Для начала обновим базу доступных для установки пакетов командой:

**sudo apt-get update**

Любой дистрибутив развивается. Это значит, что в дистрибутив могут добавлять новое ПО, а также исправлять ошибки в старом. В результате этих процессов могут меняться названия имеющихся пакетов (добавляют какую-нибудь цифру к версии), либо даже зависимости этих пакетов. И чтобы устанавливаемое ПО было заведомо работоспособным, желательно закачивать самые свежие пакеты. А для этого надо иметь самую свежую базу пакетов.

На боевом сервере лучше сделать бэкап перед тем как обновляешь что-либо

Установка призводится вот этой командой:

**sudo apt-get install samba**

Если инсталлятор начнет задавать тебе вопросы (скорее всего про winbind), то просто оставляй значения по умолчанию.

Для простоты поставь себе Midnight Commander. С консолью будем знакомиться по чуть-чуть

Идем в папку ***с конфигурационными файлами samba***. Там будет лежать демонстрационная конфигурация.

**cd /etc/samba**

Мы будем писать свою, так что дефолтную конфигурацию удаляем (или можно переименовать и почитать файл на досуге. Помимо настроек он обильно снабжен описанием этих настроек).

Сохраним его на всякий случай

**sudo mv smb.conf smb.conf.old**

если хотим сохранить файл.

Команда mv перемещает и переименовывает файл

Теперь создаем два пустых файла:

**sudo touch smb.conf**

**sudo touch smbusers**

***Файл smb.conf - конфигурация файлового сервера***

***Файл smbusers*** - будет содержать логины и пароли (зашифрованные естественно) пользователей файлового сервера.

А также создадим папку, которую будем расшаривать в сеть.

**sudo mkdir /opt/FirstTestShare**

**sudo chmod 0777 /opt/FirstTestShare**

Теперь открываем ***файл smb.conf*** в любом удобном редакторе (nano, mcedit, vim, emacs, ee или какой-то другой) и начинаем составлять конфигурацию нашего сервера.

Конфигурация сервера состоит из нескольких секций. Основная секция [global] и каждая шара будет настроена отдельной секцией, также обозначенной квадратными скобками. К примеру [share1], [share2], [Music], [Video] и т.д.

Перво-наперво настраиваем общие настройки - т.е. секцию global.

Для этого пишем

**[global]** и ниже параметры, которые вы сейчас узнаете

Настроим представление сервера в сети Windows

Для начала дадим нашему серверу имя. Под этим именем он будет виден в сетевом окружении. Обычно его делают таким же как имя самого сервера, но это необязательно. Назовем его **Test Fileserver Samba**

*server string = Test Fileserver Samba*

Далее опишем рабочую группу. В сети windows рабочей группой обозначается идентификатор, общий для всех компьютеров, которые должны видеть друг друга в сетевом окружении. По умолчанию это **WORKGROUP**. Его мы и зададим

*workgroup = WORKGROUP*

Зададим роль нашего сервера. Я упоминал ранее, что **samba** не так проста и может выполнять разные функции. Так вот мы в конфигурационном файле задаем роль - обычный файловый сервер.

*server role = standalone server*

Теперь настроим параметры авторизации

Указываем тип авторизации. Может принимать значения:

- **user** - авторизация по логину и паролю

- **domain** - авторизация по логину, паролю и домену, если сервер введен в домен.

- **share** - анонимная шара. без авторизации совсем.

Для нашей задачи идеально подходит значение user

*security = user*

Указываем тип БД пользователей.

*passdb backend = smbpasswd*

Указываем файл БД пользователей.

*smb passwd file = /etc/samba/smbusers*

Принудительно разрешаем шифрованние паролей. Это хорошо для безопасности.

*encrypt passwords = yes*

Такой строкой мы запрещаем гостевой вход. В среде windows есть специальный пользователь для гостевых подключений и входа на компьютеры. Вот такому гостю запрещена авторизация. У нас же вход по логину и паролю. Верно?

*map to guest = bad user*

И напоследок логи

Указываем файл, в который будут писаться сообщения, а также уровень логирования (от 0 до 7). Чем выше цифра уровня логирования, тем подробнее будут писаться сообщения в лог. 7-кой лучше не злоупотреблять, т.к. лог может быстро слопать все дисковое пространство. Она нужна только для отладки.

*log file = /var/log/samba/log.%m*

В первую очередь надо определить имя шары. Под этим именем расшаренная папка будет видна из сети. Можно использовать кириллицу, но лучше воздержаться от служебных символов. Не все ОС эти символы воспримут правильно =).

**[FirstTestShare]**

Комментарий это подсказка, которая будет высвечиваться если навести на расшаренную папку курсор мышки.

*comment = Our First Test Share*

Конечно же определим путь к расшариваемой папке в нашей файловой системе. Путь должен быть абсолютным.

*path = /opt/FirstTestShare*

Явно указываем, что эта папка должна быть видима. Если поставить no, то она будет скрыта, но по прямому пути будет доступна.

*browseable = yes*

Разрешаем запись в нашу шару.

*writeable = yes*

Определяем права, которые будут выдаваться файлам, залитым на шару. Записываются они аналогично тому, как мы это делаем для утилиты chmod.

*create mask = 0775*

Определяем права, которые будут выдаваться директориям, созданным или залитым на шару.

*directory mask = 0775*

Перед тем как запускать сервис, надо проверить конфигурационный файл на корректность. У многих развитых пакетов ПО есть утилита для такой задачи. Есть она и в пакете samba.

Выполняй:

testparm -s

Если оно выдало ошибку, то надо исправить. Повторять до того, пока ошибки не перестанут выводиться.

После сохранения корректного конфигурационного файла перезапускаем сервис samba. Если сервис был выключен он включится, а если был включен, то перезапустится.

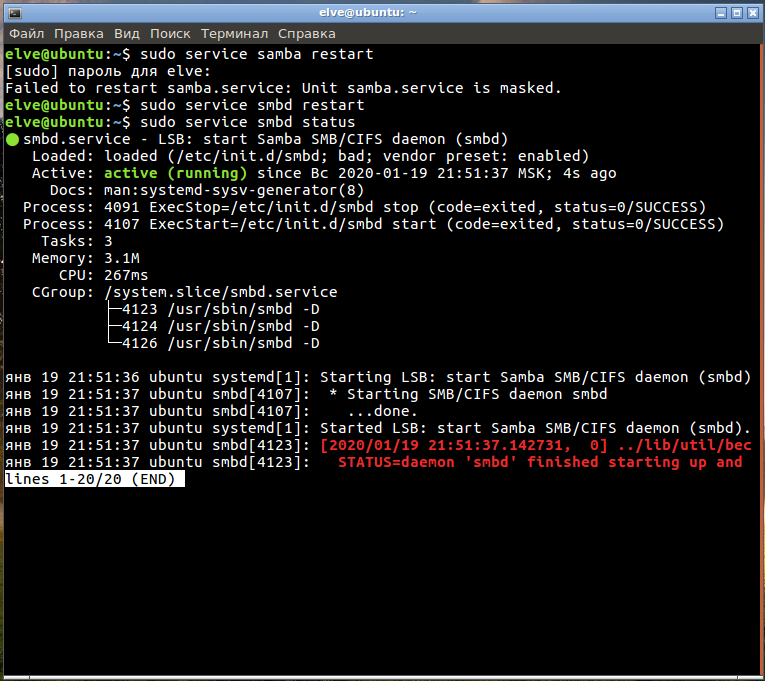
**sudo service smbd restart**

И добавляем сервис в автозагрузку, чтобы после ребута сервера не пришлось снова поднимать руками

**sudo systemctl enable smbd**

Просмотреть статус процесса можно командой

**sudo service smbd status**



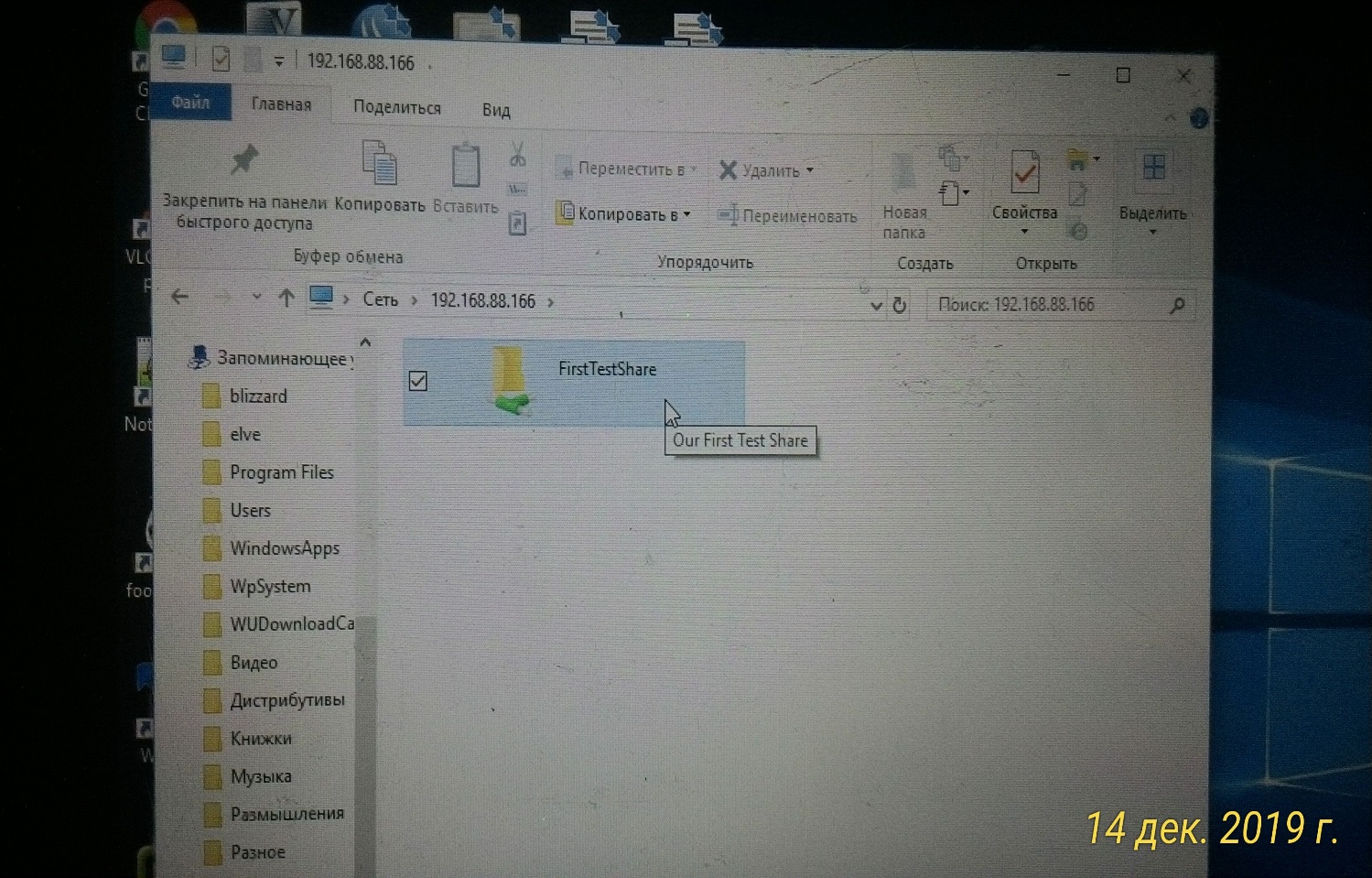
Небольшое уточнение

Раньше сервис назывался **samba**, а потом в 16-й убунте его переименовали в **smbd**. Поэтому если не получается включить и запустить сервис **smbd**, то попробуйте **samba**. Какой-то из скриптов точно добавился после установки.

Далее

Вот наш сервер настроен и запущен. В сетевом окружении вы можете его найти попробовать зайти в расшаренную папку.

Вот так это выглядит



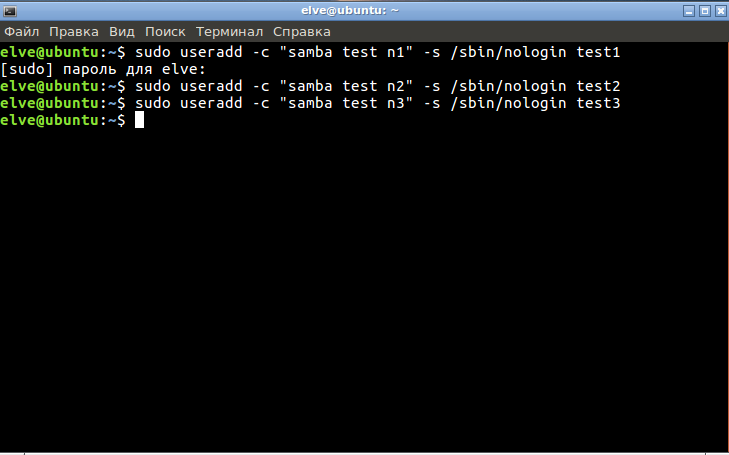
Но есть одна "проблемка" - у нас просят ввести логин и пароль, а мы их не знаем....

Значит надо их создать

Сразу предупреждаю, что у samba есть некоторая особенность при работе с пользователями без домена. Добавить доступ к шаре мы можем лишь тем пользователям, которые созданы локально на файловом сервере. При этом для доступа к шаре мы можем задать этим пользователям любой пароль. Из ОС он не копируется.

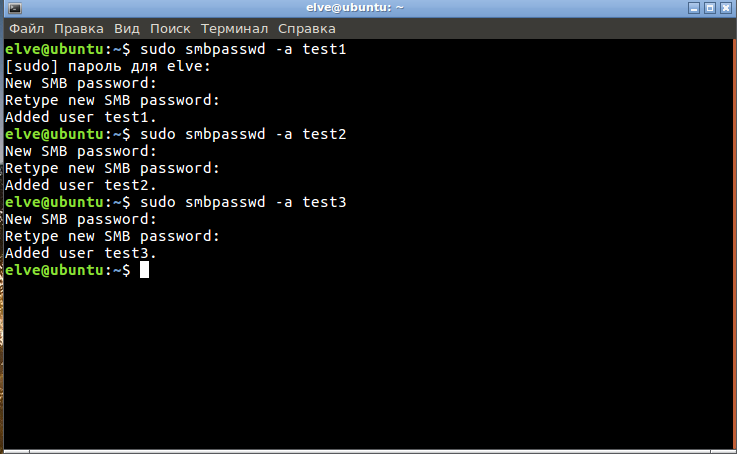
Не зря же мы создавали **файл *smbusers*** в начале урока ;)

Создадим троих пользователей



*Параметр* ***-s /sbin/nologin*** *означает, что эти пользователи не имеют права логиниться в консоль сервера. Это важно с точки зрения безопасности.*

И зададим им пароли



А вот теперь мы можем ввести учетные данные любого из пользователей и попасть на шару.

Мы можем как создавать свои папки и файлы, так и читать папки и файлы других пользователей. Однако прав на удаление и редактирование файлов и папок других пользователей у нас нет. Изменить это поведение можно через параметры **create mask** и **directory mask**

На этой радостной ноте можно было бы и закончить... в идеальном мире. Но в жизни порой возникают ситуации, которые без системного администратора будет сложно решить.

Рассмотрим ситуацию.

В организации работал бухгалтер test1. Но время идет, фирма растет, нагрузка растет.

И вот наняли ему в помощь test2.

Пользователь test2 заходит на шару в папку Buh и может все файлы просматривать,

но не может их редактировать.

Как так получилось?

Дело в том, что у нас ***create mask и directory mask*** настроены так, что файлы пользователя

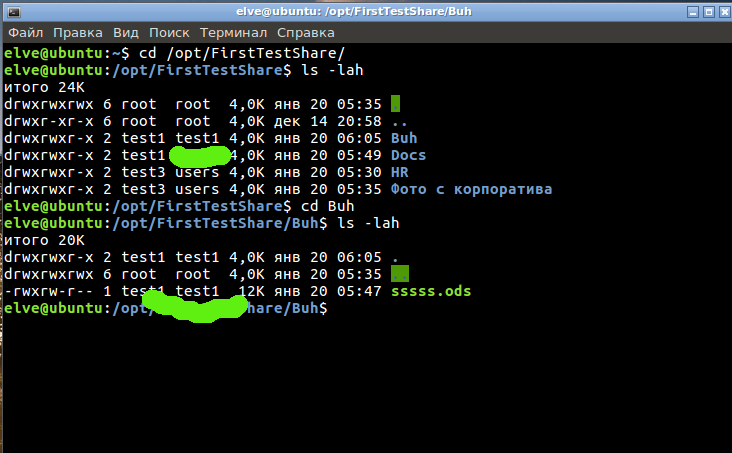
может редактировать либо он, либо кто-то из общей с ним группы. Для всех остальных только чтение.

По умолчанию при создании пользователя под него в системе создается одноименная группа.

Получается что у нас пользователь test1 в группе test1, а пользователь test2 в группе test2.

Для пользователя test1 в данной ситуации пользователь test2 относится к категории "другие".

Как это выглядит



Нам требуется групповая работа над файлами. Значит будем настраивать общую группу для

пользователей test1 и test2.

Делается это так.

1. Создаем новую группу командой:

**sudo groupadd buh**

2. Устанавливаем группу buh по умолчанию для пользователей test1 и test2

**sudo usermod -aG buh test1**

**sudo usermod -aG buh test2**

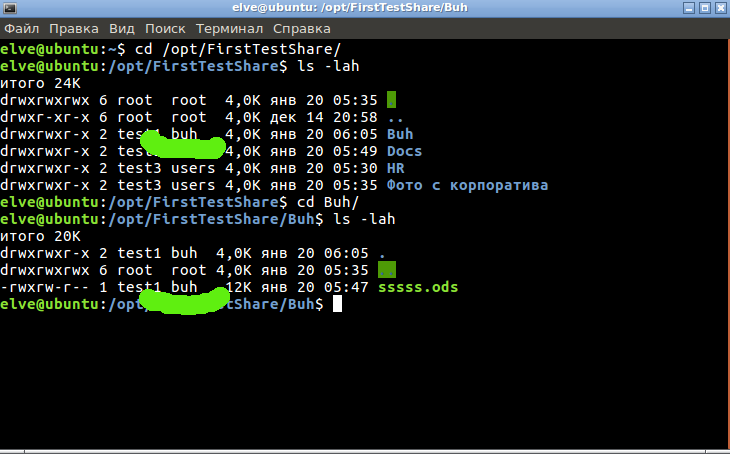
3. И перезапускаем samba, чтобы пользователи перелогинились на шару (иначе новые настройки не применятся)

**sudo service smbd restart**

4. Меняем настройки владельца для всех файлов, которые уже были созданы и для самой папки Buh

**sudo chown -R test1:buh /opt/FirstTestShare/Buh**

И вот как это выглядит теперь

Пользователь test2 теперь может редактировать файлы в папке Buh, наряду с test1. К подобной групповой работе можно подключить любого другого пользователя, добавив его в группу buh.

Не совсем. Новая ситуация.

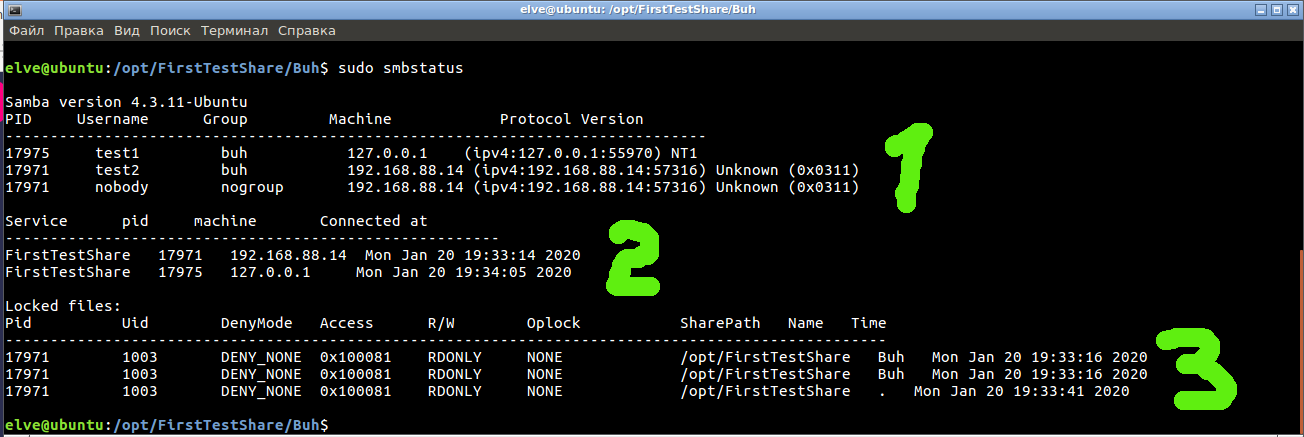
Шли дни, месяцы, недели. Пользователи test1 и test2 работали и все было прекрасно, до одного момента.

Открывает однажды пользователь test2 очень важный отчет, чтобы его отредактировать и видит уведомлениеот текстового редактора "Файл заблокирован на запись пользователем test1. Открыт в режиме только чтение". Вот это поворот!

Дело в том, что test1 любит работать с разных компьютеров. И на каком-то из них он не закрыл этот файл.

А как найти на каком именно? Тут нам поможет утилита диагностики **smbstatus.**

Давайте посмотрим как это выглядит

Вывод программы можно разделить на три части:

1. Список открытых сессий (один пользователь можешь открыть их несколько как с одного компьютера, так и с нескольких). Ключевое поле тут **PID**. Это идентификатор сессии.

2. Список открытых шар, с указанием **pid** сессии.

3. Список заблокированных на запись файлов.

Для нас будет полезно знать, что **PID сессии равнозначен Proccess ID в ОС Linux**. Запомним эту информацию. Она нам пригодится через пару мгновений

Так и какие действия мне нужно предпринять?

*Заблокированный файл в 3-м разделе будет иметь статус DENY\_WRITE в поле DenyMode.*

Смотрим PID, идем в первый раздел и видим IP-компьютера. Мы можем попросить человека закрыть файл.

Тогда мы освободим файл принудительно. Вы еще помните, что PID сессии равнозначен PID процесса в ОС?

Этим мы и воспользуемся. Находим PID залоченного файла и выполняем в консоли команду

**sudo kill -9 PID**

Тем самым мы принудительно завершили процесс, занимающий наш файл. Теперь можно его открывать и редактировать.