

Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)**

Ю.С. Белов, Е.А. Черепков, С.С. Гришунов

ОСНОВЫ СИСТЕМНОГО АДМИНИСТРИРОВАНИЯ LINUX
Методические указания к выполнению лабораторной работы
по курсу «Операционные системы»

Калуга – 2018


УДК 004.62
ББК 32.972.1
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 3 от «24» октября 2018 г.

Зав. кафедрой ИУ4-КФ

 к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 3 от «29» октября 2018 г.


Председатель методической
комиссии факультета ИУ-КФ

 к.т.н., доцент М.Ю. Адкин

- Методической комиссией

КФ МГТУ им.Н.Э. Баумана протокол № 2 от «6» ноября 2018 г.

Председатель методической комиссии
КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:

к.т.н., доцент кафедры ИУ3-КФ

 А.В. Финюшин

Авторы

к.ф.-м.н., доцент кафедры ИУ4-КФ
ассистент кафедры ИУ4-КФ
ассистент кафедры ИУ4-КФ

 Ю.С. Белов
 Е.А. Черепков
 С.С. Гришунов

Аннотация

Методические указания к выполнению лабораторной работы по курсу «Операционные системы» содержат общие сведения о компиляции в ОС Linux, синтаксис команд для работы с компилятором gcc, а также синтаксис команд для управления учетными записями и правами доступа пользователей к файлам и папкам.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2018 г.
© Ю.С. Белов, Е.А. Черепков, С.С. Гришунов, 2018 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ	6
КОМПИЛЯТОР GCC	12
УПРАВЛЕНИЕ УЧЕТНЫМИ ЗАПИСЯМИ ПОЛЬЗОВАТЕЛЕЙ.....	19
ПРАВА ДОСТУПА К ФАЙЛАМ И КАТАЛОГАМ.....	30
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ	41
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	46
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ	47
ОСНОВНАЯ ЛИТЕРАТУРА.....	48
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	48

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Операционные системы» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат краткое описание команд для работы с компилятором языков С и С++, и с учетными записями, группами пользователей, правами доступа к файлам и каталогам в ОС Linux.

Методические указания составлены для ознакомления студентов с работой с компиляторами для языков программирования С и С++ и пользователями, группами пользователей и правами доступа в ОС Linux. Для выполнения лабораторной работы студенту необходимы минимальные знания об операционной системе Linux.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является приобретение практических навыков по работе с командами для работы с компиляторами для языков программирования C и C++, и для управления учетными записями пользователей, групп пользователей, правами доступа к файлам и каталогам в ОС Linux.

Основными задачами выполнения лабораторной работы являются:

1. Получить навыки работы с конвейером в ОС Linux.
2. Получить навыки работы с компиляторами для языков программирования C и C++ в ОС Linux.
3. Получить навыки работы с командами для управления учетными записями пользователей в ОС Linux.
4. Получить навыки работы с командами для работы с группами пользователей в ОС Linux.
5. Получить навыки работы с командами для управления доступом к файлам и каталогам в ОС Linux.

Результатами работы являются:

1. Демонстрация выполнения команд для работы с компиляторами для языков программирования C и C++, управления учетными записями пользователей, групп пользователей, правами доступа к файлам и каталогам.
2. Написанная программа на языке C или C++.
3. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Компилятор — программа или техническое средство, выполняющее компиляцию.

Компиляция — сборка программы, включающая трансляцию всех модулей программы, написанных на одном или нескольких исходных языках программирования высокого уровня и/или языке ассемблера, в эквивалентные программные модули на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера) или непосредственно на машинном языке или ином двоичнокодовом низкоуровневом командном языке и последующую сборку исполняемой машинной программы. Если компилятор генерирует исполняемую машинную программу на машинном языке, то такая программа непосредственно исполняется физической программируемой машиной (например компьютером). В других случаях исполняемая машинная программа выполняется соответствующей виртуальной машиной. Входной информацией для компилятора (исходный код) является описание алгоритма или программы на предметно-ориентированном языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код).

Компилировать — проводить трансляцию машинной программы с предметно-ориентированного языка на машинно-ориентированный язык.

Структура компилятора

Процесс компиляции состоит из следующих этапов:

1. трансляция программы — трансляция всех или только изменённых модулей исходной программы.
2. компоновка машинно-ориентированной программы.

В первом случае компилятор представляет собой пакет программ, включающий в себя трансляторы с разных языков программирования и

компоновщики. Такой компилятор может компилировать программу, разные части исходно текста которой написаны на разных языках программирования. Нередко такие компиляторы управляются встроенным интерпретатором того или иного командного языка. Яркий пример таких компиляторов — имеющийся во всех UNIX-системах (в частности в Linux) компилятор `make`.

Во втором случае компилятор де-факто выполняет только трансляцию и далее вызывает компоновщик как внешнюю подпрограмму, который и компоует машинно-ориентированную программу. Этот факт нередко служит поводом считать компилятор разновидностью транслятора, что естественно неверно, — все современные компиляторы такого типа поддерживают организацию импорта программой процедуры (функции) из уже оттранслированного программного модуля, написанного на другом языке программирования. Так в программу на C/C++ можно импортировать функцию написанную например Pascal или Fortran. Аналогично и напротив написанная на C/C++ функция может быть импортирована в Pascal- или Fortran-программу соответственно. Это как правило было бы невозможно без поддержки многими современными компиляторами организации обработки входных данных в процедуру (функций) в соответствии с соглашениями других языков программирования. Например современные компиляторы с языка Pascal помимо соглашения самого Pascal поддерживает организацию обработки процедура/функцией входных в соответствии с соглашениями языка C/C++. (Чтобы на уровне машинного кода написанная на Pascal процедура/функция работала с входными параметрами в соответствии с соглашениями языка C/C++, — оператор объявления такой Pascal-процедуры/Pascal-функции должен содержать ключевое слово `cdecl`.) Примерами таких компиляторов являются компиляторы со всех без исключения языков программирования, используемые непосредственно.

Трансляция программы как неотъемлемая составляющая компиляции включает в себя:

- Лексический анализ. На этом этапе последовательность символов исходного файла преобразуется в последовательность лексем.
- Синтаксический (грамматический) анализ. Последовательность лексем преобразуется в дерево разбора.
- Семантический анализ. Дерево разбора обрабатывается с целью установления его семантики (смысла) — например, привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом», и может быть дополненным деревом разбора, новым деревом, абстрактным набором команд или чем-то ещё, удобным для дальнейшей обработки.
- Оптимизация. Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах — например, над промежуточным кодом или над конечным машинным кодом.

Генерация кода. Из промежуточного представления порождается код на целевом машинно-ориентированном языке.

Linux — система многопользовательская, а потому пользователь — ключевое понятие для организации всей системы доступа в Linux. Когда пользователь регистрируется в системе (проходит процедуру авторизации, например, вводя системное имя и пароль), он идентифицируется с учётной записью, в которой система хранит информацию о каждом пользователе: его системное имя и некоторые другие сведения, необходимые для работы с ним. Именно с учётными записями, а не с самими пользователями, и работает система. Ниже приведён список этих сведений.

Системное имя (user name)

Это то имя, которое вводит пользователь в ответ на приглашение login:. Оно может содержать только латинские буквы и знак “_”. Это имя используется также в качестве имени учётной записи.

Идентификатор пользователя (UID)

Linux связывает системное имя с идентификатором пользователя в системе — UID (User ID). UID — это положительное целое число, по которому система и отслеживает пользователей¹. Обычно это число выбирается автоматически при регистрации учётной записи, однако оно не может быть совершенно произвольным. В Linux есть некоторые соглашения относительно того, каким типам пользователей могут быть выданы идентификаторы из того или иного диапазона. В частности, UID от “0” до “100” зарезервированы для псевдопользователей².

Идентификатор группы (GID)

Кроме идентификационного номера пользователя с учётной записью связан идентификатор группы. Группы пользователей применяются для организации доступа нескольких пользователей к некоторым ресурсам. У группы, так же, как и у пользователя, есть имя и идентификационный номер — GID (Group ID). В Linux каждый пользователь должен принадлежать как минимум к одной группе — группе по умолчанию. При создании учётной записи пользователя обычно создаётся и группа, имя которой совпадает с системным именем³, именно эта группа будет использоваться как группа по умолчанию для этого пользователя. Пользователь может входить более чем в одну группу, но в учётной записи указывается только номер группы по умолчанию. Группы позволяют регулировать доступ нескольких пользователей к различным ресурсам.

Полное имя (full name)

Помимо системного имени в учётной записи содержится и полное имя (имя и фамилия) использующего данную учётную запись человека.

Конечно, пользователь может указать что угодно в качестве своего имени и фамилии. Полное имя необходимо не столько системе, сколько людям — чтобы иметь возможность определить, кому принадлежит учётная запись.

Домашний каталог (home directory)

Файлы всех пользователей в Linux хранятся отдельно, у каждого пользователя есть собственный домашний каталог, в котором он может хранить свои данные. Доступ других пользователей к домашнему каталогу пользователя может быть ограничен. Информация о домашнем каталоге обязательно должна присутствовать в учётной записи, потому что именно с него начинает работу пользователь, зарегистрировавшийся в системе.

Начальная оболочка (login shell)

Важнейший способ взаимодействовать с системой Linux — командная строка, которая позволяет пользователю вести «диалог» с системой: передавать ей команды и получать её ответы. Для этой цели служит специальная программа — командная оболочка (или интерпретатор командной строки), по-английски — shell. Начальная оболочка (login shell) запускается при входе пользователя в систему в текстовом режиме (например, на виртуальной консоли). Поскольку в Linux доступно несколько разных командных оболочек, в учётной записи указано, какую из командных оболочек нужно запустить для данного пользователя. Если специально не указывать начальную оболочку при создании учётной записи, она будет назначена по умолчанию, вероятнее всего это будет bash.

Управление доступом к каталогам и файлам

Права на доступ к файлам и каталогам могут предоставляться, во-первых, создавшим их пользователям, во-вторых, группам, в которые входят эти пользователи, а в-третьих — другим пользователям системы. Таким образом, каждый пользователь может сделать

собственные файлы и папки недоступными для остальных. Права доступа к файлам папкам делятся на три категории: право на чтение — просмотр файлов; право на запись — модификация файлов; право на выполнение — запуск программ.

Для каталога права перечисленных категорий следует трактовать следующим образом-

- право на чтение — просмотр содержимого каталога;
- право на запись — создание и удаление файлов данного каталога;
- право на выполнение — возможность входа в каталог и поиска файлов в нем.

КОМПИЛЯТОР GCC

Конвейер

Конвейер позволяет объединить команды в цепочку и использовать вывод одной команды в качестве ввода другой. Конвейер устроен очень просто:

```
# команда1 | команда2
```

Выходные данные команды 1 по конвейеру передаются на ввод команды2. В командной строке может быть несколько конвейерных передач:

```
$ cp *.c test | tar cf t.tar test | gzip t.tar | ls
```

Автоматическое дополнение

Хотя эта возможность `bash` используется не так часто, оболочка может автоматизировать ввод некоторых элементов командной строки. Например, вы много работали с файлом `test`. Допустим, нам потребовалось снова просмотреть этот файл, поэтому мы запустили команду `cat`. Но вместо того, чтобы вводить полное имя файла, остановитесь на следующей командной строке (не нажимая клавиши `Enter`):

```
$ cat te
```

Теперь нажмите клавишу `Tab`. Остальные буквы имени `test`, словно по волшебству возникают в командной строке. Дело в том, что у `bash` хватило информации для автоматического завершения командной строки. Оболочка просматривает содержимое каталога и находит имя файла, логически совпадающее с уже введенной информацией. Данная возможность используется и для завершения имен команд.

Псевдонимы

Оболочка также обеспечивает одну из самых удобных возможностей Unix — создание псевдонимов.

Псевдоним (alias) представляет собой сокращение для ввода более длинной команды. Если каталог не включен в значение переменной PATH, то для запуска находящейся в нем команды вам придется ввести абсолютный путь. Поскольку многие команды спрятаны где-то глубоко в иерархии, команда получится очень длинной.

Команда alias позволяет заменить длинную командную строку коротким псевдонимом. Кроме того, с помощью псевдонимов можно превратить потенциально опасную команду в безобидную. Хотя в bash существует команда dir (как и в MS-DOS), вы всегда можете создать в Linux псевдонимы для своих любимых команд DOS. Допустим, вы долго работали с командой DOS copy и никак не привыкнете к ее аналогу в Linux, команде cp. Создайте следующий псевдоним:

```
$ alias copy='cp'
```

После выполнения этой команды copy и cp будут делать одно и то же. При этом команда cp никуда не исчезает — вы просто сообщаете bash о том, что она должна вызываться при вводе команды copy.

Следите за правильностью синтаксиса, показанного в этом примере. Между командами не должно быть лишних пробелов, а командная строка заключается в апострофы (а не в кавычки).

Для уничтожения синонимов используется команда unalias:

```
$ unalias copy
```

Команда alias, выполненная из командной строки, действует только на протяжении текущего сеанса. Чтобы команда действовала постоянно, включите ее в файл .profile, находящийся в домашнем каталоге. Этот файл выполняется при каждом запуске Linux, поэтому именно в нем происходит основная настройка системы.

Компилятор C: gcc

Между операционной системой UNIX и языком программирования C существует тесная взаимосвязь. Общеизвестно, что язык C с самого начала предназначался для использования в качестве инструмента разработки операционной системы UNIX. И действительно, при создании программного кода операционной системы UNIX в основном применялся язык C. То же самое можно сказать относительно Linux. Большинство вариантов Linux включают GNU-версию [компилятора C](#), gcc.

Для вызова компилятора GNU C в системе Linux необходимо воспользоваться командой gcc. Команда gcc, в свою очередь, вызывает четыре других компонента.

Первый из них именуется препроцессором. В программах на языке C содержатся специальные команды препроцессора, изменяющие программный код перед его отсылкой компилятору.

В качестве второго компонента выступает сам компилятор. Он обрабатывает программный код и создает версию кода сборки для программы.

Третий компонент — ассемблер. На основании версии кода сборки ассемблер генерирует версию объектного кода.

В качестве четвертого компонента выступает редактор связей, который использует объектный код для создания исполняемой программы. Полученный в результате функционирования компилятора файл по умолчанию именуется a.out. Однако этому файлу можно присвоить и другое имя. В этом случае потребуется воспользоваться опцией -o, в качестве параметра которой следует указать название файла. В результате вместо стандартного имени a.out будет использовано новое имя. Перечень параметров команды gcc приведен в табл. 1. В следующем примере программа gcc используется для компиляции программы greet.c. Программист назвал исполняемый файл «greet». Запуск исполняемого файла осуществляется с помощью командной строки Linux точно так же, как и запуск любой другой команды.

```
$ gcc greet.c -o greet
$ ./greet
Hello, how are you
```

При работе с программами, состоящими из нескольких отдельных файлов, необходимо учитывать разницу между компилятором С и редактором связей. Компилятор С предназначен для создания объектного кода, а назначением редактора связей является создание исполняемой программы, использующей файлы с объектными кодами. Компилятор С компилирует отдельно каждый файл, содержащий исходный код, и для каждого из них создает отдельный файл с объектным кодом. Созданные файлы, включающие объектный код, имеют расширение .o, а не .c. С помощью одной команды gcc можно скомпилировать и скомпоновать программу, состоящую из нескольких файлов. Для этого нужно просто перечислить названия всех файлов, содержащих исходный код, сославшись на них как на параметры командной строки. В следующем примере программист компилирует программу bookrecs путем вызова команды gcc с указанием файлов, содержащих исходный код. С помощью опции -o указывается, что исполняемая программа именуется bookrecs.

```
$ gcc main.c io.c -o bookrecs
```

Таблица 1. Утилита gcc: компилятор языка С

Опция	Назначение
-S	Выводит исключительно код на языке ассемблера. Версии кода на языке ассемблера для скомпилированных файлов имеют расширение .s. В примере генерируется файл greet.s
-P	Выводит результаты работы препроцессора

Продолжение таблицы 1

-c	Создает исключительно файл объектного кода. Версии скомпилированных файлов объектного кода имеют расширение .o
-g	Осуществляет подготовку скомпилированной программы для использования совместно с символическим отладчиком
-o имя_файла	Задает имя исполняемого файла, <i>имя_файла</i> . По умолчанию используется a.out
-O	Выполняет оптимизированную компиляцию
-l имя_файла	Обеспечивает применение для компоновки программы системной библиотеки с указанным именем файла. Имя файла содержит префикс lib и расширение .a. В командной строке gcc эта опция не указывается. Опции -l должны всегда располагаться после исходного кода и имен файлов объектного кода в командной строке
-Idir	Задает каталоги, в которых производится поиск файлов для включения, таких как файлы заголовков (.h)
-Ldir	Задает каталоги, в которых производится поиск библиотек

Двоичные форматы ELF и a.out

Существуют два формата, которые могут использоваться при создании двоичных файлов (например, файлов исполняемых программ). Первый из них именуется a.out. Этот формат является исходным форматом, используемым для систем UNIX и первых версий Linux. Название формата в данном случае происходит от стандартного имени исполняемого файла, создаваемого компилятором C системы UNIX. Но в связи с широким применением совместно используемых библиотек возникли трудности с форматом a.out. Адаптация формата a.out с целью применения совместно используемых библиотек является весьма сложной процедурой. Поэтому, начиная с четвертой версии

UNIX System 5 и операционной системы Solaris, был предложен новый формат. Он получил название ELF (Executable and Linking Format — формат компоновки и исполнения). Целью разработки этого формата было упрощение работы с совместно используемыми библиотеками.

Формат ELF был адаптирован с целью применения в качестве стандартного формата Linux. Все двоичные файлы, создаваемые компилятором gcc, имеют формат ELF (несмотря на то, что стандартное название исполняемого файла, создаваемого в результате функционирования компиляторов, осталось прежним — a.out). Более старые программы, имеющие формат a.out, также могут выполняться системами, поддерживающими формат ELF.

Языки C++ и Objective C: команда g++

Утилита [gcc](#) выполняет также функции компилятора языка C++. Она может считывать и компилировать любую программу на языке C++. Однако она не позволяет выполнять автоматическую компоновку с вызовом библиотеки классов C++. Эта библиотека должна вызываться отдельно, с использованием режима командной строки. Для этого можно также воспользоваться командой g++, в результате чего вызывается компилятор gcc вместе с библиотекой классов C++.

Для файлов исходного кода на языке C++ используются расширения, отличающие от расширений обычных файлов кода на языке C. Компилятору C++ соответствует множество различных расширений: C, cc, sxx или srrp.



```
nano 2.6.0                                File: hi.cpp

#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World!\n";
    return 0;
}

[ Read 7 lines ]
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Рис. 1. Исходный код программы

Если не принимать во внимание это различие, то компиляция программ C++ проходит точно так же, как и компиляция программ C. Вместо команды `gcc` рекомендуется использовать команду `g++`. В следующем примере показано, как должна быть выполнена компиляция программы `tutprog.cpp`, написанной на языке C++.



```
root@darkstar:~# g++ hi.cpp -o hi
root@darkstar:~# ./hi
Hello, World!
root@darkstar:~# _
```

Рис. 2. Компиляция и запуск программы

УПРАВЛЕНИЕ УЧЕТНЫМИ ЗАПИСЯМИ ПОЛЬЗОВАТЕЛЕЙ

Для добавления пользователя используется команда **adduser**. Введите в следующую команду:

```
$ adduser
```

```
Login name for new user (8 characters or less) []: kevinr
```

Команда `adduser` делает в точности то, о чем говорит ее имя: она добавляет в систему нового пользователя. В предыдущем примере был добавлен пользователь `kevinr`. После ввода имени вам будет предложено ввести дополнительную информацию о пользователе. Если вы еще не имеете опыта работы с Linux, оставьте значения по умолчанию, указанные в квадратных скобках. Всюду, где присутствует значение по умолчанию, вместо ввода с клавиатуры можно нажать клавишу Enter (в нашем примере значения будут вводиться). Результат выглядит примерно так:

```
User if for kevinr [defaults to next available]:
```

```
Initial group for kevinr [users]: users Additional groups for kevinr []:
```

```
kevinr's home directory [/home/kevinr]: /home/kevinr
```

```
kevinr's shell [bin/bash]; /bin/bash kevinr's account expiry date  
(MM/DD/YY) []:
```

```
This is it... if you want to bail out, hit Control+C. Otherwise,
```

```
press ENTER to go ahead and make the account. Making new account:
```

```
Changing the user information for kevinr Enter the new value, or press  
return for the default
```

```
Full name [J: Kevin Reichard
```

```
Room number []:
```

```
Work phone []:
```

```
Home phone []:
```

```
Other []:
```

```
Changing password for kevinr
```

Enter the new password (minimum of 5, maximum of 8 characters) Please use a combination of upper and lower case letters and numbers.

New password: <new password1>

Re-enter new password: <new password""

Password changed.

Done...

Одна из основных задач операционной системы Linux заключается в поддержке одновременной работы большого количества пользователей. Какие пользователи работают в данный момент в системе, позволяет определить команда **who**. Дополнительную информацию о каждом подключенном пользователе (с какого терминала подключен, как долго неактивен) можно получить, добавив опцию **-u**. Команда **who** выводит регистрационное имя пользователя, системное имя терминала, дату и время входа в систему, время, в течение которого пользователь неактивен (если этот факт имеет место), а также идентификатор процесса регистрационного shell.

```
$ who -u
```

```
root console Oct 12 10:34 1219
```

```
valerie ttyl Oct 12 22:18 10 1492
```

Файл **/etc/passwd**

Когда появляется новый пользователь системы, в файле **/etc/passwd**, который называют файлом паролей, делается соответствующая запись. Каждая запись занимает одну строку, состоящую из нескольких полей, разделенных двоеточиями.

Вот перечень этих полей:

- [ИМЯ ПОЛЬЗОВАТЕЛЯ](#) — регистрационное имя пользователя;
- пароль — зашифрованный пароль учетной записи пользователя;
- [ИДЕНТИФИКАТОР ПОЛЬЗОВАТЕЛЯ](#) — уникальный номер, назначенный системой;

- [идентификатор группы](#) — номер, служащий для обозначения группы, к которой относится пользователь;
- комментарий — информация о пользователе, например его имя и фамилия;
- [начальный каталог](#) — начальный каталог пользователя;
- [регистрационный shell](#) — shell, запускаемый при регистрации пользователя в системе обычно /bin/bash.

Ниже приведен пример записи из файла /etc/passwd. В поле пароля стоит звездочка, которая демонстрирует, что для пользователя mark пароль еще не задан. При наличии таких записей необходимо с помощью команды passwd ввести пароли. Обратите внимание на то, что в данной системе идентификаторы пользователей начинаются с 500 и увеличиваются на единицу.

```
root:x:0:0:test,w,l,w:/root:/bin/bash
richlp:YOTPd3Pyy9hAc:500:500:CalderaDesktopUser:/home/richlp:/bin
/bash
mark:*:501:501:CalderaDesktopUser:/home/mark:/bin/bash
```

Файл /etc/passwd — это обыкновенный текстовый файл, слабое звено в системе защиты. Любой пользователь, получивший к нему доступ, сможет дешифровать или видоизменить пароли. Большинство дистрибутивов Linux поддерживают механизм теневого паролей, обеспечивающий достаточно мощную защиту системы: пароли шифруются и сохраняются в отдельном файле /etc/shadow, право на доступ к которому имеют только привилегированные пользователи; для групп, которым требуются отдельные пароли, создается аналогичный файл /etc/gshadow.

Добавление и удаление учетных записей

В большинстве дистрибутивов ОС Linux для управления учетными записями пользователей применяются команды **useradd**, **usermod** и

userdel. Все они получают информацию из командной строки в виде параметров. Если параметр не задан, команды используют его стандартное значение. При использовании команды **useradd** параметры вводятся в командной строке (например, имя пользователя). В результате [создаются новая учетная запись](#), а также регистрационный каталог с данным именем и стандартными свойствами.

Пользователь добавляется командой **useradd**. Из консоли это делается, например, так:

```
# useradd [-u идентификатор [-o] [-i]] [-g группа] [-G группа[,группа]...] [-d каталог] [-s shell] [-c комментарий] [-m [-k skel_dir]] [-f inactive] [-e expire] [-p passgen] [-a событие[, . . .]] рег_имя
```

Новое регистрационное имя блокируется до тех пор, пока не будет выполнена команда **passwd**.

Таблица 2. Опции команды **useradd**

Опция	Описание
-u идентификатор	Идентификационный номер пользователя UID
-o	Эта опция позволяет сдублировать UID (сделать его не
-i	Позволяет использовать устаревший идентификатор UID.

Продолжение таблицы 2

Опция	Описание
-g группа	Целочисленный идентификатор или символьное имя существующей группы. Эта опция задает основную группу (primary group) для нового пользователя. По умолчанию используется стандартная группа, указанная в файле /etc/default/useradd.
-G группа[,группа] ...]	Один или несколько элементов в списке через запятую, каждый из которых представляет собой целочисленный идентификатор или символьное имя существующей группы. Этот список определяет принадлежность к дополнительным группам (supplementary group membership) для пользователя.
-d каталог	Начальный каталог (home directory) нового пользователя. Длина этого поля не должна превосходить 256 символов. По умолчанию используется HOMEDIR/рег_имя, где HOMEDIR – базовый каталог для начальных каталогов новых пользователей, а рег_имя - регистрационное имя нового пользователя.
-s shell	Полный путь к программе, используемой в качестве начального командного интерпретатора для пользователя сразу после регистрации.
-с комментарий	Любая текстовая строка. Обычно, это краткое описание регистрационного имени и используется сейчас для указания фамилии и имени реального пользователя.

Продолжение таблицы 2

Опция	Описание
-m	Создает начальный каталог нового пользователя, если он еще не существует.
-f inactive	Максимально допустимое количество дней между использованиями регистрационного имени, когда это имя еще не объявляется недействительным.
-e expire	Дата, начиная с которой регистрационное имя больше нельзя будет использовать; после этой даты никакой пользователь не сможет получить доступ под этим регистрационным именем.
рег_имя	Строка печатных символов, задающая регистрационное имя для нового пользователя. В ней не должно быть двоеточий (:). Она также не должна начинаться с прописной буквы.

Добавление пользователя:

```
$ useradd -u 1023 -i -g newusers -G oldusers, futureusers -d /home/newuser -c 'new user from admins group' -f 99 linuxuser
```

Команда **usermod** - изменение регистрационной информации о пользователе в системе (все параметры, кроме параметра **-l** аналогичны команде **useradd**)

```
# usermod [-u идентификатор [-U] [-o]] [-g группа] [-G группа[группа] ...] [-d каталог [-m]] [-s shell] [-c комментарий] [-l новое_рег_имя] [-f inactive] [-e expire] [-p passgen] [-a [оператор]событие[, ...]] рег_имя
```


Таблица 3. Опции команды usermod

Опция	Описание
-l новое_рег_имя	Строка печатных символов, задающая новое регистрационное имя для пользователя. Она не должна содержать двоеточий (:) и переводов строк (\n). Кроме того, она не должна начинаться с прописной буквы.
рег_имя	Строка печатных символов, задающая регистрационное имя существующего пользователя. Регистрационное имя должно существовать и не должно содержать двоеточий (:) и переводов строк (\n).

Команда **userdel** - удаляет регистрационное имя пользователя из системы

userdel [-r] [-n месяцев] рег_имя

Таблица 4. Опции команды userdel

Опция	Описание
-r	Удалить начальный каталог пользователя из системы.
-n месяцев	Задаёт, сколько месяцев идентификатор пользователя должен устаревать перед повторным использованием. Задайте -1, чтобы указать, что идентификатор пользователя никогда не должен использоваться повторно. Задайте 0, чтобы указать, что идентификатор пользователя можно немедленно повторно использовать. Если опция -n не задана, идентификатор будет устаревать стандартное количество месяцев перед повторным использованием.

Продолжение таблицы 4

Опция	Описание
рег_имя	Строка печатных символов, задающая существующее в системе регистрационное имя. Она не может содержать двоеточие (:) или символ перевода строки (\n).

Пример удаления пользователя:

```
$ userdel -r -n 0 linuxuser
```

Изменение паролей

После создания нового пользователя в системе необходимо изменить/назначить для него пароль, иначе вход в систему невозможен.

Из консоли пользовательский пароль меняется командой **passwd**:

```
$ passwd userid
```

Изменить пароль другого пользователя таким способом может только root. После ввода команды вам будет предложено ввести и подтвердить устанавливаемый пароль. Если они совпадут, то данные пользователя будут обновлены, а пароль изменен. Пользователь также может изменить свой собственный пароль, написав в командной строке консоли passwd; в этом случае прежде чем вводить новый пароль, необходимо будет ввести старый.

Учетные записи групп

Системный файл, в котором находятся элементы учетных записей групп, называется /etc/group. Для каждой группы он содержит записи в виде строк, поля которых разделены двоеточиями. Запись включает четыре поля: имя группы, пароль, идентификатор и пользователи, входящие в состав группы. Поле пароля может быть пустым.

Остановимся на описании каждого поля:

- имя группы — этот параметр должен быть уникальным;
- пароль — обычно это звездочка, которая позволяет любому пользователю войти в состав группы; для управления доступом можно добавить пароль;
- идентификатор группы — номер, которым система обозначает данную группу;
- пользователи — список пользователей, относящихся к данной группе.

Ниже приведен пример записи из файла `/etc/group`. Группа называется `engines`, пароля нет, идентификатор группы — `100`, в группу входят пользователи `chris`, `robert`, `valerie` и `aleina`.

```
engines::100:chris,robert,valerie,aleina
```

Работа с учетными записями групп

Во многих дистрибутивах Linux для управления учетными записями групп применяются команды **groupadd**, **groupmod** и **groupdel**. Команда **groupadd** служит для создания новых групп. Новая группа регистрируется в файле `/etc/group` и получает идентификатор. Изначально группа, созданная командой **groupadd**, является пустой. Пользователи вводятся в группу поочередно.

```
# groupadd [-g идентификатор [-o]] группа
```

Таблица 5. Опции команды **groupadd**

Опция	Описание
-g идентификатор	Идентификатор новой группы. Этот идентификатор группы должен быть неотрицательным десятичным целым числом. Идентификаторы групп в диапазоне 0-99 зарезервированы.

Продолжение таблицы 5

Опция	Описание
-o	Эта опция позволяет задавать дублирующийся (не уникальный) идентификатор группы.
группа	Строка печатных символов, задающая имя новой группы. Она не может содержать двоеточия (:) или переводы строк (\n).

Пример добавления группы:

```
$ groupadd -g 203 students
```

Назначение команды **groupmod** — изменение имени группы и ее идентификатора. Чтобы произвести эту операцию, нужно ввести команду `groupmod -g` с новым идентификатором и именем группы. Для замены имени группы служит опция `-n`. Введите команду `groupmod -n` с новым именем группы и текущим именем. В следующем примере показано, как группу `engines` переименовать в `trains`.

```
# groupmod [-g идентификатор [-o]] [-n имя] группа
```

Таблица 6. Опции команды `groupmod`

Опция	Описание
-g идентификатор	Идентификатор новой группы. Этот идентификатор группы должен быть неотрицательным десятичным целым числом. Идентификаторы групп в диапазоне 0-99 зарезервированы.

Продолжение таблицы 6

Опция	Описание
-n имя	Строка печатных символов, задающая новое имя для группы. Она не должна содержать двоеточия (:) или переводы строк (\n).
группа	Текущее имя изменяемой группы.

Пример изменения группы:

```
$ groupmod -g 204 -n groupitd students
```

Удалить группу позволяет команда **groupdel**. Синтаксис команды:

```
# groupdel groupitd
```

ПРАВА ДОСТУПА К ФАЙЛАМ И КАТАЛОГАМ

Для каждого файла и каталога в ОС Linux задаются [права доступа](#), которые определяют, кто и какие операции может проводить над данным файлом. Эти права позволяют ограничить доступ к файлу одним из трех способов. Во-первых, вы можете предоставить доступ только себе самому. Во-вторых, вы можете предоставить доступ определенной группе. В-третьих, доступ может быть предоставлен всем пользователям системы. Для файла или каталога может быть установлено право на чтение, запись и выполнение. При создании файла автоматически устанавливаются разрешения на чтение и запись для владельца, что позволяет ему выводить файл на экран и корректировать его. Владелец может изменять права доступа как угодно. Право только на чтение предотвращает внесение в файл изменений. Для файла можно задать также право на выполнение, что позволяет выполнять его как программу.

Существуют три категории пользователей, которые могут иметь доступ к файлу или каталогу: владелец, группа и прочие. Владелец — это пользователь, создавший файл. Таким образом, вы становитесь владельцем каждого файла, который создаете. Часто пользователи объединяются в группы. Например, системный администратор может объединить в группу пользователей, работающих над одним проектом. По своему усмотрению вы можете разрешить или не разрешить доступ к файлу членам своей группы. Наконец, вы можете открыть доступ для всех остальных пользователей системы. В этом случае доступ к вашему файлу или каталогу будут иметь все, кто работает в вашей системе. Эти остальные пользователи и составляют категорию «прочие».

Команда `ls` с опцией `-l` выдает подробную информацию о файле, включая права доступа к нему. В следующем примере первый набор символов слева — это список прав доступа, установленных для файла `mydata`.

```
$ ls -l mydata
```

```
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
```

Команда `ls` с опцией `-ld`:

```
# ls -ld каталог
```

Выдает имя каталога с правами доступа

Отсутствие права доступа обозначается дефисом (-). Право на чтение обозначается буквой `r`, право на запись — буквой `w`, право на выполнение — буквой `x`. Обратите внимание на то, что в первой группе символов всего десять позиций. Первый символ обозначает тип файла. Каталог можно считать типом файла. Если первый символ — дефис, то речь идет о файле. Если это буква `d`, выдается информация о каталоге.

Следующие девять символов указывают на права доступа для различных категорий пользователей. Первый трехсимвольный набор — это права доступа к файлу для категории «владелец». Вторым трехсимвольным набором — это права доступа к файлу для категории «группа». Третьим трехсимвольным набором — это права доступа к файлу для категории «прочие». Из листинга следует, что для файла `mydata` установлены право на чтение и запись по категории «владелец», право на чтение по категории «группа» и право на чтение по категории «прочие». Это значит, что читать файл могут все члены группы и все пользователи системы, а изменять — только владелец.

Для создания различных конфигураций прав доступа применяется команда **`chmod`**. В качестве параметров в этой команде используются два списка: список изменений прав доступа и список имен файлов. Список изменений прав доступа можно задавать двумя способами. При использовании первого, который называется символическим методом, применяются символы `r`, `w`, `x`. В случае второго, который называют абсолютным методом, используется так называемая двоичная маска. Опции команды `chmod` перечислены в табл. 6.

Таблица 7. Опции команды `chmod`

Опция	Описание
+	Устанавливает право доступа
-	Отменяет право доступа
=	Присваивает весь набор прав доступа
r	Устанавливает право на чтение для файла или каталога. Файл может отображаться на экране и печататься. Для каталога может выдаваться список находящихся в нем файлов
w	Устанавливает право на запись для файла или каталога. Для файла это означает возможность его редактировать или уничтожать. Для каталога это право означает возможность создания и удаления файлов в нем
x	Устанавливает право на выполнение для файла или каталога. Если файл — сценарий командного интерпретатора, он может выполняться как программа. В каталог с таким режимом доступа можно переходить
u	Устанавливает права доступа для пользователя, который создал файл или каталог и владеет им
g	Устанавливает права доступа к файлу или каталогу для группы
o	Устанавливает права доступа к файлу или каталогу для прочих пользователей системы
a	Устанавливает права доступа для владельца, группы и прочих пользователей
s	Устанавливает бит смены идентификатора пользователя или группы; программа принадлежит владельцу и группе
t	Устанавливает sticky-бит (бит закрепления); программа с установленным sticky-битом после ее завершения остается в памяти

Утилита **chgrp** позволяет изменить группу для файлов или директорий. Она устанавливает идентификатор группы (group ID) для файлов, переданных в качестве аргумента файл, в идентификатор группы, указанной операндом группа.

chgrp [-fhv] [-R [-H | -L | -P]] группа файл

Таблица 8. Опции команды chgrp

Опция	Описание
-H	Если указана опция -R, следовать символическим ссылкам, указанным в командной строке, но не следовать символическим ссылкам, встречающимся в процессе обхода дерева каталогов.
-L	Если указана опция -R, следовать по всем символическим ссылкам.
-P	Если указана опция -R, не следовать никаким символическим ссылкам. Это режим работы по умолчанию.
-R	Изменять идентификатор группы для целых иерархий файлов, начинающихся с файлов, указанных в аргументах файл, а не только для самих файлов.
-f	Опция принудительного выполнения. Она будет игнорировать все ошибки, за исключением ошибок синтаксиса, и не будет уведомлять о странных правах доступа (кроме случаев, когда пользователю не хватает нужных полномочий).
-h	Если файл является символической ссылкой, изменить идентификатор группы самой ссылки, а не файла, на который она указывает.
-v	Выводить подробные сообщения, показывая файлы, у которых меняется группа. Если флаг -v указан более одного раза, chgrp напечатает имя файла, а затем старый и новый числовые идентификаторы групп.

Опции -H, -L и -P игнорируются, если не была указана опция -R. В дополнение, эти опции перекрывают друг друга и используется только та, которая была указана последней.

Операнд группа может быть либо именем группы из базы данных групп, либо числовым идентификатором группы. Если имя группы состоит из одних цифр и совпадает с числовым идентификатором другой группы, операнд будет интерпретирован как имя группы.

Пользователь, вызывающий `chgrp`, должен принадлежать к указываемой группе и должен быть владельцем файла, либо быть суперпользователем.

Команда **chown** - изменение владельца и группы на файлы и директории.

```
# chown [-fhv] [-R [-H | -L | -P]] владелец [:группа ] файл ...
```

```
# chown [-fhv] [-R [-H | -L | -P]] :группа файл ...
```

Программа изменяет владельца и/или группу файла или директории, то есть UserID и GroupID для указанного файла. Если не указана опция -h, символические ссылки переданные в качестве аргументов, не изменяются.

Таблица 9. Опции команды `chown`

Опция	Описание
-H	Если установлена опция -R, следовать по ссылкам из командной строки. (Ссылки найденные при обходе дерева каталогов не прослеживаются)
-L	Если опция -R установлена, следовать по всем символическим ссылкам.

Продолжение таблицы 9

Опция	Описание
-P	Если опция -R установлена, не следовать по ссылкам. Поведение по-умолчанию.
-R	Рекурсия. Изменить UserID и/или GroupID для всего дерева директорий и файлов начиная с указанной. Остерегайтесь совпадения с жесткой ссылкой на родительский каталог "..", при использовании шаблона ".*".
-f	Не сообщать о неудачной попытке изменить владельца и группу не менять код завершения операции, для сигнализирования о неудаче.
-h	Если файл является символической ссылкой, изменить UserID и/или GroupID только на саму ссылку.
-v	Режим вывода сообщений о ходе выполнения программы. Если опция указана более одного раза, chown выведет имя файла вслед за старыми и новыми UserID/GroupID.

Опции -H, -L и -P, будут проигнорированы, если опция -R не установлена. Кроме того, все эти опции, переопределяет друг друга, и поведение команды chown, будет определяться опцией, которая указана последней.

Опции владелец и группа не являются обязательными, но должна быть указана хотя-бы одна из них. Если указывается только группа, перед ней ставится знак : (двоеточие).

Параметр владелец файла, может быть представлен как в виде числового выражения UserID, так и в виде символического имени. Если имя пользователя совпадает с его UserID, операнд используется как "имя пользователя". То же самое относится к параметру группа.

Принадлежность файла может быть изменена только суперпользователем, по соображениям безопасности.

Программа `shown` возвращает 0 при удачном завершении и >0 в случае возникновения ошибки.

Установление прав доступа: символы прав доступа

При символическом методе права доступа на чтение, запись и выполнение обозначаются соответственно символами `r`, `w` и `x`. Любое из этих разрешений можно добавлять и удалять. Символом добавления права доступа является знак плюс, `+`. Символом отмены является знак минус, `-`. В следующем примере команда `chmod` добавляет право на выполнение и отменяет право на запись для файла `mydata`. Право на чтение не изменяется.

```
$ chmod +x-w mydata
```

Используются и другие символы, которые обозначают категории пользователей. Категории «владелец», «группа» и «прочие» обозначаются соответственно символами `u`, `g` и `o`. Категория «владелец» обозначена буквой `u` (от слова `user`), так как владельца можно считать пользователем. Символ категории ставится перед символами, устанавливающими права на чтение, запись и выполнение. Если символа категории нет, то подразумеваются все категории, и указанные права устанавливаются для пользователя, группы и прочих. В следующем примере первая команда `chmod` устанавливает право на чтение и запись для группы. Вторая команда `chmod` устанавливает право на чтение для прочих пользователей. Обратите внимание на то, что пробелов между спецификациями прав доступа и категорией нет. Список изменений прав доступа представляет собой одну длинную фразу без пробелов.

```
$ chmod g+rw mydata  
$ chmod o+r mydata
```

Пользователь может не только устанавливать, но и отменять права доступа. В следующем примере для прочих пользователей устанавливается право на чтение, а права на запись и выполнение отменяются.

```
$ chmod o+r-wx mydata
```

Есть еще один символ разрешения, а (от слова all), который обозначает все категории. Он действует по умолчанию. В следующем примере обе команды эквивалентны. Разрешение на чтение задается явно символом а, который обозначает все типы пользователей: прочие, группа и владелец.

```
$ chmod a+r mydata $ chmod +r mydata
```

Одна из наиболее распространенных операций с правами доступа — установление права на выполнение файла. Право на выполнение показывает, что файл содержит выполняемые команды и может непосредственно выполняться системой. В следующем примере для файла `lsc` устанавливается право на выполнение; затем этот файл выполняется.

```
$ chmod u+x lsc
$ lsc
main.c lib.c
```

Абсолютные права доступа: двоичные маски

Вместо символов разрешений многие пользователи предпочитают применять *абсолютный метод*. Абсолютный метод позволяет изменять сразу все права доступа. Здесь используется двоичная маска, которая обозначает все разрешения в каждой категории. Эти три категории, по три разрешения в каждой, представлены в восьмеричном формате. В восьмеричной системе счисления все числа имеют

основание 8. При преобразовании в двоичный формат каждый восьмеричный разряд превращается в три двоичных. Двоичное число представляет собой набор единиц и нулей. Три восьмеричных разряда числа преобразуются в три набора по три двоичных разряда в каждом. Итого получается девять двоичных цифр, что в точности соответствует количеству разрешений доступа к файлу.

Восьмеричные цифры можно использовать как маску для установления различных прав доступа к файлу. Каждая восьмеричная цифра относится к одной из категорий пользователей. При этом категории нумеруются слева направо, начиная с категории «владелец». Первая восьмеричная цифра относится к владельцу, вторая — к группе, а третья — к прочим пользователям.

Выбранные вами восьмеричные цифры будут определять разрешения на чтение, запись и выполнение по каждой категории. На данном этапе вам нужно знать, как восьмеричные цифры преобразуются в свои двоичные эквиваленты. В приведенной ниже таблице показано соответствие между восьмеричными цифрами от 0 до 7 и их трехразрядными двоичными эквивалентами. Можно считать, что восьмеричная цифра сначала преобразуется в двоичную форму, а затем группы двоичных разрядов используются для установления прав на чтение, запись и выполнение. Каждый двоичный Разряд определяет одну из составляющих прав доступа (напоминаем — в направлении слева направо). Если двоичный разряд — 0, разрешение отсутствует. Если двоичный разряд — 1, разрешение установлено. Первый слева двоичный разряд включает и выключает право на чтение, второй — право на запись, третий — право на выполнение. Например, восьмеричная цифра 6 преобразуется в двоичные цифры 110. Это соответствует установлению права на чтение и запись и отменяет право на выполнение.

Маски для разрешений:

- 0-000
- 1-001
- 2-010

- 3-011
- 4-100
- 5-101
- 6-110
- 7-111

При использовании двоичной маски нужно указывать сразу три цифры, которые обеспечивают установление прав доступа одновременно' для всех трех категорий. Это делает двоичную маску менее гибкой, чем символы разрешений. Чтобы установить право на выполнение и отменить право на запись для владельца файла mydata с сохранением права на чтение, необходимо использовать восьмеричную цифру 5 (101 в двоичном формате). В то же время нужно указать соответствующие цифры для группы и прочих пользователей. Если право на чтение для этих категорий сохраняется, то для каждой из них следует указать восьмеричную цифру 4 (100). Это дает вам три восьмеричные цифры, 544, которые преобразуются в двоичные цифры 101 100 и 100.

```
$ chmod 544 mydata
```

Права доступа к каталогам

Права доступа можно устанавливать и для каталогов. Если для каталога установлено право на чтение, пользователи могут получать список файлов, находящихся в данном каталоге. Право на выполнение позволяет переходить в этот каталог. Право на запись дает возможность пользователю создавать и удалять файлы в этом каталоге. В следующем примере для каталога thankyou устанавливается право на чтение и выполнение по категории «группа», а право на запись отменяется. Члены группы могут входить в каталог thankyou и получать список его файлов, но создавать в нем новые файлы они не могут.

```
$ chmod g+rx-w letters/thankyou
```

Как и в случае с файлами, при установлении прав доступа к каталогу можно пользоваться восьмеричными цифрами. Для указания тех же прав, что и в предыдущем примере, нужно использовать восьмеричные цифры 750 (двоичный эквивалент — 111 101 000).

```
$ chmod 750 letters/thankyou
```

Команда `ls` с опцией `-l` выдает список всех файлов, содержащихся в каталоге. Для того чтобы получить информацию только о самом каталоге, нужно использовать эту команду с опцией `d`. В следующем примере команда `ls -ld` выдает информацию о каталоге `thankyou`. Обратите внимание на то, что первым символом в поле прав доступа является `d`, а это указывает, что он представляет собой каталог.

```
$ ls -ld thankyou
```

```
drwxr-x    2 chris 512 Feb 10 04:30 thankyou
```


ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Научиться использовать команды для управления учетными записями пользователей, групп пользователей, правами доступа к файлам и каталогам, работу конвейера и псевдонимов в ОС Linux.

Продемонстрировать работу команд:

1. Добавления и удаления учетных записей
2. Изменения информации о пользователе
3. Смены пароля
4. Создания групп
5. Работы с группами
6. Изменения прав доступа к файлам и каталогам

Для демонстрации работы команд gcc и g++ написать программу на C/C++ с использованием функций ввода, вывода, использование условных и циклических операторов согласно заданию, указанному в варианте.

Вариант 1

Дано натуральное число N. Вычислить

$$S = \sum_{i=1}^N \prod_{j=1}^i \frac{j!}{i!}$$

Вариант 2

Дано натуральное число N. Вычислить

$$S = \sum_{i=1}^N \frac{i!}{(N+i)!}$$

Вариант 3

Дано натуральное число N. Вычислить

$$S = \sum_{i=1}^N \sum_{k=0}^i \frac{i-k}{i+k}$$

Вариант 4

Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^N \prod_{m=1}^{2k} \sin \frac{m\pi}{2k+1}$$

Вариант 5

Даны натуральные числа N и M. Вычислить

$$F = \frac{M! + N!}{(M + N)!}$$

Вариант 6

Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^N (-1)^{k+1} * \prod_{m=1}^{2k} \cos\left(\frac{m+1}{2k}\right)$$

Вариант 7

Даны натуральные числа N, M. Вычислить

$$S = \sum_{i=1}^N \sum_{k=1}^M \sin\left(\frac{\pi}{4}i + \frac{\pi}{8}k^2\right)$$

Вариант 8

Дано натуральное число $N > 2$. Вычислить

$$S = \sum_{k=2}^N k^2 \ln(1 + k^2)$$

Вариант 9

Дано натуральное число $N > 2$. Вычислить

$$S = \sum_{k=2}^N \prod_{i=1}^{k-1} \sin \frac{\pi * i}{k}$$

Вариант 10

Дано натуральное число N . Вычислить

$$S = \sum_{k=1}^N \frac{(\sum_{i=1}^N \sin(0.01 * k * i))}{k!}$$

Вариант 11

Поле шахматной доски определяется парой натуральных чисел, первое из которых задает номер вертикали, а второе – номер горизонтали. Даны натуральные числа k, l, m, n . Требуется выяснить, являются ли поля (k, l) и (m, n) полями одного цвета.

Вариант 12

Поле шахматной доски определяется парой натуральных чисел, первое из которых задает номер вертикали, а второе – номер горизонтали. Даны натуральные числа k, l, m, n . Требуется выяснить, угрожает ли ферзь, стоящий на поле (k, l) , полю (m, n) .

Вариант 13

Поле шахматной доски определяется парой натуральных чисел, первое из которых задает номер вертикали, а второе – номер горизонтали. Даны натуральные числа k, l, m, n . Требуется выяснить, угрожает ли конь, стоящий на поле (k, l) , полю (m, n) .

Вариант 14

Поле шахматной доски определяется парой натуральных чисел, первое из которых задает номер вертикали, а второе – номер горизонтали. Даны натуральные числа k, l, m, n . Требуется, если возможно, с поля (k, l) одним ходом ферзя попасть на поле (m, n) . Если нет – то определить, как это сделать за два хода (указать промежуточное поле).

Вариант 15

Поле шахматной доски определяется парой натуральных чисел, первое из которых задает номер вертикали, а второе – номер горизонтали. Даны натуральные числа k, l, m, n . Требуется, если возможно, с поля (k, l) одним ходом слона попасть на поле (m, n) . Если нет – то определить, как это сделать за два хода (указать промежуточное поле).

Вариант 16

Три друга были свидетелями ДТП. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий – что сумма цифр равна 30. Определить четырехзначный номер нарушителя.

Вариант 17

Во введенном тексте подсчитать количество слов, содержащих три буквы 'с' (слова разделены пробелами).

Вариант 18

По введенному символу установить, в каких позициях его двоичного кода записаны единицы.

Вариант 19

Найти наибольший общий делитель (НОД) двух введенных натуральных чисел, используя алгоритм Евклида.

Вариант 20

Даны вещественные числа A_1 , B_1 , C_1 , A , B , C . Выяснить взаимное расположение прямых $A_1 \cdot x + B_1 \cdot y = C_1$ и $A \cdot x + B \cdot y = C$. Если прямые пересекаются, напечатать координаты точки пересечения.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Раскройте понятие компиляции.
2. Опишите структуру компилятора.
3. Назовите этапы трансляции программы.
4. Опишите понятие конвейера.
5. Приведите пример использования конвейера.
6. Опишите понятие псевдонимов (alias).
7. Перечислите компоненты компилятора gcc.
8. Приведите пример команды для компиляции программы.
9. Перечислите и опишите опции компилятора gcc.
10. Назовите форматы, которые могут использоваться при создании двоичных файлов.
11. Назовите особенность компилятора gcc при использовании его для языка C++.
12. Перечислите данные о пользователе, которые хранятся в системе.
13. Опишите понятие системного имени (user name).
14. Опишите понятие идентификатора пользователя (UID).
15. Опишите понятие идентификатора группы (GID).
16. Опишите понятие полного имени (full name).
17. Опишите понятие домашнего каталога (home directory).
18. Опишите понятие начальной оболочки (login shell).
19. Перечислите права для каталогов.
20. Назовите команду, позволяющую посмотреть, какие пользователи работают в данный момент в системе.
21. Предложите вариант команды для добавления пользователя.
22. Приведите пример записи, хранящейся в файле /etc/passwd.
23. Опишите структуру записи, хранящейся в файле /etc/passwd.
24. Назовите команды для работы с учетными записями.
25. Назовите команды для работы с группами пользователей.
26. Предложите вариант команды, предоставляющей право на выполнение и отменяющей право на запись для файла file.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 3 занятия (6 академических часов: 5 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания, ответы на контрольные вопросы, описание процесса выполнения лабораторной работы, выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Вирт, Н. Разработка операционной системы и компилятора. Проект Оберон [Электронный ресурс] / Н. Вирт, Ю. Гуткнехт ; пер.с англ. Борисов Е.В., Чернышов Л.Н.. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 560 с. — Режим доступа: <https://e.lanbook.com/book/39992>

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

2. Крищенко, В.А. Сервисы Windows [Электронный ресурс] : учебное пособие / В.А. Крищенко, Н.Ю. Рязанова. — Электрон. дан. — Москва : МГТУ им. Н.Э. Баумана, 2011. — 47 с. — Режим доступа: <https://e.lanbook.com/book/52416..>

3. Войтов, Н.М. Администрирование ОС Red Hat Enterprise Linux. Учебный курс [Электронный ресурс] : учебное пособие / Н.М. Войтов. — Электрон. дан. — Москва : ДМК Пресс, 2011. — 192 с. — Режим доступа: <https://e.lanbook.com/book/1081>

4. Стащук, П.В. Администрирование и безопасность рабочих станций под управлением Mandriva Linux: лабораторный практикум [Электронный ресурс] : учебно-методическое пособие / П.В. Стащук. — Электрон. дан. — Москва : ФЛИНТА, 2015. — 182 с. — Режим доступа: <https://e.lanbook.com/book/70397>

5. Керниган, Б.В. Язык программирования С [Электронный ресурс] : учебник / Б.В. Керниган, Д.М. Ричи. — Электрон. дан. — Москва : , 2016. — 313 с. — Режим доступа: <https://e.lanbook.com/book/100543>

Электронные ресурсы:

6. Научная электронная библиотека <http://eLIBRARY.RU>

7. Электронно-библиотечная система <http://e.lanbook.com>

8. Losst - Linux Open Source Software Technologies <https://losst.ru>