Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.033 Lab, Spring 1999 Handout 8 — The NFS Loopback Server April 9

---

1 Introduction to the loopback server

The goal of the third lab is to build a real file system. Unfortunately, real file systems often live in the kernel, which is hard to modify and debug. However, for many OSes, there are hacks that allow a file system to be implemented as an application outside the kernel. The NFS loopback server is an especially portable example of such a hack.

1.1 How it works

The idea behind an NFS loopback server is simple. In the usual NFS scenario, a machine `A` will mount a network volume exported by an NFS server running on machine `B`. In the loopback scenario, the machine `A` mounts a volume from an NFS server running on machine `A` (i.e. itself).

2 Using and understanding the NFS loopback server

2.1 How to compile the NFS loopback server

Look in `/mit/6.033/lab/src/three` and decompress the archive file into a directory of your choice. In the `sfs/upt` directory, type `./configure` and then `gmake`. On Solaris machines, you may need to add the `gnu` locker to run `gmake`. In the `sfs` directory, type `./configure` and then `gmake`.

Once you start adding your own source files, you will probably need `automake` and `autoconf`. You will have to add the `outland` locker for `automake`. If you are running Linux, you will need to make sure to have those utilities installed. You can get the latest `RPM` archive of the `automake` and `autoconf` from the 6.033 file system lab source directory. To add `.c` files for the `sfsfs` binary, edit `sfs/sfsfs/Makefile.am`. Add new `.c` files to sfsfs_SOURCES and new `.h` files to `noinst_HEADERS`. Then run `gmake`. For more information, type `C-h i` in `emacs` and look for `automake`.

To summarize, the following instructions will build the package on either the x386 or Solaris platforms after you decompress the archive:

1. `add -f gnu outland`
2. `cd upt; ./configure; gmake`
3. `cd ..; ./configure; gmake`

The `-f` option will add the lockers `gnu` and `outland` to the beginning of your PATH.

As you build this system, you will probably notice a few warnings. Currently, the NFS loopback server has been tested under Linux 2.0.36 (RedHat 5.2) and Solaris 5.6.

2.2 How to run the NFS loopback server

A few things to do before you actually run the file system:

1. Make sure that the RPC portmapper (`portmap` under Linux, `rpcbind` under Solaris) is up. You can verify that the RPC portmapper is up by running the `rpcinfo` command. Portmapper is often a lead-in for security problems. We recommend you disable `portmap` when you are not using it.
2. Create a directory to be used as the mount point for your file system. For example, you can create a directory `/tmp/sfs` or `/sfs`.
3. Create a file to be used as the disk for the file system. You can use the `mkfs` command located in the `sfs/sfsfs` directory. For example, the following command creates a 1 megabyte file system in `/tmp/disk`:

```
sfsfs/mkfs 2048 /tmp/disk
```

Once you've created a mount point and a disk for the file system, you're ready to start the NFS server and mount the file system. The `sfslaunch` program will do both operations for you. You need only specify the proper parameters to `sfslaunch`. A sample invocation (assuming that you are in the `sfs/sfslaunch` directory) is

```
./sfslaunch ../sfsfs/sfsfs /tmp/disk /tmp/sfs
```

Running `sfslaunch` will require super-user privileges, which are available on Athena by using the `su` command. You can find out the password by typing `tellme root` on Athena. Do not login as root, instead `su` to root. Please do not modify the configuration of the workstation as root.

You can now browse the `/tmp/sfs` directory. For more information about how to run the system, see the README and USING files included in the distribution.

2.3 Understanding the NFS loopback server

In its current state, this distribution implements a very rudimentary file system. The file system is capable of storing only 1 regular file, but the basic read/write I/O operations have been implemented. The end of this handout shows some sample output describing how you can use the file system in this rudimentary state.

You should need to only modify `sfs/sfsfs/sfsimpl.c` and add your own `.c` files to implement your in-memory and on-disk structures. However, as always, you are free to modify anything.

3 Sample Output

Here's some sample output from the rudimentary file system that's currently implemented in the NFS loopback server distribution. The file system was started in one shell and some simple UNIX file operations were performed in another shell. The text from the shell where the file system was started is preceded by a `>` character on each line.

```
> [root@hu-home sfslaunch]# ./sfslaunch ../sfsfs/sfsfs /tmp/disk /tmp/sfs
> Using /tmp/sfs as mount point for file system.
> sfslaunch: running ../sfsfs/sfsfs (pid 4592, port 2491)
> Using /tmp/disk as disk.
> sfsfs: Reading from disk
> Hostname: (sfs)
> NFS request 1  (uid 0, gid 0)
> sfsfs: Reading from disk

[jhu@hu-home /]$ cd /tmp/sfs
[jhu@hu-home sfs]$ ls -l
total 0
[jhu@hu-home sfs]$

> Hostname: (sfs)
> NFS request 1  (uid 0, gid 0)
> sfsfs: Reading from disk
> NFS request 4  (uid 11058, gid 100)
> sfsfs: Found one in buffer cache
> NFS request 1  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 16  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)

[jhu@hu-home sfs]$ cat > file
The quick brown fox jumped over the lazy 6.033 TAs.
^D
[jhu@hu-home sfs]$

> NFS request 4  (uid 11058, gid 100)
> NFS request 9  (uid 11058, gid 100)
> sfsfs: Found one in buffer cache
> NFS request 4  (uid 11058, gid 100)
> NFS request 8  (uid 11058, gid 100)
> sfsfs: write: 0, 52
> sfsfs: Reading from disk
> sfsfs: Found one in buffer cache

[jhu@hu-home sfs]$ ls -l
total 1
-rw-------   1 jhu      users          52 Apr  7 00:06 file
[jhu@hu-home sfs]$

> NFS request 4  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 16  (uid 11058, gid 100)
> NFS request 4  (uid 11058, gid 100)
```

```
> NFS request 1  (uid 11058, gid 100)

[jhu@hu-home sfs]$ cat file
The quick brown fox jumped over the lazy 6.033 TAs.
[jhu@hu-home sfs]$

> NFS request 4  (uid 11058, gid 100)
> NFS request 4  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 6  (uid 11058, gid 100)
> sfsfs: read: 0, 1024
> sfsfs: Found one in buffer cache

[jhu@hu-home sfs]$ cat > file2
file2: No space left on device.
[jhu@hu-home sfs]$

> NFS request 4  (uid 11058, gid 100)
> NFS request 9  (uid 11058, gid 100)
> sfsfs: File exists

[jhu@hu-home sfs]$ rm file
[jhu@hu-home sfs]$

> NFS request 4  (uid 11058, gid 100)
> NFS request 4  (uid 11058, gid 100)
> NFS request 4  (uid 11058, gid 100)
> NFS request 4  (uid 11058, gid 100)
> NFS request 10  (uid 11058, gid 100)
> sfsfs: Found one in buffer cache

[jhu@hu-home sfs]$ ls -l
total 0
[jhu@hu-home sfs]$

> sfsfs: Found one in buffer cache
> NFS request 4  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
> NFS request 16  (uid 11058, gid 100)
> NFS request 1  (uid 11058, gid 100)
```

4 Random bugs about the loopback server

If you use RH 4.2, you might need to use our `rpcgen` that is included in `sfs/rpcgen`. If `gmake` complains about "int i" after the expression "(void)buf", you will need to run `cd rpcgen; gmake`. Then replace `/usr/bin/rpcgen` with your new binary.