

# Системное программирование

Лекция 9

Пользователи и группы

# Учетная запись пользователя

**Ползователь** - лицо, участвующее в функционировании автоматизированной системы или использующее результаты её функционирования [ГОСТ].

**Учетная запись пользователя** – уникальное в пределах системы имя, с которым связан набор параметров, определяющих пользователя, и набор привилегий, определяющих возможности пользователя.

- В пределах UNIX каждому пользователю соответствует уникальный идентификатор: UID (User ID, целое неотрицательное число).
- В каждой системе UNIX после установки существует как минимум один пользователь — root (UID=0).
- Пользователь root имеет все привилегии.

# Атрибуты пользователя

- UID (User ID, Идентификатор пользователя)
- Имя пользователя
- Пароль
- ID основной группы
- Домашний каталог
- оболочка (например /bin/sh)[Необязательно на некоторых ОС]
- Отображаемое имя, комментарий и пр.[Необязательно]

# Файл `/etc/passwd`

Изначально все атрибуты пользователя в открытом виде хранились в файле **`/etc/passwd`**.

Структура записи файла:

*(name):(passwd):(uid):(gid):(comment):(home):(shell)*

```
$ cat /etc/passwd
```

```
aborisov:x:1000:1000:,,,:/home/aborisov:/bin/bash
```

# Файл /etc/shadow

Хранение паролей в открытом виде достаточно быстро сочли плохой идеей.

В настоящее время зашифрованные пароли (или их хэши) хранятся в файле **/etc/shadow**.

Структура записи файла:

*(name):(passwd):<other attributes>*

```
$ sudo cat /etc/shadow
```

```
aborisov:$6$pLcK3S1Z$te.:18860:0:99999:7:::
```

Поле пароля в /etc/passwd равно «х», если пароль пользователя находится в /etc/shadow

# Атрибуты безопасности

Помимо пароля, в `/etc/shadow` находится ряд атрибутов безопасности:

- Шифрованный пароль (если начинается с «!», то учетная запись заблокирована)
- Время последнего изменения
- Минимальный «возраст» пароля в днях
- Максимальный «возраст» пароля
- Дата окончания срока действия учетной записи

# Процесс входа в систему

За вход в систему отвечает программа **/bin/login**.

- При входе login запрашивает имя пользователя. Если такого имени нет в `/etc/passwd`, возвращается ошибка.
- Если пользователь с таким именем существует, login запрашивает пароль, затем шифрует/хэширует его и сравнивает с записью в `/etc/shadow`.
- Если сравнение завершается успешно, и атрибуты безопасности не запрещают вход, login запускает новую сессию для данного пользователя.
- После N неудачных попыток ввода пароля возвращается ошибка

# Работа с учетными записями

Команды оболочки:

- вывести информацию о пользователе и его группах:

```
$ id [USER]
```

- вывести информацию о пользователе из файла /etc/passwd:

```
$ getent passwd [USER]
```

- вывести информацию о пользователе из файла /etc/shadow:

```
$ sudo getent shadow [USER]
```



# Работа с учетными записями

- Добавить нового пользователя:

```
$ sudo useradd -U [NAME]
```

- Добавить пользователя с конкретным UID:

```
$ sudo useradd -o -u [UID] -U [NAME]
```

- Изменить пароль пользователя

```
$ passwd # свой
```

```
$ sudo passwd [USER] # другого пользователя
```

- Изменить оболочку пользователя на /bin/zsh

```
$ usermod -s /bin/zsh [USER]
```

# Группы пользователей

**Группа пользователей** — именованный набор привилегий.

- Пользователь состоит в группе → пользователь имеет весь набор привилегий, связанных с группой.
- Каждый пользователь состоит как минимум в 1 группе.
- Группа, указанная в `/etc/passwd` является основной группой пользователя. Иные группы являются дополнительными.

## **Атрибуты группы:**

- GID (Group ID, Идентификатор группы)
- Имя группы
- Пароль группы [Необязательно]
- Список пользователей-членов группы

# Файл `/etc/group`

Группы вместе с их атрибутами перечислены в файле **`/etc/group`**.

Структура записи файла:

*(name):(passwd):(gid):(members)*

```
$ cat /etc/group
```

```
adm:x:4:syslog,aborisov
```

```
mygroup:x:1001:aborisov
```

# Файл `/etc/gshadow`

Пароль группы, а также администраторы группы (вместе со списком пользователей) находятся в файле **`/etc/gshadow`**.

- Администраторы группы имеют право менять пароль группы, а также состав группы — добавлять и удалять членов.
- Если у группы есть действительный пароль, то любой пользователь, знающий пароль, может к ней присоединиться на время сеанса.
- Только администратор может добавить пользователя в группу на постоянной основе.

**Структура записи файла:** *(name):(passwd):(admins):(members)*

```
$ sudo cat /etc/gshadow
```

```
adm:*::syslog,aborisov
```

```
mygroup:$6$cgT21b:aborisov:aborisov
```

# Работы с группами

- Узнать информацию о группах пользователя:

```
$ groups [USER]
```

- Изменить текущую активную группу на другую:

```
$ newgrp [GROUP]
```

- Создать новую группу:

```
$ sudo groupadd [NAME]
```

# Работы с группами

- Добавить пользователя в группу:

```
$ grasswd -a [USER] [GROUP]
```

- Удалить пользователя из группы:

```
$ grasswd -d [USER] [GROUP]
```

- Задать пароль группы:

```
$ grasswd [GROUP]
```

# Разграничение доступа

Помимо атрибутов, с пользователями также связаны привилегии — права на совершение определенных действий.

Из принципа “все есть файл” следует, что в UNIX наиболее фундаментальными привилегиями являются права на манипуляции с файлами.

**Права доступа проверяются только в момент открытия файла.**

# Владелец файла

- У любого файла есть владелец — обычно, это создатель файла.
- Помимо владельца, у каждого файла есть группа-владелец — обычно это текущая группа создателя в момент создания файла.
- Владелец файла имеет право определять права доступа к нему.
- Члены группы-владельца имеют особые права доступа.



# Изменение владельца файла

- И владелец, и группа-владелец могут быть изменены **командой *chown*** в терминале или **системным вызовом *chown()***.
- Изменение владельца файла требует повышенных привилегий (см. *man chown*, *man 2 chown*).
- Владелец обычно не может передать файл другому пользователю (в т.ч. из соображений безопасности — например, возможности подлога).
- Группа-владелец может быть изменена самим владельцем на любую группу, членом которой он является.

# Права доступа к файлу

- Существует 3 основных типа прав доступа к файлу — право на чтение (**r**), право на запись (**w**) и право на выполнение (**x**, запуск программы из файла).
- Право на выполнение (**x**) бесполезно без права на чтение (**r**).
- Существует 3 группы прав доступа — права доступа **владельца** файла, права доступа **членов группы-владельца** файла, права доступа **всех остальных пользователей**.
- Права доступа можно просмотреть утилитами ls/stat.

```
$ ls -l
```

```
-rwxrwxrwx 5 user group 4096 Sep 7 19:31 файл
```

# Права доступа к каталогу

Права доступа к каталогу имеют несколько иное значение.

- Права на чтение (**r**) и запись (**w**) дают право на просмотр каталога и создание новых элементов каталога.
- Право на выполнение (**x**) дает право на доступ к элементам каталога.
- Без права на выполнение (**x**) право на запись (**w**) бесполезно. Внизу представлен пример каталога, котором нельзя создать ни одного файла и нельзя открыть ни один файл.

```
$ ls -l
```

```
drw-rw-rw- 5 user group 4096 Sep  7 19:31 каталог
```

# Права доступа к файлу

Права доступа часто представляются в виде трех цифр. Биты каждой цифры соответствуют правам (  $7=111_2=rwx$ ,  $5=101_2=r-x$  ).

- **-rwxrwxrwx** (777)  
Обычный файл, все имеют права на чтение, запись, исполнение.
- **-rwxr-xr-x** (755)  
Обычный файл, владелец имеет полный доступ, остальные могут только читать и выполнять.
- **brw-rw----** (660)  
Файл блочного устройства, владелец и члены группы могут читать и писать.
- **drwx-----** (700)  
Каталог, только владелец имеет доступ.

# Команда chmod

- В терминале режим доступа к файлу может быть изменен владельцем файла командой *chmod*.

```
$ chmod [MODE] [FILES]
```

- MODE состоит из 3 частей и имеет вид (ugoa)(+--)(rwxst).

Пример:

```
$ chmod a=rwx file dir
```

# Команда chmod

MODE состоит из 3 частей и имеет вид (u~~g~~oa)(+~~-~~=)(rwxst).

- Часть (**u~~g~~oa**) отвечает за то, чьи права изменяются — владельца (u), группы (g), остальных (o) или всех разом (a).
- Часть (**+~~-~~=**) отвечает за то, как изменяются права — добавляются (+), отнимаются (-) или устанавливаются в точности как указано (=).
- Часть (**rwxst**) отвечает за то, какой флаг доступа изменяется — доступ на чтение (r), на запись (w), на выполнение (x), флаг set-user/group-id (s), флаг sticky (t)

# Команда chmod

*\$ chmod u-r file*

*\$ chmod go+w file*

*\$ chmod ug=rx file*

*\$ chmod a=rwx file*

*\$ chmod u+s file*

# Команда chmod

*\$ chmod u-r file*

запретить владельцу чтение файла

*\$ chmod go+w file*

разрешить группе и остальным запись в файл

*\$ chmod ug=rw file*

разрешить владельцу и группе чтение и запись, запретить исполнение

*\$ chmod a+rwx file*

разрешить всем чтение, запись и исполнение

*\$ chmod u+s file*

установить флаг set-user-id



# Вызов fchmod

Внутри программы на C флаги доступа могут быть изменены вызовом fchmod.

```
int fchmod(int fd, mode_t mode);
```

Аргументы:

fd     – файловый дескриптор;  
mode  – новый режим доступа;

В качестве значения mode могут передаваться именованные константы (S\_IRUSR, S\_IWUSR, S\_IXGRP, ...) или их объединение побитовым ИЛИ.

Вызов возвращает 0 в случае успеха и -1 в случае ошибки.

# Константы доступа

Краткий список констант флагов доступа (используются для параметра mode в вызовах creat, open, mkdir, fchmod и т.д.). Порядок: права владельца, права членов группы, права остальных

- Права на чтение: **S\_IRUSR (0400), S\_IRGRP (0040), S\_IROTH(0004)**
- Права на запись: **S\_IWUSR (0200), S\_IWGRP (0020), S\_IWOTH (0002)**
- Права на исполнение: **S\_IXUSR (0100), S\_IXGRP (0010), S\_IXOTH (0001)**
- Полный доступ: **S\_IRWXU (0700), S\_IRWXG (0070), S\_IRWXO (0007)**

S\_IRWXU | S\_IRGRP — какой набор прав? А (0666)?

# Специальные флаги доступа

Существует 3 специальных флага доступа:

- Флаг **set-user-id**: **S\_ISUID** (04000).
  - Флаг **set-group-id**: **S\_ISGID** (02000)
  - Флаг **sticky**: **S\_ISVTX** (01000).
- 
- Флаг **sticky** изначально использовался для кэширования исполняемых файлов (файл помеченный **sticky** «прилипал» к системному кэшу).
  - В настоящее время **sticky** используется для каталогов: из помеченного **sticky** каталога некоторый файл может быть переименован или удален только владельцем самого файла или владельцем каталога (даже если у других пользователей есть **wx**-права на сам каталог).

# Флаги `set-*-id`

Флаги **`set-user-id`** и **`set-group-id`** используются преимущественно для исполняемых файлов.

Исполняемый файл, помеченный *`set-user-id`*, запускается от имени владельца файла, а не от имени пользователя, реально запустившего файл. Аналогично для *`set-group-id`*.

При этом все права владельца файла (или группы) будут доступны программе (потенциальная дыра в безопасности, если использовать бездумно).

Для каталогов флаг *`set-group-id`* применяется при создании файлов — новый файл унаследует группу-владельца от каталога, а не от создавшего пользователя.

# Пользователи и программы

Когда пользователь запускает программу, с созданным процессом связываются 3 идентификатора пользователя:

- **Реальный идентификатор пользователя (UID)** — идентификатор пользователя, запустившего программу.
- **Эффективный идентификатор пользователя (eUID)** — идентификатор пользователя, участвующий в проверках безопасности (может отличаться от UID, см. флаг set-user-id).
- **Сохраненный идентификатор пользователя (sUID)** — эффективный идентификатор пользователя до системного вызова `execve()` над программой, помеченной *set-user-id*.

Для групп аналогично есть **GID, eGID, sGID**.

# Пользователи и программы

Получить идентификаторы можно вызовами `getuid()`, `geteuid()` и `getresuid()`.

```
uid_t getuid(void); //получить реальный идентификатор
uid_t geteuid(void); //получить эффективный идентификатор
/*non-POSIX – получить все идентификаторы*/
int getresuid(uid_t *ruid, uid_t *euid, uid_t *suid);
```

Для получения групповых идентификаторов существуют аналогичные вызовы.

# Вызов setuid

Для изменения эффективного идентификатора eUID пользователя используется вызов **setuid**:

```
int setuid(uid_t uid);
```

Вызов возвращает -1 при ошибке.

Всегда проверяйте результат setuid, или работа процесса может быть продолжена под учеткой предыдущего пользователя (серьезная угроза безопасности, если предыдущий пользователь - root).

См. также: seteuid()

# Пользователи и `setuid`

Обычные пользователи не могут произвольно изменить идентификатор с помощью `setuid()` (иначе возникнет дыра в безопасности)

Обычный пользователь может с помощью `setuid()` изменить эффективный идентификатор `eUID` на

- **[`eUID` := `UID`]** реальный идентификатор пользователя (можно узнать через вызов `getuid()`) ;
- **[`eUID` := `sUID`]** сохраненный идентификатор пользователя (можно узнать через не-POSIX вызов `getresuid()`).



# Root и setuid

В случае пользователя root поведение `setuid()` изменяется.

Идентификаторы могут быть изменены на идентификатор любого существующего в системе пользователя, НО изменяются все 3 идентификатора UID, eUID и sUID.

Поскольку изменятся UID и sUID, невозможно вернуться обратно к привилегиям root — они теряются навсегда (очевидно, из соображений безопасности) .

# Вызов setgid

Для изменения текущего идентификатора группы используется вызов `setgid()`:

```
int setgid(gid_t gid);
```

Вызов возвращает -1 при ошибке.

- Поведение вызова и его влияние на GID, eGID, sGID аналогично `setuid`.
- Пользователь имеет право менять текущий идентификатор группы на идентификатор любой группы в которой он состоит.

См. также: `setegid()`

# Использование `setuid` и `set-user-id`

Обычно вызов `setuid()` используется в паре с флагом *set-user-id* исполняемого файла.

- Обычно флаг *set-user-id* используется, когда в программе необходимо выполнить действия, требующие определенных привилегий (эффективный идентификатор `eUID = ID` владельца).
- Вызов `setuid()` используется для отказа от этих привилегий как только они перестанут быть необходимыми (принцип наименьших привилегий)

# Пример: su

Хорошим примером является команда su, позволяющая получить работать под именем другого пользователя.

Имеет место противоречие:

- Команда su запускается обычно не root-пользователем.
- Команде su требуется доступ к файлу /etc/shadow для проверки пароля => требуется права root.
- После успешного выполнения необходимо вернуть оболочку, работающую под именем указанного пользователя — не всегда root.

# Пример: su

Решение:

- Исполняемый файл `/bin/su` помечен флагом `set-user-id` + его владелец — `root` => программа всегда запускается от имени `root`.
- Программа читает `/etc/shadow` и проверяет пароль от имени `root`.
- В случае успеха программа использует вызов `setuid()` и отказывается от всех привилегий `root`.

# Пример: ping

Другим примером является программа ping.

- Программа ping посылает ICMP-сообщения —> необходимо использовать «сырой» сокет, что требует привилегий root.
- Выполнять ping может любой пользователь.

Решение:

- Исполняемый файл /bin/ping помечен флагом set-user-id + его владелец — root => программа всегда запускается от имени root.
- Программа открывает «сырой» сокет и сразу же выполняет `setuid(getuid())` : отказывается от привилегий root, как только они стали не нужны (минимальное временное окно для атаки).
- Программа далее работает от имени обычного пользователя. Даже если злоумышленник успешно взломает программу, привилегий root он не получит.