CSCD43 DATABASE SYSTEM TECHNOLOGY

Installing PostgreSQL in User Mode

Introduction

For this course we will be using `postgresql 7.4.13`. You can **download** the source code from

- `http://ftp-archives.postgresql.org/pub/source/v7.4.13/`
- or, go the the PostgreSQL website and follow the links to download the source code.

Extra space for installing PostgreSQL has been allocated for each student on the Math cluster. The Math cluster: `mathlab.utsc.utoronto.ca` has already accounts for you using your UTORid. When you log in you will see a symbolic link called ”cscd43<suffix>_space” (suffix identifies the semester and year of your course enrollment) that points to a directory that has ample space for your lab and assignment work for this class. Use this space for all code related assignments. You should install PostgreSQL and initialize the database in this directory only. Note that there is a quota of 5GB per student and this space will be reclaimed at the end of the semester.

If you have your own machine with root access, then you can simply follow the instructions given in the `INSTALL` file to compile and install PostgreSQL on your machine.

The purpose of this writeup is to enable installation of PostgreSQL in *user mode* on a machine where you do not have root access (such as the university computing facilities).

Installing PostgreSQL

Assume that the directory you wish to install postgresql is:

`/home/cmishra/pgsql/`

Substitute this by whichever directory you are installing into when using the procedure given below.

To **install** PostgreSQL :

1. Go to the top-level directory of the distribution and run

```
$./configure --prefix=/home/cmishra/pgsql/ --enable-depend
$gmake
$gmake install
```

PostgreSQL should now have been compiled and installed in the specified prefix directory.

2. We now need to initialize a directory where PostgreSQL stores its data. The commands:

   ```
   $mkdir /home/cmishra/pgsql/data
   $/home/cmishra/pgsql/bin/initdb -D /home/cmishra/pgsql/data
   ```

   will do the job of initializing the appropriate directory.

To **create** and **use** a database `test` we run

```
$/home/cmishra/pgsql/bin/postmaster -D /home/cmishra/pgsql/data > logfile 2 > &1 &
$/home/cmishra/pgsql/bin/createdb test
$/home/cmishra/pgsql/bin/psql test
```

These commands are written assuming the `bash` shell. In case you are using another shell, the output redirection commands will be a little different. The first command starts the `postmaster` server using the given data directory, and redirects the standard output and error to the `logfile`. The next command creates the database `test`, and the third one tells the frontend `psql` to connect to `test`. You should have now PostgreSQL running in user mode.

Suggestion: To avoid typing out the long prefixes you may add the `bin` directory to your `PATH`. Note however that there might be `postgresql` installed on the machine in advance, so make sure you are calling your own local copy of `postgresql` instead of the system one.

These instructions are pretty much the same as those given in the standard postgresql INSTALL file. The only difference is that we change the prefix of the directory tree where we will be installing PostgreSQL . This allows us to install PostgreSQL in user mode without needing any root access.

This document contains more information on how to run PostgreSQL on shared machines in a client-server mode.

Port Numbers

When trying to launch `postmaster` on a shared machine, you may have got an error like:

```
FATAL:  could not open lock file ``/tmp/.s.PGSQL.5432.lock'': Permission denied
```

This error means that there is another instance of postmaster (probably someone else doing the assignment) running on the same machine which is listening on the default port (5432).

The solution to this problem is to run `postmaster` on a different port (for instance, 6179), as in the example below:

```
$postmaster -D <DATADIR> -p 6179
```

In the case that the port number you have selected is already in use, you should choose another port number. Correspondingly, the frontend program should connect on the new port as well, as indicated below:

```
$psql <DBNAME> -p 6179
```

Similarly, `createdb` should also be run with a `-p` flag to specify which port number to connect to. For more information, check the manpages.

Shared Resources

PostgreSQL in its default configuration is fairly heavy and spawns many processes which eat up system resources. In a shared environment, where the server is being used by many instances, it might help to tune-down the settings of PostgreSQL . In particular, you should set the buffer size low (20-30 should be enough), and also allow a small number of connections to the system. An example of how to set the buffer size is given below:

```
$postmaster -D <DATADIR> -p 6179 -N 3 -B 20
```

This command states that the backend server may take at most 3 connections (`-N` flag) and the buffer size is 20 blocks (`-B` flag).

Important

It is extremely important to **always** turn off your postmaster server process when you are not using it. This can be done using the `pg_ctl` command:

```
$pg_ctl stop -D <DATADIR>
```

As always look at the `man` pages for more details.

For the first assignment, you will be running most queries using the PostgreSQL standalone back-end `postgres`; the options there are similar to the ones presented earlier.