FTP

Servers and Services:

You can run any type of server on your machine.
That server can provide any services you want.
You server can have any means of security you want.
A server run by a user (as opposed to root) cannot provide services that exceed that user's permissions.

Example: `telnet` – Allows you to login.
Telnet server (`telnetd`) runs as root –
it can do anything it wants
Our version: requires valid user name and valid password
Limits permissions to that of the valid user
Another version could: allow root priviledge without a password.
`telnetd` – responds to anyone contacting your machine on port 23.

Example: `ftpd` – allows file transfer.
Exact security/permission depends on version you use.
Responds to anyone contacting your machine on port 21.

Our version allows anonymous ftp.

File Transfer Protocol

Allows files to be transfered between two machines.

Several versions of server available

Clients include ftp, mozilla, I.E.
If you know the protocol you can use the ftp service.

Procedure:
1) login
2) locate file (or directory)
3) transfer file

Protocol listed in `/etc/services`
Server listed in `inetd.conf`
Our machines already run `ftpd`.

A) Normal account: priviledges = user log in

B) Anonymous: public repository or archive
Allow anyone to login (no password)
Allow them to copy files out.
Allowing anybody to deposit any thing is not a good idea

Anonymous ftp issues

1) limited access to files.
What you choose should be made available, nothing else.
Your system files (`passwd`) should not be accessible
Restrict anonymous users to read-only access

Problem: `/` and other directories must be readable for normal Unix usage (or you can't get to your home, can't run commands like lpr)

2) limit commands anonymous users are allowed
Allowing someone anonymous to `cc`, `vhdl`, ... is a bad.

Problem: commands you need to allow (like `ls`) are in the same directory as commands you don't want to allow.

Solution (some versions): `chroot`–change the root directory
Normally you have access starting at `/` and going down.

You specify `chroot /home/ftp`
Unix behaves as if `/home/ftp` were the top directory.
You have access from there down.
After you `chroot`, for you that is considered `/`
For you: nothing outside this subdirectory exists
Our `ftpd`: if user logs in as `anonymous`, no password is required and the `ftpd` does `chroot ~ftp`

chroot issues

You need `/etc/passwd` for Unix to work.
`~ftp/etc/passwd` acts like `/etc/passwd`
have only the ftp entry in this version of `passwd`

You need `/bin/ls`,
Now that corresponds to `~ftp/bin/ls`

Admin task: set up a mini-Unix in `~ftp`

Overview of setting up an ftp site:
because setting up anonymous ftp is so common much of this may already be setup with by the Unix distribution, a good sys-admin should verify the setup.
1) create the pseudo-user ftp
2) Create a mini-unix area in ftp's home directory
3) Set the access permissions and ownership
4) Add files to be shared to the public directory

Note: in `proftp` the `ls` command is built into the `proftpd` server, so a separate command is not needed..

1) creating the ftp pseudo-user
a) Add ftp group to `/etc/group`
b) Add ftp user to `/etc/passwd`

Often the UID is set to a large number (so it can be easily spotted).

One of the existing groups can be used
Group and other permissions match in the mini-unix area
No setgid programs exist for this group

Security recommendation special `ftp` group

2) If needed, create a mini-unix
a) create `~ftp` with subdirectories: `~ftp/etc` `~ftp/bin`
b) create `~ftp/etc/passwd` one line (copy from `passwd`)
c) create `~ftp/etc/group` one line from `group`
d) create `~ftp/bin` and copy in `/bin/ls`
e) (?) if your `ls` uses a dynamic library,
create `~ftp/lib` and copy it in.
(`ldd /bin/ls`)
our version: built in, no `bin` required
f) create a `~ftp/pub` directories
files to be shared will reside here

3) Setting permissions and ownership.

Principle: prevent user `ftp` from changing anything.

a) `ftp` owns nothing
`chown root.wheel ~ftp`
`ls` can be set to be owned by `bin`

b) Nothing is writable
Change directories to mode 555.
no one can remove files
root can, but even root will often be asked for confirmation.
Set `ls` (and any other executables) to mode 111
Set libraries to mode 555
Linux convention, dynamic libraries are executable

4) The public directory
add subdirectories, mode 555.
add files, set mode to 444.

User controlled availability
add a directory mode 755, owned by user (not ftp).
User can maintain files.

Dynamic Libraries

Review: don't want every program to include copies of library code (for large libraries).

Statically linked: compiler embeds code into program
Advantage: runs regardless of libraries

Dynamically linked: at run time the library is found and loaded.

Static advantages:
Never get "library not found"
Never get "wrong version of library"

Dynamic advantages:
Saves disk space (critical for big window handling libraries)
Fixing library bugs does not require recompile of all programs that used the old library version.

Dynamic problem:
Two programs need different versions of the same library.

`ldd` – list the dynamically loaded libraries the program needs.

`gcc` – options to build static or dynamic linkage

ftp Config files

`/etc/ftpusers` – list of users not allowed to ftp in.
Usually `root`, `uucp`, `news`.
If `ftp` is here, anonymous ftp is disabled.

`/etc/proftpd.conf` – config specific to proftp version

inetd vs standalone start.

user/group: usually it switches to the user doing the ftp so this is rarely used.

`PersistentPasswd`–off means use the standard libraries, on means `ftpd` should access `password`, `group` and `shadow` directly. On causes NIS to be ignored.

`SystemLog`–The name of a log file; all ftp logins are logged into the file.

`TransferLog`–The name of a log file; all ftp transfers are logged into the file.

Throttle controls.

proftp.conf also provides you nested control of directories using an html type syntax.

```
<Directory /*>
  AllowOverwrite on
</Directory>
<Anonymous ~ftp>
  <Limit WRITE>
    DenyAll
  </Limit>
  <Directory deposit>
    <Limit WRITE>
      AllowAll
    </Limit>
  </Directory>
</Anonymous>
```

The above is an example.
Normal users can overwrite files.
The anonymous user will chroot to ~ftp
and cannot write anywhere (regardless of permissions)
except in the deposit directory
You can also specify the location of the welcome message and messages displayed when you enter directories.

lftp client

Can handle several protocols that are used for file transfer.
ftp, ftps, http, https, fish, sftp, file

Reliable:
non-fatal errors are ignored and the operation is repeated.

uses the REST command to restart failed transfers in the middle
will retry from the start if necessary

Shell like syntax:
you can background a transfer
you can have multiple transfers active
you have job control over the transfers

Has a recursive options for downloading/uploading directory trees

Configuration files:
/etc/ltfp.conf, ~/.lftprc, ~/.lftp/rc

sftp

Client: `sftp`
Server: `sshd` with `sftp-server`

Mechanism:
Make an ssh connection to the `sshd` server
Have the server run `sftp-server`

Advantages:
1) provides an encrypted connection for the file transfer
2) uses ssh key verification

Disadvantages:
Uses the standard account system (shell access).
You can configure ssh to do only sftp and not shell.

Note: anonymous ftp doesn't usually use encryption,
the contents of the file isn't private.