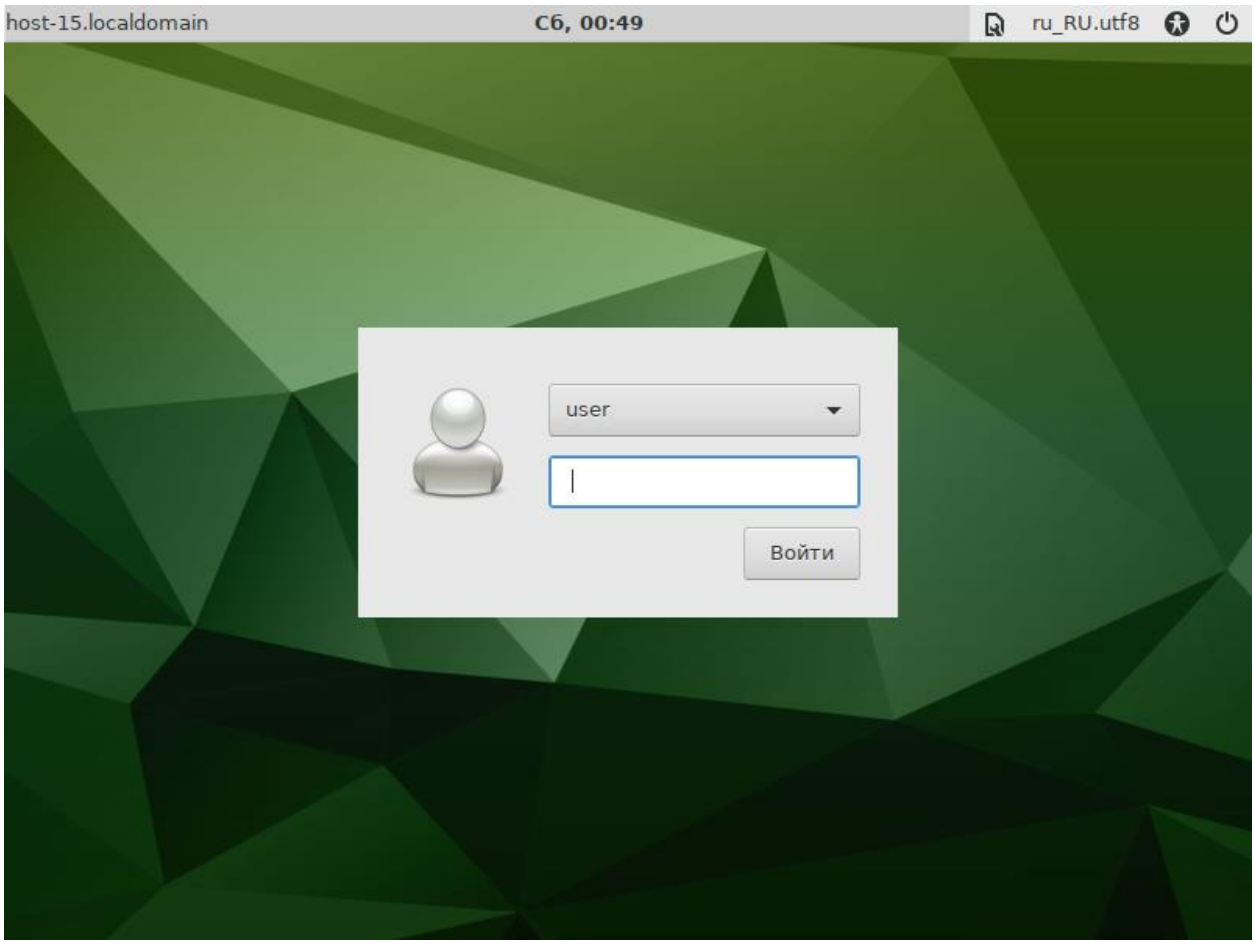

Рис. 5 – Окно входа в систему

В результате успешного прохождения процедуры аутентификации и идентификации будет выполнен вход в систему. ОС Альт 8 СП перейдет к штатному режиму работы и предоставит дальнейший доступ к графическому интерфейсу (рис. 6).

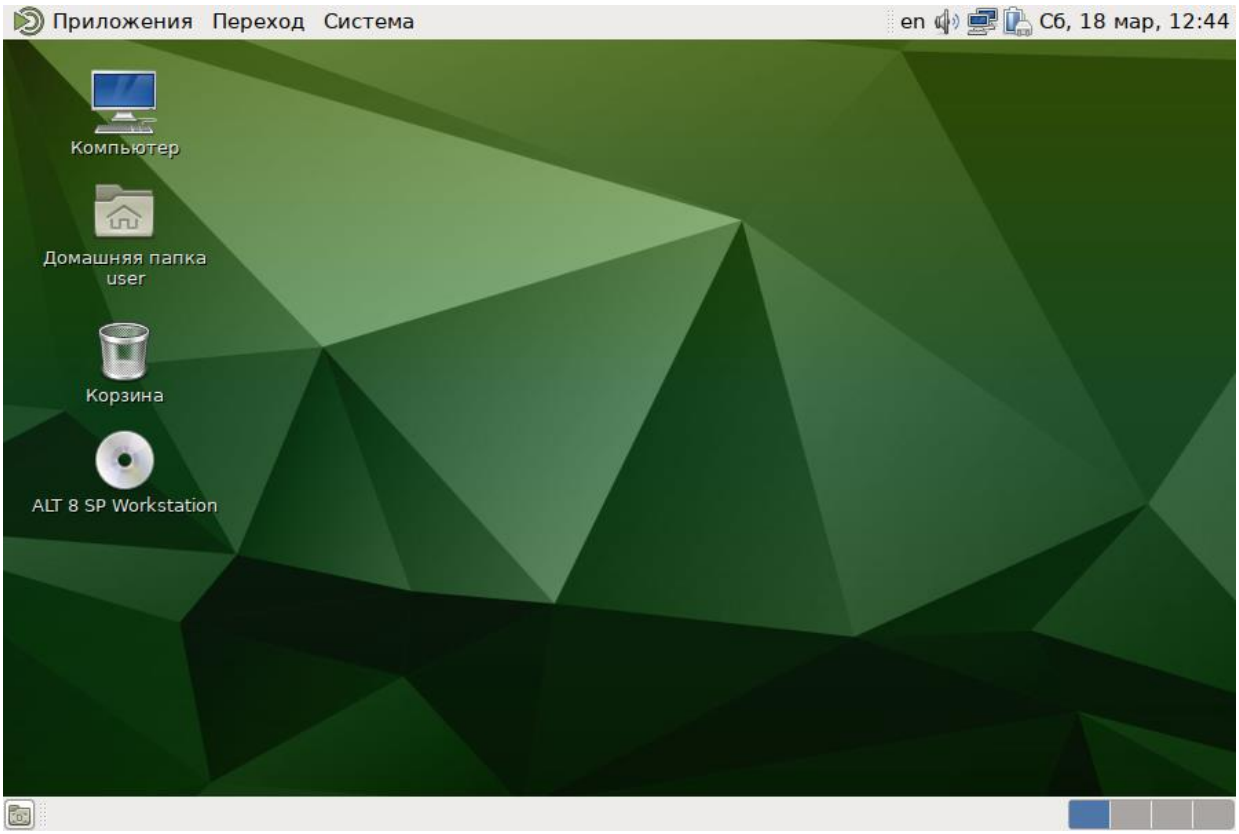

Рис. 6 – Графический интерфейс

3.2. Работа с наиболее часто используемыми компонентами

3.2.1. Файловый менеджер Caja

Файловые менеджеры предоставляют интерфейс пользователя для работы с файловой системой и файлами. Файловые менеджеры позволяют выполнять наиболее частые операции над файлами – создание, открытие/проигрывание/просмотр, редактирование, перемещение, переименование, копирование, удаление, изменение атрибутов и свойств, поиск файлов и назначение прав. Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей, например, таких как работа с сетью (через FTP, NFS и т. п.), резервное копирование, управление принтерами и прочее.

Caja – это современный файловый менеджер для рабочей среды Mate (рис. 7).

Используя файловый менеджер, можно:

- создавать папки и документы;

- просматривать файлы и папки;

- управлять файлами и папками;
- запускать сценарии и приложения;
- настраивать внешний вид файлов и папок
- получать доступ к съемным носителям.

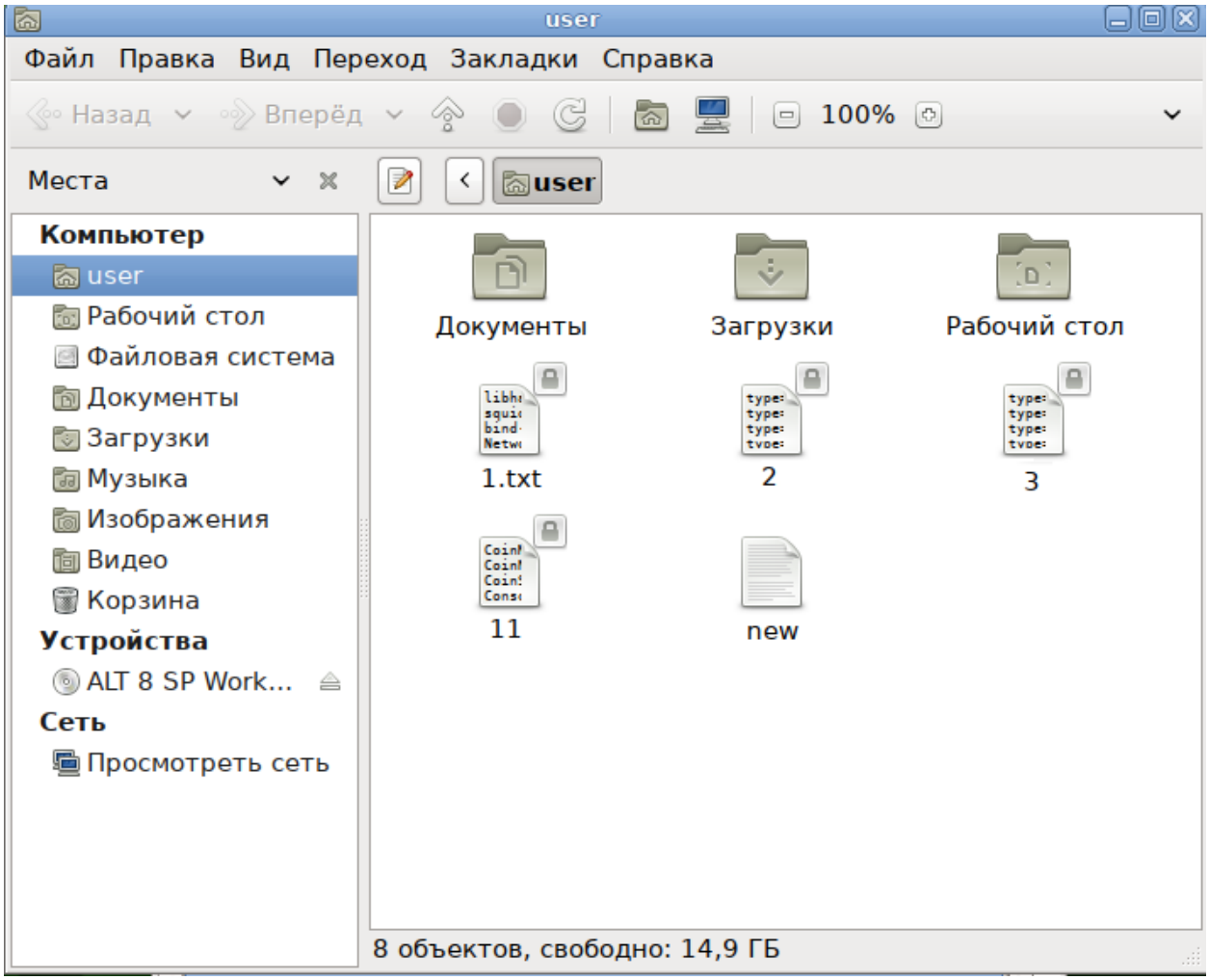

Рис. 7 – Файловый менеджер Caja

Окно файлового менеджера состоит из боковой панели слева, основной области справа и панели адреса, расположенной над основной областью. На боковой панели размещены закладки на различные папки системы. Основная область отображает содержимое текущей папки. Панель адреса всегда показывает путь к текущей папке.

Чтобы просмотреть свойства файла (папки), необходимо выделить файл (папку) и выполнить одно из следующих действий:

- в меню выбрать «Файл» → «Свойства»;
- в контекстном меню файла (папки) выбрать пункт «Свойства»;
- нажать <Alt>+<Enter>.

Окно «Свойства объекта» показывает подробную информацию о любом файле, папке или другом объекте в файловом менеджере (какие именно сведения будут доступны, определяется типом объекта):

- имя файла или папки – можно ввести новое имя, и файл или папка будут переименованы после нажатия кнопки «Закрыть»;
- тип – тип объекта (например, файл или папка);
- адрес – системный путь к объекту (указывает местонахождение объекта относительно корня системы);
- том – том, на котором расположена папка (физическое местоположение папки – носитель, на котором она находится);
- свободное место – объем свободного пространства на носителе, на котором находится папка (наибольшее количество данных, которые можно скопировать в эту папку);
- изменен – дата и время последнего изменения объекта;
- дата доступа – дата и время последнего просмотра объекта.

С помощью окна «Свойства объекта» можно выполнить следующие действия:

- изменить значок объекта;
- изменить файловые права на доступ к объекту;
- выбрать, с помощью какого приложения следует открывать данный объект и другие объекты того же типа.

Для того чтобы изменить права доступа к файлу или к папке необходимо в свойствах объекта перейти на вкладку «Права». Далее, чтобы изменить группу файла (папки), можно выбрать одну из групп, к которым принадлежит пользователь, из выпадающего списка (рис. 8).

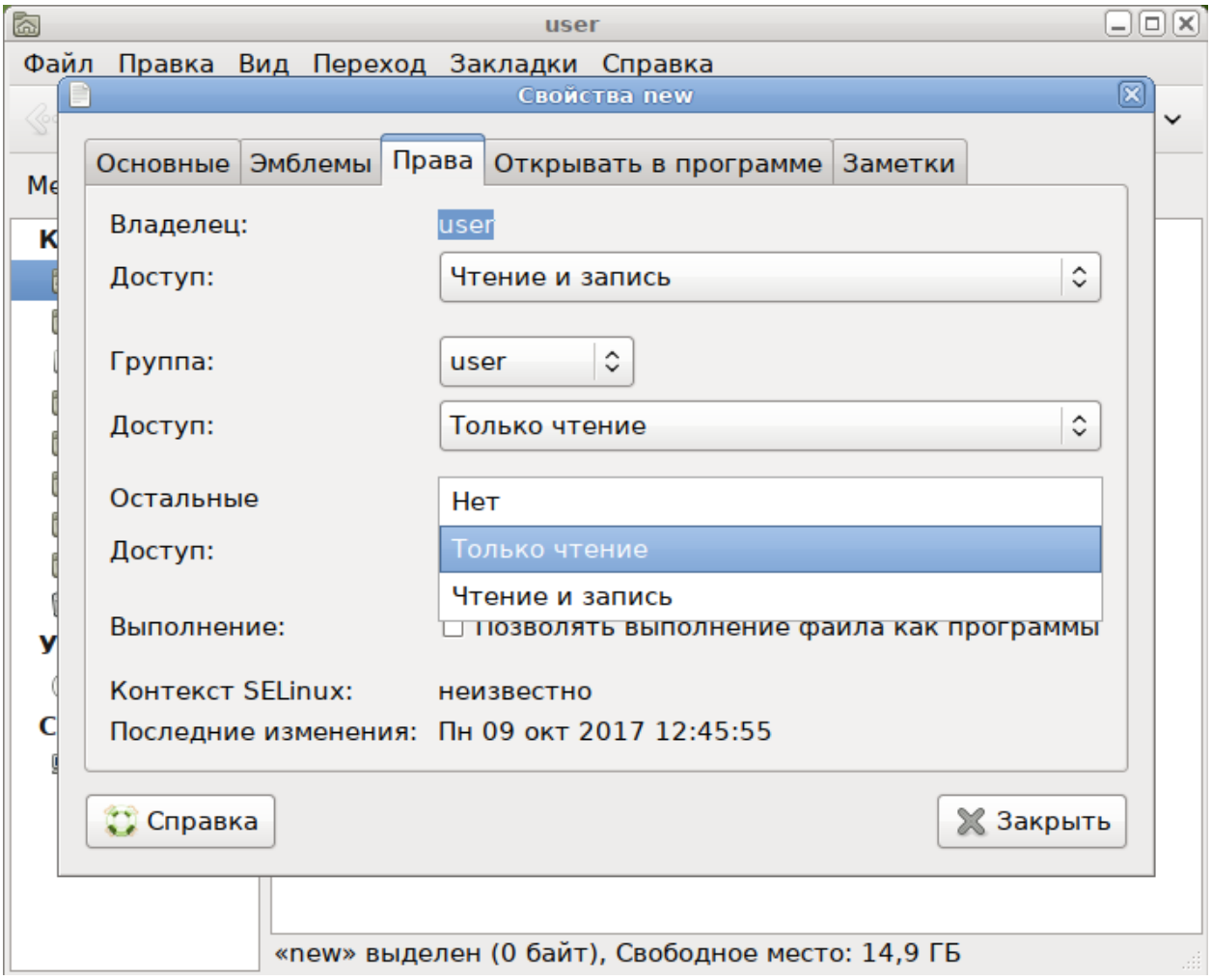

Рис. 8 – Изменение прав доступа на файл в файловом менеджере Caja

При настройке прав доступа на файл для каждой из трех категорий – владелец, группа и все остальные пользователи – возможен выбор права доступа из числа следующих:

- «нет» – нет доступа к файлу (этот режим нельзя установить для владельца);
- «только чтение» – пользователи могут открыть файл и увидеть его содержимое, но не могут вносить изменения;
- «чтение и запись» – возможен нормальный доступ к файлу — его можно открывать и сохранять.

Чтобы разрешить запуск файла как программы, необходимо отметить пункт «Выполнение».

При настройке прав доступа на папку для каждой из трех категорий – владелец, группа и все остальные пользователи – возможен выбор права доступа из числа следующих:

- «нет» – нет доступа к папке (этот режим нельзя установить для владельца);
- «только перечисление файлов» – пользователи могут видеть объекты, находящиеся в папке, но не могут открывать их;
- «доступ к файлам» – объекты в папке можно открывать и изменять, при условии, что их собственные права это позволяют;
- «создание и удаление файлов» – помимо возможности доступа к существующим в папке файлам, пользователь может удалять их и создавать в папке новые.

Чтобы установить права на все объекты, содержащиеся в папке, необходимо установить свойства «Доступ к файлу» и «Выполнение» и нажать кнопку «Распространить права на вложенные файлы».

3.2.2. Работа с командной оболочкой и основные команды

Для управления ОС Альт 8 СП используются командные интерпретаторы (shell).

Командный интерпретатор предназначен для передачи команд пользователя ОС Альт 8 СП. При помощи командных интерпретаторов можно писать небольшие программы – сценарии (скрипты).

Оболочкой по умолчанию является Bash (Bourne Again Shell) – самая распространенная оболочка под Linux, которая ведет историю команд и предоставляет возможность их редактирования.

В дальнейшем описании работы с ОС Альт 8 СП будут использоваться примеры с использованием этой оболочки.

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

- <Ctrl> + <A> – переход на начало строки;
- <Ctrl> + <U> – удаление текущей строки;

- <Ctrl> + <C> – остановка текущей задачи;

- <Ctrl> + <R> – найти конкретную команду в списке набранных.

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

Просмотреть историю команд, можно выполнив команду:

```
$ history
```

Команды, присутствующие в истории, отображаются в списке пронумерованными. Для того, чтобы запустить конкретную команду нужно набрать:

```
!номер команды
```

Для запуска последней из набранных команд:

```
!!
```

В Bash имеется возможность самостоятельного завершения имен команд из общего списка команд, что облегчает работу 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 bunzip2, можно набрать следующую команду:

```
$ bu
```

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 «bu».

В предложенном примере можно получить следующий список:

```
$ bu buildhash builtin bunzip2
```

Если ввести команду «bun» (bunzip – это единственное имя, третьей буквой которого является «n»), а затем нажать клавишу табуляции <Tab>, то оболочка самостоятельно дополнит имя.

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 `./` (если только не выбран

один из двух самых слабых уровней защиты). Поэтому для запуска программы prog из текущего каталога необходимо использовать следующую команду:

```
$ ./prog
```

3.2.3. Утилиты для управления учетными записями пользователей

3.2.3.1. Команда `su`

Команда `su` позволяет получить права администратора. При вводе команды `su`, будет запрошен пароль администратора (root), и, в случае ввода корректного пароля, пользователь получит права администратора.

Синтаксис:

```
su [-] [name [arg…]]
```

Чтобы вернуться к правам пользователя, необходимо ввести следующую команду:

```
exit
```

3.2.3.2. Команда `id`

Команда `id` – вывод идентификатора пользователя и списка групп, в которых он состоит.

Синтаксис:

```
id [пользователь]
```

3.2.3.3. Команда passwd

Команда `passwd` меняет пароль, связанный с входным_именем пользователя.

Пользователь может менять только пароль, связанный с его собственным входным_именем.

Команда запрашивает у пользователей старый пароль, а затем дважды запрашивает новый. Новый пароль должен соответствовать парольным ограниченям, наложенным системным программистом системы. Если новый пароль не удовлетворяет парольным ограничениям, он отвергается и `passwd` завершается.

Пароли, указанные по умолчанию:

- пароли из одного класса символов – запрещены;
- минимальная длина паролей из двух классов (цифр или букв, например) 24 символа;

- 1 символов для паролей-фраз минимум из трех символов;
- 8 и 7 символов соответственно для паролей из трех и четырех классов символов).

3.2.4. Основные утилиты для операций с файлами и каталогами

3.2.4.1. Команда `ls`

Команда `ls` предназначена для вывода информации о файлах или каталогах. Команда `ls` для каждого имени каталога распечатывает список входящих в этот каталог файлов; для файлов – повторяется имя файла и выводится дополнительная информация в соответствии с указанными флагами. По умолчанию имена файлов выводятся в алфавитном порядке. Если имена не заданы, выдается содержимое текущего каталога.

Синтаксис:

```
ls [параметры]... [файл]...
```

Параметры:

1) `-a`, `--all` – вывести список всех файлов (обычно не выводятся файлы, имена которых начинаются с точки);
2) `-A`, `--almos-all` – не показывать подразумеваемые «.» и «..»;
3) `--block-size=РАЗМЕР` – выдает размеры в блоках по РАЗМЕР байт. Например, `--block-size=M` для вывода объема в единицах равных 1048576 байтов;
4) `-B`, `--ignore-backups` – не показывать файлы, заканчивающиеся на «~», если они не заданы в командной строке;
5) `-c`, `--time=ctime`, `--time=status` – сортировать содержимое каталога в соответствии с временм изменения состояния файла. Если с помощью опции `-l` задан этот формат, то выдавать время изменения файла вместо времени его модификации. С опций `-t` показать время последней модификации описания файла и сортировать по имени;
6) `-C`, `--format=vertical` – вывод в несколько колонок с сортировкой по вертикали;

7) `--color[=КОГДА]` – использовать цвета в выводе. КОГДА по умолчанию `always`. Также можно использовать `never` и `auto`;
8) `-d`, `--directory` – если аргумент является каталогом, то выводить только его имя, а не содержимое. Часто используется с флагом `-l` для получения сведений о состоянии каталога;
9) `-h`, `--human-readable` – в сочетании с -l показывает размеры в удобочитаемом формате (например, 1K 234M 2G);
10) `-i`, `--inode` – показывать индекс каждого файла;
11) `-I`, `--ignore= ШАБЛОН` – не показывать записи, соответствующие ШАБЛОНУ командного интерпретатора;
12) `-k`, `--kibibytes` – использовать блоки по 1024 байта;
13) `-l` – вывод в длинном формате;
14) `-m` – показать записи в список шириной в размер терминала, имена файлов разделяются запятыми;
15) `-r`, `--reverse` – изменить порядок сортировки на обратный;
16) `-R`, `--revcursive` – рекурсивно обойти встретившиеся подкаталоги;
17) `-s`, `--size` – выдавать размер файлов в блоках;
18) `-S` – отсортировать по размеру файлов, большие сначала;
19) `--sort=СЛОВО` – сортировать по СЛОВУ, а не по имени: `none` (без сортировки) `-U`, `extension` (расширение) `-X`, `size` (размер) `-S`, `time` (время) `-t` или `version` (версия) `-v`;
20) `-t` – файлы сортируются по времени последнего изменения (сначала идут самые новые файлы);
21) `-U` – не сортировать, отображать записи в обычном порядке;
22) `-v` – сортировать по номерам (версии) в текстовом представлении;
23) `-x` – вывод в несколько колонок с сортировкой по строкам;
24) `-Z, --context` – вывести контекст для каждого файла;
25) `-1` – отображать по одному файлу в строке.

Режим доступа к файлу при указании флага `-l` выводится в виде 10 символов. При этом первый символ означает:

1) `d` – файл является каталогом;
2) `b` – файл является специальным блочным файлом;
3) `c` – файл является специальным символьным файлом;
4) `p` – файл является именованным каналом;
5) `-` – обычный файл.

Остальные 9 символов делятся на три группы по три символа: права доступа владельца, других пользователей из его группы, всех прочих пользователей. Внутри каждой группы используются три символа, обозначающие права на чтение, запись и выполнение файла соответственно. Для каталога под правом на выполнение подразумевается право на просмотр в поисках требуемого файла.

Пример:

```
ls -l /util/by
-rwxr-xr-x 1 root sys 50 Jun 22 10:42 /util/by
```

Права обозначаются следующим образом:

1) `r` – право на чтение;
2) `w` – право на запись;
3) `x` – право на выполнение (поиск в каталоге);
4) `-` – данное право доступа отсутствует;
5) `l` – учет блокировки доступа (бит переустановки идентификатора группы равен 1, бит права на выполнение членами группы равен 0). Располагается на месте права на выполнение для членов группы;
6) `s` – право переустанавливать идентификатор группы или идентификатор владельца и право выполнения файла для членов группы или владельца;
7) `S` – неопределенная комбинация бит: право переустанавливать идентификатор владельца есть, а право выполнения файла для владельца отсутствует;

8) `t` – установлен бит навязчивости у файла, который могут выполнять прочие пользователи. Располагается на месте права на выполнение для прочих пользователей;
9) `T` – бит навязчивости установлен, а права на выполнение у прочих пользователей нет. Располагается на месте права на выполнение для прочих пользователей.

Примеры:

1. Если файл доступен владельцу для чтения, записи и выполнения, а членам группы и прочим пользователям только для чтения, он имеет режим:

```
-rwxr--r-
```

2. Файл доступен владельцу для чтения, записи и выполнения, а членам группы и прочим пользователям только для чтения и выполнения. Разрешена переустановка при выполнении идентификатора пользователя на идентификатор владельца файла:

```
-rwsr-xr-x
```

3. Файл доступен для чтения и записи только владельцу и членам группы; может быть блокирован при доступе:

```
-rw-rwl--
```

4. Вывести имена всех файлов в текущем каталоге, включая и те, которые начинаются с точки и обычно не выдаются:

```
ls -a
```

5. Вывести разнообразную информацию: список всех файлов, включая те, которые обычно не выводятся (a); номера описателей файлов будут выведены в левой колонке (i); размеры файлов (в блоках) выводятся во второй колонке (s); наконец, будут выданы числовые идентификаторы владельцев и групп (n):

```
ls –aisn
```

Возможные сообщения об ошибках, при использовании команды `ls`:

```
ls: невозможно открыть каталог <путь>: Отказано в доступе
ls: невозможно получить доступ к <путь>/<файл>: Нет такого файла или каталога
```

3.2.4.2. Команда `cp`

Команда `cp` предназначена для копирования файлов и каталогов.

Синтаксис:

```
cp [ОПЦИЯ]... [-T] ИСТОЧНИК НАЗНАЧЕНИЕ
cp [ОПЦИЯ]... ИСТОЧНИК... КАТАЛОГ
cp [ОПЦИЯ]... -t КАТАЛОГ ИСТОЧНИК...
```

Копирует ИСТОЧНИК в НАЗНАЧЕНИЕ или несколько ИСТОЧНИКОВ в КАТАЛОГ.

Основные опции:

1) `--backup[=CONTROL]` – сделать резервную копию каждого целевого файла;
2) `-b` – тоже что и `--backup`, но не принимает аргументы;
3) `-f`, `--force` – если невозможно открыть существующий файл, то удалить его и попробовать еще раз (данная опция игнорируется, если используется совместно с `-n`);
4) `-i`, `--interactive` – спросить перед перезаписью (отменяет ранее указанный ключ `-n`);
5) `-H` – следовать символьным ссылкам в ИСТОЧНИКЕ;
6) `-l`, `--link` – создавать жесткие ссылки вместо копирования;
7) `-n`, `--no-clobber` – не перезаписывать существующие файлы (отменяет стоящую перед ней опцию `-i`);
8) `-R`, `-r`, `--recursive` – копировать каталоги рекурсивно;
9) `-s`, `--symbolic-link` – создать символьную ссылку вместо копирования;
10) `-u`, `--update` – копировать, только если файл ИСТОЧНИК новее, чем файл назначения или если файл назначения отсутствует;
11) `-v`, `--verbose` – выводить имя каждго файла перед копированием.

По умолчанию суффикс для резервных копий «~». Его можно переопределить при помощи опции `--suffix` или переменной окружения SIMPLE_BACKUP_SUFFIX. Способ контроля версий может быть задан через опцию `--backup` или через переменную окружения VERSION_CONTROL. Допустимые значения:

1) `none`, `off` – никогда не делать резервные копии (даже если задана опция `--backup`);
2) `numbered`, `t` – создать нумерованные резервные копии;
3) `existing`, `nil` – если существуют нумерованные резервные копии, то создавать нумерованные резервные копии, если нет, то создавать простые;

4) simple, never – всегда создавать простые резервные копии.

Следующий пример использования команды cp демонстрирует копирование файла srcfile1 в каталог dest_dir:

```
cp srcfile1 dest_dir
```

3.2.4.3. Команда rsync

Команда rsync выполняет синхронизацию файлов и каталогов, использует протокол удаленного обновления для ускорения передачи файлов, которые существуют в месте назначения.

Синтаксис:

```
rsync [ОПЦИИ] источник место_назначения
```

Опции:

1) -v – подробный режим;

2) -r – копировать данные рекурсивно;

3) -a – режим архивирования, позволяет копировать данные рекурсивно, с сохранением прав доступа на файлы, символических ссылок и другой информации);

4) -h – вывод данных в удобном формате;

5) -z – сжатие данных.

Примеры:

1. Скопировать или синхронизировать все файлы из одного каталога в другой:

```
rsync –avh /tmp/firstdir /tmp/seconddir
```

2. Копирование локальных данных на удаленный хост:

```
rsync –avzh /tmp/firstdir user@10.110.2.1:/tmp/seconddir
```

Возможные сообщения об ошибках, при использовании команды sfill:

```
rsync: change_dir#1 <каталог> failed: Отказано в доступе
rsync: change_dir <каталог> failed: Нет такого файла или каталога
```

3.2.4.4. Команда mv

Команда mv – перемещение (переименование) файлов.

Синтаксис:

```
mv [ОПЦИЯ]... [-T] ИСТОЧНИК НАЗНАЧЕНИЕ
mv [ОПЦИЯ]... ИСТОЧНИК... КАТАЛОГ
```

```
mv [ОПЦИЯ]... -t КАТАЛОГ ИСТОЧНИК...
```

Переименовать ИСТОЧНИК в НАЗНАЧЕНИЕ или переместить ИСТОЧНИК (и) в КАТАЛОГ.

Основные опции:

1) `-i, --interactive` – просит подтверждения на замену существующего файла;
2) `-n, --no-clobber` – не переписывать существующий файл. Если указано несколько опций `-i`, `-f` и `-n`, то действовать будет только последняя;
3) `-u, --update` – перемещать только, если файл ИСТОЧНИК новее, чем файл назначения или если файл назначения отсутствует;
4) `-v, --verbose` – выдавать имя каждого файла перед его переносом.

Возможные сообщения об ошибках, при использовании команды mv:

```
mv: невозможно переместить <файл> в <файл>: Операция не позволена
mv: не удалось выполнить stat для <файл>: Отказано в доступе
mv: не удалось выполнить stat для <файл>: Нет такого файла или каталога
```

3.2.4.5. Команда `dd`

Команда `dd` предназначена для копирования файла (по умолчанию из стандартного ввода на стандартный вывод), используя заданные размеры блоков для ввода и вывода, и в тоже время выполняя его преобразование.

Синтаксис:

```
dd [параметр]
```

Основные опции:

1) `if=ФАЙЛ` – читает данные из ФАЙЛа вместо стандартного ввода;
2) `of=ФАЙЛ` – пишет данные в ФАЙЛ вместо стандартного вывода;
3) `ibs=ЧИСЛО` – читает по ЧИСЛО байт за раз. По умолчанию 512;
4) `obs=ЧИСЛО` – пишет по ЧИСЛО байт за раз. По умолчанию 512;
5) `bs=ЧИСЛО` – читает и пишет по ЧИСЛО байт за раз. По умолчанию 512.

Примеры:

1. Заполнить устройство случайными данными:

```
dd if=/dev/urandom of=/dev/sda bs=4k
```

2. Скопировать раздел в другой раздел:

```
dd if=/dev/sda3 of=/dev/sdb3 bs=4096 conv=notrunc,noerror
```

Возможные сообщения об ошибках, при использовании команды `dd`:

```
dd: не удалось открыть <файл>: Отказано в доступе
```

3.2.4.6. Команда `s_rm`

Команда `s_rm` выполняет безопасное удаление целевого файла.

Синтаксис:

`s_rm ФАЙЛ...`

Возможные сообщения об ошибках, при использовании команды `s_rm`:

```
Ошибка: файл <файл>: Отказано в доступе
Ошибка: файл <файл>: Нет такого файла или каталога
```

3.2.4.7. Команда `s_fill`

Команда `s_fill` выполняет безопасную перезапись свободного пространства на разделе, в котором находится указанная директория и всех свободных индексных дескриптеров указанного каталога.

Синтаксис:

`S_fill каталог...`

Возможные сообщения об ошибках, при использовании команды `s_fill`:

```
Ошибка: не достаточно прав для <каталог>: Отказано в доступе
```

3.2.4.8. 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]
```

Опция `-L` заставляет следовать по символическим ссылкам.

Поскольку для выполнения каждой команды создается отдельный процесс, `cd` не может быть обычной командой; она распознается и выполняется командной оболочкой.

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.

3.2.4.9. Команда `pwd`

Команда `pwd` выводит абсолютный путь текущего (рабочего) каталога.

Синтаксис:

```
pwd [-LP]
```

Опции:

1) `-P` – вывод не будет содержать символических ссылок;

2) `-L` – вывод может содержать символические ссылки.

3.2.4.10. Команда `mkdir`

Команда `mkdir` предназначена для создания каталогов.

Синтаксис:

```
mkdir [опция]... каталог...
```

Опции:

1) `-m`, `--mode=РЕЖИМ` – установить права доступа для создаваемых каталогов;
2) `-p`, `--parents` – перед созданием нового каталога предварительно создаются все несуществующие вышележащие каталоги. В случае существования каталога не будет выведена ошибка;
3) `-v`, `--verbose` – выводить сообщение для каждого созданного каталога;
4) `-Z`, `--context[=CTX]` – задать контекст для каждого создаваемого каталога. Если CTX не задан, то контекст будет равным типу по умолчанию.

Чтобы создать поддерево каталогов `tmpdir/temp/dir`, надо выполнить команду:

```
mkdir -p tmpdir/temp/dir
```

Возможные сообщения об ошибках, при использовании команды `mkdir`:

`mkdir: невозможно создать каталог <каталог>: Отказано в доступе`

`mkdir: невозможно создать каталог <каталог>: Нет такого файла или каталога`

3.2.4.11. Команда `rmdir`

Команда `rmdir` предназначена для удаления каталога, при условии, что он пуст.

Синтаксис:

`rmdir [опция]... каталог...`

Для команды `rmdir` доступна опция `-p` – при указании пути к каталогу (а не просто имени каталога), команда удалит каталог и его потомков:

`rmdir –p a/b/c`

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

3.2.4.12. Команда `mount`

Команда `mount` используется для монтирования файловых систем.

Синтаксис:

```
mount [-lhV]
mount –a [опция]
mount [опция] [--source] <source> | [--target] <directory>
mount [опция] <source> <directory>
```

Опции:

1) `-t` – определение типа файловой системы раздела, предполагаемого для размещения;
2) `-o` – указание параметров монтирования.

Примеры:

1. Просмотр примонтированных устройств:

```
mount -l
```

2. Монтирование разделов жесткого диска:

```
mount –t ext3 /dev/sdb1 /home/user/test
```

Возможные сообщения об ошибках, при использовании команды `mount`:

```
mount: точка монтирования <каталог> не существует
```

3.2.5. Создание, просмотр и редактирование файлов

3.2.5.1. Команда `cat`

Команда cat позволяет просмотреть файл целиком, копируя файлы в стандартный поток вывода и объединяя их.

Синтаксис:

```
cat [ОПЦИЯ]... [ФАЙЛ]...
```

Опции:

1) `-A`, `--show-all` – тоже что и `-vET`;
2) `-e` – тоже что и `-vE`;
3) `-E`, `--show-ends` – отображать символ «$» в конце каждой строки;
4) `-n`, `--number` – нумеровать выводимые строки;
5) `-s`, `--squeeze-blank` – скрывать повторяющиеся пустые строки в выводе;
6) `-t` – тоже что и `-vT`;
7) `-T`, `--show-tabs` – отображать символ табуляции как `^I`;
8) `-v`, `--show-nonprinting` – использовать `^-` и M-нотацию для всех непечатаемых символов кроме LFD (перевод строки и табуляция) и табуляции.

Если ФАЙЛ не задан или задан как «-», то читать из стандартного ввода.

Примеры:

1. Вывести содержимое файла `f`, затем со стандартного ввода, затем – содержимое файла `g`:

   ```
   cat f - g
   ```

2. Скопировать стандартный ввод на стандартный вывод:

   ```
   cat
   ```

Возможные сообщения об ошибках, при использовании команды `cat`:

```
cat: <файл>: Отказано в доступе
cat: <файл>: Нет такого файла или каталога
```

3.2.5.2. Команда `less`

Команда `less` позволяет просматривать текст постранично.

```
less [ опции ] файл
```

Опции:

1) `-c` – очистка экран перед тем, как отобразить следующую страницу;
2) `-m` – вывод информации о том, какая часть файла выведена на данный момент (в процентах);
3) `-N` – вывод номеров строк;
4) `-r` – вывод управляющих (непечатаемых) символов;
5) `-s` – объединение несколько пустых строк в одну;
6) `-S` – урезание длинных строк до длины экрана вместо переноса.

Возможные сообщения об ошибках, при использовании команды `less`:

```
<файл>: Отказано в доступе
<файл>: Нет такого файла или каталога
```

3.2.5.3. Команда `echo`

Команда `echo` выводит текст на стандартное устройство вывода.

```
echo [опции] [строка]
```

Опции:

1) `-n` – не выводить в конце символ новой строки;
2) `-e` –включить интерпретацию управляющих символов;
3) `-E` – отключить интерпретацию управляющих символов;

Возможные сообщения об ошибках, при использовании команды `echo`:

```
<файл>: Отказано в доступе
<файл>: Нет такого файла или каталога
```

3.2.5.4. Команда `grep`

Команда `grep` предназначена для поиска текста, соответствующего регулярному выражению в файлах или потоке вывода.

Синтаксис:

```
grep [ опции ] шаблон_поиска [файл]
```

Опции:

1) `-r` – рекурсивный поиск во всех каталогах;

2) `-n` – вывод номеров строк, в которых найдено совпадение;

3) `-l` – вывод списка файлов, содержащих шаблон;

4) `-v` – поиск строк, не содержащих шаблон (инверсия);

5) `-i` – поиск с игнорированием регистра.

3.2.5.5. Команда `touch`

Создание и редактирование файлов выполняется командой touch, которая устанавливает время последнего изменения и доступа в текущее системное время у заданного файла. Если файл не существует – он создается.

Синтаксис:

```
touch [опции]... файл
```

Основные опции:

1) `-a` – изменить только время доступа к файлу;

2) `-c`, `--no-create` – не создавать файл;

3) `-d`, `--date=СТРОКА` – проанализировать строку и использовать вместо текущего времени;

4) `-m` – изменить время последней модификации файла;

5) `-r`, `--reference=ФАЙЛ` – использовать соответствующий временной штамп от ФАЙЛ в качестве нового занчения для изменяемого временного штампа;

6) `-t время` – использовать заданное время в качестве нового значения для изменяемого временного штампа.

Следующий пример использования команды `touch` создает файл `myfile.txt`:

```
touch myfile.txt
```

Возможные сообщения об ошибках, при использовании команды `touch`:

```
touch: невозможно выполнить touch для <файл>: Отказано в доступе
touch: невозможно выполнить touch для <путь>/<файл>: Нет такого файла или каталога
```

3.2.5.6. Команда `mknod`

Утилита `mknod` создает специальные блочные или символьные файлы. Специальный файл записывается в файловой системе с помощью тройки параметров: один логический и два целых. Логический параметр говорит о том, является ли специальный файл символьным или блочным. Два целых параметра задают старший и младший номера устройства. Специальнй файл практически не занимает места на диске и используется только для общения с операционной системой, а не для хранения данных.

Синтаксис:

```
mknod [опции] имя {bc} старший_номер младший_номер
mknod [опции] имя p
```

Основные опции:

1) `-m`, `--mode=РЕЖИМ` – установить РЕЖИМ доступа;

2) `-Z` – установить контекст безопасности равным типу по умолчанию.

Тип устройства может принимать следующие значения:

1) `b` – создать файл блочного устройства (буферизированный);

2) `c` – создать файл символьного устройства (небуферизированный);

3) `p` – создать именованный канал.

Возможные сообщения об ошибках, при использовании команды mknod:

```
mknod: <файл>: Файл существует
```

3.2.6. Поиск файлов

3.2.6.1. Команда `find`

Утилита `find` используется для поиска файлов.

Синтаксис:

```
find [-H] [-L] [-P] [-0уровень] [-D help | tree | search | stat |
rates | opt | exec] [путь…] [выражение]
find [путь] [опции] [критерии поиска] [действия над файлами]
```

В качестве пути для поиска можно использовать как абсолютные, так и относительные пути, а также список путей, разделенных пробелом. Путем по умолчанию является текущий подкаталог. Выражение по умолчанию `-print`.

Основные опции:

1) `-d`, `--depth` – поиск в подкаталогах перед поиском в самом каталоге;
2) `-L` – при поиске следовать по символическим ссылкам;
3) `-P` – никогда не следовать по символическим ссылкам;
4) `-maxdepth N` – при поиске проверять не более чем N вложенных уровней каталогов;
5) `-mindepth N` – не проверять вложенные каталоги уровня N и меньше;
6) `-mount` – не искать в каталогах других файловых систем.

У команды `find` может быть несколько критериев поиска (tests). Каждый критерий представляет собой определенное условие проверки, которое возвращает либо `true` либо `false`. В процессе обработки очередного файла команда `find` по очереди проверяет каждый критерий, и, если очередной критерий возвращает `false`, тогда команда `find` переходит к следующему файлу.

Основные критерии поиска:

1) `-name шаблон` – имя файла (шаблон имени) без указания пути. Рекомендуется всегда заключать шаблон в кавычки;
2) `-atime N` – последний доступ к файлу производился N дней назад. (`-atime +1` найдет файлы, доступ к которым осуществлялся как минимум два дня назад);
3) `-mtime N` – последнее изменение файла было N дней назад;
4) `-ctime N` – статус файла последний раз изменялся N дней назад;
5) `-newer другой_файл` – файл был модифицирован позднее, чем другой_файл;
6) `-size [±]N[cwbkMG]` – размер файла равен N блокам, если указано +N, тогда размер файла больше N, -N – меньше. Символ после N означает

размер блока (b – 512 байт, c – байт, w – 2 байта, k – Кбайт, M – Мбайт, G – Гбайт);

7) `-type C` – файл имеет тип c, где c есть b (блочный специальный файл), c (символьный специальный файл), d (каталог), p (именованный канал), f (обычный файл), l (символьная ссылка) или s (сокет);

8) `[-perm] [-]восьмеричное_число` – режим доступа к текущему файлу в точности равен `восьмеричному_числу`. Если перед `восьмеричным_числом` указан знак -, то для сравнения из режима файла берутся только биты, соответствующие битам `восьмеричного_числа`, равным единице;

9) `-links n` – на файл имеется `n` ссылок;

10) `-user имя_пользователя` – файл принадлежит пользователю с данным именем. Разрешены цифровые идентификаторы пользователя;

11) `-group имя_группы` – файл принадлежит группе с данным именем. Разрешены цифровые идентификаторы группы.

Критерии можно объединять, используя операторы. Ниже приведены операторы в порядке убывания их приоритета:

- унарная операция отрицания, обозначается ! (! критерий);
- логическое И, обозначается пробелом (`критерий1 критерий2`);
- логическое ИЛИ, обозначается `-o` (`критерий1 -o критерий2`).

Когда выполняется команда `find`, можно выполнять различные действия над найденными файлами.

Основные действия:

1) `-exec команда \;` – выполнить команду. Запись команды должна заканчиваться экранированной точкой с запятой. Строка «{}» заменяется текущим маршрутным именем файла;

2) `execdir команда \;` – то же самое что и `exec`, но команда вызывается из подкаталога, содержащего текущий файл;

3) `-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;

4) `-print` – вывод имени файла на экран.

Примеры:

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

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

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

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

3. 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 {} \;
```

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

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

3.2.6.2. Команда whereis

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

Опции:

1) `-b` – вывод информации только об исполняемых файлах;

2) `-m` – вывод информации только о страницах справочного руководства;

3) `-s` – вывод информации только об исходных файлах.

3.2.7. Сжатие и упаковка файлов

Сжатие и упаковка файлов выполняются 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) директории]
```

Пример использования команды упаковки архива:

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

Распаковка содержимого архива в текущий каталог выполняется следующей командой:

```
tar -xf [имя файла архива]
```

Далее приводится пример использования команды распаковки архива:

```
$ tar -xf moi_dokumenti.tar
```

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

3.2.8. Сравнение файлов

3.2.8.1. Команда `cmp`

Команда `cmp` сравнивает два файла, и, если они различаются, сообщает о первом байте и строке, где было обнаружено различие.

Синтаксис:

```
cmp OPTIONS... FROM-FILE [TO-FILE]
```

Если команды после своего выполнения возвращает «0» – различий не найдено, если «1» – найдены, если «2» – обнаружена ошибка.

Опции:

1) `-c` – печать различающихся символов;
2) `-l` – печать смещения (десятичного) и значения (восьмеричного) всех различающихся байтов;
3) `-s` – возвращение выходного статуса, показывающего отличаются ли файлы.

3.2.8.2. Команда `diff`

Команда `diff` сравнивает содержимое двух файлов и возвращает в стандартный выходной поток список изменений, необходимых для преобразования первого файла во второй. Если различий не найдено, команда `diff` ничего не возвращает.

Также команда `diff` может сравнивать каталоги.

Следующий пример использования демонстрирует использование команды `diff` для сравнения двух каталогов `directory1` и `directory`:

```
diff directory1 directory
```

3.2.9. Мониторинг и управление процессами

3.2.9.1. Команда `ps`

Команда `ps` отображает список текущих процессов. Колонка команд указывает имя процесса, колонка PID (идентификатор процесса) – номер процесса (этот номер используется, для операций с процессом).

Для просмотра всех запущенных процессов в системе используется ключ `-A`.

3.2.9.2. Команды `kill` и `killall`

Команда `kill` принудительно завершает процесс с указанным идентификатором. Команда `killall` завершает все процессы с данным именем.

Команда `kill` обладает следующим синтаксисом:

```
kill [идентификатор процесса]
killall [имя процесса]
```

Далее приводится пример использования `kill`:

```
kill 6583
killall httpd
```

Команда `kill` посылает сигнал KILL процессу, в результате чего процесс должен этот сигнал получить и соответственно повести себя. По умолчанию процессу посылается сигнал TERM, однако это не всегда помогает завершить процесс принудительно. В этих случаях имеет смысл использовать команду `kill` с ключом `-9`.

Пример использования:

```
# kill -9 2395
```

3.2.9.3. Команда `df`

Утилита `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 указаны в блоках по 1 Кбайт по умолчанию.

Синтаксис:

```
df [опция]... [файл]...
```

Опции:

1) `-a`, `--all` – включать виртуальные файловые системы;
2) `-B`, `--block-size=РАЗМЕР` – использовать блоки указанного РАЗМЕРА (в байтах). Например, укажите `-BM` для вывода блоков размером в 1048576 байт;
3) `--total` – подсчитать общий объем в конце;
4) `-h`, `--human-readable` – печатать размеры в удобочитаемом формате (например, 1K 234M 2G);
5) `-H`, `--si` – то же, но использовать степени 1000, а не 1024;
6) `-i`, `--inodes` – вывести информацию об индексных дескрипторах, а не об использовании блоков;
7) `-k` – аналог `--block-size=1K`;
8) `-l`, `--local` – ограничиться выводом локальных файловых систем;
9) `-t`, `--type=ТИП` – перечислить только файловые системы указанного ТИПА;
10) `-T`, `--print-type` – печатать тип файловой системы;
11) `-x`, `--exclude-type=ТИП` – исключить файловые системы указанного ТИПА.

3.2.9.4. Команда `du`

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

Синтаксис:

```
du [опции][файл...]
```

Опции:

1) `-0`, `--null` – вместо символа новой строки выводить нулевой байт;
2) `-a`, `--all` – выводить общую сумму для каждого заданного файла, а не только для каталогов;

3) `-B, --block-size=РАЗМЕР` – перед печатью преобразовать размеры к РАЗМЕР. Например, используйте `-BM` для печати объема в единицах равных 1048576 байтов;
4) `-c, --total` – подсчитать общий объем в конце. Может быть использовано для выяснения суммарного использования дискового пространства для всего списка заданных файлов;
5) `-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. `--max-depth=0` эквивалентно `--summarize`;
6) `-P, --no-dereference` – не следовать по символически ссылкам (по умолчанию);
7) `-S, --separate-dirs` – выдавать отдельно размер каждого каталога, не включая размеры подкаталогов;
8) `-s, --summarize` – отобразить только сумму для каждого аргумента;
9) `-t, --threshold=РАЗМЕР` – исключить элементы, которые меньше РАЗМЕРа, если это значение положительно, или значение которых больше, если значение отрицательно;
10) `-X, --exclude-from=ФАЙЛ` – исключить файлы, соответствующие любому образцу из файла ФАЙЛ;
11) `--exclude=ШАБЛОН` – исключить файлы, соответствующие образцу ШАБЛОН;
12) `-x, --one-file-system` – пропустить каталоги на других файловых системах.

ШАБЛОН – это образец командной строки (не регулярное выражение). Знак `?` соответствует любому символу. Знак `*` соответствует любой строке (ни одного, один или несколько символов). Например, `*.o` будет соответствовать всем файлам, чьи имена заканчиваются на .o. Таким образом, команда `du --exclude='*.o'` пропустит все файлы и подкаталоги, заканчивающие на `.o` (в том числе и файл `.o`).

3.2.9.5. Команда `which`

Команда `which` – отображает полный путь к указанным командам или сценариям.

`which` принимает один или более аргументов имя_программы. Для каждогоиз них она выводит тот полный путь к исполняемому файлу, который быдет использовани командной оболочкой, если имя_программы ввести в качестве команды в командной строке. Эта утилита выполняет поиск исполняемых файлов или сценариев в каталогах, перечисленных в переменной окружения PATH, используя тот же алгоритм что и bash.

Синтаксис:

```
which [опции] [--] имя_программы [...]
```

Опции:

1) `--all`, `-a` – выводит все совпавшие исполняемые файлы по содержимому в переменной окружения PATH, а не только первый из них;
2) `--read-alias`, `-i` – считывает псевдонимы, поступающие из стандартного ввода и направляет на стандартный вывод информацию по совпавшим. Эта опция полезна в сочетании с использованием псевдонима для самой команды `which`. Например: `which='alias | which -i'`;
3) `--skip-alias` – игнорирует опцию `--read-alias`. Эта опция полезна для точного поиска обычных двоичных файлов, даже если опция `--read-alias` задана в псевдониме или в функции для `which`;
4) `--skip-functions` – игнорирует опцию `--read-functions`, если таковая имееется. Это опция полезна для точного поиска обычных двоичных файлов, даже если опция `--read-functions` задана в псевдониме или в функции для `which`;
5) `--skip-dot` – пропускает все каталоги из переменной окружения `PATH`, которые начинаются с точки;
6) `--tty-only` – не обрабатывает опции, которые находятся правее этой, если они поступают не с терминала (`tty`).

Домашний каталог определяется из переменной окружения HOME, поэтому если эта переменная не определена, то это вызовет прерывание выполния данной программы. Программа `which` будет считать два эквивалентных каталога различными, если один из них содержит путь с сисмволической ссылкой.

В следующем примере показано два возможных варианте вывода одной и той же команды `which`, в зависимости от контекста ее применения. В первом случае, вмсето полного пути выводится `~/` и `./` (для исполняемого файла, расположенного в домашнем каталоге пользователя), когда команда `which` непосредственно запущена пользователем из командной строки. Во втором – команда `which` запущена из скрипта. Здесь будет выведен полный путь к указанной программе:

```
$ which q2
~/bin/q2
$echo which q2
/nome/test/bin/q2
```

3.2.9.6. Утилита `more`

Утилита `more` – осуществляет поэкранное отображение текста файла.

Синтаксис:

```
more [ОПЦИИ] ФАЙЛ [...]
```

Опции:

1) `-номер` – задает число строк на экране;
2) `-d` – печатать в конце каждого заполненного экрана сообщение `"[Press space to continue, 'q' to quit.]"` ("Нажмите пробел для продолжения, 'q' – для выхода"). В случае если нажата некорректная клавиша будет напечатано сообщение `"[Press 'h' for instructions.]"` ("Нажмите 'h' для отображения инструкций");
3) `-l` – по умолчанию more обрабатывает `^L` (прогон страницы) как специальный символ и останавливается после каждой строки с таким символом. Опция `-l` отключает это поведение;
4) `-f` – считать логические строки, а не экранные строки (как будто длинные строки не были перенесены);

5) -p – не прокручивать текст на экране. При этом экран будет очищен и на нем будет выведен новый текст;
6) -c – не прокручивать текст на экране. При этом текст будет построчно заменен на новый;
7) -s – сжать несколько пустых линий до одной;
8) -u – скрыть подчеркивание;
9) +/ – указать строку для поиска в файле перед его отображением;
10) +номер – начать с заданного номера строки.

Опции more также они принимаются из переменной окружения MORE. Опции, указанные в командной строке, имеют больший приоритет.

Команды:

1) h или ? – отобразить справку по интерактивным командам;
2) ПРОБЕЛ – показать следующие k строк текста. По умолчанию – размер экрана;
3) z – показать следующие k строк текста. По умолчанию – размер экрана. Аргумент становится новым значением по умолчанию;
4) ВВОД – показать следующие k строк текста. По умолчанию – 1. Аргумент становится новым значением по умолчанию;
5) d или ^D – прокрутить k строк. По умолчанию – 11. Аргумент становится новым значением по умолчанию;
6) q или Q или ПРЕРЫВАНИЕ – выход;
7) s – пропустить вперед k строк. По умолчанию – 1;
8) f – пропустить вперед k экранов текста. По умолчанию – 1;
9) b или ^B – прокрутить назад k экранов текста. По умолчанию – 1. Работает только с файлами, но не с каналами;
10) ′ – перейти в место, в котором был начал предыдущий поиск;
11) = – показать текущий номер строки;
12) /шаблон – искать k-тое вхождение регулярного выражения. По умолчанию – 1;

13) `n` – искать k-тое вхождение последнего регулярного выражения. По умолчанию – 1;
14) `!команда` или `:!команда` – выполнить команду оболочки;
15) `v` – открыть текущий файл в редакторе (с текущей позиции). Редактор ищется в переменных окружения VISUAL и EDITOR (в этом порядке). Если обе переменные не заданы, то будет использован редактор vi;
16) `^L` – перерисовать экран;
17) `:n` – перейти к k-тому следующему файлу. По умолчанию – 1;
18) `:P` – перейти к k-тому предыдущему файлу. По умолчанию – 1;
19) `:f` – показать имя файла и номер строки;
20) `.` – повторить предыдущую команду.

Утилита more использует следующие переменные окружения:

- MORE – позволяет задать опции;
- SHELL – текущий интерпретатор команд (обычно задается самим интерпретатором при входе в систему);
- TERM – указывает тип терминала. Используется для определения символов управления экраном;
- VISUAL – указывает предпочитаемый редактор;
- EDITOR – указывает редактор, в случае если не задана переменная окружения VISUAL.

3.2.9.7. `chmod`

`chmod` – изменяет права доступа к файлу.

Синтаксис:

```
chmod [ОПЦИЯ]... РЕЖИМ[,РЕЖИМ]... ФАЙЛ...
chmod [ОПЦИЯ]... ВОСЬМЕРИЧНЫЙ-РЕЖИМ ФАЙЛ...
chmod [ОПЦИЯ]... --reference=ИФАЙЛ ФАЙЛ...
```

Опции:

1) `-c`, `--changes` – тоже что и `--verbose`, но сообщено будет только о выполненных изменениях;

2) `-f`, `--silent`, `--quiet` – не показывать большинство сообщений об ошибках;
3) `-v`, `--verbose` – выводить диагностическую информацию для каждого файла;
4) `--no-preserve-root` – не обрабатывать «/» специальным образом (по умолчанию);
5) `--preserve-root` – не выполнять рекурсивные операции с «/»;
6) `--reference=ИФАЙЛ` – использовать режим файла `ИФАЙЛ`;
7) `-R`, `--recursive` – рекурсивно изменять файлы и каталоги.

`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). Буква `a` эквивалетна `ugo`. 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 имеющися 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.

Числовой режим состоит из не более четырех восьмиричных цифр (от нуля до семи), которые складываются из битовых масок 4, 2 и 1. Любые пропущенные разряды дополняются лидирующими нулями. Первая цифра выбирает установку идентификатора пользователя (setuid) (4) или идентификатора группы (setgid) (2) или stiky-бита (1). Вторая цифра задает права доступа для пользователя, владеющего данным файлом: чтение (4), запись (2) и выполнение (1). Третья – права доступа для группы (с теми же значениями, что и вторая цифра). Четвертая – права доступа для остальных пользователей (с теми же значениями, что и вторая цифра).

Примеры:

1. Чтобы установить права, позволяющие владельцу читать и писать в файл, а членам группы и прочим пользователям только читать, надо сложить 0400, 0200, 0040 и 0004. Таким образом, команду можно записать двумя способами:

```
chmod 644 f1
chmod u=rw,go=r f1
```

2. Позволить всем выполнять файл `f2`:

```
chmod +x f2
```

3. Запретить удаление файла `f3`:

```
chmod +t f3
```

4. Дать всем права на чтение запись и выполнение, а также на переустановку идентификатора группы при выполнении файла `f4`:

```
chmod =rwx,g+s f4
chmod 2777 f4
```

Возможные сообщения об ошибках, при использовании команды `chmod`:

```
chmod: изменение прав доступа для <файл>: Операция не позволена
chmod: невозможно получить доступ к <путь>/<файл>: Нет такого файла или каталога
```

3.2.9.8. `chown`

Команда `chown` изменяет владельца и (или) группу для каждого заданного файла.

Синтаксис:

```
chown [КЛЮЧ]…[ВЛАДЕЛЕЦ][:[ГРУППА]] ФАЙЛ ...
```

Опции:

1) `-c`, `--changes` – тоже что и `--verbose`, но сообщено будет только о выполненных изменениях;
2) `-f`, `--silent`, `--quiet` – не показывать большинство сообщений об ошибках;
3) `-v`, `--verbose` – выводить диагностическую информацию для каждого файла;
4) `--no-preserve-root` – не обрабатывать «/» специальным образом (по умолчанию);
5) `--preserve-root` – не выполнять рекурсивные операции с «/»;
6) `--reference=ЭФАЙЛ` – использовать владельца и группу ЭФАЙЛа;
7) `-R`, `--recursive` – рекурсивно обрабатывать файлы и каталоги.

Изменить владельца может только владелец файла или пользователь с идентификатором root. 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:

1. Поменять владельца `/u` на пользователя test:

   ```
   chown test /u
   ```

2. Поменять владельца и группу `/u`:

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

3. Поменять владельца `/u` и вложенных файлов на test:

   ```
   chown -hR test /u
   ```

Возможные сообщения об ошибках, при использовании команды `chown`:

```
chown: изменение владельца <файл>: Операция не позволена
```

chown: невозможно получить доступ к <файл>: Нет такого файла или каталога

3.2.9.9. umask

Утилита umask задает маску режима создания файла в текущей среде командного интерпретатора равной значению, задаваемому операндом режим. Эта маска влияет на начальное значение битов прав доступа всех создаваемых далее файлов.

Синтаксис:

umask [-p] [-S] [режим]

Пользовательской маске режима создания файлов присваивается указанное восьмеричное значение. Три восьмеричные цифры соответствуют правам на чтение/запись/выполнение для владельца, членов группы и прочих пользователей соответственно. Значение каждой заданной в маске цифры вычитается из соответствующей «цифры», определенной системой при создании файла. Например, umask 022 удаляет права на запись для членов группы и прочих пользователей (у файлов, создававшихся с режимом 777, он оказывается равным 755; а режим 666 преобразуется в 644).

Если маска не указана, выдается ее текущее значение.

Команда umask распознается и выполняется командным интерпретатором bash.

Команду umask целесообразно включить в пользовательский профиль, тогда она будет автоматически вызываться при входе в систему и установит нужный режим доступа к создаваемым файлам и каталогам.

3.2.9.10. chattr

chattr – изменяет атрибуты файлов файловой системы ext2fs.

Синтаксис:

chattr [-Rvf] [+-=aAcCdDeijsSTtu] [-v версия] файлы...

Оператор '+' означает добавление выбранных атрибутов к существующим атрибутам; '-' означает их снятие; '=' означает определение только этих указанных атрибутов для файлов.

Символы `'ASacDdijsTtu'` указывают на новые атрибуты для файлов: не обновлять время последнего доступа (atime) к файлу (A), синхронное обновление (S), только добавление к файлу (a), сжатый (c), синхронное обновление каталогов (D), не архивировать (d), неизменяемый (i), журналирование данных (j), безопасное удаление (s), вершина иерархии каталогов (T), нет tail-merging (t), неудаляемый (u).

Опции:

1) `-R` – рекурсивно изменять атрибуты каталогов и их содержимого. Символические ссылки игнорируются;
2) `-V` – выводит расширенную информацию и версию программы;
3) `-f` – подавлять сообщения об ошибках;
4) `-v` версия – установить номер версии/генерации файла.

Когда изменяется файл с атрибутом (A) время последнего доступа к нему не изменяется. Это позволяет избежать некоторого количества дисковых операций ввода/вывода на портативных персональных компьютерах.

Файл с атрибутом (a) можно открыть для записи только в режиме добавления. Только пользователь с идентификатором root или процесс с возможностью CAP_LINUX_IMMUTABLE может устанавливать и снимать этот атрибут.

Файл с атрибутом (c) автоматически сжимается на диске ядром. Чтение из такого файла возвращает несжатые данные. При записи в такой файл данные перед записью на диск сжимаются.

Когда изменяется каталог с атрибутом (D) изменения синхронно записываются на диск. Это эквивалентно опции монтирования `dirsync' примененной к подмножеству файлов.

Файл с атрибутом (d) не является кандидатом на архивирование при использовании команды `dump`.

Атрибут (E) используется экспериментальными сжимающими патчами для того, чтобы показать, что сжатый файл содержит ошибки сжатия. Он не может быть

установлен или сброшен с помощью `chattr`, хотя его можно просмотреть с помощью `lsattr`.

Атрибут (I) используется кодом `htree` для того, чтобы показать, что каталог индексируется с использованием хэши-рующих деревьев. Он не может быть установлен или сброшен с помощью `chattr`, хотя его можно просмотреть с помощью `lsattr`.

Файл с атрибутом (i) не может быть изменен: он не может быть удален или переименован, к этому файлу не могут быть созданы ссылки, и никакие данные не могут быть записаны в этот файл. Только пользователь с идентификатором root или процесс с возможностью CAP_LINUX_IMMUTABLE может устанавливать и снимать этот атрибут.

При записи в файл с атрибутом (j) все данные записываемые в такой файл записываются в журнале ext3, прежде чем они будут записаны непосредственно в файл, если файловая система смонтирована с опциями `data=ordered` или `data=writeback`. Если файловая система смонтирована с опцией `data=journalled`, то все данные журналируются, и этот атрибут не дает никакого эффекта. Только пользователь с идентификатором root или процесс с возможностью CAP_SYS_RESOURCE может устанавливать или снимать этот атрибут.

Когда удаляется файл с атрибутом (s), все его блоки заполняются нулями.

Когда изменяется файл с атрибутом (S), все изменения синхронно записываются на диск; это эквивалентно опции монтирования 'sync' примененной к подмножеству файлов.

Каталог с атрибутом (T) будет поднята на вершину иерархии каталогов для целей Orlov block allocator.

У файла с атрибутом (t) в конце не будет partial block fragment соединенного с другими файлами (для тех файловых систем, которые поддерживают tail-merging). Это необходимо для приложений, таких как LILO, которые читают файловую систему напрямую, и которые не понимают tail-merged файлы.

Когда удаляется файл с атрибутом (u) его содержимое сохраняется. Это позволяет пользователю восстановить файл.

Атрибут (X) используется экспериментальными сжимающими патчами, чтобы показать, что исходное содержимое сжатых файлов доступно напрямую. В данное время он не может быть установлен или переустановлен с помощью `chattr(1)`, но может быть показан с помощью `lsattr`.

Атрибут (Z) используется экспериментальными сжимающими патчами, чтобы показать, что сжатый файл не сохранен. Он не может быть установлен или переустановлен с помощью `chattr`, но может быть показан с помощью `lsattr`.

Возможные сообщения об ошибках, при использовании команды `chattr`:

```
chattr: Отказано в доступе While reading flags on <файл>
chattr: Нет такого файла или каталога While trying to stat <файл>
```

3.2.9.11. `lsattr`

`lsattr` – выдает список атрибутов файлов на Linux ext2fs.

Синтаксис:

```
lsattr [ -RVadv ] [ файлы... ]
```

Описание:

`lsattr` выдает список атрибутов файлов на ext2fs. В `chattr` описаны все атрибуты и их назначение.

Опции:

1) `-R` – рекурсивно изменять атрибуты каталогов и их содержимого. Символические ссылки игнорируются;
2) `-V` – выводит расширенную информацию и версию программы;
3) `-a` – просматривает все файлы, в каталоге включая те, имена которых начинаются с '.';
4) `-d` – отображает каталоги также, как и файлы вместо того, чтобы просматривать их содержимое.
5) `-v` – просматривает номера версий/генераций файлов.

Возможные сообщения об ошибках, при использовании команды `lsattr`:

```
lsattr: Отказано в доступе While reading flags on <файл>
```

```
lsattr: Нет такого файла или каталога While trying to stat <файл>
```

3.2.10. Текстовый редактор Vi

Текстовый редактор – это программа, которая предназначена для редактирования (составления и изменения) файлов, содержащих только текст.

Vi (visual editor) является стандартным текстовым редактором Unix для редактирования текста.

3.2.10.1. Запуск vi

Vi может быть запущен из командной строки разными способами:

- запуск в командном режиме:

```
vi
```

при этом будет запущен vi с пустым буфером;

- запуск для существующего файла:

```
vi filename
```

где filename – имя файла, который надо редактировать;

- запуск на определенной строке файла:

```
vi +47 /usr/src/linux/init/main.c
```

vi запускается на 47-й строке `/usr/src/linux/init/main.c`. При этом vi покажет на экране указанный файл и поместит курсор на заданной строке. В случае, если указана строка за пределами конца файла, vi поместит курсор на последней строке.

3.2.10.2. Режимы работы vi

vi работает в различных режимах, которые используются для выполнения различных задач:

- «Командный режим» – выполнение различных команд для работы с текстом, перемещения по файлу, сохранения, выхода и изменения режимов;
- «Режим ввода текста» – вставка и замена текста;
- «Режим строчного редактора» – используется для управления файлами.

3.2.10.3. Открытие/создание файла

Для открытия или создания нового файла в командном режиме необходимо ввести команду:

```
:e filename
```

Для сохранения файлов используется ряд команд.

Следующая команда сохраняет файл с существующим именем:

```
:w
```

Следующая команда сохраняет файл с заданным именем:

```
:sav filename
```

В случае попытки выполнить запись в файл, владельцем которого является другой пользователь, операция сохранения не может выполниться, о чем будет выдано соответствующее предупреждение.

3.2.10.4. Перемещение по файлу

Перемещение по файлу происходит с помощью клавиш со стрелками (в случае правильного описания терминала) либо с помощью следующих клавиш:

- <h> – перемещение на позицию влево;
- <j> – перемещение вниз;
- <k> – перемещение вверх;
- <l> – перемещение на позицию вправо;
- <Ctrl>+<F> – перемещение на страницу вниз;
- <Ctrl>+<B> – перемещение на страницу вверх.

Также можно использовать следующие быстрые клавиши:

- <0> – перемещение в начало текущей строки;
- <$> – перемещение в конец текущей строки;
- <w> – перемещение на слово вправо;
- <b> – перемещение на слово влево;
- <g> – перемещение в начало файла;
- <G> – перемещение в конец файла.

3.2.10.5. Редактирование файла

Для редактирования текста в файле необходимо перейти в режим ввода. Чтобы перейти из командного режима в режим ввода текста необходимо воспользоваться командой:

```
i
```

после чего можно приступать к вводу текста.

Для возврата к командному режиму нужно использовать клавишу <ESC>.

При редактировании файла допускается использовать следующие команды:

1) `R`, `i` – переход в режим ввода – замена текста под курсором;

2) `I` – переход в режим ввода с начала текущей строки;

3) `o` – переход в режим ввода с новой строки под курсором;

4) `O` – переход в режим ввода с новой строки над курсором;

5) `a` – переход в режим ввода после курсора;

6) `x` – удаление символа под курсором;

7) `X` – удаление символа перед курсором;

8) `dd` – удаление текущей строки;

9) `d<число>d` – удаление числа строк, начиная с текущей;

10) `yy` – копирование текущей строки в неименованный буфер;

11) `y<число>y` – копирование числа строк, начиная с текущей, в неименованный буфер;

12) `p` – вставка строки из неименованного буфера под курсор;

13) `P` – вставка строки из неименованного буфера над курсором;

14) `J` – слияние текущей строки со следующей;

15) `u` – отмена последней команды;

16) `.` – повтор последней команды.

Для перехода в режим строчного редактора ED необходимо нажать `Shift+:`.

3.2.10.6. Сохранение и выход

При работе в режиме ввода необходимо предварительно нажать <ESC> для перехода в командный режим.

Для выхода из редактора vi без сохранения изменений необходимо воспользоваться следующей командой:

```
q!
```

Для выхода из редактора с сохранением изменений, сделанных в файле, используется следующей команда:

```
:wq
```

Чтобы сохранить файл, но не выходить из редактора vi, используется следующей команда:

```
:w
```

3.2.11. Редактор Vim

Редактор Vim – свободный режимный текстовый редактор, созданный на основе более старого vi. Одна из главных особенностей редактора – применение двух основных, вручную переключаемых, режимов ввода: командного (после запуска редактор находится в нем) и текстового (режим непосредственного редактирования текста).

3.2.11.1. Режимы работы

В Vim существуют четыре основных режима работы:

- основной;
- режим непосредственного редактирования текста;
- режим командной строки;
- визуальный режим.

По умолчанию Vim начинает свою работу в основном режиме, который также называют командным. Нажатие клавиш в этом режиме воспринимается как команды (копирования, удаления, перемещения текста и других команд).

Основной режим предназначен для просмотра файлов, ввода команд и перехода из него в другие режимы. Из любого режима в командный режим можно перейти, нажав (в некоторых случаях дважды) клавишу <Esc>.

При нажатии клавиши <:> происходит переход в режим командной строки Vim, в которой можно вводить команды. За двоеточием следует сложная команда (например, поиска или замены), которая после ввода передается на исполнение нажатием клавиши <Enter>. После выполнения команды редактор возвращается в нормальный режим. К этому режиму также относятся команды поиска дальше по тексту «/», поиск назад по тексту «(?)» и команда-фильтр «!» для передачи данных внешнему фильтру.

Другие примеры команд:

- команда выхода `quit` (Vim принимает сокращения, поэтому можно ввести просто `q`);
- команда сохранения `write` (или `w`), параметром которой может быть имя файла;
- команда вызова справки `help` (или `h`).

Для перехода из командного режима в режим непосредственного редактирования текста можно нажать клавишу <i> (для начала вставки текста на месте курсора) или клавишу <a> (для начала вставки текста после курсора).

В этом режиме по умолчанию набранные символы не воспринимаются как команды, а вставляются в существующий текст. Однако даже в этом режиме можно задать особые действия редактора, выполняемые при нажатии определенных клавиш, или их сочетаний.

Для вставки, удаления, изменения и автодополнения текста могут использоваться клавиатурные сочетания вида <Ctrl>+<R>.

Визуальный режим предназначен в первую очередь для выделения блоков текста. Для перехода используются следующие сочетания клавиш:

- <v> – для посимвольного выделения текста;
- <Shift>+<v> – для построчного выделения текста;
- <Ctrl>+<v> – для блочного выделения текста.

К выделенным фрагментам текста затем можно применить команды нормального режима (например, удаление выделенного текста или его замена).

3.2.11.2. Основные возможности

Перечисленные ниже команды вводятся в основном режиме. Все они имеют команднострочные аналоги и могут быть легко переопределены.

3.2.11.2.1 Переходы

Для перехода на строку с номером n необходимо воспользоваться командой:

```
G
```

Для перехода к началу текста необходимо воспользоваться командой:

```
1G
```

Для перехода к концу текста необходимо воспользоваться командой:

```
$G
```

Для перехода на n символов в нужную вам сторону можно использовать клавиши со стрелками.

3.2.11.3. Метки

Используются для отметки позиции и быстрого к ней перехода. Метки нижнего регистра действительны в пределах данного файла, метки верхнего регистра действуют во всех открытых файлах. Список всех меток можно получить следующей командой:

```
marks
```

3.2.11.4. Сессии

При ведении группы проектов нередко желательно сохранить текущее состояние и настройки редактора, чтобы в дальнейшем продолжить работу с того же места. Для этого предназначены сессии, которые можно создать следующей командой:

```
:mksession /path/to/Session.vim
```

Сессии читаются следующей командой:

```
:so /path/to/Session.vim
```

Сохранение текущего контекста (например, положение курсора в тексте) выполняется следующей командой:

```
:mkview
```

Для открытия сохраненного состояния используется следующая команда:

```
:loadview
```

3.2.12. Служба xinetd

Служба xinetd запускает процессы, которые предоставляют различные сервисы Интернет. В отличие от сервисов, которые запускаются во время инициализации системы и находятся в режиме ожидания запросов, xinetd представляет собой только один процесс, который прослушивает все порты сервисов, перечисленных в файле конфигурации `xinetd.conf`. При поступлении запроса производит xinetd запуск соответствующего сервера.

Сервисы, перечисленные в конфигурационном файле xinetd, можно разделить на две группы. Сервисы из первой группы называются multi-threaded (многопоточными) и они требуют разветвления нового серверного процесса для каждого нового запроса на соединение. Далее соединением управляет новый сервер. Для таких сервисов xinetd продолжает прослушивать сеть для приема новых запросов, чтобы вызвать новые серверы. Вторая группа включает службы single-threaded (однопоточные), для которых xinetd не будет управлять новыми запросами, пока сервер не завершит свою работу. Службы в этой группе обычно основаны на передаче данных через датаграммы (UDP).

Служба xinetd применяется в основном для того, чтобы сохранить системные ресурсы через недопущение разветвления огромного числа процессов, которые могут бездействовать в течение большей части времени своей работы. В то же время, выполняя эту функцию, xinetd работает согласно идее суперсервера, предоставляя такие возможности, как управление доступом и протоколирование. Кроме того, xinetd не ограничена сервисами, перечисленными в файле `/etc/services`, поэтому данная служба может использоваться для запуска сервисов специального назначения.

3.2.12.1. Опции xinetd

Синтаксис:

```
xinetd [опции]
```

Параметры:

1) `-d` – активирует режим отладки. Указание этой опции приводит к большому количеству отладочных сообщений, которые делают возможным использование отладчика на xinetd;
2) `-syslog syslog_facility` – разрешает протоколирование создаваемых xinetd сообщений через syslog с заданным syslog facility. Поддерживаются следующие имена facility: daemon, auth, user, local[0-7] (посмотрите syslog.conf для того, чтобы понять их назначение). Данная опция неэффективна в режиме отладки, так как все необходимые сообщения отправляются на терминал;

3) `-filelog файл_журнала` – сообщения, создаваемые xinetd, будут помещаться в указанный файл. Сообщения всегда добавляются к уже существующему файлу. Если файл не существует, то он будет создан. Данная опция не действует в режиме отладки;
4) `-f файл_настроек` – задает файл, который xinetd использует для настройки. По умолчанию это `/etc/xinetd.conf`;
5) `-pidfile pid_файл` – в этот файл записывается идентификатор процесса. Данная опция неэффективна в режиме отладки;
6) `-stayalive` – xinetd будет оставаться запущенным, даже если не задано никаких служб;
7) `-loop rate` – устанавливает верхнюю величину цикла, по которой определяется, что служба работает с ошибками и по которой она отключается. Величина цикла задается в терминах количества серверов в секунду, которое может быть запущено в обработку (fork). Для этой опции, корректное значение определяется скоростью вашей машины. По умолчанию равно 10;
8) `-reuse` – xinetd будет устанавливать опцию сокета SO_REUSEADDR перед привязкой сокета службы к какому-либо интернет адресу. Это позволяет привязать адрес, даже если есть программа, которая уже использует его, например, в том случае, если некоторые серверы были запущены во время предыдущего запуска xinetd и еще не завершили свою работу. Данная опция не оказывает влияния на службы RPC;
9) `-limit proc_limit` – устанавливает ограничение на количество одновременно запущенных процессов, которые может запустить xinetd. Ее назначение предотвращать переполнение таблицы процессов;
10) `-logprocs limit` – устанавливает ограничение на количество одновременно запущенных серверов на один идентификатор удаленного пользователя;

11) `-shutdownprocs limit` – устанавливает ограничение на количество одновременно запущенных серверов для завершения работы службы;

12) `-version` – вывести информацию о версии xinetd;

13) `-cc interval` – xinetd будет выполнять периодические проверки своего внутреннего состояния каждые `interval` секунд.

Опции `syslog` и `filelog` являются взаимно исключающими. Если ни одна из них не задана, то по умолчанию используется `syslog` с `daemon facility`. Не путайте сообщения xinetd с сообщениями, которые создаются службами, последние протоколируются, только если это задано в файле с настройками.

Файлы xinetd:

- `/etc/xinetd.conf` – стандартный конфигурационный файл;
- `/var/run/xinetd.dump` – стандартный файл дампа.

3.2.12.2. Управление xinetd

xinetd выполняет определенные действия при получении определенных сигналов. Действия, ассоциированные с соответствующими сигналами, могут быть переопределены путем редактирования файла `config.h` и последующей компиляции.

Сигналы:

- SIGUSR2 – заставляет выполнить жесткую перенастройку, которая означает, что xinetd перечитает файл с настройками и завершит работу серверов для тех служб, которые больше не доступны. Управление доступом выполняется снова на уже запущенные сервера через проверку удаленных подключений, времени доступа и копий серверов. Если количество копий серверов уменьшается, то некоторые произвольно выбранные сервера будут убиты, чтобы соблюсти ограничение; это случится после завершения работы тех серверов, которые попадают под ограничение доступа с удаленных адресов или ограничение времени доступа. Также, если флаг INTERCEPT был сброшен и происходит его установка, то будет завершена работа любых запущенных серверов для служб с этим флагом. Цель такого поведения –

убедиться, что после жесткой перенастройки не будет запущено серверов, которые могут принимать пакеты с тех адресов, которые не соответствуют критериями управления доступом;

- SIGQUIT – приводит к завершению работы;
- SIGTERM – завершает работу всех запущенных серверов перед завершением работы xinetd;
- SIGHUP – приводит к снятию дампа внутреннего состояния (по умолчанию файл дампа это `/var/run/xinetd.dump`, чтобы изменить данное имя файла, нужна правка `config.h` и перекопиляция);
- SIGIOT – производит внутреннюю проверку того, что структуры данных, используемые программой, не повреждены. Когда проверка завершится, xinetd сгенерирует сообщение, которое скажет успешно прошла проверка или нет.

При реконфигурации файлы журналов закрываются и вновь открываются. Это позволяет удалять старые файлы журналов.

3.2.13. Crontab

Crontab – служба таблиц, управляющих работой службы cron. Crontab управляет доступом пользователя к службе cron путем копирования, создания, выдачи содержимого и удаления файлов crontab, таблиц заданий.

При вызове без опций crontab копирует указанный файл или стандартный входной поток (если файл не указан) в каталог, в котором хранятся пользовательские таблицы заданий cron.

Для создания, изменения и удаления файлов cron следует использоваться специальную утилиту crontab.

Синтаксис:

```
crontab [имя_файла]
crontab [ -elr ] имя_пользователя
```

Опции:

1) `-e` – редактирует копию файла crontab текущего пользователя или создает пустой файл для редактирования, если соответствующего файла crontab не

существует. Когда редактирование завершается, файл устанавливается в качестве пользовательского файла crontab. Переменная среды EDITOR задает редактор, вызываемый при указании опции `-e`. Все задания в файле crontab должны создаваться с помощью утилиты crontab;

2) `-l` – отображает текущий файл crontab на стандартный вывод;

3) `-r` – удаляет текущий файл crontab.

3.2.13.1. Контроль доступа к crontab

Доступ пользователя к crontab разрешен, если:

- имя пользователя указано в файле `/etc/cron.d/cron.allow`;
- файл `/etc/cron.d/cron.allow` не существует и имя пользователя не указано в файле `/etc/cron.d/cron.deny`.

Доступ пользователя к crontab не разрешен, если:

- файл `/etc/cron.d/cron.allow` существует и имя пользователя в нем не указано;
- файл `/etc/cron.d/cron.allow` не существует и имя пользователя указано в файле `/etc/cron.d/cron.deny`.

Правила разрешения и запрещения выполнения заданий применимы к пользователю root только если существуют файлы `allow/deny`.

В файлах `allow/deny` надо задавать по одному имени пользователя в строке.

3.2.13.2. Формат записи файла crontab

Редактировать crontab пользователя можно используя команду:

```
crontab -e
```

Файл crontab состоит из строк, содержащие шесть полей. Поля разделяются пробелами или символами табуляции. Первые пять полей – целочисленные шаблоны, задающие:

- минуту (0 – 59);
- час (0 – 23);
- день месяца (1 – 31);
- месяц года (1 – 12);
- день недели (0 – 6, причем 0=воскресенье).

Каждый из этих шаблонов может представлять собой звездочку (которая обозначает все допустимые значения) или список элементов через запятые. Элемент – число или два числа через дефис (что обозначает закрытый интервал). Обратите внимание, что дни можно указывать в двух полях (день месяца и день недели). Оба поля учитываются, если заданы в виде списка элементов (запись: 30 4 1,15 * 5 приведет к выполнению команды в 4:30 пополуночи первого и пятнадцатого числа каждого месяца, плюс в каждую пятницу). При указании диапазона можно пропускать некоторые его значения, указав шаг в форме `/число`. Например: «0-23/2» для поля час означает запуск команды через два часа. Шаг можно указывать также после звездочки: «каждые два часа» соответствует значению «*/2». Для задания полей месяц и день_недели можно использовать имена. Указывайте первые три буквы нужного дня или месяца на английском, регистр букв не имеет значения. Диапазоны или списки имен не разрешены.

Служба cron запускает команды, когда значения полей минута, час, месяц и хотя бы одно из полей число и день_недели, совпадают с текущим временем. Служба cron сверяет директивы с текущим временем раз в минуту.

Вместо первых пяти полей допустимо указание одного из восьми специальных триггеров:

- @reboot – выполнить команду один раз, при запуске cron;
- @yearly – выполнять команду каждое 1 января, «0 0 1 1 *»;
- @annually – эквивалентно @yearly;
- @monthly – выполнять команду в начале каждого месяца, «0 0 1 * *»;
- @weekly – выполнять команду каждое воскресенье, «0 0 * * 0»;
- @daily – выполнять команду в полночь, «0 0 * * *»;
- @midnight – эквивалентно @daily;
- @hourly – выполнять команду раз в час, «0 * * * *».

Шестое поле в строке файла crontab – строка, выполняемая командным интерпретатором в указанные моменты времени. Символ % (процент) в этом поле, если он не замаскирован \ (обратной косой), преобразуется в символ новой строки.

Только первая строка (до символа % или до конца строки) поля команды выполняется командным интерпретатором. Другие строки передаются команде как стандартный входной поток. Пустые строки, ведущие пробелы и символы табуляции игнорируются. Строки, начинающиеся с символа ('#') считаются комментариями и игнорируются. Комментарии не допускаются в тех же строках, где расположены команды cron, так как они будут распознаны как части команды. По этой же причине комментарии не разрешены в строках, задающих переменные среды.

Строка-директива представляет собой либо задание переменной среды, либо команду `cron`.

Демон cron предоставляет каждому командному интерпретатору стандартную среду, задавая переменные `HOME`, `LOGNAME`, `SHELL(=/bin/sh)`, `TZ` и `PATH`. Стандартное значение переменной `PATH` для пользовательских заданий cron – `/usr/bin`, а для заданий cron пользователя root – `/usr/sbin:/usr/bin`.

Если стандартный выходной поток и стандартный поток ошибок команд не перенаправлены, любые сгенерированные результаты или сообщения об ошибках будут отправлены пользователю по электронной почте.

3.2.13.3. Примеры

Пример 1

```
$ crontab -e
#minute (0-59),
#| hour (0-23),
#| | day of the month (1-31),
#| | | month of the year (1-12),
#| | | | day of the week (0-6 with 0=Sunday).
#| | | | | commands
# Каждые 5 минут записывать результат вывода
# команды date в файл date.txt в домашнем каталоге
*/5 * * * * date > ~/date.txt
# Выполнять задание в 18 часов 7 минут 13 числа
# каждого месяца и по пятницам
7 18 13 * 5 /home/www/myscript.pl
# Выполнять задание по воскресеньям в 10 час 30 минут
```

```
30 10 * * 0 /home/www/myscript.pl
crontab: installing new crontab
```

Вывод `crontab: installing new crontab` означает, что новый crontab успешно установлен.

Пример 2

```
# использовать для запуска команд /bin/sh
# не обращая внимание на то, что написано в /etc/passwd
SHELL=/bin/sh
# отправлять вывод выполнения команд по электронной
# почте пользователю 'paul'
# не обращая внимания на то, чей это crontab
MAILTO=paul
#
# запускать пять минут пополуночи, каждый день
5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# запускать в 14:15 первого числа каждого месяца
15 14 1 * * $HOME/bin/monthly
# запускать в 22.00 каждый рабочий день
0 22 * * 1-5 mail -s "Уже 10 вечера"
23 0-23/2 * * * echo "запуск в 00:23, 2:23, 4:23 ..., каждый
день"
5 4 * * sun echo "запуск в 4:05 каждое воскресенье"
```

3.2.14. Служба передачи файлов FTP

В ОС Альт 8 СП передача файлов обеспечивается с помощью программы lftp. Данная команда реализует протокол передачи файлов FTP. Для копирования файлов необходимо знать имя и пароль пользователя, которому принадлежат файлы на сервере службы FTP.

Для запуска `lftp` необходимо в консоли ввести команду:

```
lftp
```

После появления приглашения `lftp :~>` становятся доступными для использования внутренние команды `lftp`.

Основные внутренние команды `lftp`:

- `open` – подключение к серверу;
- `user` – идентификация при удаленном подключении;
- `close` – отключение от сервера;
- `ls` – просмотр списка файлов;
- `lcd` – смена локального каталога;
- `mkdir` – создание нового каталога;
- `lpwd` – просмотр имени каталога на локальном компьютере;
- `get` – копирование файла с сервера;
- `put` – копирование файла на сервер;
- `help` – просмотр списка доступных команд и справки по ним;
- `exit` – выход из `lftp`.

3.2.15. Защищенный интерпретатор команд SSH

Защищенный интерпретатор команд SSH – клиент-серверная система для организации защищенных туннелей для удаленного доступа к другим компьютерам.

SSH реализует соединение с удаленным компьютером, которое позволяет защититься от следующих угроз:

- прослушивание данных, передаваемых по этому соединению;
- манипулирование данными на пути от клиента к серверу;
- подмена клиента либо сервера путем манипулирования IP-адресами, DNS либо маршрутизацией.

Для создания защищенного туннеля используется программа ssh. Инициировать соединение с сервером можно командой:

```
ssh <имя_клиента>@IP_addr
```

где `IP_addr` – IP-адрес компьютера с запущенной службой sshd.

При использовании идентификации по паролю на сервере должна существовать учетная запись с указанным именем клиента.

Параметры, относящиеся к способу аутентификации, а также все прочие настройки ssh указываются в конфигурационном файле `/etc/ssh/ssh_config`.

Конфигурационные файлы разбиты на разделы, установки которых относятся к отдельному компьютеру, группе компьютеров или ко всем компьютерам, при этом установки разных разделов могут конфликтовать друг с другом. Предпочтение в данном случае будет отдаваться тому параметру, который указан раньше.

3.2.16. Работа со справочной информацией

В состав ОС Альт 8 СП входят справочные страницы, предназначенные для представления подробной информации по функциям ОС и командам, которые позволяют работать с функциями ОС, а также по конфигурационным файлам или библиотекам.

Для получения подробной справочной информации об интересующей программе или команде пользователю необходимо вызвать соответствующую справочную страницу. Вызов справочных страниц осуществляется с консоли с помощью команды `man` в следующем формате:

```
man command_name
```

где `command_name` – имя нужной программы или команды.

По умолчанию после вывода первой найденной справочной страницы команда man завершит свою работу. Для изменения работы команды man могут применяться дополнительные опции.

Опции команды `man`:

1) `-a` – выводит все доступные для запрашиваемой программы или команды справочные страницы. Например:

   ```
   man -a command_name
   ```

2) `-f` – ищет и выводит краткое описание всех справочных страниц, где имеются ссылки на запрашиваемую программу или команду. Например:

   ```
   man -f command_name
   ```

3) `--warning`-включает предупреждения
4) `-I` – включает чувствительность к регистру
5) `-h` – выводит справку по опциям командной строки и завершает работу
6) `-H` – активирует вывод в HTML и просмотр в браузере, который определен в `$BROWSER` или определен по умолчанию во время компиляции.

Например:

```
man -H[browser]
```

3.2.16.1. Управление справочными страницами

Так как большинство справочных страниц состоят более чем из одной страницы, для их вывода на экран используется команда `less`, которая позволяет перелистывать страницы с помощью клавиш <PgUp> и <PgDown>, или <Space> и <Esc> + <V>.

Выйти из справочной страницы можно в любое время с помощью клавиши <q>.

Поиск в справочной странице вызывается с помощью клавиши </>, каждое последующее найденное вхождение можно просмотреть с помощью клавиши <n>, а вернуться к предыдущему вхождению можно с помощью комбинации клавиш <Shift>+ <n>.

Справочные страницы команд можно находить по категориям, к которым они принадлежат. Для этого используется команда apropos, где цифрами можно задать категорию, к которой принадлежит команда.

Категории команд:

- «Команда 1 » – исполняемые программы и команды оболочки;
- «Команда 2» – системные вызовы;
- «Команда 3» – библиотечные вызовы;
- «Команда 4» – файлы устройств (обычно расположены в `/dev`);
- «Команда 5» – форматы файлов;
- «Команда 6» – игры;
- «Команда 7» – макропакеты и соглашения;
- «Команда 8» – программы системного администрирования;
- «Команда 9» – процедуры ядра.

На справочные страницы принято ссылаться по имени, с указанием номера категории в скобках. Часто существуют сразу несколько справочных страниц с одинаковыми именами, но в разных категориях, например man(1) и man(7). В таком

случае, команде man необходимо передать номер конкретной категории перед именем справочной страницы, например:

```
man 5 passwd
```

отобразит справочную страницу по файлу `/etc/passwd` вместо утилиты `passwd`.

3.2.16.2. Печать справочных страниц

Если необходимо распечатать справочную страницу, то предварительно нужно удалить форматирование страницы с помощью команды `col`.

Например, чтобы напечатать страницу руководства по `man`, необходимо выполнить следующую команду:

```
man man | col -b | lpr
```

Команда, приведенная выше, пропускает выводимую информацию через фильтр `col`, который форматирует текст для принтера. Затем данные, выводимые `col`, посылаются на принтер.

3.3. Завершение работы ОС

Для корректного завершения работы ОС (перезагрузки) во время ее работы запрещается выключать питание компьютера или перезагружать компьютер нажатием на кнопку «Reset», так как для корректного завершения работы требуется размонтирование файловой системы.

Перед окончанием работы с ОС необходимо завершить все работающие программы.

Для завершения работы ОС можно воспользоваться несколькими различными способами остановки системы:

- нажать комбинацию клавиш <Ctrl>+<Alt>+<Del>, что на рабочей станции приведет к вызову диалога завершения работы системы, а на сервере – к перезагрузке системы, при этом необходимо дождаться появления на экране сообщения «Reboot» (перезагрузка) и выключить питание системы;
- воспользоваться специальной командой `shutdown`, доступной пользователю с правами root;

- при наличии графической оболочки следует воспользоваться диалогом доступным в меню «Система».

3.3.1. Завершение работы ОС с помощью консоли

Для завершения работы ОС в консольном режиме рекомендуется использовать команду `shutdown`, вызывающую остановку, выключение и (или) перезагрузку ПЭВМ.

Синтаксис:

```
shutdown [опции...] [время] [сообщение...]
```

Опции:

1) `--help` – печатает краткую справку по использованию;

2) `-H`, `--halt` – остановка компьютера;

3) `-P`, `--poweroff` – выключение компьютера (по умолчанию);

4) `-r`, `--reboot` – перезагрузка компьютера;

5) `-h` – то же что и `--poweroff`, если не указана опция `--halt`;

6) `-k` – не останавливать, не выключать и не перезагружать компьютер, а просто напечатать сообщение;

7) `--no-wall` – не печатать сообщение перед остановкой/выключением/ перезагрузкой;

8) `-c` – отменить запланированное выключение. Эта опция может быть использована для отмены эффекта ранее выполненной команды `shutdown` с указанием времени отличного от +0 или now.

Аргумент [время] может быть строкой времени (в том числе, используется значение «now»). Время может быть указано в формате «чч:мм» для задания часа и минуты в формате 24 часа.

Также допускается использовать синтаксис «+м», указывая количество минут от текущего момента. «now» является синонимом для «+0» и может быть использовано для немедленного выключения компьютера. Значение времени по умолчанию (если не задано явно) считается «+1». Необходимо учитывать, что для задания сообщения также потребуется явно указать время. Если задана строка

времени, то за пять минут до выключения системы будет создан файл `/run/nologin` для того чтобы гарантировать, что никакие новые пользователи не войдут в систему.

После указания времени допускается указать аргумент [сообщение] и ввести текстовую строку, которая будет напечатана во всех терминальных сессиях.

3.3.2. Завершение работы ОС с помощью инструментов графической оболочки

Для завершения работы ОС в графическом режиме необходимо перейти в меню «Система» и выбрать пункт «Выключить...» (рис. 9).

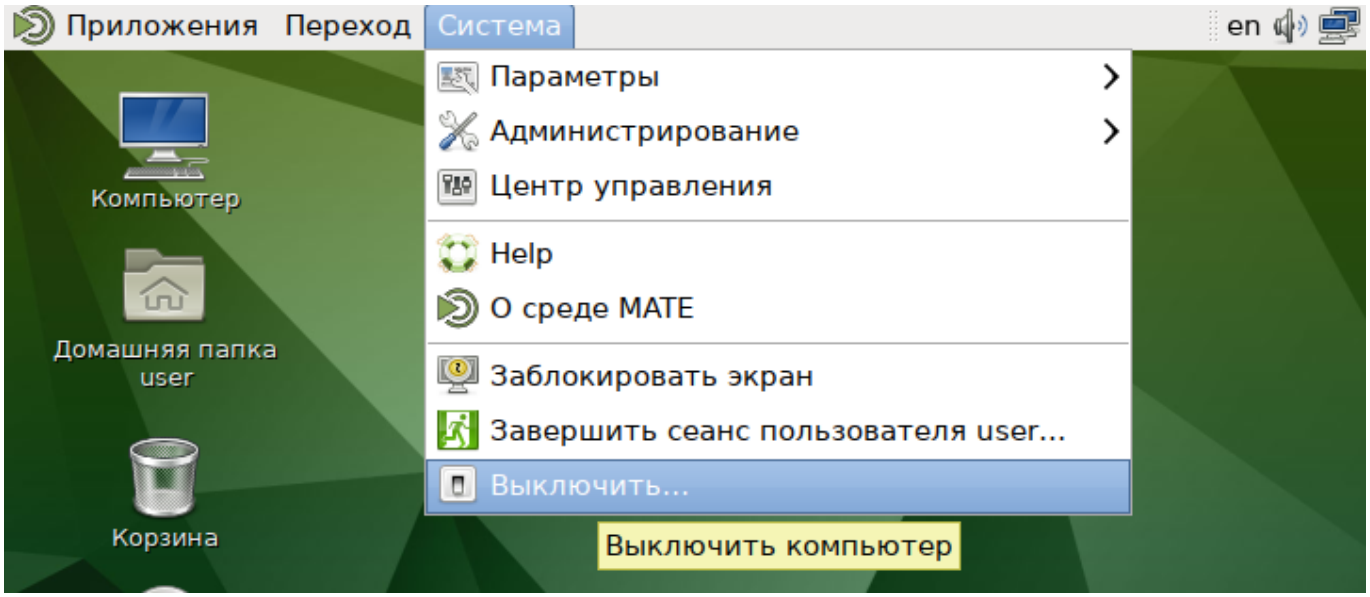

Рис. 9 – Завершения работы ОС

Далее откроется окно, в котором предоставляется выбор дальнейших действий (рис. 10).

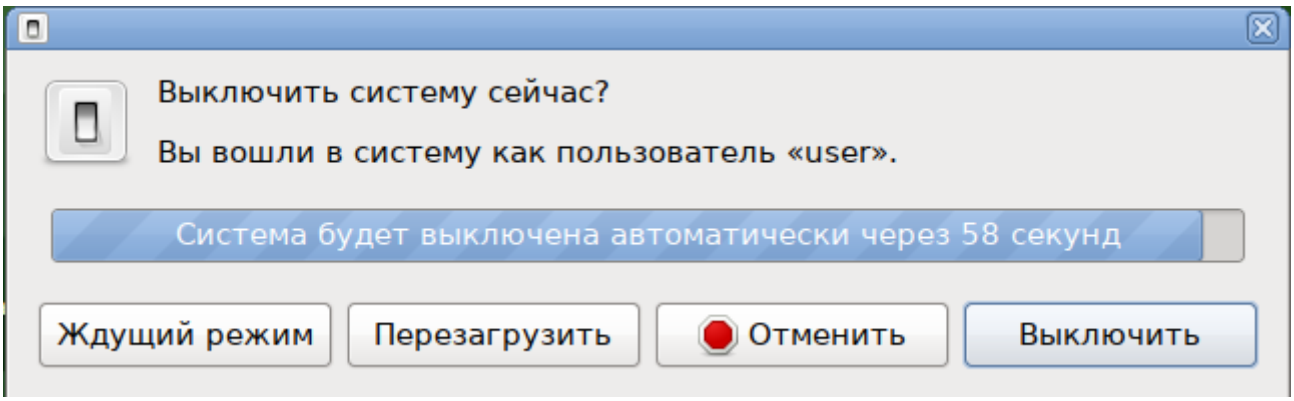

Рис. 10 – Выбор действий

После выбора действий и нажатия кнопки ОС перейдет в спящий режим, выполнит перезагрузку или выключение ПЭВМ соответственно.

Если не предпринимать никаких действий, Система будет автоматически выключена через 60 секунд.

4. ОБЩИЕ ПРАВИЛА ЭКСПЛУАТАЦИИ

4.1. Включение компьютера

Для включения компьютера необходимо:

- включить стабилизатор напряжения, если компьютер подключен через стабилизатор напряжения;
- включить принтер, если он нужен;
- включить монитор компьютера, если он не подключен к системному блоку кабелем питания;
- включить компьютер (переключателем на корпусе компьютера либо клавишей с клавиатуры).

После этого на экране компьютера появятся сообщения о ходе работы программ проверки и начальной загрузки компьютера.

4.2. Выключение компьютера

Для выключения компьютера надо:

- закончить работающие программы;
- выбрать функцию завершения работы и выключения компьютера, после чего ОС самостоятельно выключит компьютер, имеющий системный блок формата ATX;
- выключить компьютер (переключателем на корпусе AT системного блока);
- выключить принтер;
- выключить монитор компьютера (если питание монитора не от системного блока);
- выключить стабилизатор, если компьютер подключен через стабилизатор напряжения.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АРМ – автоматизированное рабочее место;

ИРС – интерактивная рабочая среда;

КСЗ – комплекс средств защиты;

ОС – операционная система;

ПРД – правила разграничения доступа;

ПЭВМ – персональная электронная вычислительная машина.

Лист регистрации изменений									
Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм	*измененных*	*замененных*	*новых*	*анулированных*					