# Unix

Алексей Зубаков
*на основе лекций Д. Халанского*

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

# Разграничение прав

Удалять файлы операционной системы или читать список паролей нельзя почти никогда, а запускать команду `ls` можно всегда.
В Unix есть два основных механизма для управления тем, что разрешено делать: пользователи и группы пользователей.
Каждый файл принадлежит какому-то пользователю и какой-то группе.

# Пользователи

Unix ”— многопользовательская система. Список пользователей обычно[1] находится в файле `/etc/passwd`. Каждая строка там ”— это описание пользователя в виде разделённых двоеточиями полей (см. `passwd(5)`:

- Имя;
- Пароль (в `/etc/passwd` не пишется);
- Номер пользователя;
- Номер *основной группы* пользователя;
- Описание;
- *Домашняя директория* (домашний каталог);
- Путь к shell, используемому пользователем.

[1] Иногда, например, он может приходить по сети.

# Использование пользователей

Важный пользователь ”— `root` (рут), с номером 0. Это суперпользователь, ему всё можно.
Есть пользователи, которые связаны с реальными людьми.
Запускать какую-то долго работающую программу с доступом в сеть от рута ”— плохо: если её взломают, то получат доступ ко всему. Хотя бы по этой причине создаётся много пользователей-болванок, от имени которых запущены сервисы и которым нельзя почти ничего.

# Домашний каталог

Каждому пользователю присвоен домашний каталог (и обычно ему же принадлежит). Там обычно хранятся его личные файлы, а используемые программы ищут/сохраняют свою конфигурацию где-то относительно домашнего каталога.
`cd` без аргументов переходит в домашний каталог.
Рабочая директория программы, запускаемой при входе в систему, будь то shell или графическая оболочка, ”— домашний каталог.
Домашний каталог обозначается символом ~.

# Группы

Каждый пользователь относится к набору групп. Созданные пользователем файлы будут принадлежать его основной группе.
`id` выводит имя и список групп данного пользователя.
Частая проблема: добавление пользователя в группу будет иметь эффект *только при следующем входе в систему*.
Запомните это!

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

# Права

В выводе команды `ls -l` есть следующие колонки: права на файл, число ссылок на него (это понятие мы не знаем), владелец файла, группа-владелец файла, размер файла, дата последнего редактирования, имя.

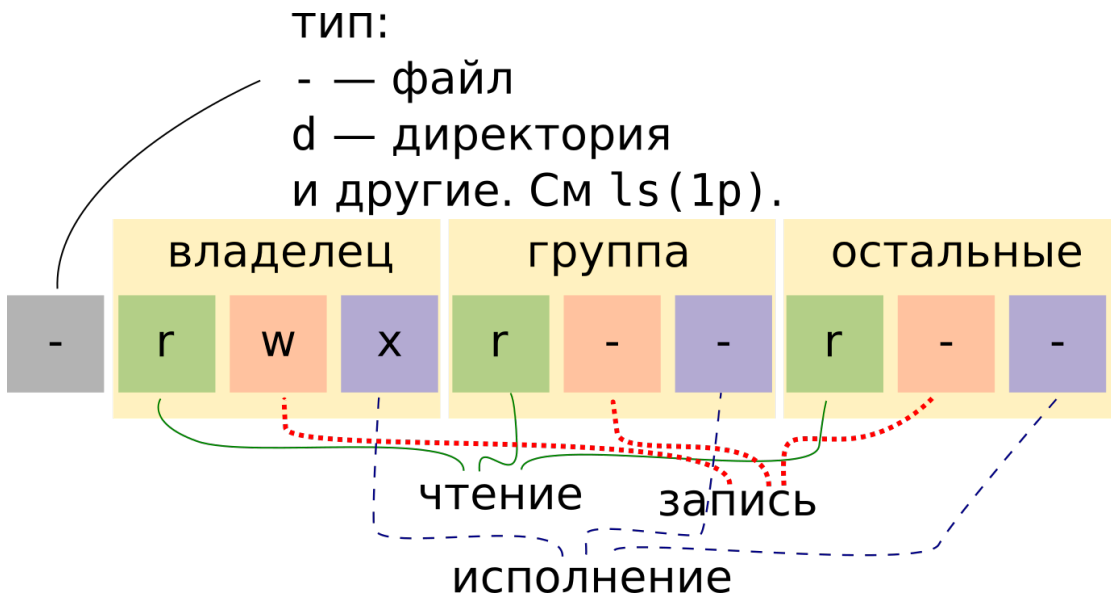


Рис. 1: Права в Unix.

# Смысл прав

**Чтение** Читать файл / Получать список имён содержимого директории.

**Запись** Писать в файл / Изменять содержимое директории: добавлять, удалять, переименовывать файлы.

**Исполнение** Исполнять как программу / Использовать данную директорию как текущую.

# SUID, SGID, sticky bit

В третьем поле каждой триады может быть не только – и `x`.

**`s` в первой триаде** SUID (set-user-id) ”— программа будет исполняться так, будто её запустил *её владелец*.

**`S` во второй триаде** SGID (set-group-id) ”— программа будет исполняться так, будто нынешний пользователь состоит в группе, которой она принадлежит.

**`t` в третьей триаде** sticky bit ”— на файлах не имеет смысла, а на директории означает, что удалить файл из неё сможет только владелец файла или самой директории.

# Примеры SUID

SUID нужен, когда хочется пользователям дать ограниченный объём власти. Несколько SUID-приложений, принадлежащих руту:

**su** (super user) запускает shell от имени другого пользователя (обычно ”— рута).

**sudo** (super user do) от имени другого пользователя (обычно ”— рута) выполняет поданную в аргументах команду. Пример:

```
1  $ sudo rm -rf /*
```

# Временная директория

`/tmp` ”— директория, в которой хранятся временные файлы. Обычно очищается при перезапуске компьютера. Обычно её содержимое вообще находится в оперативной памяти.

На `/tmp` стоит sticky bit: директория публичная, каждый должен иметь право в неё писать, но мы не хотим каждому позволять что угодно из неё удалить.

См. `mktemp(1)` или документацию к своему языку программирования, чтобы узнать, как создавать временные файлы и директории и никому этим не мешать.

# Числовое представление прав

Права часто пишут четырёхзначным числом в восьмеричной системе счисления. Младшие три разряда устроены одинаково: число 4 означает право на чтение, 2 ”— на запись, 1 ”— на исполнение, и разряд сотен ”— это права владельца, десятков ”— права группы, младший разряд ”— права остальных. `4000` ”— SUID, `2000` ”— SGID, `1000` ”— sticky bit.

Примеры:

- `4755` означает “установлен SUID; владелец имеет право на чтение, запись и исполнение (4 + 2 + 1 = 7), а все остальные ”— только на чтение и исполнение (4 + 1 = 5)”.
- `644` ”— владелец читает и пишет, а остальные только читают.
- `750` ”— владелец и группа читают и исполняют, а владелец может также писать.

См. `chmod(1p)`.

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

# Скрытые файлы

В каждой директории обязательно находятся две особых: “`.`” ”— это сама она же, “`..`” ”— её родительская.

Ранняя реализация `ls` хотела скрыть их из вывода такой проверкой:

```
1  if (name[0] == '.') continue;
```

Другие программисты посчитали, что это означает, что файлы, имя которых начинается с точки, ”— скрытые. Так пошла традиция.

`ls -a` показывает полное содержимое директории.

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

# Протоколы

Протокол ”— набор правил для общения между программами (обычно по сети).
Примеры:

- IP (Internet Protocol) ”— протокол, описывающий, как компьютерам из разных сетей находить друг друга.
- HTTP (HyperText Transfer Protocol) ”— протокол, задающий взаимодействие с веб-сайтами.
- HTTPS (HTTP (Secure)) ”— протокол, который добавляет поверх HTTP шифрование, чтобы никто не мог подменить текст или подглядеть, что пишут.
- FTP (File Transfer Protocol) ”— протокол для передачи файлов. Устарел.
- POP3, IMAP ”— протоколы для электронной почты.
- ... тысячи их.

# URL

URL (Uniform Resource Locator) ”— это адрес “ресурса”, чем бы он ни был.

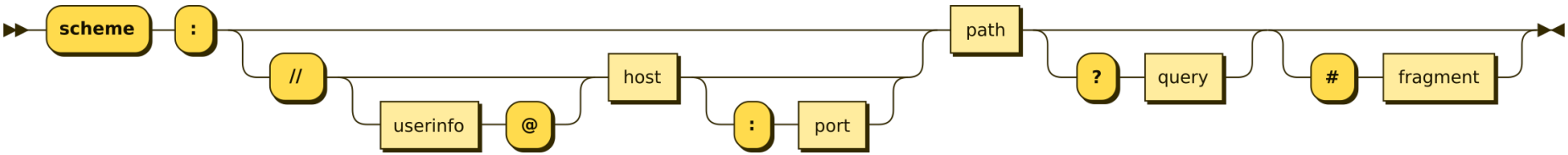


Рис. 2: форма URL (wiki).

- `https://github.com/torvalds/linux` ”— по протоколу HTTPS, по адресу github.com, по пути `torvalds/linux`.
- `ssh://git@github.com/torvalds/linux` ”— по протоколу SSH, от имени пользователя git, по адресу github.com, по пути `torvalds/linux`.
- `http://admin:password@127.0.0.1:8080/index.php?id=15#name` ”— по протоколу HTTP, от имени пользователя `admin`, пароль которого ”— `password`, по адресу `127.0.0.1`, на порту 8080, по пути `index.php`, со строкой запроса `id=15`, фрагмент `name`.

# Порт

Чтобы связываться по сети, недостаточно знать IP-адрес компьютера: на одном компьютере может быть много общающихся по сети программ: веб-сервер, база данных, интерфейс для удалённого подключения… Программы различаются *портом*, который *слушают*.
Порт редко надо указывать, так как для протоколов задаётся порт по умолчанию. У HTTP ”— 80, у HTTPS ”— 443, у FTP ”— 21, и так далее.

# ssh

ssh (Secure SHell) ”— протокол для неперехватываемого (Secure) получения доступа к Unix shell другого компьютера. Порт по умолчанию ”— 22. Аналоги: TeamViewer, VNC, Remote Desktop.
OpenSSH ”— реализация протокола ssh.
Команды:

- `ssh -p 22 level1@io.netgarage.org` ”— подключиться к компьютеру по адресу `io.netgarage.org` и порту 22 от имени пользователя `level1`.
- `scp -p 22 level1@io.netgarage.org:/levels/level01 level01` ”— скопировать файл `/levels/level01` с компьютера `io.netgarage.org`, подключившись по порту 22 от пользователя `level1`, в локальный файл `level01`.

`scp` по возможности лучше не использовать, он неудобный; выучите `rsync`.

# ssh-ключи

Для подключения по ssh используется пара из *публичного* и *приватного ключей* со стороны “клиента” (подключающегося) и “сервера” (того, куда подключаются).
Ключи рассматриваются во втором семестре на лекции по алгоритмам. Суть вкратце: каждый знает (и хранит в тайне) свой приватный ключ; каждый знает публичный ключ каждого; сообщения можно зашифровать чьим-то публичным ключом так, что расшифровать их можно только соответствующим приватным ключом.
`ssh-keygen -t rsa -b 4096` сгенерирует пару из публичного и приватного ключа “мощностью” 4096 для общения по протоколу RSA. Чем “мощнее” ключ, тем сложнее врагам будет его подобрать, но тем медленнее (де)шифрование.
Публичный ключ по умолчанию лежит в `~/.ssh/id_rsa.pub`, приватный ”— в `~/.ssh/id_rsa`.
См. `ssh-copy-id(1)`.

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

- Unix
  - Пользователи
  - Права
  - Скрытые файлы
- Связь по сети
- Всякое

# Хэши

Хэш ”— очень короткое и неполное представление каких-то данных. Это не сжатие: из хэша нельзя получить исходные данные. Пример хэша ”— md5-сумма.
Хэши стараются подбирать так, чтобы у схожих файлов был разный хэш. Хэши часто используют для проверки того, правильно ли скачался файл, не произошло ли с ним чего при передаче по сети.
Подробно (и более корректно) про хэши скажут на алгоритмах.

# Глобы

Вне кавычек в shell можно использовать специальные конструкции "— глобы "— для лёгкого перечисления файлов. Строка, в которой есть глобы, раскроется в столько аргументов, сколько есть файлов, подходящих под глоб.

- Символ `?` подразумевает любой символ.
- Символ `*` подразумевает сколько угодно любых символов.

Примеры:

- `abc*e` "— все файлы, имена которых начинаются с `abc` и заканчиваются на `e`.
- `???` "— все файлы с именами из трёх букв.
- `.??*` "— все файлы, имена которых начинаются с точки и состоят хотя бы из трёх символов.

См. `glob(7)`.

# diff

Есть утилита `diff file1 file2`, которая описывает, как надо изменить файл `file1`, чтобы в результате получился `file2`. Работает только на текстовых файлах: не на картинках или видео.

Имя нарицательное. *диффать* (to diff) ”— искать разницу между файлами (или версиями файла). *дифф* (a diff) ”— либо разница между файлами, либо, в узком смысле, результат выполнения команды `diff`.