SSH FileSystem

- there is a filesystem client based on the SSH File Transfer Protocol

- advantages
  - since most SSH servers already support this protocol it is very easy to set up: i.e. on the server side there's nothing to do
  - on the client side mounting the filesystem is as easy as logging into the server with ssh

- features:
  - implemented in userspace (although there is a kernel implementation if required)
  - multithreaded: more than one request can be on it's way to the server
  - allows large reads (max 64k)
  - caches directory contents

How to mount a filesystem

- once sshfs is installed, running it is very simple:

```
$ mkdir mountpoint
$ sshfs hostname: mountpoint
```

Example of ssh filesystem

```
$ df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda3              57G  2.7G   51G   6% /home
$ mkdir foo
$ sshfs fred@mcgreg.comp.glam.ac.uk:/home/fred/Sandpit /home/fred/foo
Password:
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1              9614116   2372668   6753076  26% /
udev                     10240        52     10188   1% /dev
devshm                  518260         0    518260   0% /dev/shm
/dev/sda3             59020704   2816436  53206140   6% /home
sshfs#fred@mcgreg.comp.glam.ac.uk:/home/fred/Sandpit
                    7999999992         0 7999999992   0% /home/fred/foo
$ cd foo
$ ls
autopassword                          MorlocSounds.tar.gz
autopassword.tar.gz                   MPlayer-1.0pre7try2
build                                 MPlayer-1.0pre7try2.tar.bz2
build-32                              oa051
```

Example of ssh filesystem

```
$ cd ..
$ fusermount -u foo
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1              9614116   2372668   6753076  26% /
udev                     10240        52     10188   1% /dev
devshm                  518260         0    518260   0% /dev/shm
/dev/sda3             59020704   2816436  53206140   6% /home
$
```

Summary of ssh

- ssh uses public/private key encryption
  - all usernames, passwords and data are encrypted
- can be used to tunnel any TCP port across the ssh port (22)
  - contents of this port is encrypted
  - can also tunnel X11 traffic across this port
  - ssh has a basic ftp client (`sftp`)
- sshfs exists and can be installed (client side on GNU/Linux, MacOS, FreeBSD)
  - no server side additions are needed

rsync

- a fast, versatile, remote (and local) file-copying tool
  - it computes the differences between the various files and only transmits these
  - thus minimising the network throughput required
  - uses the `ssh` protocol

-
```
$ rsync -t *.c foo:src/
```

- `-t` preserves time

- copies all `*.c` files onto remote machine `foo` into directory `src`

rsync

```
$ rsync -az username@remote.machine.com:/directory /localdir
```

- copy all files (recursively)
  - use archive format, preserve ownerships, symbolic links, devices
  - compress data before transmission, uncompress data after reception
- above mechanism is a very useful method of providing per user remote backups
- can install a rsync daemon (if desired)
  - easier to perform system backups or multiple user backups