Link NPL, NDM, and PostgreSQL

How to create rtip and rtpp.

Creating the rtip and rtpp is not difficult but it is different from creating rtic and rtpc, there are many details that have to be in place for success. If any of the details are wrong the compile will fail.

PostgreSQL Version 7.3.3 must be installed correctly and functioning properly.

PostgreSQL is not available for SCO as a precompiled package. It is necessary to download the source code and build it.

The official PostgreSQL web site is <http://www.postgresql.org/>. Manuals and FAQs can be found in the "Docs" section. The "Administrators Guide" has installation instructions. The "SCO FAQ" covers some SCO OpenServer issues.

This has been developed & tested using PostgreSQL 7.3.3.

Preparation:

Before you attempt to build PostgreSQL, make sure the following optional software is installed on the SCO System:

- SCO UnixWare and OpenServer Development Kit (UDK)
- SCO Skunkware - Development Tools
- SCO Skunkware - File and Shell Utilities

When building PostgreSQL, you should be logged in as an unprivileged user; some steps will not work correctly if you are logged in as 'root'.

Make sure the directories containing the UDK executables (cc, ld, etc.) and the GNU development tools (gmake, gzip, etc) are included in your PATH environment variable; by default, these are installed in /udk/usr/ccs/bin and /usr/local/bin. If you have the SCO OpenServer Development System installed too, ***make sure they do not*** precede the UDK tools in the search path.

The version of the GNU C compiler (gcc) included on the SCO Skunkware CD appears to be incompatible with the PostgreSQL configuration & build scripts; if you have it installed, you may need to rename it so that the scripts won't find it and try to use it.

Kernel Configuration:

The PostgreSQL server requires a large amount of shared memory. The default value of the SHMMAX kernel parameter on SCO OpenServer is too low; it should be set to at least 1.5MB (1572864 bytes).

You can inspect and change kernel parameters in the graphical desktop by running the Hardware/Kernel Manager, or from a text-mode window by running the command "scoadmin hardware/kernel manager", or by following the SCO OpenServer instructions in the PostgreSQL Administrator's Guide.

After changing SHMMAX, it will be necessary to rebuild the kernel and reboot the system.

For details, see the PostgreSQL Administrator's Guide, "Managing Kernel Resources".

Building & Installing PostgreSQL on SCO

Following are instructions on building and installing PostgreSQL on SCO OpenServer. For more information, see the PostgreSQL Administrators Guide. The following assumes you are installing PostgreSQL version 7.3.3; if you choose a different version, the file & directory names will differ.

Make sure you are logged in on your SCO OpenServer system as a non-privileged user (i.e. not root), and go to the directory where you want to put the PostgreSQL source code.

1) Download and extract the source code:

Download the PostgreSQL source code (e.g. "postgresql-7.3.3.tar.gz") from the PostgreSQL web site to the current directory.

Extract the contents of the .tar.gz file using the commands:

```
gunzip postgresql-7.3.3.tar.gz

tar xf postgresql-7.3.3.tar
```

This will create a directory named postgresql-7.3.3 under the current directory. Go to this subdirectory.

2) Edit the int8.c source file:

The UDK's C compiler has a bug which is triggered by a line in one of the PostgreSQL source files. To avoid this bug, edit the source file ./src/backend/utils/adt/int8.c. Find the function "int8fac", and look for the following line in that function:

```
for (i = arg1, result = 1; i > 0; --i)
```

Remove the "> 0", so that the line now reads:

```
for (i = arg1, result = 1; i; --i)
```

3) Run the configure script:

Run the PostgreSQL "configure" script with appropriate options:

```
./configure \
--with-libs=/udk/usr/lib:/usr/local/lib \
--with-includes=/udk/usr/include:/usr/local/include \
--disable-largefile \
--without-readline \
--without-zlib
```

(Note: The backslashes "\" indicate that the command is continued on the next line.)

The --with-libs and --with-includes options specify where the UDK and Skunkworks library and include files can be found. If for some reason they were installed in other locations, these options will have to be changed accordingly.

Additional parameters may be specified to control where the PostgreSQL executables, libraries, etc. will be installed. The default is to put them in subdirectories of /usr/local/pgsql. See the PostgreSQL Administrator's Guide for details.

If the configure script reports any errors, it will be necessary to examine the resulting log file and figure out how to correct the problem.

4) Build PostgreSQL:

Run the GNU make command:

```
gmake
```

There may be some warning messages, but should be no errors.

5) Regression Tests:

Test the newly built PostgreSQL server by running the command:

```
gmake check
```

You may get an error if the SHMMAX kernel parameter is too low.

See the PostgreSQL Administrator's Guide for information on how to interpret the results of the regression test. If one or more tests fail, you will have to examine the results closely to determine whether it's important. For example, the Geometry tests may fail due to a slight difference in the precision of floating-point results, but this is unlikely to cause any problems.

6) Install PostgreSQL:

Install PostgreSQL by running the command:

```
gmake install
```

Setting up PostgreSQL:

Setup is much the same as for PostgreSQL on SuSE Linux, but with some minor changes:

For details on setting up PostgreSQL, see the Administrator's Guide, "Server Runtime Environment".

For details about PostgreSQL user accounts, see the Administrator's Guide, "Database Users and Privileges".

To set up a user account for the PostgreSQL daemon, use the SCO Account Manager in the graphical desktop.

After creating the "postgres" user account, edit the login script for that user so that the PATH environment variable will include the directory where the PostgreSQL executables were installed (default is /usr/local/pgsql/bin).

Note that the instructions in the PostgreSQL manual use "/usr/local/pgsql/data" as the directory where the PostgreSQL data will be kept, while my instructions used "/usr/ndm/data". The location is up to the administrator to decide.

In previous instructions for Linux, the PostgreSQL server is started manually by using the /usr/bin/pg_ctl command. On SCO OpenServer, the PostgreSQL executables are normally installed in /usr/local/pgsql/bin instead of /usr/bin, so the command will have to be changed accordingly:

```
/usr/local/pgsql/bin/pg_ctl start -l /usr/ndm/data/logfile -D /usr/ndm/data
```

The method for starting the PostgreSQL server automatically upon system startup is different for SCO OpenServer. First, create a file called /etc/init.d/postgresql containing the following line:

```
su - postgres -c "startup-command"
```

where *startup-command* is the command that was used to manually start the server.

Then create a symbolic link to this new file in /etc/rc2.d as S99postgresql:

```
ln -s /etc/init.d/postgresql /etc/rc2.d/S99postgresql
```

To make rtip and rtpp you need:

NDM BESDK for PostgreSQL installed in the NDM directory.
NDM DEVELOPER KIT for PostgreSQL installed in the NDM directory.
NPL BESDK for PostgreSQL (ELF version of the BESDK) installed in the NPL directory.

1. Log in as root
2. Run umask 0000
3. Create a directory called ndm in /usr
4. Place both ndmbesp51037.sco and ndmdevp51037.sco into /usr/ndm
5. Use "tar xvf " to expand both files.
6. Change directories to /usr/BASIC2C
7. Make sure NIAKWA_RUNTIME is set to /usr/BASIC2C
8. Make sure LD_LIBRARY_PATH is set to the location of the PostgreSQL library.
9. Make sure PGSQLDIR is set to the location of the PostgreSQL directory.
10. Make sure PGUSER is set to the value configured for the PostgreSQL user.
11. Copy ELF_besdk.sco into /usr/BASIC2C
12. Use "tar xvf" to expand the file.
13. There is a problem with the "installb" shell script. It will be corrected in the final. For now you will need to edit the shell script and substitute "./" for "${start_dir}".
14. Run ./installb to install the NPL Besdk.
15. Make sure the $PATH begins with "/bin:/usr/bin:/usr/local/bin:"
16. Make sure "/udk/usr/ccs/bin" is at the beginning of the $PATH.
17. Change directories to /usr/BASIC2C/uextrn/cexam and run "make" as a test. This will assure you are using the correct compiler. This will compile a runtime called rtix.
18. If the above step is successful, change directories to "/usr/ndm/ndmx" and run make to make the rtip and rtpp.

Compiling rtic and rtip

The error messages that start with "UX:cc", "UX:make", and "UX:ld" indicate that they're using the SCO UDK, which is not compatible with the C-ISAM libraries. To work with C-ISAM, they have to use the SCO OpenServer Development System; to work with PostgreSQL, they have to use the SCO UDK.

The SCO OpenServer Development System is included on the same CD as the operating system. The CD is labeled "SCO OpenServer" with a release number (e.g. "Release 5.0.6"). The Development System requires a separate license code.

The SCO UDK is on a CD labeled "UnixWare and OpenServer Development Kit".

It's possible to install both the Development System and the UDK on one system, but you have to use environment variables to determine which one will be used.

They appear to be using the development tools from the SCO UDK, instead of the default SCO OpenServer development tools. (As I'm sure you remember, NDM for C-ISAM requires the OpenServer tools, and NDM for Postgres requires the SCO UDK tools.)

The easiest way to find out which tools are being used is to check the version information by typing the following commands:

```
cc -V
ld -V
```

(note that's an uppercase 'V')

Both version messages should say "SCO UNIX Development System", which means that they're the default SCO OpenServer development tools.

If either or both messages say "Optimizing C Compilation System", then they're using the SCO UDK. If this is the case, double-check the PATH environment variable. Try to identify which directory contains the copy of 'cc' and/or 'ld' that it's using, and remove that directory from PATH.

It might also be necessary to type the command "make clean" before typing "make", to force it to recompile everything.

Link NPL, NDM, and C-ISAM

How to create rtic and rtpc

Creating the rtic and rtpc is not difficult but there are many details that have to be in place for success. If any of the details are wrong the compile will fail.

To make rtic and rtpc you need:

NDM BESDK for C-ISAM installed in the NDM directory.
NDM DEVELOPER KIT for C-ISAM installed in the NDM directory.
NPL BESDK for C-ISAM (COFF version of the BESDK) installed in the NPL directory.
SCO Open Server Development tools must be installed.
Informix C-ISAM must be installed to the default location.

1. Log in as root
2. Run umask 0000
3. Create a directory called ndm in /usr
4. Place both ndmbesc51037.sco and ndmdevc51037.sco into /usr/ndm
5. Use "tar xvf " to expand both files.
6. Change directories to /usr/BASIC2C
7. Make sure NIAKWA_RUNTIME is set to /usr/BASIC2C
8. Copy COFF_besdk.sco into /usr/BASIC2C
9. Use "tar xvf" to expand the file.
10. There is a problem with the "installb" shell script. It will be corrected in the final. For now you will need to edit the shell script and substitute "./" for "${start_dir}".
11. Run ./installb to install the NPL Besdk.
12. Make sure the $PATH begins with "/bin:/usr/bin:/usr/local/bin:"
13. Make sure "/udk/usr/ccs/bin" is ***NOT*** in the $PATH.
14. Change directories to /usr/BASIC2C/uextrn/cexam and run "make" as a test. This will assure you are using the correct compiler. This will compile a runtime called rtix.
15. If the above step is successful, change directories to "/usr/ndm/ndmx" and run make to make the rtic and rtpc.

Richard Lindas

Date sent:	**Tue, 20 Sep 2005 16:30:49 -0500**
From:	**"Wayne C. Morris" <wcmorris@shaw.ca>**
Subject:	**Re: C-ISAM on SCO 6**
To:	**Richard Lindas <ral@niakwa.com>**

I was able to get NDM for C-ISAM working on SCO 6 without much difficulty.

To make it work, you have to do two things:

1) Set the LD_LIBRARY_PATH environment variable to:

/osr5/usr/ccs/lib:CISAMDIR/lib :/INFORMIX/LIB

where 'CISAMDIR' is the directory to which C-ISAM was installed (normally "/usr").

2) Edit ndmx/makefile, ndmserv/makefile, and ndmexam/makefile.
Change all instances of "cc" to "cc -K osr".

The COFF version of BESDK ('uextrn') also needs some changes to work on SCO 6:

1) Set LD_LIBRARYPATH as above.

2) Edit include/rtpparms.c, include/rtpparms.h, and cexam/myrtpext.c.
Change all instances of "stricmp" to "stricmpx".

3) Edit cexam/makefile, parmtest/makefile, and noextern/makefile.
Change all instances of "-b coff" to "-K osr".

Changed extern/BIN/objlists CC_BESDK from -b coff to cc -K osR

I'll check whether it's feasible to incorporate these changes into the next release for SCO, but it may be difficult to do this while still keeping it compatible with both SCO 5 and SCO 6.