Guidelines for Installing and Customizing Moira

Requirements

This distribution has only been tested on Vaxen and IBM PC/RT's running 4.3 BSD, Decstation 3100's running Ultrix 3.1, and Sun3's running SunOS 4.0. It should run on other systems, but there are probably some porting issues that have been missed.

It requires:

- the Kerberos authentication system
- RTI Ingres, version 5.0 or later

It can also make use of the zephyr notification system and hesiod nameservice.

Compiling the System

The makefiles should be complete. If the file */usr/rtingres/bin/eqc* exists when you make the system, both the client and server programs will be built. If that file does not exist, then only the client programs will be built. This enables clients to be built for machines which do not have Ingres available.

Some local configuration may be required. First make any appropriate changes to *include/moira_site.h*:

1. **MOIRA_SERVER** is the default server to connect to. Note that clients will user command line arguments, environment variables, and hesiod to find the server before falling back on the compiled in default.
2. Define **HESIOD** here if you want to allow clients to use hesiod to find the Moira server. If you don't define **HESIOD** and don't have a libhesiod.a on your system, put an empty library with this name in lib/libhesiod.a.
3. Define **ZEPHYR** to have errors and status messages broadcast using zephyr. If you don't define **ZEPHYR** and don't have a libzephyr.a on your system, put an empty library with this name in lib/libzephyr.a.
4. Define **SYSLOG** to have errors and status messages logged using syslog.
5. Undefine **ATHENA** unless you really want all of the local hacks we use. Many parts of our system have evolved and have various quirks for the sake of backwards compatibility. You probably don't need these.
6. Define **INGRESVER** to be the major version number of the Ingres release you are using (i.e. 4, 5, or 6). This is used by some server code which attempts to work around certain Ingres bugs.
7. At the end of the file are a number of defines of filenames and directories that the server will use. These directories are referred to by name below. You may change them to arrange the disk usage on your Moira server differently.

You may also need to edit *gen/nfs.sh* to change the string **MIT.EDU** to match your local domain.

First run a "make depend" in the top level directory, then "make all".

Installing Binaries

After building, a "make install" will put server-specific binaries in *bin* and the client binaries where we keep them at Athena: user clients in *${DESTDIR}/usr/athena*, maintenance clients in *${DESTDIR}/usr/etc*, manual pages in *${DESTDIR}/usr/man/man?*, header files in *${DESTDIR}/usr/include*, libraries in *${DESTDIR}/usr/athena/lib*, and lint libraries in *${DESTDIR}/usr/athena/lib/lint*. The include files and libraries are so that users may compile their own moira clients without the rest of the source tree.

Next the server machine must be set up.

1. Install Ingres on the machine, and configure */etc/rc* so that it verifies the databases at reboot. For example:

   ```
   if [ -f /usr/rtingres/bin/restoredb ]; then
       echo -n "Recovering Ingres databases..." >/dev/console
       /usr/rtingres/bin/restoredb -s -p
       echo "done." >/dev/console
   ```

2. Create this set of directories:
   - SMS_DIR[1]
   - BIN_DIR
   - DCM_DIR
   - LOCK_DIR
   - DCM_DIR/hesiod, DCM_DIR/nfs

   Copy all of the binaries installed in *bin* to BIN_DIR.

Finally, make entries in /etc/services:

```
sms_db          775/tcp         # Moira database
sms_update      777/tcp         # Moira update protocol
sms_ureg        779/udp         # Moira user registration
```

These are the numbers we use at Athena. They are not officially allocated, and you may change them at your site.

Creating the Database

Create an Ingres database called moira with the command "createdb sms".

Run the script *db/newdb* in quel, the Ingres query interpreter, creating a completely empty database.

Run the *newmoira* program to initialize the database. This program will ask:

1. For the name of a user to be given access to the database. Root will automatically be given access; if you want someone else to be able to access the database through quel, then enter their name(s) here. It will keep asking you for names until you enter an empty string.
2. The starting UID to assign. This is the user id to be assigned to the next account created. Each time an account is created, this number is incremented as long as it conflicts with an existing UID in the database, then the resulting value is used.
3. The starting GID to assign. This is used just like the UID above.

[1] Directory names are as defined in *include/moira_site.h*.

4. The default NFS quota. This is the quota that new users are assigned.
5. The name of a privileged user. This user will be put on the list dbadmin, which will be on the access control list for every query. This user may then make any change to the database, including putting other users on the dbadmin list.

At some point, the database must be optimized. If a lot of data is to be added, wait until afterwards. Otherwise, you may do the optimization now. The script *db/dbopt* contains quel commands to modify some of the structures and create additional indexes on the tables. You should also run the ingres commands */usr/rtingres/bin/optimizedb* and */usr/rtingres/bin/sysmod* after data is loaded into the database.

Make sure the necessary Kerberos keys are in your */etc/srvtab* file on the server. For the initial server testing, only **moira.***machinename@localrealm* is needed, but **sms**.[NULL]@*localrealm* is needed as well for updating servers, and **register.***machinename@localrealm* is needed to run the registration service.

Testing the Services

At this point the server and clients may be tested. Run BIN_DIR*/startmoira* to get a server running. It will log everything to SMS_DIR*/moira.log*. The privileged user chosen above may run the moira client, and make changes to the database. To make more privileged users, put the new users on the "dbadmin" list.

When this much is satisfactory, then service extraction may be initialized. Enable the desired services from the **services** sub-menu of the **dcm** menu of the moira client. Then run the DCM by invoking BIN_DIR*/startdcm*, and verify that it produces the desired file. It will log messages to SMS_DIR*/dcm.log*. If the data files are OK, then you may enable automatic update to the other servers by making additions in the **hosts** sub-menu of the **dcm** menu in the moira client. Each host must be running */usr/etc/update_server* and have a Kerberos key for **rcmd**.*hostname@localrealm* in its */etc/srvtab* to be able to receive updates.

When service updates are satisfactory, you can set up */usr/lib/crontab* to invoke BIN_DIR*/startdcm* every 15 minutes. When the DCM is started, it will determine what work there is to do.

The only remaining part of the system is the registration server. This may be invoked by starting BIN_DIR*/startreg*. Note that the registration client (*userreg*) expects to find its data files in */mit/register*. Before you can successfully register new accounts, you must have

- People entered as users in state 0
- 1 or more hosts as POP service providers listed in the **hosts** sub-menu of the **dcm** menu in the moira client. *Value2* for each of these is the maximum number of poboxes that may be allocated on that server, *value1* is the number currently allocated.
- 1 or more NFS partitions listed in the **nfs** menu of the moira client. The declared *size* of the partition must be greater than the sum of the quotas that will be assigned to it. The status must be "Student" for register to automatically assign new users there.
- **register**.*moiraserver@localrealm* must be on the "add" and "modify" access control lists for the Kerberos admin server.

When you are confident that the system works, you may have the daemons started by */etc/rc*. We prefer to start them by hand after verifying that the database survived any system crash. We also have */etc/rc* create the file */etc/nodcm* at boot time, so that the DCMs started every 15 minutes won't do anything until we removed that file.

A final piece of configuration is regular backups. We backup the database over the net, rather than backing up the filesystem. This allows us to restore the database on another machine architecture, if necessary. We do this by having *cron* run BIN_DIR*/nightly.sh* at 5am each morning.

Customization

This is difficult to give advice on, but I'll start by describing how we add new services and change things.

1. First determine what data must be recorded for this service. What is the type of each datum: integer, string, date, username, listname, machinename, other?
2. Design the Ingres table to record this information. Integers may be 1, 2, or 4 bytes long. Strings may be ascii character strings (storage compresses out whitespace) or binary byte fields. Other moira objects such as users, lists, machines, etc. should be referenced by their internal ID numbers. If you're creating a new type that may be referenced by other tables, assign each object a unique ID number as well as its name. Add this table specification to *db/newdb*, so that recompiling the backup and restore programs will cause them to know about the new table(s).
3. If a new object type is added to the database, also update *server/incremental.qc* to know about incremental updates to this type of object.
4. Specify the queries that will be used to access this data. Most datatypes have 4 queries: retrieve, add, update, and delete. Some have multiple kinds of retrieves, and/or multiple kinds of updates. If you make it similar to other existing queries, you are more likely to be able to share code in the server.
5. Make the additional entries in *server/queries2.c* for the new queries. See the document *doc/server-internals* for some idea of how the query table is coded. Then recompile and re-install the server. You can test the new queries with the mrtest client.
6. It may be necessary to write a new dcm extraction program (for many services, this is the only thing required: no new data or queries). The source to these is in the *gen* directory. Note that they touch the database directly, rather than going through the moira server. This is for performance reasons. These routines use the queue and hash table routines from the moira library. The easiest way to write a new one that does all of the time checking correctly is to start with one of the existing ones.