Installing the CCSO Nameserver

by
Steven Dorner   s–dorner@uiuc.edu
Computer and Communications Services Office
University of Illinois at Urbana

March 4, 1992

updated by
Paul Pomes   paul–pomes@uiuc.edu
Computer and Communications Services Office
University of Illinois at Urbana

August 2, 1992

Introduction

This document covers the installation of the CCSO Nameserver. Included are a description of the distribution, instructions on configuring the software, and some suggestions for building a database. Detailed descriptions of the various parts are left for other documents.

The Nameserver requires a UNIX system with a reasonable amount of Berkeley-ness. If you have a pure System V machine, you're in for a lot of fun. It also requires a C compiler (ANSI C preferred), and perl.

A Word About Support

The word about support is, "no". This software is provided as-is, and neither I nor the University of Illinois nor CSNet nor even your mother takes any responsibility for anything bad that happens because of it.

On the other hand, we do use the software extensively, and are interested in bug reports and suggestions. As time permits, I will answer email questions about the software, provided those questions aren't answered in the supplied documentation, or available through a quick perusal of the source code.

The Distribution

This section describes the various pieces of the distribution. Each piece is marked with one of several codes, which are listed in **bold**. The codes and their meanings are:

vital Things you must use/understand/modify to get the Nameserver up and running.

important Things you had better become familiar with, but can be safely skipped or taken for granted during initial installation.

optional Things you may or may not wish to use someday.

uiuc Things we use at UIUC that may be of little or no use to you, except as models.

Two general notes. First, *Makefile* in the various subdirectories are generated from the *Makefile.templ* files in those same directories, by *Configure*. Second, the RCS subdirectories do contain RCS files, but there are almost no useful log messages; the files are used for checkpointing only.

Converted to portable n/troff format using the -me macros from funky Next WriteNow format (icch).

README.NOW **vital** — Release notes, general instructions, warnings, last-minute changes, etc. Please read it before you go any further. Then, please read this entire document.

buildmisc **uiuc** — This directory contains the Makefile we use to do database updates. While it's certainly instructive, much of it is UIUC-specific. Saying "touch s.tape.raw f.tape.raw s.tape.all old.dir old.dov; make −n" in this directory is a good way to get an idea of what our update process looks like.

configs **vital** — This directory contains configuration files (perl fragments) for use in configuring the software. These fragments are divided into two major classes; operating-system specific fragments and setup-specific fragments. More about these in the Configure section below.

configs/defaults **vital** — Defaults for the configuration process.

configs/{aix,convex,dynix,next,ultrix} **important** — These are OS-specific configuration files. Use these to get basic parameters for the flavors of UNIX involved.

configs/{garcon,net−nav,ux2,uxa} **uiuc** — These are specific configuration files for our setups. They may be instructive, but you'll not be able to use any of them directly.

Configure **vital** — This perl script configures the source tree. **N.B.**, you **must** read the **Configure** section below before trying to use *Configure*; it's not like the Configure that comes with (eg) rn or perl.

olddoc **ptional** — This directory contains older documents, of varying relevance and utility, in a variety of formats. This directory will be removed when its contents have been completely superseded.

help **important** — A directory which contains help files for the server's use.

help/native **important** — A directory of help files related to the server and its database, but not to any particular client.

help/{macph,ph} **important** — Directories for client-specific help.

include **important** — This directory contains include files for the Nameserver.

lib **important** — Some library routines for common use.

Makefile **important** — This is the master Makefile for the whole system, and is generated by *Configure*.

doc **vital** — This directory contains the most up-to-date documents in n/troff format using the −me macro package. The man pages, *ph.1* and *qi.8*, use the −man macros.

doc/install.me **vital** — You're reading it now.

ph **important** — The UNIX *ph* client lives here.

qi **important** — And here is the server.

util **vital** — This directory contains files that are useful for building or manipulating Nameserver data. You will probably have to modify some of these programs for use in building your own database. Which ones depend on your situation.

util/age **uiuc** — We use this to get rid of people who have been in the database for a year after they've actually left UIUC.

util/aliasassign **important** — This is a perl script that takes the output of *aliasprepare* and assignes unique aliases (and kerberos fields). It produces a file in *maked* format (see below).

util/aliasprepare **important** — A perl script that takes input in *maked* format, and produces input for *aliasassign*.

util/border.c **important**	This program reorders the bytes in a Nameserver database. This allows databases to be moved between machines with VAX and 68000 byteorders.
util/build.c **important**	Build takes the *.idx* and *.iov* files and generates from them the *.seq* and *.bdx* files.
util/credb.c **important**	Creates an empty database.
util/f.unblock **uiuc**	Perl script that takes a UIUC staff dataset and puts it into *maked* format.
util/id.c **uiuc**	Functions for dealing with real id <−> fake id mapping.
util/maggie **uiuc**	A perl script to produce input for the UIUC printed phone book.
util/maked.c **important**	This program turns *maked* format files into *.dir* and *.dov* files.
util/makei.c **important**	*Makei* generates the hash table (*.idx* and *.iov*) from the *.dir* and *.dov* files.
util/mdump.c **important**	Dumps the database according to various criteria and into various forms.
util/merge3 **uiuc**	We use this perl script to reconcile the old database with new student and/or staff information. Pray you never, ever, have to get near it.
util/nsck.c **important**	Runs some consistency checks on the database.
util/phify **optional**	A script that turns *maked* format data into something that looks like *ph* output.
util/phoneaddr **uiuc**	Perl script that copies either office or home phone and address into phone and address fields. Uses *maked* format.
util/qierrs **optional**	Perl script that sifts the output of *qi*, looking for errors.
util/s.unblock **uiuc**	Perl script that takes a UIUC student dataset and puts it into *maked* format.
util/ssndump.c **uiuc**	Dumps a dbm real id <−> fake id database into ASCII form.
util/ssnid.c **uiuc**	Uses a dbm real id <−> fake id database to map real id's to fake id's, and to assign fake id's.
util/ssnload.c **uiuc**	Loads a dbm real id <−> fake id database from ASCII form.
util/testqi.csh **important**	A script that tests *qi*, at least minimally.
whoi **optional**	A "whois" server that actually uses *qi*.
xtra **optional**	Stuff related to the Nameserver, but not integrated into the distribution.

Configure

Configure is a perl script that gets the source ready for compilation. This process includes setting up compilation and linking options, choosing database locations, deciding where binaries go, and determining which features to enable. It does this by building *Makefile* from the *Makefile.templ* and building the *conf.h* and *conf.c* source files. *Configure* makes use of files in the *configs* subdirectory. It reads *configs/defaults* first, and then read in turn each of its argument files. These files should contain perl scripts.

The scripts supplied are separated into three categories. In the first category is defaults, which is read first, and contains global defaults. Insofar as possible, I suggest you leave defaults alone; if you wish to alter the environment it creates, do so by overriding the defaults with your own configuration files.

The second category of scripts are OS-specific scripts. These scripts set compiler options and defines for use with various flavors of UNIX.

The third category are installation-specific scripts. These scripts are used to define options for a particular databases. Use of these scripts make it easy to run multiple *qi* databases on a single host, with different features enabled on each database.

The scripts you write should primarily set perl variables. The values of these variables will later be used when *Configure* is actually run. The variables you may set and what you may set them to are described in the *defaults* script. I will highlight a few of the most important here. You should, of course, review all the variables; just be doubly sure not to miss these.

- The @Features array can be used to enable optional features in the code. If you want to run with encrypted passwords, this array is the place to say so.
- $CC is your C compiler. This should be an ANSI C compiler; I use gcc.
- $Owner and $Group own the nameserver binaries and database.
- If you have some extra libraries you need, put them in $MoreLib.
- $ExecDir is where executables will be put.
- $DefineStrings{"Database"} is the name of your database, shorn of suffices, but with the leading path component.
- $OtherDefines{"Drecsize"} and $OtherDefines{"Doversize"} must be correct for your database, as must $OtherDefines{"NIChars"}.

The Field Configuration File

The field configuration file is *qi*'s key to interpreting your database. In this file you associate names with field numbers, and determine the properties of fields. The file should be named the same as your database, but with a *.cnf* extension (older versions of source and documents may refer to this as the *prod.cnf* file). It consists of lines of the form:

```
3:name:256:Full name.:FS:Indexed:Lookup:Public:Default:
```

The first item is the field id (in this case, 3). This number identifies the field in an entry, or in a *maked* format file. The second item is the field name (in this case, “name”), which should be used in commands, and will be printed in query responses. The third item is the maximum length of the field (in this case, 256 characters); maximum lengths should be less than 4096 characters. The fourth item is a brief description of the field. The fifth item contains instructions for the *merge3* program; if you don't use *merge3*, put an “O” in this item. The final items contain a list of field attributes. Only the first character of the attributes are significant. The attributes and their functions are:

I Indexed; the contents of this field will be put in the database index. At least one Indexed field must be included in every query.

L Lookup; users may use this field in queries.

P Public; the contents of this field may be displayed to anyone (but see “F” below).

D Default; this field will be returned for queries that do not specify which fields to return.

C Change; users may change the contents of this field.

F ForcePub; users may not suppress this field. Fields not marked “F” may be hidden from view by putting something (anything) in the F_SUPPRESS field.

N NoPeople; users may change this field, but only if their F_TYPE field does not contain “person”.

E Encrypt; this field may not cross the network, nohow, noway.

W Any (Wild); fields so marked may be searched collectively by specifying an “any” field in a query.

There are other defined attributes, but they are not used at this time.

You have a great deal of freedom in how you manage your field configuration file. You may have as many fields as you like, and give them whatever names, numbers, and attributes you like. There is, however, a relatively small set of “core” field names and numbers. If you change these field names or numbers, or omit them from the database, you are likely to have to make changes to the source to accommodate the change. These fields are:

```
2:email
3:name
4:type
5:id
6:alias
```

```
7:password
8:proxy
23:nickname
25:all
30:hero
43:suppress
```

Furthermore, there are some other fields that are used by some of the utilities, or auxilliary programs like *phquery*. If you modify these names or numbers, some such programs may have difficulty.

```
0:address
1:phone
9:department
10:title
11:curriculum
20:home_address
21:permanent_address
22:office_address
26:callsign
31:no_update
32:office_phone
33:home_phone
35:high_school
37:permanent_phone
42:left_uiuc
```
(only the number is important for this one)

Our field configuration file is included in the *qi* distribution, for reference.

This document is incomplete. Sorry.