Лабораторная работа № 1.1
«Подсистема разграничения доступа в ОС Linux»

Цель: получить навыки работы с подсистемой разграничения доступа ОС Linux. Рассмотреть реализацию дискреционной и мандатной модели доступа в ОС Linux.

1. Теоретическая часть

Основу политики безопасности для компьютерной системы любой организации составляют правила разграничения доступа (ПРД) к объектам компьютерной системы. Разграничение доступа к компьютерным ресурсам базируется на различных моделях управления доступом. В ОС Microsoft Windows NT и ОС семейства Unix обычно применяется дискреционное и ролевое управление доступом к объектам. Однако есть расширения для ОС, а также специализированные ОС, включающие мандатную систему разграничения доступа (такие ОС, как МСВС, Astra Linux Special Edition; программно-аппаратный комплекс Secret Net для Windows NT)

Дискреционная модель разграничения доступа предполагает назначение каждому объекту списка контроля доступа, элементы которого определяют права доступа к объекту конкретного субъекта. Правом редактирования дискреционного списка контроля доступа обычно обладают владелец объекта и администратор безопасности. Эта модель отличается простотой реализации, но возможна утечка конфиденциальной информации даже в результате санкционированных действий пользователей.

Мандатная модель разграничения доступа предполагает назначение объекту метки (грифа) секретности, а субъекту – уровня допуска. Доступ субъектов к объектам в мандатной модели определяется на основании правил «не читать выше» и «не записывать ниже» своего уровня секретности. Использование мандатной

модели, в отличие от дискреционного управления доступом, предотвращает утечку конфиденциальной информации, но снижает производительность компьютерной системы.

Ролевая модель разграничения доступа основана на конструировании ролей и назначении их пользователям на основании выполняемых ими конкретных должностных обязанностей. При назначении и использовании ролей возможно наложение динамических и статических ограничений на совмещение разных ролей одним субъектом, одновременное использование одной роли разными субъектами и т.п. Подобный подход к разграничению доступа к объектам позволяет разделить обязанности между конструктором ролей и диспетчером ролей, а также более точно связать права доступа пользователей к объектам компьютерной системы с их обязанностями в организации, исключить избыточность полномочий.

Поскольку Linux – система многопользовательская, вопрос об организации разграничения доступа к файлам и каталогам является одним из существенных вопросов, которые должна решать ОС. Механизмы разграничения доступа, разработанные для системы UNIX в 70-х годах, очень просты, но они оказались настолько эффективными, что просуществовали уже более 40 лет и по сей день успешно выполняют стоящие перед ними задачи.

В основе механизмов разграничения доступа лежат имена пользователей и имена групп пользователей. В Linux каждый пользователь имеет уникальное имя, под которым он входит в систему. Кроме того, в системе создается некоторое число групп пользователей, причем каждый пользователь может быть включен в одну или несколько групп. Создает и удаляет группы суперпользователь (Root), он же может изменять состав участников той или иной группы. Члены разных групп могут иметь разные права по доступу к файлам, например, группа администраторов может иметь больше прав, чем группа программистов.

В индексном дескрипторе каждого файла записаны имя так называемого владельца файла и группы, которая имеет права на этот файл. Первоначально, при создании файла его владельцем объявляется тот пользователь, который этот файл создал. Точнее – тот пользователь, от чьего имени запущен процесс, создающий

файл. Группа тоже назначается при создании файла – по идентификатору группы процесса, создающего файл. Владельца и группу файла можно поменять в ходе дальнейшей работы с помощью команд chown и chgrp. Изменить владельца и группу может только суперпользователь (root).

Выполнение команд в Linux происходит через терминал, также можно написать файл с расширением «sh», который будет содержать команды для терминала. Чтобы получить информацию о конкретном файле в linux, в терминале можно использовать команду ls –l (рисунок 1). Первое поле в этой строке отражает права доступа к файлу. Третье поле указывает на владельца файла (user). Четвёртое поле указывает на группу, которая владеет этим файлом (users). Последнее поле указывает имя файла. Для вызова справки команда «ls --help».

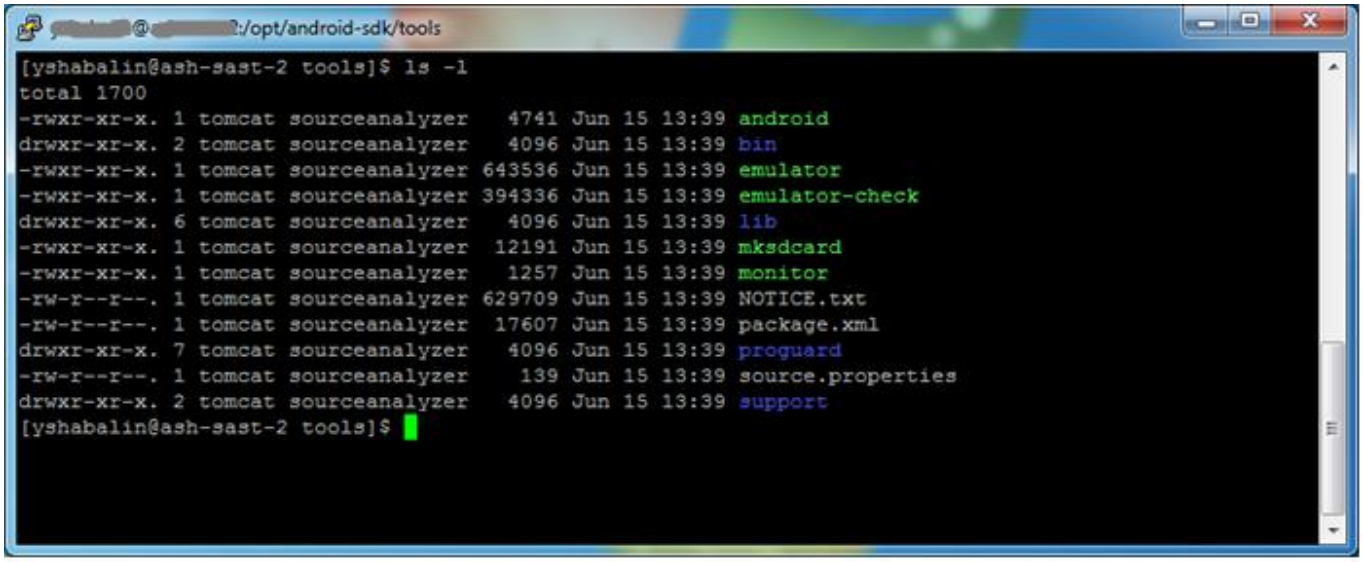

Рисунок 1 – Выполнение оманды «ls -1»

Первый символ означает тип файла (рисунок 2). Три следующие группы по три символа определяют права доступа к файлу соответственно для владельца файла, для группы пользователей, которая сопоставлена данному файлу, и для всех остальных пользователей системы. В нашем примере права доступа для владельца определены как rw, что означает, что владелец (Mem) имеет право читать файл (r), производить запись в этот файл (w), но права запускать файл на выполнение (x) у него отсутствует. Замена любого из этих символов прочерком будет означать, что пользователь лишается соответствующего права.

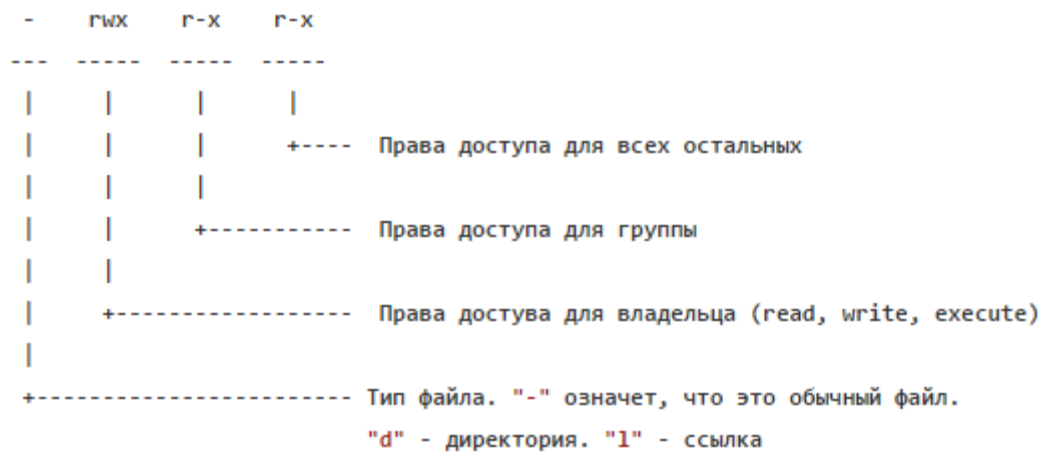

Рисунок 2 – Форма вывода прав доступа

Права доступа и информация о типе файла в UNIX-системах хранятся в индексных дескрипторах в отдельной структуре, состоящей из двух байтов, т. е. из 16 бит. Четыре бита из этих 16-ти отведены для кодированной записи о типе файла. Следующие три бита задают особые свойства исполняемых файлов, о которых мы скажем чуть позже. И, наконец, оставшиеся 9 бит определяют права доступа к файлу. Эти 9 бит разделяются на 3 группы по три бита. Первые три бита задают права пользователя, следующие три бита – права группы, последние 3 бита определяют права всех остальных пользователей (т. е. всех пользователей, за исключением владельца файла и группы файла). Если соответствующий бит имеет значение 1, то право предоставляется, а если он равен 0, то право не предоставляется. В символьной форме записи прав единица заменяется соответствующим символом (r, w или x), а 0 представляется прочерком.

Право на чтение (r) файла означает, что пользователь может просматривать содержимое файла с помощью различных команд просмотра, например, командой more или с помощью любого текстового редактора. Но сохранить изменения в файле на диске возможно, только если есть права на запись (w) в этот файл. Право на выполнение (x) означает, что вы можете загрузить файл в память и попытаться запустить его на выполнение как исполняемую программу. Конечно, если в действительности файл не является программой (или скриптом shell), то запустить

этот файл на выполнение не удастся, но, с другой стороны, даже если файл действительно является программой, но право на выполнение для него не установлено, то он тоже не запустится.

В Linux каталоги – те же файла, поэтому если выполнить команду ls –l для каталога, то, также как для файла, получим список прав доступа, причем они задаются теми же самыми символами rwx. Естественно, что по отношению к каталогам трактовка понятий "право на чтение", "право на запись" и "право на выполнение" несколько изменяется. Право на чтение по отношению к каталогам легко понять, если вспомнить, что каталог – это просто файл, содержащий список файлов в данном каталоге. Следовательно, если вы имеете право на чтение каталога, то вы можете просматривать его содержимое (этот самый список файлов в каталоге). Право на запись тоже понятно – имея такое право, вы сможете создавать и удалять файлы в этом каталоге, т. е. просто добавлять в каталог или удалять из него запись, содержащую имя какого-то файла и соответствующие ссылки. Право на выполнение интуитивно менее понятно. Оно в данном случае означает право переходить в этот каталог. Если вы, как владелец, хотите дать доступ другим пользователям на просмотр какого-то файла в своем каталоге, вы должны дать им право доступа в каталог, т. е. дать им "право на выполнение каталога". Более того, надо дать пользователю право на выполнение для всех каталогов, стоящих в дереве выше данного каталога. Поэтому в принципе для всех каталогов по умолчанию устанавливается право на выполнение как для владельца и группы, так и для всех остальных пользователей. И, уж если вы хотите закрыть доступ в каталог, то лишите всех пользователей (включая группу) права входить в этот каталог. Разница прав на чтение и выполнения есть. Если задать только право на выполнение, вы сможете войти в каталог, но не увидите там ни одного файла.

Алгоритм проверки прав пользователя при обращении к файлу можно описать следующим образом. Система вначале проверяет, совпадает ли имя пользователя с именем владельца файла. Если эти имена совпадают (т. е. владелец обращается к своему файлу), то проверяется, имеет ли владелец соответствующее право доступа: на чтение, на запись или на выполнение (не удивляйтесь, суперпользователь может

лишить некоторых прав и владельца файла). Если право такое есть, то соответствующая операция разрешается. Если же нужного права владелец не имеет, то проверка прав, предоставляемых через группу или через группу атрибутов доступа для остальных пользователей, уже даже не проверяются, а пользователю выдается сообщение о невозможности выполнения затребованного действия.

Если имя пользователя, обращающегося к файлу, не совпадает с именем владельца, то система проверяет, принадлежит ли владелец к группе, которая сопоставлена данному файлу. Если принадлежит, то для определения возможности доступа к файлу используются атрибуты, относящиеся к группе, а на атрибуты для владельца и всех остальных пользователей внимания не обращается. Если же пользователь не является владельцем файла и не входит в группу файла, то его права определяются атрибутами для остальных пользователей. Таким образом, третья группа атрибутов, определяющих права доступа к файлу, относится ко всем пользователям, кроме владельца файла и пользователей, входящих в группу файла.

Для управления правами в Linux существуют несколько команд:

- chmod - изменяет права доступа на файл;
- su - позволяет временно стать супер юзером;
- chown - изменяет владельца файла;
- chgrp - изменяет группу файла.

Для изменения прав доступа к файлу используется команда chmod. Вызов справки в терминале – «man chmod». Команду можно использовать в двух вариантах. В первом варианте вы должны явно указать, кому какое право даете или кого этого права лишаете:

[user]$ chmod wXp имя-файла.

Вместо символа w подставляется один из следующих символов:

- u (пользователь, который является владельцем);
- g (группа);
- (все пользователи, не входящие в группу, которой принадлежит данный файл);
- a (все пользователи системы, т. е. и владелец, и группа, и все остальные).

Вместо X ставится:

- + (предоставляем право);
- – (лишаем соответствующего права);
- = (установить указанные права вместо имеющихся),

Вместо p – символ, обозначающий соответствующее право:

- r (чтение);
- w (запись);
- x (выполнение).

Примеры использования команды chmod:

[user]$ chmod a+x file_name

предоставляет всем пользователям системы право на выполнение данного файла.

[user]$ chmod go-rw file_name

удаляет право на чтение и запись для всех, кроме владельца файла.

[user]$ chmod ugo+rwx file_name

дает всем права на чтение, запись и выполнение.

Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо [user]$ chmod a+x file_name

можно записать просто

[user]$ chmod +x file_name

Второй вариант задания команды chmod (он используется чаще) основан на цифровом представлении прав. Для этого мы кодируем символ r цифрой 4, символ w – цифрой 2, а символ x – цифрой 1. Чтобы предоставить пользователям какой-то набор прав, надо сложить соответствующие цифры. Получив, таким образом, нужные цифровые значения для владельца файла, для группы файла и для всех остальных пользователей, задаем эти три цифры в качестве аргумента команды chmod (ставим эти цифры после имени команды перед вторым аргументом, который задает имя файла). Например, если надо дать все права владельцу (4+2+1=7), право

на чтение и запись – группе (4+2=6), и не давать никаких прав остальным, то следует дать такую команду:

[user]$ chmod 760 file_name

Если вы знакомы с двоичным кодированием восьмеричных цифр, то вы поймете, что цифры после имени команды в этой форме ее представления есть не что иное, как восьмеричная запись тех самых 9 бит, которые задают права для владельца файла, группы файла и для всех пользователей.

Выполнять смену прав доступа к файлу с помощью команды chmod может только сам владелец файла или суперпользователь. Для того, чтобы иметь возможность изменить права группы, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл.

Пример цифровых прав:

rwx rwx rwx = 777

rw- rw- rw- = 666

rwx --- --- = 700

Есть также 3 возможных атрибута файла, устанавливаемые с помощью той же команды chmod. Это те самые атрибуты для исполняемых файлов, которые в индексном дескрипторе файла в двухбайтовой структуре, определяющей права на файл, занимают позиции 5-7, сразу после кода типа файла.

Первый из этих атрибутов – так называемый "бит смены идентификатора пользователя". Смысл этого бита состоит в следующем.

Обычно, когда пользователь запускает некоторую программу на выполнение, эта программа получает те же права доступа к файлам и каталогам, которые имеет пользователь, запустивший программу. Если же установлен "бит смены идентификатора пользователя", то программа получит права доступа к файлам и каталогам, которые имеет владелец файла программы (таким образом, рассматриваемый атрибут лучше называть "битом смены идентификатора владельца"). Это позволяет решать некоторые задачи, которые иначе было бы трудно выполнить. Самый характерный пример – команда смены пароля passwd. Все пароли пользователей хранятся в файле /etc/passwd, владельцем которого является

суперпользователь root. Поэтому программы, запущенные обычными пользователями, в том числе команда passwd, не могут производить запись в этот файл. А, значит, пользователь как бы не может менять свой собственный пароль. Но для файла /usr/bin/passwd установлен "бит смены идентификатора владельца", каковым является пользователь root. Следовательно, программа смены пароля passwd запускается с правами root и получает право записи в файл /etc/passwd (уже средствами самой программы обеспечивается то, что пользователь может изменить только одну строку в этом файле).

Установить "бит смены идентификатора владельца" может суперпользователь с помощью команды

[root]# chmod +s file_name

Аналогичным образом работает "бит смены идентификатора группы".

Еще один возможный атрибут исполняемого файла – это "бит сохранения задачи" или "sticky bit" (дословно – "бит прилипчивости"). Этот бит указывает системе, что после завершения программы надо сохранить ее в оперативной памяти. Удобно включить этот бит для задач, которые часто вызываются на выполнение, так как в этом случае экономится время на загрузку программы при каждом новом запуске. Этот атрибут был необходим на старых моделях компьютеров. На современных быстродействующих системах он используется редко.

Если используется цифровой вариант задания атрибутов в команде chmod, то цифровое значение этих атрибутов должно предшествовать цифрам, задающим права пользователя:

[root]# chmod 4775 file_name

При этом веса этих битов для получения нужного суммарного результата задаются следующим образом:

- ✓ 4 – бит смены идентификатора пользователя;
- ✓ 2 – бит смены идентификатора группы;
- ✓ 1 – бит сохранения задачи (sticky bit).

Если какие-то из этих трех битов установлены в 1, то несколько изменяется вывод команды ls -l в части отображения установленных атрибутов прав доступа.

Если установлен в 1 "бит смены идентификатора пользователя", то символ "x" в группе, определяющей права владельца файла, заменяется символом "s". Причем, если владелец имеет право на выполнение файла, то символ "x" заменяется на маленькое "s", а если владелец не имеет права на выполнение файла (например, файл вообще не исполняемый), то вместо "x" ставится "S". Аналогичные замены имеют место при задании "бита смены идентификатора группы", но заменяется символ "x" в группе атрибутов, задающих права группы. Если равен 1 "бит сохранения задачи (sticky bit)", то заменяется символ "x" в группе атрибутов, определяющей права для всех остальных пользователей, причем "x" заменяется символом "t", если все пользователи могут запускать файл на выполнение, и символом "T", если они такого права не имеют.

Важные для практической команды для работы в терминале Linux:

- ✓ man название_функции – вызов справки;
- ✓ man man – вызов общей справки;
- ✓ dir – отображает содержимое текущего каталога (папки);
- ✓ mkdir название_каталога – создание каталога;
- ✓ cd – команда перехода в другой каталог;
- ✓ echo – команда вывода текста; для ввода данных в файл используем команду, например, так: echo “данные в кавычках” > название_файла.расширение (если название файла содержит пробел, то также вместе с расширением название в кавычках; если мы хотим дописывать информацию, а не писать с затиранием, то используем двойной знак перенаправления потока >>);
- ✓ more название_файла.расширение – команда для вывода данных;
- ✓ stat – выводит статистику о файле;
- ✓ ls -l название_файла.расширение – команда для вывода владельца (пользователя и группы), кому принадлежит файл в выводе команды;
- ✓ whoami – команда для вывода имени текущего пользователя;
- ✓ groups – команда, чтобы узнать в какие группы входит текущий пользователь;

✓ groups польз_1 польз_2 – команда, позволяющая узнать пользователю 1 в какие группы входит пользователь 2;

✓ umask $x_2x_3x_4$ – маска режима создания пользовательских файлов (прав);

✓ chown польз_1 файл_1 – команда для изменения прав доступа пользователя польз_1 к файл_1 (параметр –R позволит изменить права также на все файлы в подкаталогах);

✓ chgrp груп_1 файл_1 – команда для изменения прав доступа группы груп_1 к файл_1 (параметр –R позволит изменить права также на все файлы в подкаталогах);

✓ chmod =rx – команда для установки прав к файл_1 только на чтение и запуск;

✓ chmod $x_1x_2x_3x_4$ файл_1 – команда для изменения прав доступа к файл_1, x1x2x3x4 – набор определяемых прав (принцип формирования комбинации описан выше);

✓ chmod u+s файл_1 – команда для установки бита suid;

✓ chmod g+s файл_1 – команда для установки бита sgid;

✓ chmod g-s файл_1 – команда для удаления бита sgid;

✓ chmod u-s файл_1 – команда для удаления бита suid;

✓ getfacl файл_1 – команда получения прав доступа к файлу файл_1;

✓ setfacl -m user:[пользователь]:права[user:пользователь:права] или -m group:[группа]:права[group:группа:права] файл_1– команда установки прав доступа к файлу файл_1, заметим:

- параметр «–m» означает модификацию, чтобы убрать права используем «–r»;
- для файла 1.txt установить право чтения для пользователя aaa_user, право чтение/запись для bbb_user, чтение выполнение для группы alphabet_users, запретить группе num_users выполнять какие либо действия: «setfacl -m u: aaa_user:r,u:bbb_user:rw,g:num_users:-,g: alphabet_users:rx 1.txt»;

- для файла 1.txt установить запрет доступа всем, полные права root и право запуска для группы root: «setfacl -m "u::-,o::-,g::-,u:root:rwx,g:root:x" 1.txt)».

Популярные значения прав доступа (2-4 символ в цифровой маске доступа):

✓ 400 (-r--------) владелец имеет право чтения, никто другой не имеет права выполнять никакие действия;

✓ 644 (-rw-r--r--) все пользователи имеют право чтения, владелец может редактировать;

✓ 660 (-rw-rw----) владелец и группа могут читать и редактировать, остальные не имеют права выполнять никаких действий;

✓ 664 (-rw-rw-r--) все пользователи имеют право чтения, владелец и группа могут редактировать;

✓ 666 (-rw-rw-rw-) все пользователи могут читать и редактировать;

✓ 700 (-rwx------) владелец может читать, записывать и запускать на выполнение, никто другой не имеет права выполнять никакие действия;

✓ 744 (-rwxr--r--) каждый пользователь может читать, владелец имеет право редактировать и запускать на выполнени;

✓ 755 (-rwxr-xr-x) каждый пользователь имеет право читать и запускать на выполнение, владелец может редактировать;

✓ 777 (-rwxrwxrwx) каждый пользователь может читать, редактировать и запускать на выполнение;

✓ 1555 (-r-xr-xr-t) каждый пользователь имеет право читать и запускать на выполнение, удалить файл может только владелец этого файла;

✓ 2555 (-r-xr-sr-x) каждый пользователь имеет право читать и запускать на выполнение с правами группы (user group) владельца файла;

✓ 4555 (-r-sr-xr-x) каждый пользователь имеет право читать и запускать на выполнение с правами владельца файла.

2. Практическая часть

Linux

1) Зайти по ssh или через PuTTY на сервер Linux под пользователем root
2) Создать пользователя student, задать ему пароль и выйти из сессии:

```
# useradd –m student
# passwd student
# exit
```

3) Зайти по ssh или через PuTTY на сервер Linux под пользователем student и выполнять последующие действия относительно домашнего каталога пользователя
4) В директории owner/dir_1 выдать права на файлы следующим образом:
   - file_1 - доступен только для чтения владельцу
   - file_2 - доступен для записи и чтения владельцу
   - file_3 - полный доступ для владельца
5) Выдать права на директорию owner/dir_2
   - subdir_1 - доступен только для чтения владельцу
   - subdir_2 - доступен для записи и чтения владельцу
   - subdir_3 - полный доступ для владельца
6) В директории owner/dir_3
   - file_1 - изменить владельца файла на любого другого пользователя
   - file_2 - доступен для записи и чтения владельцу, содержимое файла - UID, GID и группы в которых находится пользователь
7) В директории group/dir_1 выдать права на файлы следующим образом:
   - file_1 - доступен только для чтения владельцу и группе
   - file_2 - доступен для записи и чтения владельцу и группе
   - file_3 - полный доступ для владельца и чтение + выполнение для группы
8) Выдать права на директорию group/dir_2

- subdir_1 - доступен только для чтения владельцу и для чтения\записи группе
- subdir_2 - доступен для записи и чтения владельцу и для чтения группе
- subdir_3 - полный доступ для владельца и исполнение\чтение для группы

9) В директории group/dir_3
   - file_1 - изменить владельца файла и группу на любого другого пользователя и группу
   - file_2 - доступен для записи и чтения владельцу и группе, содержимое файла - список процессов текущего пользователя
10) В директории others/dir_1 выдать права на файлы следующим образом:
   - file_1 - доступен только для чтения владельцу и группе, всем остальным - чтение
   - file_2 - доступен для записи и чтения владельцу, чтения - группе и полный доступ всем остальным
11) Выдать права на директорию others/dir_2
   - subdir_1 - доступен только для чтения владельцу и для чтения\записи группе, на чтение всем остальным
   - subdir_2 - доступен для записи и чтения владельцу и для чтения группе, для чтения\выполнения всем остальным
   - subdir_3 - полный доступ для владельца и исполнение\чтение для группы, чтение и запись для всех остальных
12) В директории others/dir_3
   - file_1 - символьная ссылка на file_2
   - file_2 - полный доступ для всех, содержимое файла - список файлов с правами из текущей директории
13) Вывести список процессов для текущего пользователя / всех пользователей, обратить внимание на параметр UID, за что он отвечает?