WWFS Reference Manual

version 109.5

Youki Kadobayashi
youki@wide.ad.jp
Department of Information and Computer Sciences
Osaka University, Japan

December 19, 1993

Contents

Chapter 1

Introduction

1.1 Description

WWFS (WorldWide File System, or Waku-Waku File System) is a filesystem which extends NFS over the Internet, while achieving simplicity, scalability, and compatibility. WWFS users can access existing FTP archives without expert knowledge of the Internet and other network applications.

WWFS is not wizard's vehicle. It eliminates burden of users and network administrators. There is no need to worry about hostname, pathname, or geographical location of the host. Once network administrator arranged "volumes" for you, it's no longer necessary to remember the details of FTP archives. Also, WWFS supports network administrators by letting them estimate, optimize or regulate file transfer.

WWFS is not technology with another dozen buzzword. Every existing UNIX command works fine, just as ordinary UNIX file systems, apart from some strange behaviors resulting from large delays in the Internet. Because it is built on top of a filesystem, many existing command works fine with little or no modification; MULE does great job with dozen Emacs lisp jewels, and NCSA MOSAIC now saves your time and network bandwidth with direct WWFS support. You can use it "off-the-shelf", without any modification to the UNIX kernel. Server and client runs on a number of BSD-derivatives, since I used widely available features only. Client workstations require minimal addition: `wwmount` and `wwumount`, which is modified versions of mount/unmount commands. With the cluster server (`csd`) running on a well-endowed, well-managed machine, everyone can share those cached information with little performance overhead.

So just try it. It's technology of today — enabling future applications to come out now.

1.2 Organization of this distribution

This release contains source code and documentation for our intermediary file-server software named Cluster Server (`csd`), client software, direct WWFS support in NCSA MOSAIC for X, API (application programming interface), API examples, some perl scripts for administration, and volume files for variety of information resources in the Internet.

Directories are organized as follows:

- `caps/`
  `perl` scripts and shell scripts. CAPS stands for Client Administration Perl Scripts.

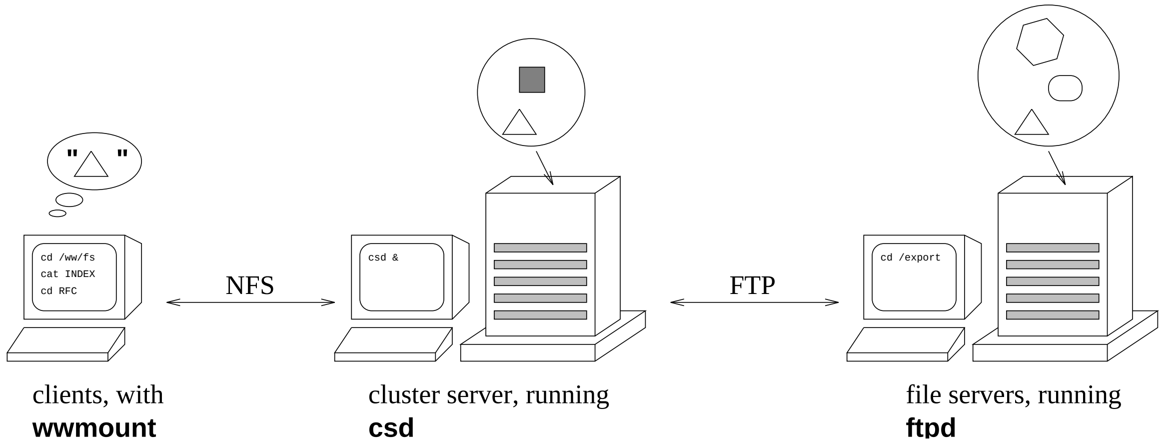

Figure 1.1: Cluster Server: An NFS server, an NFS-FTP gateway software, or the Internet file cache.

- `config/`
  Machine-, architecture- and operating-system dependent files.
- `csd/`
  An NFS server, an NFS-FTP gateway software, the Internet file cache, or the substrate for scalable information systems, that runs as a daemon in the user mode. Modification to the UNIX kernel is not necessary at all. It can be accessed as an ordinary NFS server from virtually any client workstations with NFS capability. It talks with anonymous ftp servers using FTP protocol (see Figure 1.1).
- `doc/`
  Documentation, both finished and unfinished, including memorandum and preliminary ideas. Some may be only useful to interested developers.
- `gtr/`
  A package to rearrange servers by geographical / topological region.
- `include/`
  Header files for C programs.
- `libww/`
  Application programming interface, for bypassing NFS when necessary.
- `mosaic/`
  Patches to add direct WWFS support to NCSA Mosaic for X, the multi-protocol information browser. Also included here are migration tools from Internet Gopher Protocol and HyperText Transfer Protocol.
- `mule/`
  Emacs lisp package to access WWFS within multi-lingual EMACS.
- `rpc/`
  Subroutines for remote procedure call.

- `saps/`
  SAPS stands for Server Administration Perl Scripts; some of which has been used for day-to-day server administration.

- `vol/`
  Volume files, that describe server-to-location mappings and alternative servers.

- `wwalk/`
  Point-and-click style graphical user interface, which is actually a rework of my good-old-days Tk script. You need Tk2.3, TclX6.4c, and tclRawTCP-1.0 to run this script. *It's not updated in this release.* If you don't know what Tcl and Tk is, just get the `comp.lang.tcl` FAQ from USENET newsgroups (or just go to `/news.answers/tcl-faq` on WWFS).

- `wwmount/`
  A modified version of `mount` and `umount` command, based on BSD Net2.

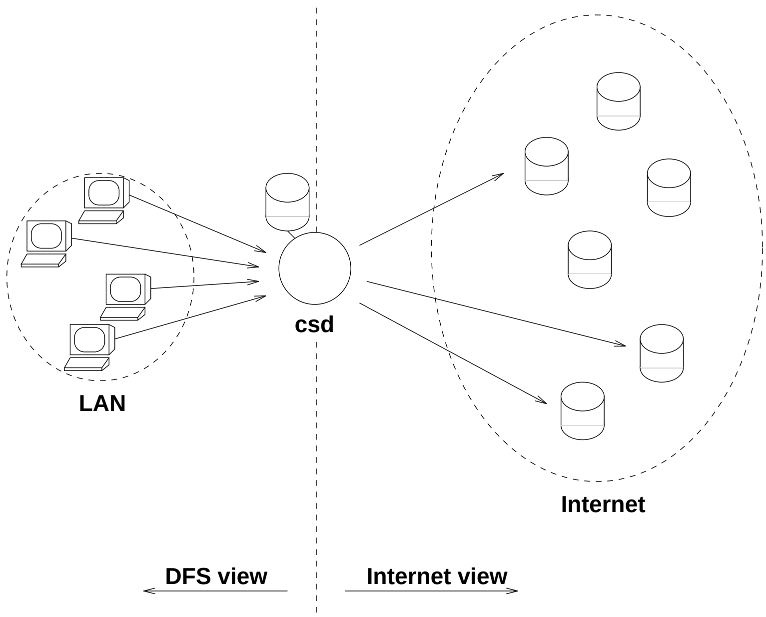

Figure 1.2: WWFS: shared namespace, replicated services, binding mechanism, and right protocol for the right place.

Chapter 2

Installation

This chapter describes installation procedure of this software suite. Before beginning installation, you must make sure that you have necessary tools handy, then decide where you install this software.

2.1 Bootstrapping environments

The following tools are necessary to generate executables and to perform administrative tasks:

name	requirements
make	bundled `make`, or GNU `make`.
cc	bundled `cc`, or GNU `cc`.
perl	version 4.0 or later.

Basically any brain-damaged `make` should work. If it doesn't work, please let me know.

If you are planning to make MOSAIC/WWFS, you will need Motif/X11 libraries. You can grab binaries from `WWFS/binaries` on WWFS and original Mosaic source from `NCSA/Web/Mosaic-source` on WWFS.

If you are planning to hack `wwalk`, a point-and-click style directory browser for the X Window System, you need Tcl and Tk. Note that this stuff is no longer maintained up to date. Bunch of Tcl stuff is available from `tcl-tk` on WWFS.

If you are willing to adapt, improve or port this software by yourself, I strongly recommend the use of GNU software suite to facilitate development. If you are planning to modify the source code, I recommend the use of additional tools which I have been using for day-to-day programming tasks:

name	description
cextract	C prototype extractor
gdb	GNU debugger
lsof	process file-table inspector
malloclib	malloc debugging library
mkid	source code indexing tool
rpcsrc-4.0	Sun RPC source

Other brand-new tools or commercial programming environments might be of great help. I am very much interested to share any programming tips with other developers (preferably via mailing list).

2.2 Planning server installation

Now you must determine where to put related material. You need sufficient amount of storage because our daemon cache accessed files. Necessary storage space varies depending on the number of users, user activity, and granularity of cached files, but you can start with relatively small space (e.g., 40Mbytes) and change the partition later, when demands exceed supply. Other extras, such as logs and maintenance scripts, do not occupy much space as compared with cache directories.

If you have a relatively large empty partition available and you are planning to use WWFS regularly, it would be better to allocate the entire partition to WWFS, since cache occasionally grows quickly when users access WWFS in a sudden burst[1]. If you are planning to adopt this option, simply `newfs` it, create mountpoint and `mount` it as usual. No special operation is necessary since the caching mechanism is built on top of UNIX file system.

WWFS requires a directory tree to store various information. Name the top directory appropriately (e.g., `/var/wwfs`) and make sure it is writable by root. It can reside either on a local disk or on an NFS-mounted disk, but I recommend the use of local disk for better reliability and performance.

2.3 Installing the Cluster Server

The `csd` included in this release is known to work on the following platforms:

hardware	operating system
Sun [34]	Sun OS 4.1.[123]
DECstation	Ultrix 4.3
Gateway 2000	BSD/386
OMRON Luna 2	BSD / Mach 2.5
SUMI STATION	RISC/os 4.x
Sony NEWS	NEWS-OS 4.[012]

Basically any 4.3BSD, 4.4BSD or OSF/1 machines is OK (well, supposed to be).

This software is not tested on other platforms. Port to other platforms will be easy however, if the target platform implements Sun NFS, the Berkeley family of filesystems (FFS or FFFS) and 4.[34]BSD system calls. I haven't made further attempt to port this software to more esotelic platforms (largely due to lack of motivation for it and familiarity with unseen beasts), but I hope we can support many more platforms without pain.

2.3.1 Edit `csd.conf` and `Makefile.config`

If you are using one of these supported platforms, skip to the next paragraph. If you are not, then read comments in the `config/Makefile.config` and edit flags accordingly. There are lots of configuration files in the `config/` directory, which has been mostly stolen from the `amd`, the 4.4 BSD automounter. If your platform has been supported by the `amd`, you can avoid re-inventing the wheel by starting there.

Next, modify `config/csd.conf` to reflect your decision in Section 2.2. Let us call it `WWFSDIR`. `csd.conf` is referenced by various utilities including make, shell scripts, perl scripts and tcl scripts.

[1]Also, WWFS has had a strange bug which filled up entire partition, although it was known to be harmless and is now exterminated.

2.3.2 Just type `make`

At the top directory of the distribution, just type `make`. It automagically guesses your architecture and operating system.

`make` will start compilation, and eventually generate the server program (`csd`) and client commands (`wwmount` and `wwumount`), among other things. You can ignore a lot of warnings during compilation procedure, especially if your compiler conform to the ANSI standard.

NOTE: It's not desirable to hand-edit C files generated by `rpcgen`. If you have encountered any difficulty with compiling `*_prot_*.c` files, you must regenerate these files using `rpcgen`. To do this, step down to `rpc/` then just type `make`.

2.3.3 Edit `/etc/passwd`

If this is the first time for your server installation, you must create a local account with user ID `wwfs` and group ID `operator` (or its counterpart in your system), at the machine designated as your cluster server, since `csd` operates under this account, and since all files and directories beneath `WWFSDIR` will be owned by the account. Choice of user ID and group ID is arbitrary, but you must choose IDs carefully so that other filesystems can be protected from unexpected behaviour of `csd` (which is very unlikely though, but serious admins wants firewall). An excerpt from our password database follows.

```
wwfs:*:40:5:WWFS manager:/var/wwfs:/bin/csh
```

Since `csd` uses `wwfs@name.of.this.machine` for e-mail address, this account should be reachable by e-mail and every received message should be forwarded to the administrator. The e-mail address has been used for statistics reports and password (contact address, that is) for anonymous ftp login.

2.3.4 `make install-server`

As root, go to the top directory of the distribution, then type:

```
make install-server
```

This will install the `csd` and shell/perl scripts appropriately, creating the following directories beneath `WWFSDIR`:

- `bin/` — executables are installed here.
- `cache/` — cached information is stored here.
- `etc/` — configuration files for perl scripts.
- `lib/` — for library files used by perl scripts.
- `log/` — logs and event traces go here.
- `stale/` — stale files and directories are moved here.
- `ongoing/` — all interim data, including files, are stored here.
- `vol/` — volume files are stored here.

Now you finished server installation procedure. Please read Sections 3.1 and 3.4 for starting up the Cluster Server.

2.4 Installing mount commands

Now you compile `wwmount` and `wwumount` for use on your workstation clusters. Or you can grab binaries from `WWFS/binaries` on WWFS.

The mount commands is known to work on BSD/386 1, SunOS 4, Ultrix 4, OSF/1 1.3, IRIX 4, HP-UX 8, NEWS-OS 4, RISC/os 4, NeXT OS, and Luna family of operating systems. Bascially any 4.3BSD, 4.4BSD or OSF/1 machines should be OK (say VAX or Tahoe which I don't have access to). Also, it is relatively easy to port the client software to non-BSD platforms. I would like to know if it runs on Solaris 2 and NEWS-OS 6; please let me know if it runs on one of these (and other exotic operating systems :-).

NOTE: BSD/386 and possibly other operating systems with 'Guelph NFS' may suffer from the symptom that previously uncached directories remain empty even when FTP transfer completes and directories get filled. I don't have any smart idea on how to fix it except modifying the BSD kernel. I hope 4.4BSD have fixed this bug.

NOTE: Luna2/Mach users must steal `mount.o` from Luna/UniOS-B `libc.a`.

You must modify `config/wwfs.conf` accordingly. It is referenced by various utilities including shell scripts, perl scripts and tcl scripts.

You may need to edit `config/Makefile.config` and correctly set the following flags.

name	description
`ETCDIR`	`wwmount` and `wwumount` goes here
`LIBDIR`	`libww.a` goes here
`INCDIR`	`libww.h` goes here

Just type `make install-client` then it will install `/etc/wwfs.conf`, mount commands, libraries and header files.

Chapter 3

Administration

This chapter is written for the administrator of the Cluster Server (`csd`). Please read Sections 3.1 and 3.4 for starting up `csd`.

3.1 Server administration

The Cluster Server, `csd`, can be restarted anytime. Basically, there is no loss of information due to termination, since cached information is kept on the disk. You should be careful however, since there might be ongoing file transfer, and there might be some users accessing WWFS.

Starting server process

To start the server process, become superuser and just type `csd-nanny` to start `csd`. (presumably installed at `WWFSDIR/bin`).

NOTE: WWFS provides non-stop operation with `csd-nanny`, an iterative and automatic debugger front-end specifically written for non-stop programs such as `csd`. `csd-nanny` is included in `saps/` directory of this distribution. For more information on `csd-nanny`, read the source, which is actually a small perl script and is well-documented.

Terminating server process

`csd` can be killed just as other daemon processes. You can use either `SIGINT`, `SIGTERM` or `SIGKILL`. Termination is logged as such, using `syslog` facility. `csd-nanny` detects termination and automatically restarts `csd`.

If you really want to stop `csd`, first kill `csd-nanny` then `csd`. But beware: if someone was accessing WWFS when you stopped the daemon, his shell (or other accessing processes, say Emacs) will become frozen. This is normal NFS behaviour and not a bug in WWFS.

Managing cache

Currently, `csd` does not reclaim secondary storage by itself. The shell script `expire.sh`, which has been included in this distribution, should be invoked regularly via `crontab` (typically once a week). It expires "inactive" files that haven't been accessed for more than two weeks, so that cache partition will not be filled up with "no longer referenced" files. Directories do not expire, but obsolete directory listings are automatically discarded and superseded by newer ones.

NOTE: It's okay to delete cached files selectively, but it's not recommended to delete cached directory files (i.e., files within `dir/` directory) unless you want to remove cached information for the whole volume. Even if all cached directory files were lost, `csd` continues to function but `csd` will occasionally forget which file is cached and which is not.

Other management tools, such as file system consistency checker, will be available in future versions.

If `csd` was accidentally killed by someone, even with `SIGKILL`, the chances are that user's ongoing file transfers weren't lost; just take a look at the `ongoing` subdirectory of `WWFSDIR`, where interim files are kept. Significant amount of effort have been paid so that daemon can survive virtually eternally, and can be restarted anytime without loss of data.

Snooping cache with perl

I have written `libmm.pl`, the Maru-Maru library (counterpart of Waku-Waku in Japanese) in the hope that perl scripts built upon this package complements features not implemented in `csd`, thereby enabling much more advanced features such as cache strategy selection, cache statistics and cache consistency management. I have also added UIP commands that enables close collaboration of `csd` and these perl programs.

NOTE: This package works only on localhost (that is, the machine running csd) for security reasons.

Several sample perl scripts are provided in the `libww/` subdirectory of this distribution; I hope these examples facilitate writing perl scripts for `csd` administration.

3.2 Client administration

`wwclient`, a perl script, is included in this distribution to facilitate the administration of clients. It should be already present in `/etc` if you have done `install-client`.

If you want a client machine to automatically mount WWFS, you can do it with `wwclient`. To do this, add the following lines to `/etc/rc.local`:

```
# WWFS
if [ -f /etc/wwclient -a -f /etc/wwfs.conf ]; then
        /etc/wwclient
fi
```

This does not work if `csd` does not respond to client's request for some reason (e.g., temporarily terminated for reboot).

Sites using `acct` should beware so that it does not traverse beneath WWFS mount point. Some sites report that `diskusg`, the disk accounting command of System V, makes `csd` crash.

Also, `/usr/adm/daily`, `/usr/spool/cron/crontabs/root` and other periodically kicked maintenance scripts should be checked so that `find` do not traverse beneath WWFS mount point. Traversal should be limited to local disks with appropriate `find` options such as `-xdev` and/or `-fstype nfs -prune`.

3.3 Troubleshooting `csd`

Reporting bugs

Although it is very unlikely, if your daemon runs out of CPU or memory, please be prepared to send bug reports to me. I would like to particularly know:

- if it is reproducible.
- peculiarities of your network configuration.
- peculiarities of your administrative tools.
- debug logs produced by `csd-nanny`.

If you could get snapshots of debug session, then please e-mail them to me. I am very grateful to receive your bug report.

Other troubles: send e-mail

If you have faced any other problem concerning the server software, please send e-mail to me (or the WWFS mailing list). If you have any problem with the client software, please make sure to read chapter 4 before asking any question to me, since it might be the inevitable oddities of global filesystems.

3.4 Volume administration

I have introduced the notion of *volume* that is a group of files with a specific purpose and organized within a subtree, since every anonymous ftp server contains tons of files with different purpose and since it is difficult to locate where those files have been categorized, while it is relatively easy to find desired files within the directory subtree.

Volume makes it easy to find necessary information, since volumes have been indexed with keywords and no extra information is necessary to access them.

The separation of `hostname:directory` and archived files yields several benefits. An archive administrator can provide alternative access point for the same files, so that alternative server is available when the original server become unavailable for some reason, such as network troubles and disk crashes. In other words, the task of locating file server, exploring directory trees, and switching between file servers become automatic, once the administrator creates volumes.

Traditionally, these tasks have been done by the people called “ftp hacker”. Now that we have WWFS, their efforts can be publicized in the form of *volume file*, that will benefit every users of the filesystem. If you are not ftp type and you are not willing to administer volumes, ask your friendly ftp hackers to do the task instead.

The beautiful point of volume files is that they can be shared among Internet sites. However, their uncontrolled distribution will lead to totally chaotic situation, which is no better than current ftp archives. Thus coordination by authoritative organization should be necessary in near future.

Installing volumes

Volume files must be installed properly to enable access to volumes. You must hold write permission on the volume directory, which is normally `WWFSDIR/vol`.

Creating new volumes

If you or your clients want to access other ftp directories not listed in existing volumes, you can create a new volume for that. You can write a volume file with your favorite text editor. The syntax of volume file can be described as follows:

file	::= line | line file
line	::= key: value
key	::= { “volume-id”, “ftp-server”, “ftp-directory” }

Although `map.pl` tries to assign unique volume ID for each volume file, it cannot make sure that given volume ID is unique. `map.pl` maintains the next free volume id in the `seq` file, which is located at `WWFSDIR/vol`. However, if the volume id conflicts for some reason, `csd` may behave incorrectly. You must guarantee the uniqueness of volume IDs. The most viable solution is probably that you learn `perl` and troubleshoot your problem yourself.

It is desirable to list all alternative servers that provide the same volume. If there are alternative servers, `csd` will switch to secondary server if the primary server became unavailable due to network problems or hardware problems. An example, `X-contrib.vol`, is shown below, which enumerates alternative servers.

```
volume-id:      34
ftp-server:     ftp.center.osaka-u.ac.jp
ftp-directory:  X-contrib
ftp-server:     ftp.iij.ad.jp
ftp-directory:  pub/X/contrib

ftp-server:     ftp.x.org
ftp-directory:  contrib
ftp-server:     ftp.germany.eu.net
ftp-directory:  pub/X11/contrib
ftp-server:     nic.switch.ch
ftp-directory:  mirror/X-contrib

description:    Contributed software for X Window System
#category-youki:        free/gui
```

The priority of access is determined by the order of listing; the topmost one becomes primary server.

NOTE: “ftp-server:” can be either hostname or IP address, so that `csd` can run on corporate networks without DNS.

Modifying volumes: `make reload`

You can accommodate volume files to your network configuration by sorting server information in volumes files so that file transfer can be optimized; response time can be improved, load can be reduced, and bandwidth can be saved.

This release comes with GTR, which automates selection of nearest server by GTR configuration file (`WWFSDIR/etc/gtr.conf`) and GTR resource records stored in DNS. As this server selection scheme is not standardized, GTR resource records can be stored in a flat file (`WWFSDIR/etc/gtr.txt`). Please look at `gtr/` directory for more detailed information on GTR.

To install modified volumes, reload them using `WWFSDIR/Makefile`. Volumes can be reloaded while `csd` is running.

Deleting volumes: `make unload`

Volumes can be deleted, either because they are reorganized into other volumes or because they became obsolete and no longer necessary. You can delete the volume files in `WWFSDIR/vol`. Use `WWFSDIR/Makefile` to reflect the removal.

Organizing volumes

You can organize volumes by creating subdirectories under WWFSDIR/vol and moving appropriate[1] volumes into that directory. You must think twice before doing this, since we do not provide any smart scheme to pick up desired volumes from hundreds of volumes that are organized in subdirectories. The number of volumes does not change no matter how you organize it, and after all, how good is subdirectories that have hundreds of volumes in each?

We are merely escaping from the namespace explosion problem; but at least we learned one thing: flat namespace with carefully annotated INDEX does help.

If you have MOSAIC/WWFS running, try looking at `wwfs:/WWFS/web/WWFShome.html`.

Generating INDEX: `make index`

Since volumes are initially invisible, users must be informed of their availability. The list of available volumes can be generated from volume files using `makelist.pl`. To do this, simply type `make index` in `WWFSDIR`.

Automating administration with Makefile

Some of procedural tasks tend to be commonly performed by administrators. Such tasks can be facilitated by preparing a `Makefile`. Sample `Makefile` have been provided for these purpose; they are copied to `WWFSDIR` upon correct installation. You are encouraged to tweak it and create `perl` scripts for smarter administration.

3.5 Access control and security

For some reason, you may want to make the filesystem visible to certain subnets but keep it inaccessible from the rest of the world. You can do this with access control file named `access`. Access control files can be placed in arbitrary subdirectory of WWFSDIR/vol; in other words, if you want to make some sensitive information accessible as volumes, make a subdirectory under WWFSDIR/vol, move sensitive volumes there, and put `access` there. I do not go into detail about its grammer; here is an example `access` file.

```
163.221.32-163.221.68        nfs,uip    # address range
163.221.80.*-163.221.96.*    nfs,uip    # same effect
163.221.74.25                nfs,uip    # specific host
163.221.78.*                 nfs,uip    # 0 to 255
```

[1] The real question is that what's appropriate.

Basically it's combination of *address range* and *capability*. Host names and domain names are not allowed here, since there are only tricky and non-standard way to retrieve subnet masks and subnet addresses via DNS.

NOTE: You may want to ensure security at your firewall gateways (i.e., Internet routers at the outermost segment) by letting them drop packets of UDP port 8000 (csd NFS), UDP and TCP port 8002 (csd UIP).

Chapter 4

Using WWFS

This chapter is intended for casual UNIX users who can understand what shell is and what `ls` is. Even if you are wizard, you are recommended to glance through this chapter at least once. Also, you may want to ask your local WWFS administrator for local configuration information and/or local guide.

4.1 `wwclient`: starting to be a WWFS client

If you or your system administrator has already mounted WWFS, you can skip this section. You can see if WWFS have been mounted with the `mount` command:

```
youki-k@dec416$ /etc/mount | grep wwfs
fsa2:wwfs on /ww/fs type nfs (ro,soft)
```

If it's not mounted, then `grep` will display nothing. In that case, you can mount WWFS by yourself (if you don't have superuser privilege, ask your system administrator to do so). There should be a perl script named `wwclient` somewhere in your filesystem (presumably `/etc`, `/usr/local/etc` or `/usr/local/ww/etc`). You can mount WWFS by invoking `wwclient` as root without any arguments. If you can't find it, ask your local WWFS administrator for help.

4.2 Stopping to be a WWFS client

If you decided to stop using WWFS for some reason, you don't have to unmount the filesystem. However, if your WWFS administrator asked you to unmount the filesystem, you would better do that. To unmount the filesystem, simply give the mount point as its argument. An example follows.

```
dec416# wwumount /ww/fs
```

4.3 Accessing volumes

Volumes are represented as ordinary directories, but it is invisible at first time. A volume is made visible only after it has been accessed explicitly, since users will get bewildered if thousands of volumes have been visible at once.

```
youki-k@dec416$ ls /ww/fs
CERN-web/  CTAN/      GNU/       INDEX      WWFS/      www/
```

4.4 Accessing directories

Directories in WWFS can be accessed just like other filesystems using your favorite tools, with one noticeable difference: you must sometimes retry. It will take longer when `csd` establish FTP connection with anonymous ftp server. You must be patient for this, since delays are inevitable in the Internet and delays increase as the logical distance between client and ftp server increase.

NOTE: If you feel it really irritating to type `ls` repeatedly, you should plan to re-compile `ls` with `libww`. See Chapter 5 for more detail.

4.5 Accessing files

Files suffer the same problem with directories. Generally, you cannot expect a file transfer to complete real soon now unless you have enough bandwidth available for you. Therefore, the initial access to the designated file is likely to fail especially under poorly connected networks. After certain amount of time, access to the same file will succeed since the file has been cached on the local filesystem.

Failure to instantly deliver the requested file will result in an error "no such file or directory" (`NFSERR_NOENT`), so that `csd` deliberately avoids your shell being blocked (i.e., become irresponsible) almost eternally by filesystem request. This was a design choice because of poor semantics provided by NFS.

NOTE: You can virtually forget the problem by re-compiling your favorite programs with `libww`. See Chapter 5 for more detail.

4.6 Other operations

Write operation is not supported for now, since WWFS is designed for widely shared information, which is rarely modified, according to statistical studies.

Chapter 5

Useful hacks

Due to strong demand from the user community, I devised several ways to bypass limitations of NFS with minimal effort. This chapter is intended for aggressive WWFS users who want to accommodate existing tools to WWFS. The basic idea is that you have WWFS-ready command sets, say `emacs`, `ls`, `tar`, `cat`, and so on.

5.1 MULE: `wwfs-access-hook.el`

There is `mule/` subdirectory in this distribution. For more information on WWFS-ready `mule`, follow the README in that directory which I have written for `mule` addicts.

So what is MULE? MULE stands for Multilingual enhancement to GNU Emacs. For those who cares, I do not support the US-centric version of Emacs (because it's mostly useless to me). But I think it's easy to add `access-hook` to Emacs 19 as well.

5.2 Recompiling other utilities

Most of GNU software can be compiled WWFS-ready and installed without modifying Makefiles; use `libww/wwconfigure` in this distribution instead of the bundled `configure`.

Basically, it should be fairly easy to make existing commands WWFS-ready, as long as you have source code of them. Just redefine `open` to `ww_open`, `opendir` to `ww_opendir`, `fopen` to `ww_fopen` by adding C flags like this:

```
CFLAGS= -Dopen=ww_open -Dopendir=ww_opendir -Dfopen=ww_fopen
```

Next, make them link with `libww.a` by adding flags for `ld` like this:

```
LIBS= -L/usr/local/ww/lib -lww
```

5.3 Splicing the C library

You can rename arbitrary symbols in `libc.a` with `renamesym.pl`, my brute-force perl script, which may be useful when you have object files but not source files (e.g., binary licensed software). You can generate WWFS-ready C library, `libc_wwfs.a`, with this perl script; to do this, just step down

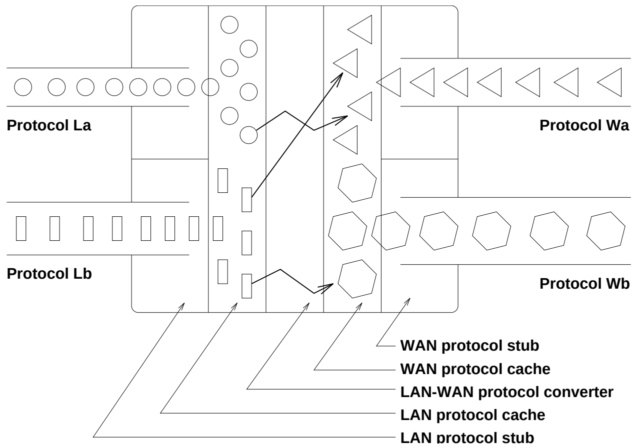

Figure 5.1: For those who cares: yes, `csd` is a 1) protocol conversion engine for information access, 2) shared file cache for the Internet, and 3) an information delivery mechanism.

to `libww/` subdirectory and type `make libc`. Note that this perl script assumes BSD-style nm output format (that is, it does not work on IRIX, HP-UX and other SysV variants for now).

5.4 Monitoring server status

There are several ways to monitor the server status. You can use traditional tools such as `netstat`, `ps` and `pstat`. Both `ps` and `pstat` are useful to see if `csd` is occupying too much memory or too much CPU time.

`netstat` may be useful to see how many file transfer have been taking place. But there is much smarter tool for this purpose, called `lsof`, developed at Purdue University. Snapshot of a sample `lsof` session follows:

```
youki@raicho$ /usr/local/etc/lsof -u wwfs -s
COMMAND     PID     USER   FD  TYPE     DEVICE   INODE/NAME
csd        1887     wwfs  cwd  VDIR      7,   6   53658 /usr (/dev/sd0g)
csd        1887     wwfs    0  VCHR     20,   0    2864 /dev/ttyp0
csd        1887     wwfs    1  VCHR     20,   0    2864 /dev/ttyp0
csd        1887     wwfs    2  VCHR     20,   0    2864 /dev/ttyp0
csd        1887     wwfs    4  inet 0xff66090c      UDP *:791
csd        1887     wwfs    6  inet 0xff651a8c      TCP *:8002
csd        1887     wwfs    7  inet 0xff65158c      UDP *:2272
csd        1887     wwfs    8  inet 0xff660e0c      UDP *:8000
csd        1887     wwfs   10  VREG      7,  11   20542 /space (/dev/sd1d)
csd        1887     wwfs   11  VREG      7,  11   20485 /space (/dev/sd1d)
csd        1887     wwfs   12  inet 0xff66450c      TCP 133.1.84.1:3681
                                                        ->192.41.197.14:21
```

There is an experimental feature called “debug connection” in `csd`. You can talk to csd directly with `telnet` using TCP port 8002. A new protocol, UIP (WWFS User Interface Protocol) has been introduced for the administrative purpose, among others. A sample session is shown below.

```
youki@nozomi$ telnet raicho 8002
Trying 133.1.84.1...
Connected to raicho.rd.ecip.osaka-u.ac.jp.
Escape character is '^]'.
DIAG file /tahoe/bin/cat/cat.1
220-009 fil /tahoe/bin/cat/cat.1
213-009 name = cat.1
213-009 link = (null)
213-009 id = 1e5a2ca7
213-009 crc = 1e5a2ca7
213-009 flag = 0
213-009 type = 1
213-009 fsid = d
213.009 fileid = 1e5a2ca7
QUIT
Connection closed by foreign host.
```

The draft document for UIP protocol specification has been included in this distribution. It doesn't document all available features however, and may not reflect latest changes to the protocol. The corresponding source code, `csd/uip.c`, might be more informative.

The `telnet` debug connection can never be a security hole, but if the connection peer used privileged port (i.e., `sin_port` of the connection peer is smaller than 1024), `csd` logs the event for later perusal.

```
raicho csd[1887]: privileged connection established: 133.1.84.13
raicho csd[1887]: privileged connection closed: 133.1.84.13
```

5.5 Mounting WWFS using `amd`

It's possible to mount WWFS using `amd`'s program file system. One possible pitfall is that `amd` will fall into deadlock if you misconfigured it. The rule of thumb is that *do not put wwmount and wwumount under filesystems managed by amd*. The deadlock occurs as follows:

1. `amd` executes `wwmount` and block until it completes.
2. exec routine in the kernel attempts to read `wwmount` binary from its directory, say `/usr/local`, which in turn is served by `amd`.
3. `amd` deadlocks.

Now you know how to avoid `amd` deadlock, so write your own map file. Here is our working example:

```
world   type:=program;\
        mount:="/etc/wwmount wwmount raicho ${fs}";\
        unmount:="/etc/wwumount wwumount ${fs}";
```

Chapter 6

Contribution

6.1 Sending comments, bug reports and improvements

WWFS is a still-ongoing research project. I think this software suite can become a building block for large scale networked environments in near future. You can contribute to our research and development by sending comments, bug reports, and ideas for improvements; I am very grateful to receive your feedback.

Please feel free to send your feedbacks to me, or to the following e-mail address:

`wwfs@is.aist-nara.ac.jp`

If you want to join the WWFS bandwagon, just send an e-mail to the following address, so that you will be added to the WWFS mailing list:

`wwfs-request@is.aist-nara.ac.jp`

Although we adopt English as our primary communication language, we accept Japanese as well, since this research project and this mailing list has been wheeled by Japanese folks, including me.

6.2 Acknowledgements

I would especially like to thank Noritoshi Demizu for his useful perl hacks, impressive jokes and his serious attitude toward layer violation. Also, I would like to thank Akira Kato, Hiroyuki Ohno, Kazumasa Utashiro, Kouichirou Eto, Motonori Nakamura, Shuji Ishii, Toshihiko Shimokawa, Yoshitaka Tokugawa and other alpha testers for their helpful comments, bug reports and discussions on the WWFS mailing list and private talks. I would like to thank folks at Osaka University, Nara Institute of Science and Technology, and WIDE Project, especially Dr. Suguru Yamaguchi, Prof. Hideo Miyahara and Prof. Jun Murai for their fundamental support — including environmental, financial, and spiritual support — for my research activities. I would also like to thank the administrators of ftp archives in the Internet.