1. Общие понятия

Операционная система (ОС) — это основное программное обеспечение, которое управляет компьютером. Она управляет аппаратными ресурсами (процессор, память, устройства ввода-вывода) и предоставляет платформу для выполнения программ. ОС работает как посредник между пользователем и оборудованием.
Основные функции ОС:
Управление файлами и директориями;
Управление памятью (какие программы используют память, сколько и когда);
Управление процессами (выполнение программ, многозадачность);
Управление устройствами (диски, принтеры, сетевые устройства);
Обеспечение безопасности и управления доступом.
Что такое Unix?
Unix — это многозадачная, многопользовательская операционная система, разработанная в 1969 году в компании Bell Labs (AT&T). Изначально она была создана для работы на серверах и рабочих станциях, но также используется в научных и образовательных целях. Unix повлиял на многие современные операционные системы, включая Linux, macOS и другие.
Основные особенности Unix:
Многозадачность — позволяет выполнять несколько задач (программ) одновременно.
Многопользовательская — поддерживает работу нескольких пользователей на одном компьютере.
Модульность — системы Unix состоят из множества небольших программ, каждая из которых выполняет одну задачу, и программы можно комбинировать для выполнения сложных операций.
Файловая система — в Unix всё рассматривается как файл (файлы данных, устройства, программы и даже сетевые ресурсы).
Отличие Unix и Linux
Unix и Linux похожи, потому что Linux был создан как "клон"Unix. У них одинаковый графический интерфейс (командная строка), но разные ядра. Также между ними есть важные различия.
Происхождение:
Unix был разработан в Bell Labs и лицензирован как проприетарное программное обеспечение. Linux был разработан Линусом Торвальдсом в 1991 году как свободная альтернатива Unix, под открытой лицензией GPL (General Public License).
Тип лицензии:
Unix — это коммерческая операционная система, хотя существуют открытые реализации, такие как FreeBSD.
Linux — это свободное программное обеспечение, доступное с открытым исходным кодом.
Совместимость:
Unix имеет множество коммерческих реализаций (например, AIX от IBM, HP-UX от HP).
Linux представлен множеством дистрибутивов (Ubuntu, CentOS, Debian), но все они используют одно ядро Linux и часто совместимы между собой.
Поддерживаемые архитектуры:
Linux поддерживает широкий спектр аппаратных платформ: от мобильных устройств до серверов и суперкомпьютеров.
Unix больше фокусируется на серверах и крупных вычислительных системах, хотя существуют и настольные версии (например, macOS).

2. Про файловые системы

Файловая система в UNIX — это способ организации данных на диске и управления файлами и директориями.
1. Что такое файл?
Файл — это минимальная единица данных в файловой системе. Это может быть текстовый документ, программа, изображение или даже каталог (директория). Все в UNIX (файлы, каталоги, устройства) рассматривается как файл.
2. Что такое директория (каталог)?
Директория — это специальный тип файла, который содержит другие файлы и директории. Директории создают структуру файловой системы, похожую на дерево. Корневая директория — это верхний уровень этой структуры. Она обозначается символом /.
3. Что такое файловая система?
Файловая система — это структура, которая определяет, как данные хранятся и организуются на диске. В UNIX файловая система организована как древовидная структура:
В самом верху — корневая директория /.
Под ней находятся другие директории и файлы.

Пример:

```
/ (root directory)
|-- home/
|   |-- user1/
|   |   |-- file1.txt
|   |   |-- file2.txt
|-- etc/
|   |-- config.conf
|-- var/
    |-- log/
```

Здесь /home/user1/file1.txt — это путь к файлу file1.txt.
4. Абсолютные и относительные пути
Абсолютный путь — это путь от корня файловой системы (от /). Он всегда начинается с /. Пример: /home/user1/file1.txt.
Относительный путь — это путь от текущей директории, в которой вы находитесь. Например, если вы находитесь в директории /home/user1/, то относительный путь к файлу file1.txt будет просто file1.txt.
5. Что такое . и ..?
Когда создается директория, в ней автоматически появляются два специальных файла:
. — это ссылка на саму себя (текущую директорию).
.. — это ссылка на родительскую директорию (директорию, на уровень выше).
Например, если вы находитесь в директории /home/user1/, то:
. — это путь к самой директории /home/user1/,
.. — это путь к родительской директории /home/. Эти файлы нужны для удобной навигации в файловой системе.
6. Жесткие ссылки и символические ссылки
В UNIX файловой системе можно создать несколько ссылок на один и тот же файл:
Жесткая ссылка — это другая ссылка на файл, которая хранится в той же файловой системе. Когда вы создаете жесткую ссылку, создается еще одно имя для файла, но файл сам не дублируется. Удалив одно имя (ссылку), файл все равно останется доступен через другую ссылку, пока не удалите все ссылки. Жесткие ссылки нельзя создавать на директории, чтобы не нарушить структуру дерева.
Символическая ссылка (или "симлинк") — это файл, который указывает на другой файл или директорию. Это как ярлык в Windows. Симлинк может указывать на файлы и директории, находящиеся даже в других файловых системах. Если удалите исходный файл, симлинк станет "битым т. е. будет указывать на несуществующий файл.
Пример:
Файл /home/user1/document.txt
Создаем жесткую ссылку: ln /home/user1/document.txt
/home/user2/doc_copy.txt
Создаем символическую ссылку: ln -s /home/user1/document.txt
/home/user2/doc_symlink.txt
7. Shell (оболочка)
Shell — это программа, которая позволяет пользователю взаимодействовать с операционной системой через командную строку. В Shell вы можете:
Навигировать по файловой системе (с помощью команд cd, ls и т.д.),
Выполнять программы,
Создавать, удалять файлы и директории и так далее.
Примеры команд:
cd /home/user1/ — перейти в директорию /home/user1/.
ls — показать список файлов в текущей директории.
mkdir new_folder — создать новую директорию.
rm file.txt — удалить файл.
8. Особенности файловой системы UNIX
Простота: все является файлом. Даже устройства, такие как жесткие диски и принтеры, рассматриваются как файлы.
Разделение прав доступа: каждая директория и файл имеет права доступа для владельца, группы и всех остальных пользователей (чтение, запись и выполнение).
Монтирование файловых систем: UNIX позволяет подключать разные файловые системы к одной иерархии (например, вы можете подключить USB-накопитель в определенную директорию).
Как записывается имя файла?
Имя файла в Unix (и Linux) записывается как последовательность символов, и оно может включать буквы, цифры, символы и даже пробелы (хотя пробелы нежелательны для удобства работы с командной строкой). Имя файла чувствительно к регистру, то есть file.txt и File.txt — это разные файлы.
Имя файла — это просто метка, которая используется для обращения к файлу через файловую систему. Полный путь

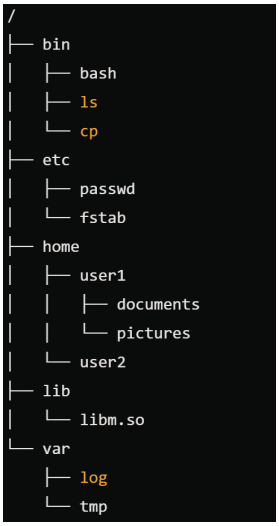
к файлу включает путь к его местоположению в файловой системе.
Пример пути к файлу: /home/user/documents/file.txt, где:
/home — это директория (каталог),
/home/user — подкаталог внутри /home,
/home/user/documents/ — подкаталог в директории user,
file.txt — это имя самого файла.
Что такое номер файла (inode)?
Номер файла (inode) — это уникальный идентификатор каждого файла в файловой системе Unix/Linux. В файловой системе каждый файл хранится не только по имени, но и имеет связанный с ним inode. В inode хранится важная информация о файле:
Размер файла,
Время последнего изменения,
Права доступа,
Количество жестких ссылок,
Физическое расположение данных на диске.
Важно: В inode не хранится имя файла. Имя файла хранится в директории и связано с соответствующим inode. Пример: Если два файла с разными именами имеют одинаковый inode, это значит, что это один и тот же файл с двумя именами.
Почему у файла может быть множество имен? У файла в Unix/Linux может быть множество имен благодаря тому, что существует понятие жесткой ссылки (это дополнительное имя для одного и того же файла). Так как несколько имен могут указывать на один и тот же файл через один и тот же inode, файл будет иметь несколько "имен"или "ссылок но физически это один и тот же файл. Пример:
Файл file1.txt имеет inode 12345. Мы создаем жесткую ссылку ln file1.txt file2.txt. Теперь у нас есть два файла file1.txt и file2.txt, но оба ссылаются на один и тот же файл через один и тот же inode 12345. Удаление одного из файлов не удаляет сам файл, так как пока остается хотя бы одна жесткая ссылка, файл продолжает существовать.

Жесткие ссылки, имена файлов и inode в Unix/Linux взаимодействуют следующим образом:
Как хранится информация о файле в директории?
Представь директорию как специальный файл (да, директория — это тоже файл), который содержит таблицу с записями. В этой таблице каждая запись соответствует файлу или поддиректории, находящейся в этой директории.
Эта запись не содержит сами данные файла (например, текст или изображение), а только имя файла и номер inode, который является уникальным идентификатором для этого файла. То есть, в директории не хранятся файлы целиком — там только ссылки (в виде имен файлов и связанных inode).
Пример структуры директории
Предположим, у нас есть директория /home/user, в которой лежит файл file1.txt. Когда ты создаешь файл, система создает запись в этой директории, которая связывает имя файла с его inode. Пример:
В директории /home/user может храниться запись:
file1.txt -> inode 12345
Это означает, что файл file1.txt имеет номер inode 12345. Вся подробная информация о файле (где он хранится на диске, размер файла, права доступа) находится в inode 12345, но имя файла хранится только в директории.
Таким образом, в директории на каждое имя файла есть запись, которая связывает это имя с inode.
Жесткие ссылки и несколько имен для одного файла
Теперь представь, что ты создаешь жесткую ссылку на файл. Жесткая ссылка — это еще одно имя для уже существующего файла. При этом создается новая запись в директории, которая указывает на тот же самый inode, что и исходный файл. Пример:
Ты создаешь файл file1.txt, который имеет inode 12345:
file1.txt -> inode 12345 Затем ты создаешь жесткую ссылку на этот файл, например:
ln file1.txt file2.txt Теперь в директории /home/user будет две записи:
file1.txt -> inode 12345
file2.txt -> inode 12345
Обе записи указывают на один и тот же inode (12345). Это значит, что у файла есть два имени — file1.txt и file2.txt, но это один и тот же файл на уровне файловой системы. Они ссылаются на одни и те же данные, расположенные в inode 12345.
Что происходит при удалении имени файла?
Если ты удаляешь одно из имен файла (например, file1.txt), система удаляет запись об этом имени в директории, но сам файл не удаляется, если на inode всё еще есть другие ссылки. В данном примере:
Если ты удаляешь file1.txt, остаётся запись file2.txt, которая по-прежнему ссылается на inode 12345. Файл не исчезнет с диска, так как inode 12345 ещё связан с file2.txt. Файл удаляется полностью только тогда, когда все ссылки (имена), которые ссылаются на inode, удалены. Когда удаляется последнее имя файла, счетчик ссылок inode становится равным нулю, и система удаляет файл с диска. Важный момент: в директории хранятся только ссылки на файлы. Теперь важно понять, что в самой директории не хранится никакая информация о данных файла. В директории есть только

записи, которые связывают имя файла с inode. В inode содержатся все данные о файле, но не в самой директории. Когда ты работаешь с файлами через командную строку или файловый менеджер, система находит файл по имени в директории, затем по номеру inode находит данные файла и показывает их тебе.

3. Дерево

Файловая система в Unix и Unix-подобных системах представляется в виде дерева, где каждая директория и файл являются узлами этого дерева. Такая структура позволяет удобно организовать данные и обеспечивает логическую иерархию, что упрощает навигацию и управление файлами. Давайте подробно рассмотрим, как это работает.
1. Корневая директория (/)
Вся файловая система начинается с корневой директории, обозначаемой /. Она является верхним уровнем дерева, и все остальные директории и файлы расположены под ней. Корневая директория не имеет родителя, и все другие элементы файловой системы являются её подчинёнными.
2. Деревовидная структура
Каждая директория может содержать другие директории и файлы. Это создаёт иерархическую структуру, где директории могут быть: Листовыми узлами: Директории, которые не содержат других поддиректорий, только файлы. Неполными узлами: Директории, которые содержат как поддиректории, так и файлы.
Вот пример иерархии файловой системы:



/bin: Содержит исполняемые файлы (программы).
/etc: Содержит конфигурационные файлы.
/home: Содержит домашние директории пользователей.
/lib: Содержит библиотечные файлы.
/var: Содержит изменяемые данные, такие как журналы.
3. Путь к файлу
Каждый файл или директория в файловой системе может быть доступен по полному пути (absolute path) или относительному пути (relative path).
4. Узлы дерева и файлы Каждый файл и директория в Unix-файловой системе представляются как узлы в дереве, и каждый узел содержит:
Имя: уникальное имя файла или директории в пределах родительского узла.
Местоположение: указатель на место, где фактически хранится файл на диске.
Метаданные: унформация о файле, такая как размер, права доступа, дата создания и последнего изменения, тип файла и т.д.
5. Логические устройства и точки монтирования
Unix позволяет монтировать файловые системы из разных устройств в дерево. Это означает, что различные устройства (жёсткие диски, USB-накопители и т.д.) могут быть доступны через структуру дерева, позволяя пользователю обращаться к ним как к обычным директориям.

4. Права

Задание прав пользователям в Unix и других операционных системах имеет несколько важных целей. Основной причиной является обеспечение безопасности и управление доступом к данным и ресурсам. **При использовании в целях безопасности разрешения и атрибуты (это свойства файла или директории, которые определяют его основные характеристики и поведение в операционной системе. Атрибуты предоставляют дополнительную информацию о файле или директории) защищают только от атак, запускаемых изнутри запущенной системы.** Вот основные причины, зачем задавать права пользователям:
1. Безопасность данных
Права доступа позволяют защитить данные от несанкционированного использования или модификации. Например, если все пользователи смогут свободно изменять важные файлы системы или приложения, это может привести к нестабильности или уязвимости системы. Система прав ограничивает доступ к файлам и ресурсам только тем, кому это необходимо. Пример: системные файлы или конфигурационные файлы могут быть доступны только для чтения большинству пользователей, а изменять их могут только администраторы. Это предотвращает случайное или намеренное нарушение работы системы.
2. Разделение ответственности и функций
В многопользовательских системах важно разграничить доступ к ресурсам между пользователями. Не все пользователи должны иметь одинаковые права на доступ к определенным файлам и программам. Это позволяет каждому пользователю выполнять свою задачу, не вмешиваясь в работу других. Пример: в корпоративной среде бухгалтер должен иметь доступ к финансовым данным, но не обязательно иметь права на редактирование конфигураций серверов или баз данных, которые обслуживает ИТ-отдел.
3. Защита конфиденциальности
Некоторые данные в системе могут быть конфиденциальными и предназначены только для конкретных пользователей или групп. Права доступа помогают сохранить конфиденциальность, ограничивая чтение, запись или выполнение файлов. Пример: в личной директории пользователя хранятся документы, которые не должны быть доступны другим пользователям системы. Права могут быть настроены таким образом, чтобы другие пользователи не могли просматривать эти файлы.
4. Обеспечение целостности системы
Права предотвращают случайное или намеренное повреждение критически важных системных файлов. Это особенно важно для файлов операционной системы, которые отвечают за её работоспособность и безопасность. Пример: обычные пользователи не должны иметь возможность модифицировать исполняемые файлы системы, так как это может привести к их повреждению и, в результате, к неработоспособности системы.
5. Эффективное использование системных ресурсов
Права доступа помогают администрировать системы, разделяя ресурсы между пользователями и группами. Это предотвращает случайное или умышленное использование ресурсов (процессоров, памяти, дискового пространства) теми, кому это не положено. Пример: если несколько пользователей одновременно работают на одном сервере, у каждого могут быть свои права на запуск программ, использование памяти или дискового пространства, что предотвращает конкуренцию за ресурсы.
6. Аудит и отслеживание действий
Система прав доступа также позволяет вести аудит того, кто и когда производил действия с файлами или другими ресурсами. Это важно для выявления потенциальных проблем с безопасностью, расследования инцидентов или мониторинга работы системы. Пример: журналы доступа могут показать, что определенный пользователь изменил важный конфигурационный файл, что привело к сбою в системе. Настройка прав помогает установить, кто был ответственен за это изменение.

В системе прав Unix для файлов и директорий используется довольно простая, но мощная модель, основанная на трех уровнях прав доступа для трех категорий пользователей. Unix разделяет всех пользователей на три категории:
Владелец (user, u) — обычно это создатель файла или директории.
Группа (group, g) — группа пользователей, которым может быть предоставлен доступ к файлу или директории.
Прочие (others, o) — все остальные пользователи, которые не являются ни владельцем, ни частью группы.
Права доступа в Unix можно увидеть с помощью команды ls -l, которая выводит права файла в следующем формате:
-rwxr-xr–
Первый символ указывает на тип файла: - — обычный файл, d — директория, l — символическая ссылка и так далее.
Следующие девять символов делятся на три блока по три символа:
Первый блок: права для владельца файла (user).
Второй блок: права для группы.
Третий блок: права для всех остальных (others).
Каждый блок содержит три символа:
r — право на чтение файла.
w — право на запись в файл.

x — право на исполнение файла.
Права доступа к директориям (r, w, x) в Unix имеют немного другое значение, чем для обычных файлов. Важное отличие заключается в том, что директории — это контейнеры для других файлов и поддиректорий, поэтому права на директорию определяют, что пользователь может делать с её содержимым (просматривать, изменять или перемещаться по ней).
Право на чтение директории позволяет пользователю просматривать список файлов и поддиректорий, находящихся внутри этой директории. Если у пользователя есть право на чтение (r) директории, он сможет выполнить команду ls, чтобы увидеть, какие файлы находятся в этой директории. Но при этом, если у пользователя есть право на исполнение (x), он всё равно сможет работать с файлами внутри директории, зная их точные имена (например, открыть файл напрямую с помощью cat или less).
Право на запись (w) в директорию даёт возможность добавлять, удалять и переименовывать файлы в этой директории. Однако для успешного выполнения этих операций, обычно требуется также наличие права на исполнение (x) (это сделано для безопасности: пользователь может иметь право изменять содержимое директории, но система должна быть уверена, что он также имеет доступ к этому содержимому). Если у пользователя есть право на запись (w) в директорию, он сможет создавать новые файлы или поддиректории, удалять или переименовывать существующие файлы (если пользователь знает точные имена файлов внутри директории).
Право на исполнение (x) директории даёт возможность "входить"в директорию, т.е. открывать её и перемещаться по её содержимому. Оно не связано с запуском программ, как для файлов, но означает, что пользователь может получить доступ к самой директории и выполнять действия над её содержимым, зная его точные имена. Если у пользователя есть право на исполнение (x) директории, он сможет перейти в неё с помощью команды cd или получить доступ к файлам внутри директории (например, открыть файл с cat или nano). Если у пользователя нет права на исполнение, то, даже если у него есть право на чтение, он не сможет перемещаться по директории или обращаться к её содержимому напрямую.
Если право отсутствует, на его месте будет стоять -. Например, запись rwxr-xr– расшифровывается так:
Владелец файла (user) имеет полные права: чтение, запись и исполнение. Группа пользователей (group) имеет права на чтение и исполнение, но не на запись. Прочие пользователи (others) имеют только право на чтение.
кроме символа - (отсутствие прав) в выводе прав доступа к файлам и директориям в Unix и Linux можно встретить также символы + и .
Эти символы добавлены для более детального отображения прав доступа, связанных с расширенными атрибутами (ACL) и контекстом безопасности. Символ + в выводе команды ls -l указывает на то, что для файла или директории заданы расширенные списки контроля доступа (ACL, Access Control List). ACL позволяют задавать более детальные права на уровне пользователя и группы, которые выходят за рамки стандартной системы прав. Здесь символ + в конце строки прав (-rw-r–r–+ 1 user group 4096 Oct 2 14:23 myfile.txt) показывает, что для файла myfile.txt определён ACL, который может предоставлять или ограничивать доступ конкретным пользователям или группам, помимо стандартных прав (например, можно указать дополнительные права для конкретных пользователей, кроме владельца и основной группы). Символ . указывает на то, что у файла или директории нет расширенных прав доступа (ACL), но может быть назначен контекст безопасности (это дополнительные метаданные, которые используются для определения правил доступа к файлам, процессам и другим ресурсам в системе на основе политик безопасности. Они особенно важны в системах с усиленными мерами контроля доступа), связанный с системами, такими как SELinux (Security-Enhanced Linux).
Примеры:
drwx—— 6 archie archie 4096 июл 5 17:37 Документы
Archie имеет полный доступ к каталогу Документы. Он может просматривать, создавать, переименовывать и удалять любые файлы в Документах, независимо от прав доступа к файлам. Его возможность получить доступ к файлам зависит от разрешений самого файла.
dr-x—— 6 archie archie 4096 июл 5 17:37 Документы
Archie имеет полный доступ, за исключением создания, переименовывания и удаления файлов. Он может просматривать список файлов и (если позволяют разрешения к файлам) может получить доступ к существующему файлу в Документах.
d-wx—— 6 archie archie 4096 июл 5 17:37 Документы
Archie не может выполнить ls в каталоге Документы, но если он знает имя существующего файла, то он может просмотреть, переименовать и удалить или (если позволяют разрешения) получить доступ к нему. Также он может создавать новые файлы.
d–x—— 6 archie archie 4096 июл 5 17:37 Документы
Archie может только (если позволяют разрешения файла) получить доступ к тем файлам в каталоге Документы, о которых он знает. Он не может просмотреть уже существующие файлы или создавать, переименовывать и удалить любой из них.

chmod — это команда в Linux и других Unix-подобных операционных системах, которая позволяет изменять права доступа к файлам или каталогам.
Давайте разберём каждую часть строки, которую вы видите в выводе команды ls -l:

-rw-r–r–. 1 user group 4096 Oct 2 14:23 myfile.txt
-rw-r–r– — Права доступа
Символ . в конце строки прав (-rw-r–r–.) указывает на то, что у файла нет расширенных прав доступа (ACL), но может существовать дополнительная информация, связанная с контекстом безопасности.
1 — Количество жёстких ссылок
user — Владелец файла
group — Поле указывает на имя группы, которая владеет файлом. Пользователи, входящие в эту группу, могут иметь доступ к файлу в зависимости от установленных для группы прав доступа (в данном случае r–, то есть только чтение).
4096 — Размер файла в байтах
Oct 2 14:23 — Дата и время последнего изменения
myfile.txt — Имя файла
Текстовый метод
Для изменения прав доступа — или режима доступа — к файлу используйте команду chmod в терминале. Ниже приведена общая структура команды:
chmod кто=разрешения имя_файла
Где кто — любая из нескольких букв, каждая из которых обозначает, кому дано разрешение. Они следующие:
u (user): пользователь, который является владельцем файла.
g (group): группа пользователей, которой принадлежит этот файл.
o (other): другие пользователи, то есть все остальные.
a (all): все сразу; используйте вместо ugo.
Предположим, вы захотели сильно защитить каталог Документы и отказать всем, кроме себя, в разрешении на чтение, запись и выполнение (или, в данном случае, поиск/просмотр) внутри него:
До: drwxr-xr-x 6 archie web 4096 июл 5 17:37 Документы
chmod g= Документы
chmod o= Документы
После: drwx—— 6 archie web 4096 июл 6 17:32 Документы
Здесь, поскольку вы хотите отказать в разрешениях, вы не ставите никаких букв после =, где будут введены разрешения. Из этого видно, что только разрешения владельца — это rwx, а все остальные разрешения — это -.
Рассмотрим ещё один пример: предположим, вы хотите изменить файл foobar так, чтобы у вас были разрешения на чтение и запись, коллеги из группы web, которые хотят совместно работать с файлом foobar, также могли читать и записывать файл, а все остальные пользователи могли только читать его:
До: -rw-r–r– 1 archie web 5120 июн 27 08:28 foobar
chmod g=rw foobar
После: -rw-rw-r– 1 archie web 5120 июн 27 08:28 foobar
Это точно такой же пример, как и первый, но с файлом, а не каталогом, и вы предоставляете разрешение на запись (просто для того, чтобы привести пример предоставления каждого разрешения).
Сокращения для текстового метода
Команда chmod позволяет добавлять и вычитать разрешения из существующих, используя + или - вместо =. Это отличается от описанных выше команд, которые по сути полностью заменяют разрешения (например, чтобы изменить разрешения с r– на rw-, вам всё равно нужно указать r и w после = в вызове команды chmod. Если вы пропустите r, то = перезапишет разрешения и таким образом удалит разрешение r. Использование + и - позволяет избежать этого, добавляя или отнимая разрешения из текущего набора разрешений).
Пример, который запрещает запись абсолютно всем:
До: -rw-rw-r– 1 archie web 5120 июн 27 08:28 foobar
chmod a-w foobar
После: -r–r–r– 1 archie web 5120 июн 27 08:28 foobar
Также есть специальный режим X: это не настоящий файловый режим, но он часто используется вместе с опцией -R, чтобы добавить бит выполнения только каталогам, но оставить его нетронутым у файлов. Типичный пример использования: chmod -R a+rX ./data/ (опция -R у команды chmod в Unix используется для рекурсивного изменения прав доступа к файлам и директориям. Это значит, что команда не только изменяет права доступа для указанной директории, но и для всех её содержимых, включая вложенные директории и файлы внутри них.)
Копирование разрешений С помощью chmod можно взять разрешения у одного класса, например владельца, и выставить те же разрешения группе или даже всем. Для этого вместо r, w или x после = поставьте нужную вам букву кто. Например:
До: -rw-r–r– 1 archie web 5120 июн 27 08:28 foobar
chmod g=u foobar
После: -rw-rw-r– 1 archie web 5120 июн 27 08:28 foobar
Эта команда по сути означает «изменить разрешения группы (g=), чтобы они были такими же, как у владельца (=u)». Обратите внимание, что вы не можете скопировать одновременно несколько разрешений или добавить новые, то есть такая команда chmod g=wu foobar выдаст ошибку.
Числовой метод Использование чисел — это ещё один метод, который позволяет редактировать разрешения одно-

временно для владельца, группы и остальных. Основная структура такова:
chmod xxx file
Где xxx это три цифры, каждая из которых может иметь значение от 0 до 7. Первая цифра задаёт разрешения для владельца, вторая — для группы, а третья — для всех остальных.
Права r, w и x соответствуют следующим числам:
r=4
w=2
x=1
Чтобы объединить нужные права в одно трёхзначное число, нужно суммировать соответствующие значения. Например, если вы хотите предоставить владельцу каталога права на чтение, запись и выполнение, а группе и всем остальным — только права на чтение и выполнение, то числовые значения будут выглядеть следующим образом:
Владелец: rwx = $4 + 2 + 1 = 7$
Группа: r-x = $4 + 0 + 1 = 5$
Остальные: r-x = $4 + 0 + 1 = 5$
chmod 755 file
Это эквивалентно следующим двум командам:
chmod u=rwx file
chmod go=rx file
Также двоичный метод, при котором каждое разрешение рассматривается как двоичное число, а затем они объединяются в обычное десятичное число.
Массовое изменение разрешений