### **Задание 3. Резервное копирование данных с помощью rsync**

Будем копировать данные с одной машины на другую. Для этого нам понадобятся 2 операционные системы. В примере используется **ubuntu 22.04** и система контейнеризации **lxc** (управляемая **lxd**).

Уже предполагается, что **lxc** был настроен через команду **sudo lxd init**.

Создаём 2 контейнера, которые будут для нас 2-мя операционными системами. Можно использовать контейнеры, виртуальные машины или реальные устройства, соединённые в сеть. Также можно использовать любую их комбинацию.

sudo lxc launch images:ubuntu/22.04 **host-1**

sudo lxc launch images:ubuntu/22.04 **host-2**

Для lxd есть проблема. В новых версиях систем её работу блокирует сети (моста) блокирует межсетевой экран. Подробнее смотрите https://discuss.linuxcontainers.org/t/lxd-uses-iptables-legacy-even-if-iptables-nft-is-default/10837/8:   
***iptables --flush -t mangle  
systemctl reload snap.lxd.daemon***

Для входа в контейнер **host-1** используем команду:

sudo lxc exec **host-1** -- bash

Таким образом запускается оболочка терминала **bash** на хосте **host-1**. Есть 2 способа передачи данных на другую машину: через xinetd и rsync, через удалённый терминал и rsync, будем использовать второй вариант. Устанавливаем нужные пакеты: **openssh-server** и **rsync**.

apt-get install rsync openssh-server

Настроим сервер для того, чтобы он заходил только по ключу. Для этого в файле конфигурации нужно прописать PasswordAuthentication no

Можно это сделать через консольный текстовы редактор, например

vi /etc/ssh/sshd\_config

или через команду

echo "PasswordAuthentication no" >> /etc/ssh/sshd\_config

Которая добавит такую строчку в конец (в самом файле она обычно закомментированна и с yes на конце). Такой способ составления конфигураций часто применяется для автоматизации, если нет возможности просто загрузить нужный конфигурационный файл, или это не целесообразно.

Перезагрузим openssh командой. Он работает в виде сервиса, поэтому используем:

systemctl reload sshd

Создадим и настроим пользователя для логирования. Используем команду создания пользователя и явно укажем место для домашней папки:

useradd backuper --create-home --home-dir /home/backuper

Перейдём в каталог /home/backuper

cd /home/backuper

И далее создадим ключи для доступа. Создаём пользователю папку для ключей и вводим команду для создания ключа длиной 4096 бит в нужном месте.

mkdir /home/backuper/.ssh

ssh-keygen -b 4096 -f /home/backuper/.ssh/backuper

Нас попросят ввести пароль для разблокировки закрытого ключа. Вводим и запоминаем его. Для примера используется **пароль backup**.

Посмотрим на место с ключами.

ls /home/backuper/.ssh

Открытый ключ с расширением **.pub** должен находиться на сервере. Закрытый ключ – на клиенте.

У openssh-server ключи открытые ключи должны находиться в каталоге пользователя в папке .ssh с названием authorized\_keys. Отправим ключ по нужному адресу

mv /home/backuper/.ssh/backuper.pub /home/backuper/.ssh/authorized\_keys

После этого можно заходить на сервер по ssh с использованием соответствующего закрытого ключа. Данный ключ нужно доставить на клиента (обычно это делается по-другому: клиент должен сгенерировать ключи, свой открытый ключ отдать на сервер и закрытым ключом орудовать как инструментом проникновения в удалённую систему, но у нас особый путь, да и не все пользователи так умеют).

Копируем закрытый ключ любым способом. Можно через буфер обмена, можно любым другим способом. Можно воспользоваться

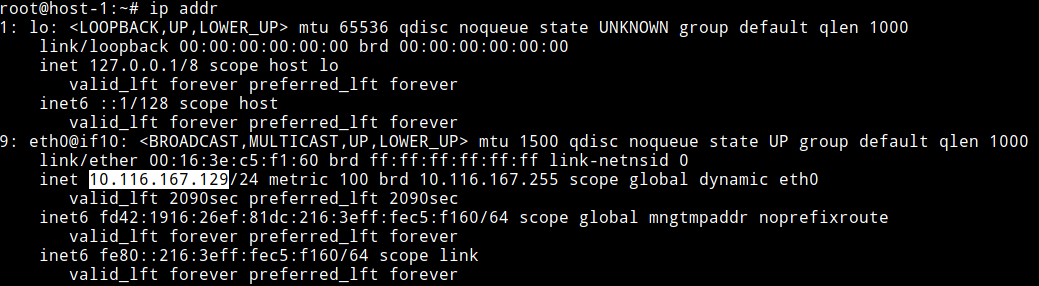
cat /home/backuper/.ssh/backuper

И скопировать из консоли через CTRL + SHIFT + C.

Напоследок узнаем адрес машины с помощью команды

ip addr

Нужен адрес, не похожий на 127.0.0.1. Для примера в одном конкретном случае адрес 10.116.167.129



Далее настроим клиент. Переходим к другому серверу (из консоли можно выйти через посылание завершающего символа CTRL + D).

Переходим к клиенту.

sudo lxc exec host-2 -- bash

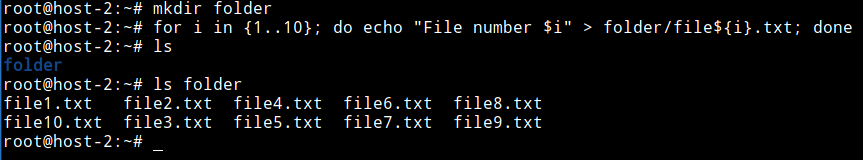
На нём обновляем список пакетов и устанавливаем rsync

apt-get update && apt-get install rsync

Создаём папку и несколько файликов

mkdir folder

for i in {1..10}; do echo "File number $i" > folder/file${i}.txt; done



В итоге эти файлы будем копировать через rsync на наш настроенный сервер.

mkdir ~/.ssh

cat > ~/.ssh/backuper

Дальше вставляем туда текст ключа. В любом случае, нужно как-то перенести ключ. Можно использовать команды pull и push от lxc (вне контейнера), можно хоть голубинной почтой файл доставить. Главное, чтобы он был.

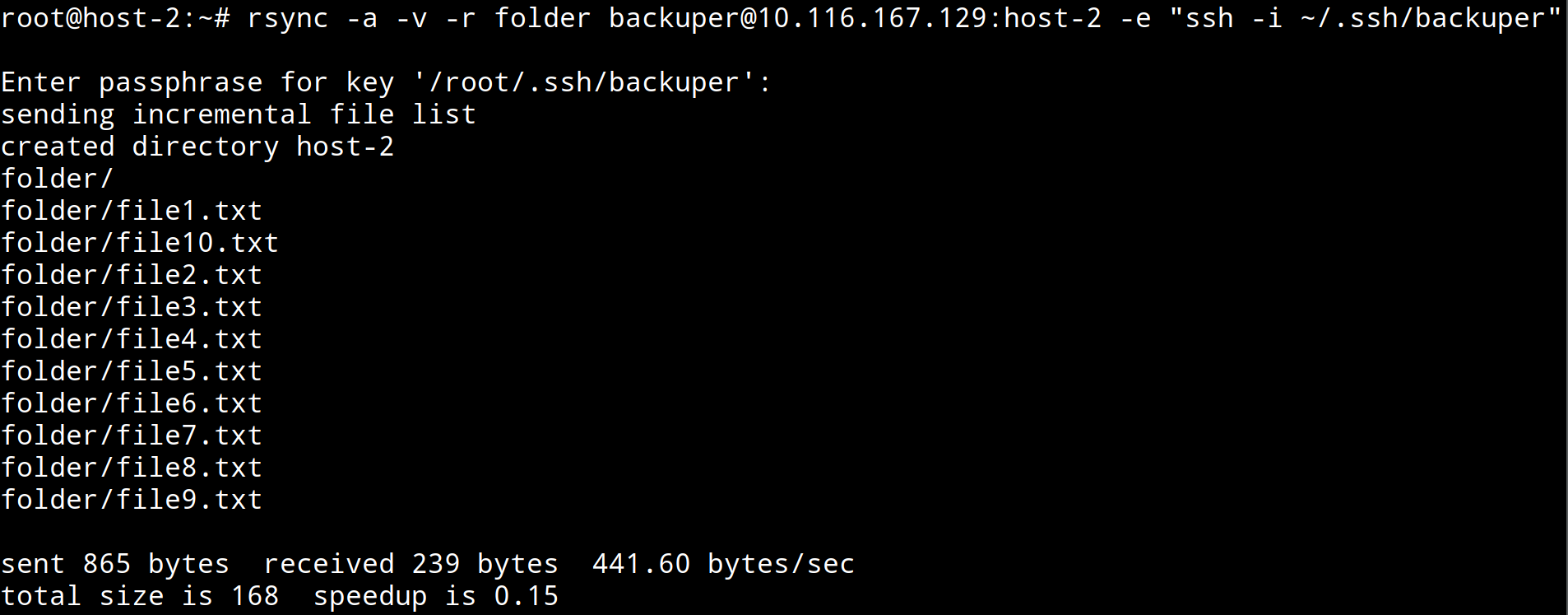
Всё, можно копировать, наверное. Единственная проблема, ключ имеет неправильные настройки доступа. Установим ограничение на файл для всех, кроме владельца:

chmod 600 ~/.ssh/backuper

rsync -a -v -r folder backuper@**10.116.167.129**:host-2 -e "ssh -i ~/.ssh/backuper"

**Адрес** в команде нужно заменить на собственный. -r означает, что копируем папку (recursive), -v означает, что выводим полонительную информацию (verbose).

Вам потребуется ввести пароль для расшифрования закрытого ключа.



Мы настраивали сервер ssh, поэтому можем зайти через него на другую машину. У программы есть возможность сразу выполнить какую-то команду и вернуться. Для этого команда указывается после.

ssh backuper@**10.116.167.129** -i ~/.ssh/backuper

Далее можно посмотрить, что лежит в папке через команду **ls**.

Папка с содержимым скопировались на другую машину. Мы указали после двоеточия host-2, поэтому скопировалось всё в папку host-2, которую создал rsync для нас.

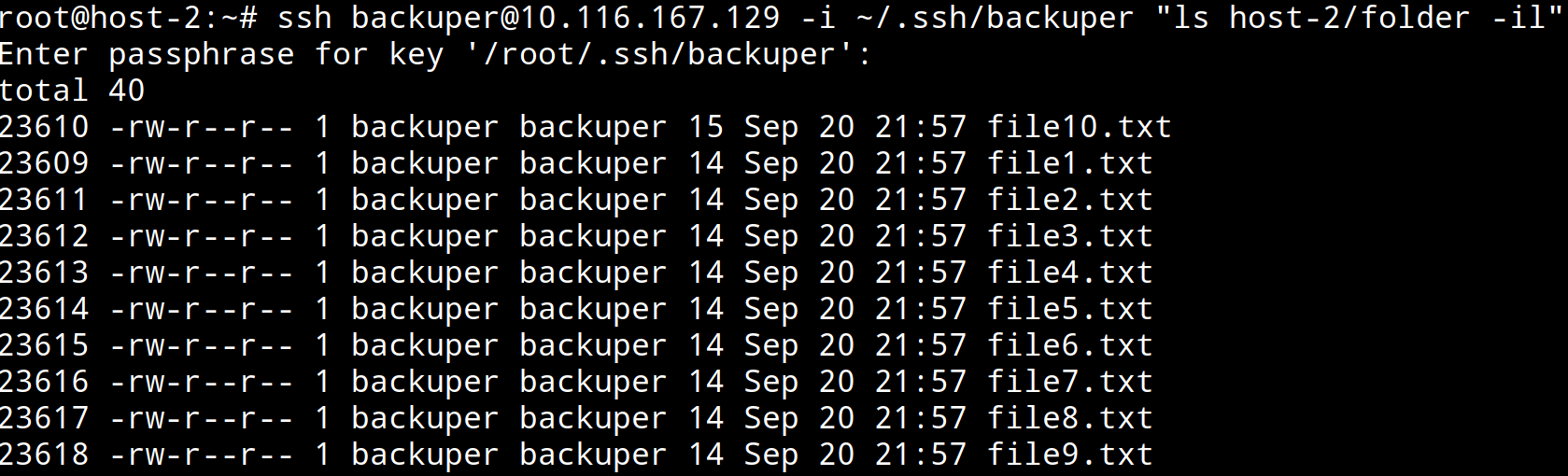
**Мы сделали полную резервную копию.**

Через CTRL+D пошлём символ конца ввода данных и выйти в свою систему.

Нам хотелось посмотреть, что произошло и нужно было выполнить только одну команду. SSH может работать и по-другому, например просто выполнить команду и вывести результат, он не остаётся в удалённой системе. Нужно просто указать команду в конце строки записи.

ssh backuper@**<ваш-IP-адрес>** -i ~/.ssh/backuper "ls host-2/folder -il"

Получим такую картинку:



#### Разностная резервная копия

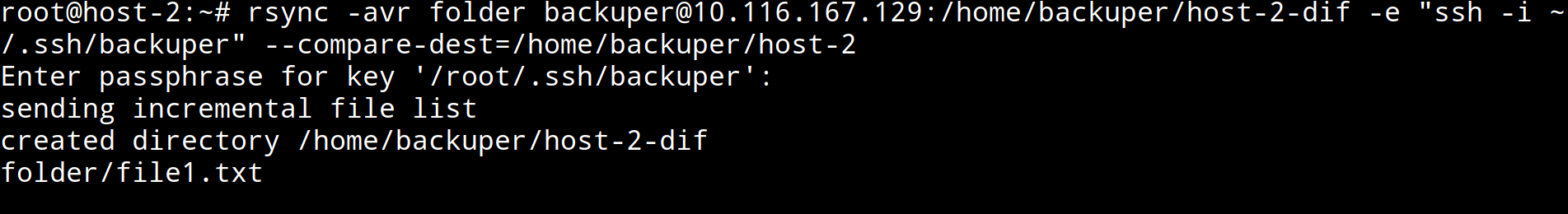
Часто хранят не только последнюю копию, а копии за последние несколько раз. Чтобы можно было восстановить данные на определённый момент времени.

Допустим, что у нас есть почтовый сервер, в котором 100 гигабайт писем и мы хотим сделать вторую копию. Скопируется ещё раз 100 гигабайт и так изо дня в день. При этом большинство файлов просто не менялось – нет смысла их копировать ещё раз. Значит, нужно скопировать только то, что оказалось новым. Для этого и существует разностное копирование.

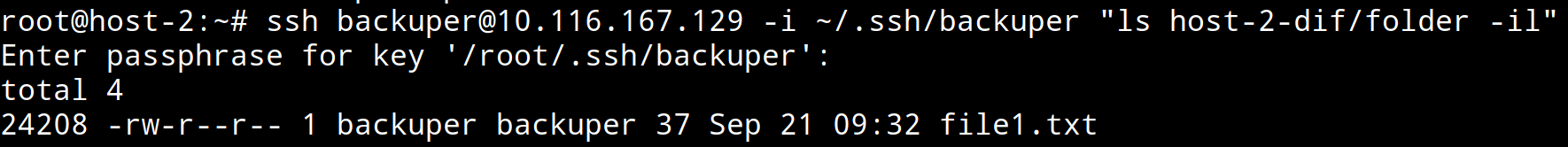
Для начала изменить в папке folder один или несколько файлов. Можете воспользоваться командой echo hello >> folder/file1.txt для добавления строки hello в конец файла file1.txt

Выполним копирование. Выполним команду:

rsync -avr folder backuper@**<ваш-IP-адрес>**:/home/backuper/host-2-dif -e "ssh -i ~/.ssh/backuper" --compare-dest=/home/backuper/host-2

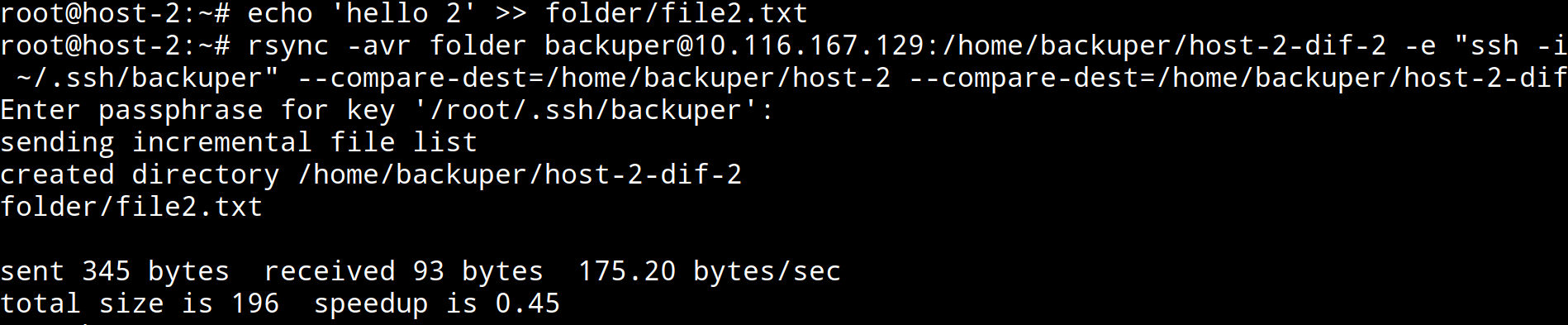


Программа написала, что отправила только один изменённый файл. Это и требуется. Проверим, что у нас получилось в папке host-2-dif, в которой хранится разностная копия. Там должна быть подпапка folder и в ней файл. Смотрим:



Действительно, получился один файл. Давайте проделаем те же самые действия ещё раз. Изменим второй файл и попробуем скопировать. Если будем сравнивать с первой папкой, то получим обычный результат - 2 изменённых файла. Но нужно же учитывать разностную копию... Да, это тоже не сложно, просто добавим ещё один ключ   
--compare-dest.

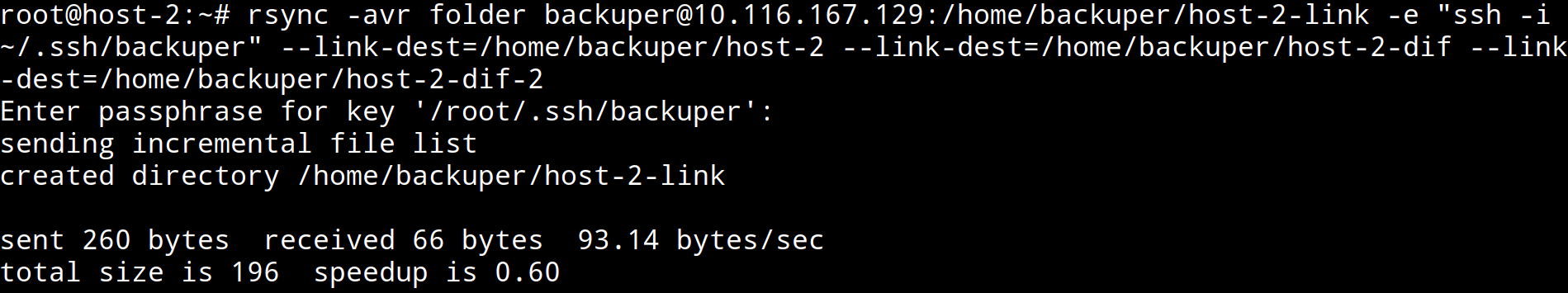
rsync -avr folder backuper@**<ваш-IP-адрес>**:/home/backuper/host-2-dif-2 -e "ssh -i ~/.ssh/backuper" --compare-dest=/home/backuper/host-2 --compare-dest=/home/backuper/host-2-dif



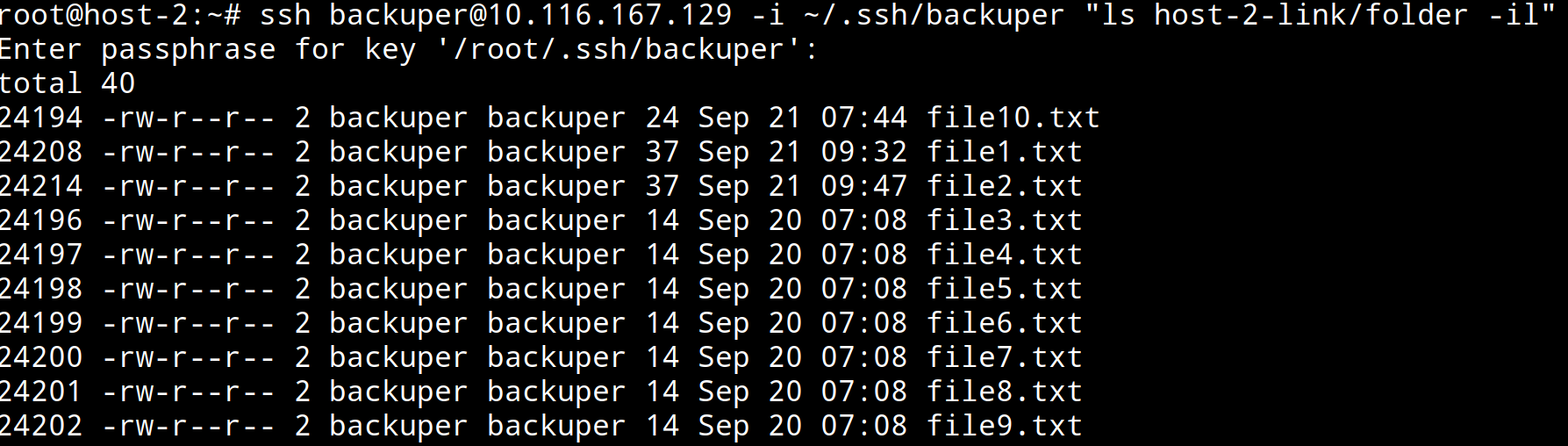
Разностная копия – это очень хорошо, но вдруг мы хотим всё восстановить? Нам нужно тогда скопировать файлы из первой папки, после из второй с разностной копией, далее из третьей и т.д. А если их штук 100? Это уже не очень весело, хотя и, на самом деле, особых проблем не доставляет.

У rsync есть интересная возможность по созданию жёстких ссылок на файлы. Таким образом копия вроде бы является полной, но при этом она разностная, то есть передаются только те файлы, которые изменялись или появлялись. Попробуем её. Вместо того, чтобы указывать папки для сравнения нужно указать папки, откуда брать ссылки. По сути это то же самое, что и было ранее.

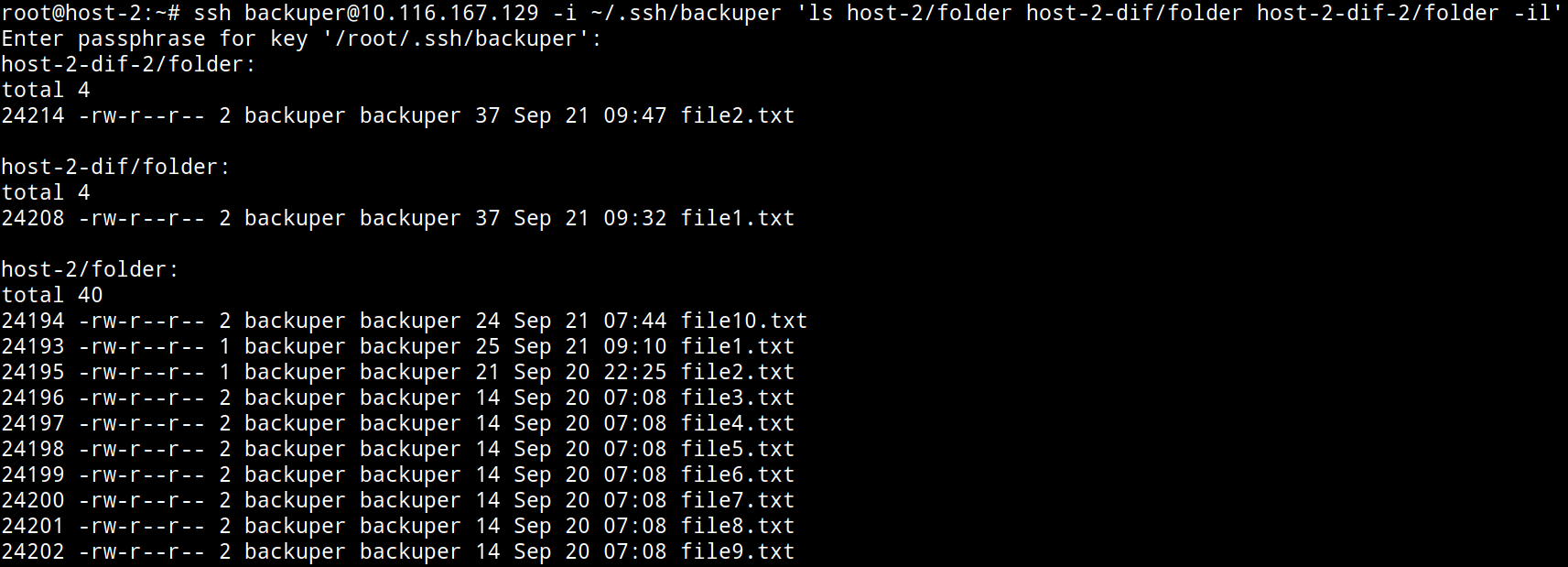
rsync -avr folder backuper@**<ваш-IP-адрес>**:/home/backuper/host-2-link -e "ssh -i ~/.ssh/backuper" --link-dest=/home/backuper/host-2 --link-dest=/home/backuper/host-2-dif --link-dest=/home/backuper/host-2-dif-2



Видно, что не написано, что передан хоть один файл. Но если посмотреть на новую папку, то увидим:



При этом можно сверить inode с другими файлами. Они совпадают, то есть это точно просто жёсткая ссылка.



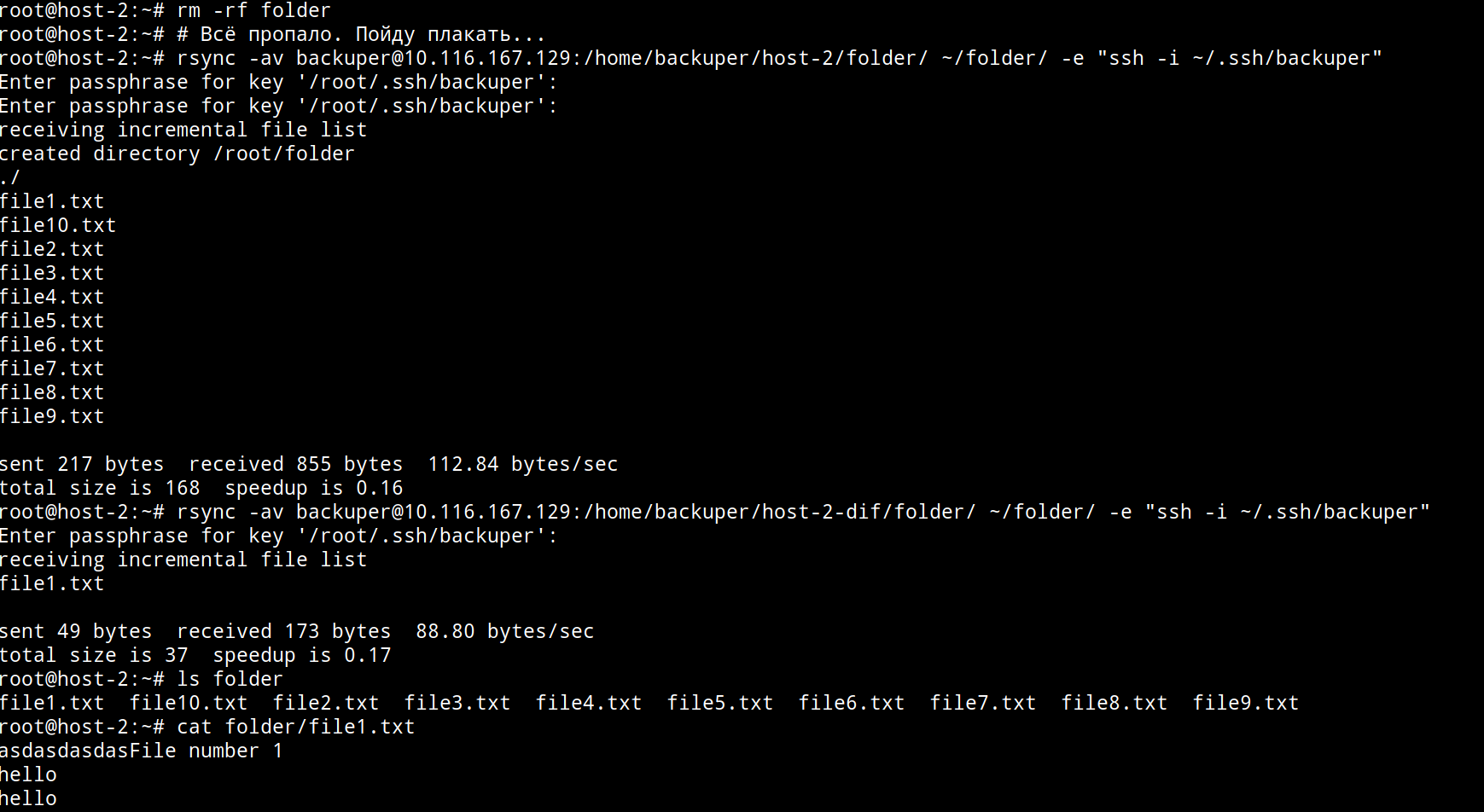
Есть дополнительная особенность – это удалённые файлы. Но это уже очень страшная особенность. Можно поддерживать актуальную папку с последней копией, но отразить сам факт удаления с помощью таких копий, к сожалению, не предоставляется возможным. Но вот при использовании жёстких ссылок такое сделать всё же можно. У вас просто не появится ссылка на файл в очередной копии. Как же это удобно.

Для восстановления системы нужно копировать всё обратно. Просто достаточно поменять адреса. Давайте восстановим вторую копию. Мы случайно выполнили команду rm -rf ~/folder и теперь места себе не находим – где же наши файлики. Они в небытие. Печально, но не время для печали – можно просто расстоиться и пойти поплакать. После этого собрав волю в кулак используем rsync для восстановления данных.

rsync -avr backuper@**<ваш-IP-адрес>**:/home/backuper/host-2/folder/ ~/folder/ -e "ssh -i ~/.ssh/backuper"

rsync -avr backuper@**<ваш-IP-адрес>**:/home/backuper/host-2-dif/folder/ ~/folder/ -e "ssh -i ~/.ssh/backuper"

**Внимание!** Слеши / на конце папок в данном случае обязательны! Программа их распознаёт так: если указана папка со слешом на конце, то нужно работать с содержимым папки, если она сама, то именно начинать с неё.



А слёз было то. Посмотрим на наши файлы внимательно. Файл первый должен быть изменённым, остальные – как были при создании.