Postgres Setup on the CSIF

Introduction

For those unfamiliar with the Computer Science Instructional Facility (CSIF), please read **The CSIF** section.

To get everything set up on the CSIF quickly, see the **Quick Setup Commands** section. If you want more information, see the **Detailed Setup Instructions** section.

Either way, please read the **Important Note**, **Using Postgres**, and **CSIF Disk Quota Note** sections below.

Note that the setup commands only have to be done **one time** to get everything ready. Once everything is set up, the **only thing you'll have to do is to start and stop the server.** Once the server is started, you can access your databases with the `psql` command (see the **Using Postgres** section for details).

The CSIF

To submit the Postgres assignments in this course (the assignments will include specific submission directions), you will need to log in to the CSIF, which consists of the Linux machines named `pc1.cs.ucdavis.edu` through `pc20.cs.ucdavis.edu` in the basement of Kemper Hall. You will need to log in to them using your campus Kerberos username and password either physically at the machines themselves, or remotely as follows (note that you will have access to the same instance of your home directory `~/` that contains your files regardless of which machine you log into, since each CSIF user's home directory is mounted via NFS).

If you're using a Unix-like operating system (e.g. Ubuntu, Fedora, Mac OS X, etc.) or Cygwin on Windows (you'll need to install the "`openssh`" package under "`Net`"), you can connect via SSH. Open a terminal window and type the following to connect to a CSIF machine, where `username` is your Kerberos username and`pc20` is one of the CSIF machines:

```
ssh username@pc20.cs.ucdavis.edu
```

and then enter your Kerberos password when prompted. Note that there are around 60 machines to choose from - please pick a machine other than `pc20` so that it isn't overloaded!

To upload a file to the CSIF, use SCP as follows (note that there must **not** be a space after the colon):

```
scp source_file username@pc20.cs.ucdavis.edu:path_to_destination
```

Note that if the `path_to_destination` is a relative path (i.e. not an `/absolute_path` with a leading slash) it will be relative to your home directory. This means that if you want to upload a file to your home directory then you can leave the destination blank (if you don't want to explicitly type out `~/`), but you **must** still use a colon ("`:`") at the end, otherwise you'll locally copy `source_file` to a file named `username@pc20.cs.ucdavis.edu`!

To upload a directory, use the `-r` flag:

```
scp -r source_directory username@pc20.cs.ucdavis.edu:path_to_destination
```

Simply reverse the source and destination if you want to download instead, e.g.:

```
scp -r username@pc20.cs.ucdavis.edu:path_to_source path_to_destination
```

When downloading, you'll often want to download to the current directory, denoted by a period ("`.`"), e.g.:

```
scp -r username@pc20.cs.ucdavis.edu:path_to_source .
```

If you're on Windows and don't want to use Cygwin, you can use PuTTY to get an SSH terminal and WinSCP to transfer files via SCP.

Quick Setup Commands

The following instructions are based on the user comments from an old Postgres Tutorial.

If you just want to get everything running, simply copy and paste the following commands into a terminal on the CSIF. This assumes that your shell is `tcsh` (this is the default shell on the CSIF). If you're using `bash`, see the "`export`" commands in the **Detailed Setup Instructions** section. Note that the "`alias start_postgres ...`" command must be on one line in your `~/.cshrc` file and was broken up here to work around `tcsh` quoting issues (note that the `-n` option for `echo` suppresses the line break when outputting its message).

```
mkdir ~/postgres
echo 'setenv LOCAL_DB ~/postgres' >> ~/.cshrc
echo 'setenv PGHOST $LOCAL_DB' >> ~/.cshrc
echo 'setenv PGDATA $LOCAL_DB/data' >> ~/.cshrc
echo -n "alias start_postgres 'pg_ctl -w -l ~/postgres_server.log start " >>
~/.cshrc
echo -n '-o "-c listen_addresses= ' >> ~/.cshrc
echo -n '-c unix_socket_directories=$LOCAL_DB"' >> ~/.cshrc
echo "'" >> ~/.cshrc
echo "alias stop_postgres 'pg_ctl stop'" >> ~/.cshrc
source ~/.cshrc
initdb
```

You can now (and any time you log in in the future) start and stop the server with the

- **`start_postgres`**

and

- **`stop_postgres`**

commands.

The server will log messages to the `~/postgres_server.log` file. If the server doesn't start, try another CSIF machine.

If you want to double check everything, the commands should look like this is your `~/.cshrc` file (again, note that the `start_postgres` command should be on one line):

```
setenv LOCAL_DB ~/postgres
setenv PGHOST $LOCAL_DB
setenv PGDATA $LOCAL_DB/data
alias start_postgres 'pg_ctl -w -l ~/postgres_server.log start -o "-c
listen_addresses= -c unix_socket_directories=$LOCAL_DB"'
alias stop_postgres 'pg_ctl stop'
```

To access your databases, please see the **Using Postgres** section. If you want to know more about the above commands, see the next section.

Detailed Setup Instructions

This section explains the setup commands in detail.

First, create your local `~/postgres` directory:

```
mkdir ~/postgres
```

Now add the following environment variables to your `~/.cshrc` file (just put them at the bottom if you don't know where else to put them) if you're using `tcsh`:

```
setenv LOCAL_DB ~/postgres
setenv PGHOST $LOCAL_DB
setenv PGDATA $LOCAL_DB/data
```

The Postgres server requires that the above variables be defined in order for it to know what data to work with.

If you're using `bash`, those lines will instead be the following, which would go in your `~/.profile` file:

```
export LOCAL_DB=~/postgres
export PGHOST=$LOCAL_DB
export PGDATA=$LOCAL_DB/data
```

If you don't know how to use an editor like `vi` to add them to your `~/.cshrc` file, these lines will add them to the end of the file for you (just copy and paste them into the terminal):

```
echo 'setenv LOCAL_DB ~/postgres' >> ~/.cshrc
echo 'setenv PGHOST $LOCAL_DB' >> ~/.cshrc
echo 'setenv PGDATA $LOCAL_DB/data' >> ~/.cshrc
```

Now bring them into your current environment before you proceed by typing:

```
source ~/.cshrc
```

Or by closing and re-opening the terminal, since `tcsh` reads `~/.cshrc` on startup.

Now type:

```
initdb
```

and Postgres will be set up after a few moments.

This ends the one-time startup instructions.

To start a server that logs messages to `~/postgres_server.log` (this must be on one line):

```
pg_ctl -w -l ~/postgres_server.log start -o "-c listen_addresses= -c
unix_socket_directories=$LOCAL_DB"
```

To stop the server:

```
pg_ctl stop
```

You can add these aliases to your `~/.cshrc` file to make starting and stopping the server easier (again, the "`alias start_postgres ...`" command must be on one line):

```
alias start_postgres 'pg_ctl -w -l ~/postgres_server.log start -o "-c
listen_addresses= -c unix_socket_directories=$LOCAL_DB"'
alias stop_postgres 'pg_ctl stop'
```

Now you can just use `start_postgres` and `stop_postgres` instead.
Again, to add the aliases to your `~/.cshrc` file without using `vi`, just copy and paste these lines into the terminal:

```
echo -n "alias start_postgres 'pg_ctl -w -l ~/postgres_server.log start " >>
~/.cshrc
echo -n '-o "-c listen_addresses= ' >> ~/.cshrc
echo -n '-c unix_socket_directories=$LOCAL_DB"' >> ~/.cshrc
echo "'" >> ~/.cshrc
echo "alias stop_postgres 'pg_ctl stop'" >> ~/.cshrc
```

And bring them into your current environment with:

```
source ~/.cshrc
```

or just restart `tcsh`.

If you're using `bash` instead of `tcsh`, your `~/.profile` file should contain:

```
export LOCAL_DB=~/postgres
export PGHOST=$LOCAL_DB
export PGDATA=$LOCAL_DB/data
. ~/.bashrc
```

and your `~/.bashrc` file should contain:

```
alias start_postgres='pg_ctl -w -l ~/postgres_server.log start -o "-c
listen_addresses= -c unix_socket_directories=$LOCAL_DB"'
alias stop_postgres='pg_ctl stop'
```

Important Note

The server will remain running on your CSIF machine even after you log out, so you should shut it down if you don't intend to come back to your CSIF machine after you log out with the **`stop_postgres`** command.

However, the nightly reboot of the CSIF machines will stop it if you forget! Note also that the server is running **only** on the CSIF machine that you start it on, and if you move to another machine you'll need to **stop** it on the first machine and **start** it on the new machine. Failing to stop the old server first doesn't seem to produce any ill effects other than "stranding" the original process: if you move back to the original machine you'll have to restart the server there again. Again, the nightly reboot should clear out any of these "stranded" processes, but if you don't like the idea of leaving them around, simply type this (and ignore the errors) to kill any process named "`postgres`" (this will include your current non-stranded process, if any, so you'll need to do a `start_postgres` if you need to use the database):

```
killall postgres
```

Using Postgres

By default, Postgres creates a database named "`postgres`". Once your server is started, you can access the "`postgres`" database with `psql` (the PostgreSQL interactive terminal) by typing:

```
psql postgres
```

and you can now create tables, etc. To quit, type:

```
\q
```

and hit enter. Type `help` and hit `enter` for help.

If you don't give `psql` a database name like "`postgres`" above, it will look for a database named after your **username**. If you want to, you can create such a database with (type this into `tcsh`, not `psql`):

```
createdb $USER
```

Note that `$USER` is an environment variable that contains your username. Now you can just type:

```
psql
```

to connect to the database named after your username.

Note that `createdb` (and its counterpart `dropdb`) is a program that you run from your shell, although you can also create and drop databases from within `psql` with the `CREATE DATABASE` and `DROP DATABASE` commands. See the Postgres documentation for more. Note that the CSIF has version 9.2.4 installed as of Fall 2013.

If you want to create a different database, named, say, "`test`", you can type:

```
createdb test
```

and connect to it via:

```
psql test
```

To delete the database named, say, "`test`", type:

```
dropdb test
```

CSIF Disk Quota Note

Be aware that your disk quota on the CSIF is limited. If you run low on space you will be unable to create new files, which includes new databases.

To see your current usage, type:

```
quota
```

Note that the CSIF provides a script to help clean out common cache files, etc. from your home directory. If you're low on space, type:

`fquota`

and follow the prompts. `mquota.pl` is another script that contains a trick to reduce the size of your Mozilla Firefox profile, so you may also want to run that script as well.

If you still don't have enough free space, even after deleting unused files, we can ask the CSIF staff to increase your disk quota.

Regarding the "pg_ctl: could not start server" error message

If you get this error message when trying to start the Postgres server with `start_postgres`, the cause of the error will be logged in the `~/postgres_server.log` file.

Deleting your `~/postgres` folder and starting over seems to fix most errors:

```
rm -rf ~/postgres
mkdir ~/postgres
initdb
```

You'll need to re-create any databases you had afterwards, of course.

Some of the causes that we've seen are: "FATAL: could not create shared memory segment" and "FATAL: incorrect checksum in control file". The "checksum" error can be caused by running `initdb` on a 32-bit machine

and then trying to run `start_postgres` on a 64-bit machine, and vice versa.