90 - Network File Transfer

Single Host to Single Host secure copy

`gftp` FTP Client

Settings in SSH Server:
- `sshd` must be running
  - `/etc/ssh/sshd_config` (at the end of file)
    `Subsystem sftp /usr/lib/ssh/sftp-server`

Settings in `gftp` client:
- (Menu)FTP---->`Options`---->Tab SSH ---->(enable)`Use SSH2 SFTP subsys`
- Normal server and user entries, the port is 22, Transport type is SSH2
  no password needed for now: it is asked later

If BlaBlaBla.... (yes/no) window appears:
- type `yes` and press <Enter>

SCP - The **S**ecure remote **C**o**p**y program is also a secure program used to copy files to and from a remote host. It replaces the insecure `rcp`.
The **ssh** or **Openssh** package MUST be installed on both client and server.
The **sshd** Daemon must be started on the remote host.

The syntax is quite similar to the normal copy (`cp`) program except that the source or destination can include the user (optional) and remote hostname with the destination path. Before executing the copy `scp` will ask for the remote user's password.

Syntax:

```
scp [-options] [user@]remotehost:/sourcefile(s) /local/destination
                                        or
scp [-r] /local/source/directory [user@]remotehost:/destination/path
```

`-r` = Recursive

eg.

```
scp root@dozlinux.linux.local:/etc/*.conf /home/michel/confdata
```

Logs-in as `root` on `dozlinux.linux.local` and copies all of the `.conf` files from `/etc` directory to local `/home/michel/confdata` directory.

Single Host to Single Host secure copy/update/mirroring

rsync - The program runs just the same way as the `scp` except that it allows also to only update the remote host instead of copying the whole lot. It can use the **ssh** for secure transfer. Although there exists an `rsyncd` server daemon available for the remote host, it is not necessary for normal operation as host to host copy/update. It is necessary to have the program `rsync` in the PATH of the remote host.

Syntax:

```
rsync [-options] -e ssh /Source [user@]remotehost:/remotepath
rsync [-options] -e ssh [User@]Host:/RemoteSource /LocalDestination
```

eg.

```
rsync -vazu --delete --force -e ssh /home/ joe@dozlinux:/temp/ad
```

Updates recursively all files from my `/home/` directory to `/temp/ad/` of `dozlinux` host. The `/temp/ad/` directory MUST exist before copying.
Note: if the source directory is ended by '/' then the transferring will be recursive.

eg. `rsync -avz foo:src/bar/ /data/tmp`
A trailing slash on the source changes this behavior to transfer all files from the directory `src/bar` on the machine foo into the `/data/tmp/`.
A trailing / on a source name means: "Copy the contents of this directory".
Without a trailing slash it means: "Copy the directory".
This difference becomes particularly important when using the `--delete` option.

Options:

`-a`	Archive Mode (preserves all attributes,like `-a` in `cp`)
`-u`	Update (Do not overwrite newer files in destination)
`-v`	Verbose
`-z`	gzip compression is applied before transfering
`--delete`	Deletes all the files existing in the destination that are not present in the source.
`--force`	Forces deletion of directories even if not empty.

`--exclude` *`dir/to/exclude`*
Excludes the directories/files from being copied to destination.
Note: This `--exclude` path is relative to the source path.
eg. to prevent to copy the subdirectory `/usr/local/httpd`

```
rsync -vazu -e ssh --delete --force \
      --exclude local/httpd /usr/ remote:/tmp/
```

Will copy recursively all the files in the `/usr/` directory including subdirs except the subdirectory `/usr/`**`local/httpd`**.

Notes: 1) `--force` together with `--delete` allows for full **mirroring** of directories.
2) `--exclude` can be used more than once as argument.
3) BE CAREFUL! adding a / at the end of the source directory means the same as `/*` except that `/*` doesn't honor the `--delete` for files at this directory level.

File listing : We can get a listing of files on the remote host:
eg. `rsync -e ssh dozlinux:/etc/*`
Makes a listing of all the files in the `/etc` directory of `dozlinux` host.

Single Host to Single Host secure directory mirroring

unison

Unison is a file synchronizing program that can be used for mirroring.
Conditions to mirror over the network using ssh:

- `sshd` must be running on the remote host
- `unison` must be in the local host and in the path of the remote host

The following command formats ensures that the destination directories are exact copies of the original and that the files or directories existing in the destination directory that don't exist in the source directory are erased. The advantage of this one over `rsync` is that the timestamp is not the factor that helps deciding which files will get updated.
This provides full mirroring of 2 directories.
Syntax for mirroring directories:

From local to remote:

```
unison -ui text -batch -force localdir localdir ssh://rhost//remotedir
```

eg.

```
unison -ui text -batch -force /data /data ssh://laptop//databackup
```

From remote to local:

```
unison -ui text -batch -force ssh://rhost//remotedir \
                                 ssh://rhost//remotedir  localdir
```

eg.

```
unison -ui text -batch -force ssh://laptop//databackup \
                                ssh://laptop//databackup  /data
```

From local to local:

```
unison -ui text -batch -force localdir1 localdir1  localdir2
```

eg.

```
unison -ui text -batch -force /data /data  /databackup
```

Note: The parameter after the option `-force` MUST be the same as the source directory(next parameter) to ensure the direction of the mirroring.
As noticed in the examples, the source directory is written twice:

- once as the parameter of `-force` option to specify the source of copy.
- once as the directory of files to mirror.

Interactive Single Host to Single Host

Midnight Commander

mc (as client) and sshd(as server)

The combination of mc as client and sshd as server allows to transfer,edit,change access rights,etc interactively on remote files and directories. mc uses the fish protocol for all transactions with sshd.

Single Host to Multi Host secure copy/update/mirroring

rdist - Allows to copy/update/mirror a full diretory tree to multiple remote hosts simultaneously. This program can use the **ssh** for secure transfer.
It uses a configuration file in which list of hosts and files to transfer are defined.
The **ssh** or **Openssh** package MUST be installed on both client and server.
Although there exists an rdistd server daemon available, it is not necessary for normal operation as host to multi-host copy/update/mirroring.

---

eg. content of the config file /root/.rdist/rdistfile

```
HOSTS = ( moon.linux.local john@dozlinux.linux.local )
FILES = ( /home/public /var/log/httpd/*.conf)
${FILES} -> ${HOSTS}
install -oremove /home/pub ;
```

Command: **rdist -P /usr/bin/ssh -f /root/.rdist/rdistfile**

The result of this command with the would be to mirror all files in the local /home/public/ directory to remote hosts directory /home/pub at hosts moon.linux.local (logged in as the same user as local) and john@dozlinux.linux.local.

Options

The option remove deletes the files that are present in the remote destination but not present locally in the source.(perfect for **mirroring**)
NOTE:If no destination path is given in the install line then the destination path is the same as the local source path.
The option nodescend option can be used to turn the default recursive off.
This nodescend option doesn't work together with the remove option.
Options can be combined in the install line separeted with a comma.
eg. install -oremove,verify /tmp/pub

To find out which files would be updated but not do it (debugging), use the option: -verify
See man rdist for more info and options.

Logging of `rdist` actions:

This program allow for transfer logging as well.
The **`-l` *`logopts`*** option to `rdist` tells `rdist` what logging options to use locally
The form of *`logopts`* should be of form:

`facility=types:facility=types...`

The valid facility names are:

`stdout` Messages to standard output.

`file` Log to a file. To specify the file name, use the format `file=filename=types`. e.g. ``file=/tmp/rdist.log=all,debug''.

`syslog` Use the syslogd(8) facility.

`notify` Use the internal rdist notify facility. This facility is used in conjunction with the notify keyword in a distfile to specify what messages are mailed to the notify address.

Types should be a comma separated list of message types. Each message type specified enables that message level. This is unlike the syslog(3) system facility which uses an ascending order scheme. The following are the valid types:

`change` Things that change. This includes files that are installed or updated in some way.

`info` General information.

`notice` General info about things that change. This includes things like making directories which are needed in order to install a specific target, but which are not explicitly specified in the distfile.

`nerror` Normal errors that are not fatal.

`ferror` Fatal errors.

`warning` Warnings about errors which are not as serious as `nerror` type messages.

`debug` Debugging information.

`all` All but debug messages.

Here is a sample command line option:

```
rdist -P /usr/bin/ssh -f /root/.rdist/rdistfile \
      -l stdout=all:syslog=change,notice:file=/tmp/rdist.log=all
```

This entry will set local message logging to have all but debug messages sent to standard output, change and notice messages will be sent to `syslog`(3), and all messages will be written to the file `/tmp/rdist.log`.

Single Host to Single Host NON-secure copy
Using `netcat`

For quick and dirty file transfers which dont require a bit of security, `netcat` is your friend. Using `netcat` you perform a network `tar`:
These 2 commands will copy recursively a directory from `host2` to `host1`
host1 > `netcat -l -p <port> | tar -zxv`
host2 > `tar -zcv <directory to transfer> | netcat host1 <port> -vv`

NOTE: `Host 1` must initiate its command before the host 2 does his.
When all the files of tared directory are listed in `Host2` screen then and nothing moves any more, then host2 presses `Ctrl-C` to break the connection. The transfer is done.

<port>	Can be any unused port. Must be the same in both hosts.
<Directory>	Is the diredtory to send from `Host2` to `host1`

These 2 commands act as follows:
- `Host1`'s command starts and waits.
- `Host2` Tar recursively the full directory and shows its file list to screen.
- `netcat` receives the content of tar file and sends it to `netcat` on Host1
- `netcat` on host1 receives the tar file and sends it to `tar` to extract it.
  The transfered directory is this way saved in the current directory of host1.

H D Moore says:
This is exactly what the FTP protocol does, except you dont need to send your password across the network to do it. You also have all the features of a unix shell pipe combination. If you have a command-line encryption utility, you can use it to send your files encrypted across the network.