OpenACS Documentation

by Roberto Mello.

This is the documentation for OpenACS. Feedback and suggestions are appreciated, through our Software Development Manager.

- Documentation in PDF format
- OpenACS Installation Guide
- OpenACS Getting Started Guide
- Simple AOLserver Installation Guide
- Sample nsd.tcl Config File
- Simple PostgreSQL Installation Guide
- Guide to porting Oracle SQL to PostgreSQL
- Porting From Oracle PL/SQL is a guide written to help OpenACS (and PostgreSQL) developers to port applications from Oracle to PostgreSQL. It has been incorporated to the PostgreSQL documentation along with improved PL/pgSQL documentation.

---

rmello@cc.usu.edu ben@adida.net mcmullan@alum.mit.edu

OpenACS Installation Guide

Roberto Mello (rmello@fslc.usu.edu) and the OpenACS Team

This is the Installation Guide for the ArsDigita Community System port for the PostgreSQL Relational Database Management System. This documentation is based on Philip Greenspun's ACS installation docs with parts of it (marked with [1]s) being quoted here.

This is the April 2001 revision of the documentation.

Table of Contents

Next
What is OpenACS

1. What is OpenACS

According to Philip Greenspun:

The ArsDigita Community System it is a toolkit of software that will help you build Web services with a collaborative dimension, ranging from knowledge management within companies to B2C ecommerce to product support and community among the customers. The software is free and open-source and has been tested in heavy use since 1995.

The software and underlying philosophy are documented in a full-length textbook for Web developers (also free and available online at http://photo.net/wtr/thebook) (Using The ACS). We highly recommend you to read this book. It will save you lots of trial-and-error, time, grief and will help you understand how to build a great web service. Chapter 3 is of special interest, because that's where the data model and ideas of the ACS are explained, but the whole book is great.

If you are serious about building a web service, you should also read ArsDigita Server Architecture, a way of building and delivering reliable web services.

The port for the PostgreSQL RDBMS was started and coordinated by Ben Adida (who also helped to write the original ACS) and Don Baccus, and is maintained as a community project.

The combination of GNU/Linux, AOLserver/OpenNSD, PostgreSQL and OpenACS enables you to build very sophisticated web sites with completely free software. The development and demo web site for OpenACS is http://openacs.org.

2. What you need to run OpenACS

There are now two main ways to install everything you need to run OpenACS. The first is to make use of packages designed to make installation easy. This is only useful if a set of packages exists for the particular operating system and OS distribution which you are using. The second way is the more traditional one, obtaining source code in tarballs and installing from those.

2.1. Installing Using Binary Packages (RPM or .deb)

There are binary packages available for Debian GNU/Linux and for Red Hat Linux 6.2. RPM packages for Red Hat Linux 7.1 are expected to become available soon.

2.1.1. Debian GNU/Linux .deb packages

Debian GNU/Linux has OpenACS .deb packages all ready to install. Thanks to Brent Fulgham for providing the Debian packages.

2.1.2. Red Hat Linux 6.2 RPM packages

Very recently, a set of binary RPM packages have been created which make installing OpenACS on a Red Hat Linux 6.2 machine quick and simple. These are still being tested but look very promising. They are currently available at http://www.xc.org/jonathan/openacs but may soon become available at the OpenACS software page. Thanks to Jonathan Marsden for providing these Red Hat Linux RPM packages.

2.1.3. RPMs using Apache and mod_aolserver from ArsDigita

ArsDigita is providing RPM packages for OpenACS with mod_aolserver on Apache. They are available at the OpenACS software page. Note that installing these will require recompiling some of these RPMs from their source RPMs, though an install script simplifies this.

2.2. Installing from Source Code

- AOLserver - A free, Tcl-enabled, multi threaded, powerful web server that is behind huge sites such as aol.com and digitalcity.com. AOLserver/OpenNSD is also a complete web development platform being designed from start to have efficient access to databases. We support AOLserver/OpenNSD 3. NOTE: In february a fork of the AOLserver project was started to better address user contributions and requests. The project is called OpenNSD and its home page is at http://www.opennsd.org)

(Note: ArsDigita and Robert Thau came up with a mod_aolserver for the Apache web server, which intends to replicate the AOLserver API on Apache. It has performance issues when compared to AOLserver, but if you have to use Apache, it gives you a way to use OpenACS. It's available at the OpenACS Software page.

- PostgreSQL - A free, powerful SQL92-based Relational Database Management System with advanced features including Multi-variant Concurrency Control (unlike in the more common table-locking model, readers don't wait for writers and writers don't wait for readers, very crucial in a high-volume web server environment). Version 7 is the supported version.

- *Some flavor of UNIX or Windows*. Our development is almost entirely done on GNU/Linux (and when not done in GNU/Linux, it is tested on it) and for now, this is the only operating system we can help you with. Some very big web sites run on GNU/Linux and it has proven itself as a great platform for web servers, plus it is free software. Red Hat Linux and Debian GNU/Linux are the reference distributions for AOLserver and OpenACS, but if you know what you are doing, you can use any GNU/Linux distribution (and many people do).

To get OpenACS working on Windows you will have to get PostgreSQL working on it first, which is beyond the scope of this document. Please refer to the PostgreSQL documentation for that information. The OpenACS-specific instructions should need very little adaptation. Please contribute the documentation for Windows if you are going to do it.

- *The PostgreSQL driver* for AOLserver, available at OpenACS software page..

- The OpenACS distribution, available at OpenACS software page..

---

3. Getting Ready to Install

This section describes the filesystem location conventions of OpenACS, as well as how to initially unpack the files. These locations vary, depending on which installation approach is used.

Untarring is what those installing from source code will do; those installing from RPMs will do something using `rpm`, those installing from .deps will use `apt-get`.

3.1. Getting Ready to Untar [1]

We recommend rooting Web server content in /web (typically a symlink to a large mirrored disk drive). Since most servers these days are expected to run multiple services from multiple IP addresses, each server gets a subdirectory from /web. For example, http://scorecard.org would be rooted at /web/scorecard on one of our machines and if http://jobdirect.com were on the same box then it would be at /web/jobdirect.

For the sake of argument, we're going to assume that your service is called "yourdomain", is going to be at http://yourdomain.com and is rooted at /web/yourdomain in the Unix file system. Note that you'll find our definitions files starting out with "yourdomain.com".

- download openacs-3.2.?.tar.gz into /tmp/openacs-3.2.?.tar.gz (where ? is the version number)
- cd /web
- tar xzvf /tmp/openacs-3.2.?.tar.gz (creates a directory "openacs-3.2.?")
- mv openacs-3.2.? /web/yourdomain

3.2. Installing OpenACS from RPMs

The use of `/web` is conventional in the ACS community, but conflicts with the Linux File Hierarchy Standard. When installing from binary RPMs, the service is installed under `/var/lib/aolserver/servers/` and the default OpenACS service is therefore installed at `/var/lib/aolserver/servers/defaultacs/` -- all this happens transparently as the RPMs are installed. With the appropriate OpenACS RPM file in the current directory, the single command

```
rpm -Uvh *.rpm
```

should be all that is required.

3.3. Installing OpenACS from Debian GNU/Linux .deb files

OpenACS Debian packages are included in Debian's unstable distribution (woody), so you'll need the appropriate entries in your /etc/apt-sources.list to get it. All you need to do is some apt-getting:

```
apt-get install openacs
```

and that's it. NOTE: At the time of writing, the OpenACS Debian packages are outdated.

Prev | Home | Next
What you need to run OpenACS | AOLserver/OpenNSD

4. AOLserver/OpenNSD

4.1. Installing AOLserver

The from source approach places AOLserver under `/home/aolserver`. The RPMs and DEBs use `/var/lib/aolserver`.

4.1.1. Installing AOLserver from source

We wrote a Simple AOLserver Install Guide. For other instructions, see the AOLserver docs.

It was verified that, as of June 2000, AOLserver has problems compiling with pgcc (used by GNU/Linux distributions such as Mandrake and Stampede). I could only get AOLserver 3 to compile once I got rid of pgcc in my Mandrake 6.1 box and loaded egcs (standard in Red Hat.)

You must create a user that AOLserver is going to run as, since it will not run as root. I usually use nsadmin or ns (for NaviServer, AOLserver's original name).

4.1.2. Installing AOLserver from RPMs

This is already taken care of by the `rpm` command mentioned in a previous section. Note that the `nsd.tcl` configuration file is installed into `/etc/aolserver/nsd.tcl`, and is adapted to work with no hand editing for most OpenACS sites, bringing up the web server on port 8000 by default.

4.1.3. Installing AOLserver from .deb files

This is taken care of by the `apt-get` command in a previous section.

4.2. Configuring AOLserver

Note: AOLserver 3 can use both .ini and .tcl files for initialization. We are dropping support for old .ini files because .tcl files are much more flexible. Those who are used to .ini files and want to use them will know what to do :-)

We are including a sample working nsd.tcl file as reference that you can use, but you must configure it for your own needs. Just moving it to your AOLserver directory and running AOLserver will not work. The file is well commented so you should be able to understand it. Look at the AOLserver docs for other parameters you could have.

Pay special attention to the following sections in the nsd.tcl file:

```
ns_section "ns/db/drivers"
ns_section "ns/db/pools"
ns_section "ns/server/${server}/tcl"
ns_section "ns/server/${server}"
```

You can only have one ArsDigita Community System running from a single nsd process (though you can have as many ACS servers as you like on a physical machine; each just needs its own process).The reason reason for this is that each ACS installation needs to source its own parameter file (see second to last line in the nsd.tcl file).

In the ns/server/${server} section, if you want to use our fancy custom error responses and such, uncomment the lines

```
ns_param NotFoundResponse             "/global/file-not-found.html"
ns_param ServerBusyResponse           "/global/busy.html"
ns_param ServerInternalErrorResponse "/global/error.html "
ns_param ForbiddenResponse            "/global/forbidden.html"
ns_param UnauthorizedResponse         "/global/unauthorized.html"
```

then go into the www/global/ directory and edit these files to suit. [1]

4.3. Sample nsd.tcl file

You can find well-commented, almost ready nsd.tcl file right here. Rename it to nsd.tcl, chown it to your AOLserver user and edit to your needs. There are only a few things that need changes. Read the comments.

---

5. PostgreSQL

5.1. Which version: Postgresql 6.5.3 or 7 or 7.1?

PostgreSQL 7 is out at the time of this writing. It comes with some exciting new features such as referential integrity, to_char, optimizer overhaul and an enhanced psql.

The more advanced features of PG 7 are critical to OpenACS (such as lztext which will allow bigger comments, static pages, and referential integrity) so that's the version we support. Upgrading from PG 6.5 to PG 7 is not too hard, although you'll need to pg_dump and restore your database, plus an initdb.

Postgresql 7.1 is even more featureful and provides useful performance gains too. However it is new enough that few OpenACS installations use it yet. OpenACS 3.2.4 had a few SQL queries which do not work under Postgresql 7.1, these are fixed in OpenACS 3.2.5.

5.2. Installing PostgreSQL

5.2.1. Installing Postgresql from Source

5.2.1.1. Installing Postgresql itself from Source

I wrote a bare-bones installation guide for PostgreSQL, see Simple PostgreSQL Install Guide (simple-pg-install.html).

For more generic downloading and compiling instructions see http://www.postgresql.org/docs/admin/install855.htm.

5.2.1.2. Compiling the PostgreSQL driver for AOLserver

> **Note:** The `nspostgres.so` driver included in the AOLserver distribution *will not work* with OpenACS. Please use the driver available at openacs.org/software.adp. You will need the AOLserver source distribution to compile the driver. We plan to offer binaries at some point.

Also, to do any compiling in C, you'll need a compiler and the right libraries installed in your system. AOLserver, PostgreSQL and the PG driver were tested with gcc (the GNU Compiler Collection) and gmake (GNU Make). You will need gcc (egcs in Red Hat, pgcc in Mandrake/Stampede) and the glibc (GNU C library) installed (for GNU/Linux distributions this usually means packages like glibc and glibc-devel).

The PostgreSQL driver now comes with the AOLserver distribution (nspostgres.so), but if you are having problems, you can always get the latest PostgreSQL driver from OpenACS software page and compile it. If you are using the driver that comes with AOLserver, skip the next step.

Edit the Makefile to include the correct path to your PostgreSQL and AOLserver directories. If you are using the RPM version of PostgreSQL, make sure you have the devel package installed as well. You need to pay attention to these lines:

```
PGLIB=/usr/local/pgsql/lib # Where your PG libraries are installed
PGINC=/usr/local/pgsql/include # Where your PG includes are installed
NSHOME=/home/aolserver # Where your AOLserver is installed
NSINC=/usr/local/src/aolserver3_0/include # Where you untarred AOLserver
```

Do a *make* and then *make install*. The file *postgres.so* will be copied to the AOLserver's bin directory.

If you are running PG 7, make a symbolic link from libpq.so.2.0 pointing to libpq.so.2.1 because AOLserver looks for

libpq.so.2.0 when loading the driver:

```
cd /usr/local/pgsql/lib
ln -s libpq.so.2.1 libpq.so.2.0 (as user postgres)
```

5.2.1.3. Loading the data model

Make sure PostgreSQL is running fine with all the environment variables set (the RPM version does that all for you).

- Login as postgres (the PostgreSQL super user) and create a user for AOLserver in PostgreSQL. If your AOLserver runs as nsadmin , that should be the user to create with the command createuser nsadmin. In PG 6.5, you will be asked if the user is a super user and allowed to create dabatases, respond YES (y) to both. In PG 7 it will ask you if this user is allowed to create databases and if this user is allowed to create new users, respond YES (y) to both as well.

From now on, become the user AOLserver will connect to PostgreSQL as (e.g. nsadmin).

- Come up with a name for your Database (Usually it will be the name of the web service you're setting up. I'll use yourdb as example). Then create the database with the command: createdb yourdb.

- cd to the www/install directory of the OpenACS distribution and load the country/state/zip codes with the command :

```
./load-geo-tables yourdb
```

- cd to the *www/doc/sql* directory. If you are running the RPM version of PostgreSQL, edit the file postgres.sql and uncomment the following lines, commenting the two similar lines right below them:

```
--create function plpgsql_call_handler() RETURNS opaque
--as '/usr/lib/pgsql/plpgsql.so' language 'c';
```

- Edit the file load-data-model.sql. Uncomment the line \i postgres65.sql only if you are running PG 6.5.x.

(Optional - Deprecated) If you are running PG 6.5.3 and have the Tcl package loaded (or compiled --with-tcl) you may comment the \i postgres65.sql line and uncomment the \i postgres-pgtcl.sql line.

- Load the data model into yourdb with the command:

```
psql -f load-data-model.sql yourdb
```

Alternatively and for debugging purposes you can do the following (NOTE: The PDF output for some reason, turns the "greater than" sign that comes after 2, to "62". It should read 2> datamodel.txt):

```
psql -f load-data-model.sql yourdb 2> datamodel.txt
```

to save PG's output to a file called datamodel.txt, which you can review and look for errors. If you have a bunch of ERROR messages in this file, then you forgot to configure one of the OpenACS files.

If anything goes wrong, it is easier to simply destroy the db (command dropdb yourdb) and recreate it after you've reviewed your steps.

- Do a *psql yourdb*. You should end up with a prompt after a couple of messages indicating that it has successfully connected with the database. Do a \d to see all the tables in your db. Once you're certain you can connect from the account via psql, you should have no problem connecting via AOLserver. [DRB]

5.2.2. Installing Postgresql from .deb files

The `apt-get` command mentioned earlier should install all the packages you need. You'll need the `postgresql` and `postgresql-pl` packages and if you'd like bboard searches to work, you'll also need the `postgresql-tcl`.

5.2.3. Installing Postgresql from Red Hat RPMs

If you want to use RPMs (the suggested approach on Red Hat machines), please use the official PostgreSQL RPMs. They are available at Postgres' website. Lamar Owen maintains those RPMs and is also a member of the OpenACS team, so we know exactly how those packages are done, and our tests and documentation are based on that.

Those installing the aolserver binary RPM should *also* install the companion aolserver-postgresql RPM instead of compiling the driver from source code. Be consistent with this -- either install both from RPMs, or both from source code.

5.3. Some PostgreSQL tips from Don Baccus

Note: These are not absolutely necessary for running OpenACS, but I included here because it tells you how to get more out of PostgreSQL. It is good reading especially if you want to do something serious with your copy of OpenACS.

You'll need to make sure the Postgres postmaster is running, first. This is the process that forks a Postgres backend when AOLserver (or any other application, including PSQL) initiates a backend connection. I've got my .ini file configured so idle connections never get released by AOLserver, so this forking happens only once per connection per lifetime of the server (the MaxOpen and MaxIdle in the pools section of the nsd.ini, as in the example above).

Here's the command I use to run Postmaster from my */etc/rc.d/init.d/postgresql* script:

```
su -l postgres -c '/usr/local/pgsql/bin/postmaster -B 1000 -o "-S 2000" -S -D
/usr/local/pgsql/data'
```

For the RPM version should be something like this:

```
su -l postgres -c '/usr/bin/postmaster -B 1000 -o "-S 2000" -S -D
/var/lib/pgsql/data'
```

Jonathan Marsden suggests that for the RPM version of Postgresql 7.1, it should be instead something more like this:

```
su -l postgres -c "LC_ALL=C /usr/bin/pg_ctl  -D $PGDATA -p /usr/bin/postmaster -o '-B
2000 -o -S2000' start >/dev/null 2>&1" < /dev/null
```

Some explanations - "*-B 1000*" tells it to allocate 1000 blocks of shared memory (rather than the default 64, which is way puny). I've compiled my copy of postgres with a 16K blocksize, so this is 16MB of shared memory space, i.e. the most postgres will use without a kernel recompile. If you've compiled with the default 8K blocksize (RPM version), "-B 2000" will work. You needn't do this for testing, but for an active system helps a lot.

The *'-o "-S 2000" '* tells each backend to use up to 2 MB (2000 x 1KB) of RAM for sorting, etc before spilling to disk.

The other "*-S*" (to the postmaster itself, don't confuse with the above where -o is used to pass flags to forked backends) tells it to run "silently", in the background.

-D is used to pass the path to the database which you've hopefully already run initdb on, etc.

5.3.1. How to increase the blocksize in PostgreSQL

Again, this is not required to run OpenACS.

By default PostgreSQL is compiled with a blocksize of 8 Kb. You can compile PostgreSQL to have 16 Kb blocksize instead,

which will allow for bigger text and lztext data types. (NOTE: This is completely unnecessary in PostgreSQL 7.1, as it will be rid of this limitation)

Refer to the Simple PostgreSQL Installation Guide for instructions on how to do this for versions of Postgresql before 7.1.

5.4. Notes on PostgreSQL 7.1

The long-awaited 7.1 version of PostgreSQL, as usual, brings some exciting features and improvements like OUTER JOINs, optimizer improvements, fixes in PL/PgSQL, etc. The purpose of this section is to let you know what OpenACS 3.2.x items will behave differently under PG 7.1. This is what we know so far, if you find something else, please let us know.

- In *bookmarks.sql* you can comment the lines where the *chr(int4)* function is declared, for PG 7.1 comes with an identical one (contributed by Krzysztof Kowalczyk).

---

6. Configuring OpenACS itself

Although installing AOLserver and OpenACS in different ways will put its files under a different directory (`/home/aolserver/servers/yourdomain/` or `/var/lib/aolserver/servers/defaultacs/`, for example), the directory structure within OpenACS remains the same, no matter what method was used to install OpenACS.

6.1. What is where (OpenACS directories and what is inside each of them)

- `/bin` - executables used by the toolkit (e.g. WatchDog)
- `/parameters` - OpenACS configuration file(s)
- `/tcl` - OpenACS Tcl scripts library. Your AOLserver config needs to point to here as its Tcl library. The definitions for the modules are here.
- `/templates` - where templates are stored. Templates modify how a page is displayed according the user's preference (e.g. text x graphics) and language (e.g. with language extensions .fr, .pt, etc.) Not all modules are template-enabled yet, but expect that to change in ACS 4.
- `/users` - user specific files. Used by the home page module for example.
- `/www` - Where all the pages live. Each module has a subdirectory here (e.g. www/bboard)
- The `www/register` directory contains the login and registration scripts. You can easily redirect someone to `/register/index.tcl` to have them login or register. [1]
- The `www/pvt` directory is for user-specific pages. They can only be accessed by people who have logged in. [1]

6.2. Configuring OpenACS

Note: Starting with version 3.2.5, the supplied default configuration file `ad.tcl` has been made generic. It may work fine completely unchanged for many installations, and in any event should now need less editing that in previous versions.

Also, RPM users wishing to further customize their OpenACS installation should edit `defaultacs.tcl` rather than renaming `ad.tcl`.

- In the parameters directory of the OpenACS tree, rename the ad.tcl file to the name of the virtual server you are running in AOLserver (*server1.tcl* for example)
- Each module of the ACS is configured in the sections of this file, with a heading such as *ns/${server}/acs* (if you are using the included ad.tcl file). If you are using .ini file, than the headers will look like *[ns/server/yourservername/acs]*. In this case, replace all ocurrences of yourservername with the actual name of the virtual server configured in AOLserver (such as photonet, or server1).
- Edit the parameters to fit your needs, otherwise your website will show Yourdomain Network and webmaster@yourdomain.com all over. If you want to change how some of these are used, a good place to look is

/web/yourdomain/tcl/ad-defs.tcl. [1] There are lots of comments in the file to help you out and the documentation of each individual module can be found at http://openacs.org/doc.

- READ THE CAVEAT in section 6.5 if you choose to save encrypted password in the db (EncryptPasswordsInDBP=0 or ns_param EncryptPasswordsInDBP "0" in nsd.ini or nsd.tcl respectively).

6.3. Configuring Permissions [1]

You need to protect the proper administration directories of the ACS. You decide the policy. Here are the directories to consider protecting:

- /doc (or at least /doc/sql/ since some AOLserver configurations will allow a user to execute SQL files)
- /admin (this directory is already protected in latter OpenACS releases).
- any private admin dirs for a module you might have written that are not underneath the /admin directory

6.4. Adding Yourself as a User and Making Yourself a Sysadmin [1]

The ArsDigita Community System will define two users: system and anonymous. It will also define a user group of system administrators.

After starting AOLserver, you'll want to:

- add yourself as a user to the system, at http://yourservername.com/register/
- add yourself as as member of the site-wide administration group.

To do this, log out as yourself (there's a link at Your Workspace http://yourservername.com/pvt/home.tcl) and then log in as the system user (email of "system"). Change the system user's password (the default is changeme). Visit the the User Groups Admin pages at *http://yourservername.com/admin/ug/* and add your personal user as a site-wide administrator.

Now you're bootstrapped!

6.5. CAVEAT for those that want encrypted passwords in the DB

If you want to save encrypted passwords in the database, you'll have to do some things manually to get ACS working because the default users system and anonymous come with plain text passwords.

This is what you need to do:

1. create a login for you (as described above). Your password will be saved encrypted in the database. Go into psql and do a *select user_id,first_names,password from users;* to see all the users in your database.
2. Next, change the system user password for the password of the user you just created. Let's say the encrypted password for your user was something like 0xabcdef (or whatever), then do a *update users set password='0xabcdef' where user_id=1;*
3. Now go back to your browser, and logout as your user (from http://yourservername.com/pvt/home.tcl), login as system with the same password you used for your user, add your user to the administration group as described above, and then change the system and anonymous passwords (from http://yoursername.com/admin/users).

6.6. Closing Down Access [1]

The ACS ships with a user named "anonymous" (email "anonymous") to serve as a content owner. If you're operating a restricted-access site, make sure to change the anonymous user's password (the default is changeme).

6.7. Making sure that it works (and stays working)

Run the acceptance tests in http://photo.net/doc/acceptance-test.html.

Note: The first part of the above page is aimed at the original version of ACS for Oracle. You can replace that first part by going to psql (PostgreSQL interactive SQL tool) and doing some tests:

```
$ su - youraolserveruser
$ psql yourdb
yourdb# \d
yourdb# select * from users;
```

The first psql command is going to list all your tables (under PG 6.5) or all your relationships (under PG 7) and the second will show all the records in the users table.

The other sections of the acceptance-test can be used either under the Oracle or the PostgreSQL versions of the ACS.

6.8. Ensure that your service automatically starts on boot (or any other time the service dies).

> **Note:** The binary RPMs of Postgresql and AOLserver use the Red Hat init script system. So users of these RPMs can simply do
>
> ```
> /sbin/chkconfig postgresql on
> /sbin/chkconfig aolserver on
> ```
>
> to ensure that these services are restarted at boot time.

This section was taken from The Hitchhiker's Guide to the ACS , written by the ArsDigita folks.

This step should be completed as root. This can break every service on your machine, so proceed with caution.

- Copy this restart-aolserver into /tmp/restart-aolserver.txt
- This script needs to be SUID-root, which means that the script will run as root. This is necessary to ensure that the aolserver processes are killed regardless of who owns them. However the script should be in the *web* group to ensure that the users updating the web page can use the script, but that general system users cannot run the script. You also need to have Perl installed and also a symbolic link to it in /usr/local/bin.

```
$ su - ; Enter root password.
# cp /tmp/restart-aolserver.txt /usr/local/bin/restart-aolserver # chown root.web
/usr/local/bin/restart-aolserver
# chmod 4750 /usr/local/bin/restart-aolserver
# ln -s /usr/bin/perl /usr/local/bin/perl
# su - nsadmin
```

- Test the *restart-aolserver* script by making sure all servers are dead, starting a new server, and then killing it. You should see the following lines. nsadmin and typing

```
$ killall -9 nsd
nsd: no process killed
$ /home/aolserver/bin/nsd -u nsadmin -g web -t /home/aolserver/service_name.tcl
$ restart-aolserver service_name
Killing 23727 23728 23729 23730
$ killall -9 nsd nsd: no process killed
```

The numbers indicate the process ids (PIDs) of the processes being killed. It is important that no processes are killed by the second call to killall. If there are processes being killed, it means that the script is not working.

- Assuming that the restart-aolserver script worked, login as root and open /etc/inittab for editing.

```
$ su - ; Enter root password
# emacs -nw /etc/inittab
```

- Copy this line into the bottom of the file as a template, making sure that the first field nss is unique.

```
nss:2345:respawn:/home/aolserver/bin/nsd -u nsadmin -g web -i -t
/home/aolserver/service_name.tcl
```

- *Important:* Make sure there is a newline at the end of the file. If there is not a newline at the end of the file, the system may suffer catastrophic failures.
- Still as root, enter the following command to re-initialize /etc/inittab.

```
# killall -9 nsd
# /sbin/init q
```

- *Important:* See if it worked by running the restart-aolserver script again.

```
# restart-aolserver service_name Killing 23750 23753 23754 23756
```

If the processes were killed, congratulations, your server is now automated for startup and shutdown.

---

7. Everything works, now what ?

- Add your site to the OpenACS Sites list: http://www.openacs.org/sites.html
- Read the OpenACS Getting Started Guide.
- Read the Installing Monitors section of The Hitchhiker's Guide to the ACS. It will teach you how to install programs that will monitor your web server and take actions in case something goes wrong. It is written for ACS/Oracle, but it can be easily adapted to OpenACS (except for Cassandracle). We intend to port those documents to OpenACS in the near future.

7.1. Backing up your PostgreSQL databases

See *Backup Tips from Don Baccus* , in the *OpenACS Getting Started Guide*.

7.2. If you are using the e-commerce module

You should be aware that the e-commerce module makes use of the *ImageMagick* suite of graphics manipulation programs to generate picture thumbnails, especifically the *convert* program. Unfortunately the path to this program is hard-coded into the OpenACS code and this path may not match where your copy of *convert* is installed.

To find out where your *convert* is and make a symlink from it to where OpenACS thinks it is installed, do the following:

```
# which convert
/usr/X11R6/bin/convert
ln -s /usr/X11R6/bin/convert /usr/local/bin/convert
```

8. Where do go I for help and how can I help ?

- http://openacs.org/bboard - the OpenACS installation and configuration bulletin board.
- http://openacs.org/sdm - all bug reports and feature requests should go here (including corrections/suggestions to this documentation).
- If you want to help with the OpenACS project, email ben@mit.edu
- http://yourservername.com/doc. Once you have your OpenACS system setup, the /doc directory contains documentation about all modules and many other aspects of the toolkit.

8.1. Why Not MySQL ?

We get this question many times. Ben Adida wrote a short article explaining why OpenACS doesn't use MySQL (which made into Slashdot, giving openacs.org 37,000+ hits on that day). Read it at http://www.openacs.org/why-not-mysql.html.

You can read more about this on the article that explains why Sourceforge.net has moved away from MySQL, at http://www.phpbuilder.com/columns/tim20001112.php3.

8.2. Some Useful Links

- aolserver.com and especially aolserver.com/doc for AOLserver info and documentation (also included in the source distribution).
- ArsDigita's AOLserver Presentation
- ArsDigita Systems Journal for lots of good quality information on AOLserver, the ACS and building great web services.
- OpenForce Inc, provides commercial support, customization and other services using OpenACS.
- PostgreSQL's TechDocs, has lots of useful extra documents, tips and tricks for PostgreSQL.

9. Contributed Items

Several user have contributed to the OpenACS project. These are the items that our community has found useful and requested to be included in the documentation:

- Backup Tips from Don Baccus- Tips for optimizing and backing up your PostgreSQL databases, including a script that can be used from within OpenACS.
- Red Hat Updater- Michael Cleverly has written this Tcl script to keep your Red Hat box up-to-date.

10. Acknowledgements

Several people have contributed to this document in either content or error reporting and I would like to thank them. Please let me know if I forgot to include your name.

Contributors (in no specific order):

Philip Greenspun, Ben Adida, Don Baccus, Michael Cleverly, Janne Blonqvist, Jonathan Ellis, Janine Sisk, Jade Rubick, Chris Hardy, Jonathan Marsden.

OpenACS Getting Started Guide

Roberto Mello (rmello@fslc.usu.edu)

This is a guide to help you get started with OpenACS. This is not a complete guide to OpenACS and its modules... it would be too long. For detailed documentation on a module, refer to the www/doc directory in the OpenACS distribution. This guide assumes you have a working installation of OpenACS.

This is the February 2001 revision of the documentation.

Table of Contents

Next
Thinking about your web service

1. Thinking about your web service

OpenACS is a toolkit for online communities. This is a pretty powerful concept. If you think that most of the successful websites are community-oriented, you'll begin to see the power that OpenACS brings to you.

Most people get too focused on how their website is going to look, rather than thinking what it's going to provide for the users. Looks are important, but if you don't have content and services well organized, you can have as much looks as you want. Yahoo.com is pretty much all text-based, but it's fast and provides lots of content to users, or magnet content as Philip Greenspun calls.

OpenACS has over 40 modules to facilitate collaboration among users. You need to think which of these modules you can use in your web service and what's the best way to do it. A complete list of OpenACS modules is at http://www.arsdigita.com/pages/toolkit. After you have that planned and defined, you can start worrying about look and feel. You should also read the Webmasters Guide, which is at /doc/webmasters.html.

Refer to Philip and Alex's Guide to Web Publishing for more info on how to build great web services.

2. OpenACS Basics

2.1. Admin pages are your friends

For most of the OpenACS modules, you can configure them through the admin pages, accessible through */admin* from any browser. You must be registered as a site-wide administrator to access those pages (refer to the installation guide to know how to do that).

By visiting the /admin pages, you'll have most of the modules in front of you, and you'll have the opportunity to set them up. This is a great way to explore the possibilities that OpenACS opens to you.

If you see a module that you don't understand how to setup, just visit the /doc directory to learn how that module works and what it's intended to do.

For the ecommerce module, type in */admin/ecommerce* to be taken to its admin pages.

2.2. Content Sections

URL: /admin/content-sections

The content sections module allows you to add a section of the website, static or dynamic, to the users' Workspace and tell if that's publicly available or only to registered users, plus provide an introduction and help to the users.

2.3. User Groups

URL: /admin/ug

User groups are a great feature of OpenACS and gives you lots of flexibility. Unfortunately, as of this release, not all modules are user groups-ready, but that should change with OpenACS 4 (coming this summer).

There are *User Group Types* and *User Groups*. One user group type can have several user groups, which in turn can have several *subgroups*. Each user group can have modules associated with it, that members of that group will be able to use and that will be specific to that group.

For example, in the USU Free Software and GNU/Linux Club we had several projects that were being carried by club members and we wanted to provide a way for them to collaborate. Something besides mailing lists. So we created a group type called projects and several groups inside it, one for each project. Each group had an administrator assigned to it, who had control over that group an*d that could create subgroups if he/she thought necessary. OpenACS provides group pages, all ready, in* /groups.

There's much power in user groups and I highly recommed you to read the documentation about it on /doc/ug.

Prev Home Next
Thinking about your web service Look and feel

3. Look and feel

There's quite a bit of information about this on *Establishing Style and Supporting Multi-Lingualism* at /doc/style.html.

Basically you have four ways of modifying the look and feel of your OpenACS website:

1. Editing ad_header and ad_footer
2. Customized ADP tags
3. Templates
4. ArsDigita Templating System [1]

3.1. Editing ad_header and ad_footer

Almost all of the tcl pages shipped with OpenACS make calls o ad_header and ad_footer, two procedures defined in *tcl/ad-defs.tcl.preload*

ad_header returns the initial HTML tags and title to the page, and optionally returns extra stuff for the <head>. This is its usage:

```
ad_header page_title { extra_stuff_for_document_head "" }
```

ad_footer stuff an <hr> and a e-mail signature (defaults to system owner) and then it closes the body and html tags. Its usage is:

```
ad_footer { signatory "" } { suppress_curriculum_bar_p "0" }
```

For example, you'd call them from an adp page like this:

```
<%= [ad_header  My First OpenACS Page ] %>
<%= [ad_footer  me@mydomain.com ] %>
```

The <%= means evaluate this and then return as a ns_puts . The above would return a page with the title My First OpenACS page and an e-mail signature in the bottom saying me@mydomain.com .

Ths disadvantages of this approach is that it's very limited and requires restarting AOLserver for changes in the ad-defs file to take effect.

3.2. Customized ADP tags

OpenACS has some utility procedures to help you. One of them is *ad_register_styletag* (defined in tcl/ad-style.tcl).

With ad_register_styletag you can register a tag can will be available for use under ADP and TCL pages. It will also register documentation for that tag through the proc_doc OpenACS procedure.

Usually what I do is rename the file *tcl/photonet-styles.tcl* to *tcl/myservice-styles.tcl* and then edit it. In that file I define some tags like pagetop , pagebottom , pageside . Then I call these tags from ADP pages just like regular HTML tags (e.g. <pagetop></pagetop>). From .tcl pages, you'd call these tags through ad_style_pagetop (or [ad_style_pagetop] if you are calling it from inside a ns_write).

ad_register_styletag used the AOLserver API call to ns_register_adptag, which will give you more flexibility on defining your tags (e.g. you can define tags that take arguments). Read the documentation for ns_register_adptag for more info.

Although this approach is more flexible, it also requires you to restart AOLserver to make changes.

3.3. Templates

Templates are very flexible and do not require an AOLserver restart. You can find full documentation on this at /doc/style.html.

Basically, if you have a .tcl page that has the Tcl and SQL code in /test/mypage.tcl, you have this page call *ad_return_template* at its bottom. ad_return_template will search for an .adp template at */templates/test/mypage.*.adp* and return it.

You can also use *ad_return_template template_file*, where *template_file* is a .adp file in your template subdirectory that should be returned to the user.

You can have several templates with different interfaces or even languages, such as mypage.plain.adp (for users than want text-only) or mypage.fancy.adp (for users that want graphical site) or mypage.plain.pt.adp (for users that speak Portuguese). ad_return_template will server the page according to the user's preferences.

This means that programmers will edit the .tcl pages and define some variables that HTML designers will then use in the the .adp templates they will handle. If I am not mistaken, there are mods for Dreamweaver to handle adp pages.

Unfortunately, because the templating module is fairly new, only a few modules are template-enabled, ecommerce being one of them.

3.4. ArsDigita Templating System (ATS)

ArsDigita has written a very powerful publishing system that was included in ACS/Oracle 3.2.3. Older incarnations of this system could be used with OpenACS with very few modifications. Newer versions -designed for ACS 4.x- however, are not so easy. We are looking into a port of the templating system done by Vlad Seryakov, and the templating system will probably be included in the next release of OpenACS. Stay tuned!

The ATS uses XML and defines some special tags that completely separates programming from presentation. People that have been using it told me that it's really nice.

You can find more information about the ATS at http://developer.arsdigita.com/doc/acs-templating.

Notes

[1]

---

4. Backups tips from Don Baccus

4.1. The Strategy

The need for making backups should be self-explanatory. There are several strategies you can use. My own strategy for minimizing the odds that I'll lose all my data is quite simple:

- The database is stored on a mirrored (RAID 1) disk.
- The machine has battery backup.
- Backups are made nightly onto a third disk on another controller
- ftp is used to copy the resulting backup to two separate remote servers in two locations

Rather than making remote copies, you might choose to dump to tape or writeable CD media. Whatever strategy you use, it is important to routinely check dumps to make sure they can be reloaded. The strategy outlined above means that in the case of catastrophic failure, I'll lose at most one day's data.

By mirroring disks and using a battery backup, preferably one that can trigger an automatic and controlled shutdown of the system when the battery runs low, you greatly lower the odds of ever having to use your nightly backup. Despite this, it is important to take backups seriously if the data stored at your site is valuable to you or your users.

It is also important that you use the Postgres dump utility, *pg_dump*, rather than simply copy the files in the database directory unless you stop the Postmaster while making your copy. If you copy the files while the Postmaster is running and updates to the database are being made, you'll end up with inconsistent tables and a potentially unusable database. *pg_dump* makes a consistent dump even when the database is in use.

I find it convenient to use AOLserver's schedule proc routine to run a Tcl script which generates my nightly backups, rather than use cron. The major benefit is that you can very easily make a web page that calls this script, and link to it from an admin page. If there are problems making your backups (connectivity problems seem to crop up a few times a year, in my experience), you can just click on the link to force a backup with no need to remember where you placed the script, what you called it, any arguments that might be needed, etc.

4.2. The *vacuum analyze* command

Currently, Postgres doesn't automatically reclaim space in a database table when an existing row is deleted or updated.

The "vacuum" command must be run periodically to reclaim space. The "vacuum analyze" form additionally collects statistics on the disbursion of columns in the database, which the optimizer uses when it calculates just how to execute queries. The availability of this data can make a tremendous difference in the execution speed of queries. I run this command as part of my nightly backup procedure - if "vacuum" is going to screw up my database, I'd prefer it to happen immediately after (not before!) I've made a backup! The "vacuum" command is very reliable, and has never caused me a problem, but conservatism is the key to good system management.

4.3. Sample Tcl script

Here's a sample script based on the one used to back up the database backing my most important personal site, "birdnotes.net". If you're wondering why this procedure doesn't backup the scripts for the site as well, it is because they're developed on a local machine, which is backed up separately.

```
# Back up the database, scheduled to be run nightly. As written, it     # keeps a
month's worth of daily backups, cycling over the same files   # which are suffixed
with the day of the month on which the backup is   # created.      # This version:
ftp only.       # NOTE: The indenting gets screwed up during conversion to HTML
proc backup {} {
  # Set these to the appropriate values for your installation.
  set b "/usr/local/pgsql/bin"
  set bak "/home/birdnotes_backup/"
  set db [ns_db gethandle]
  set sql "select date_part('day','today'::date) as day"
  set selection [ns_db 1row $db $sql]
  set_variables_after_query
  set data "birdnotes_$day.dmp"
  ns_log Notice "Backup of [ad_system_name] starting."
  ns_log Notice "pg_dump beginning..."
  if [catch {append msg [exec "$b/pg_dump" "birdnotes" ">$bak/$data"]} errmsg] {
    ns_log Error "pg_dump failed: $errmsg"
   ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : pg_dump
failed..." "$errmsg"
     ns_db releasehandle $db
     return
}
  append msg "\n"
  ns_log Notice "gzip of data beginning..."
  if [catch {append msg [exec "gzip" "-f" "$bak/$data"]} errmsg] {
     ns_log Error "gzip of data failed: $errmsg"
  ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : gzip of data
failed..." "$errmsg"
     ns_db releasehandle $db
     return
  }
  append msg "\n"
  ns_log Notice "ftp data beginning..."
  set fd [open "$bak/ftp_data.tmp" w]
  # Replace "your_username", "your_password", and "your_remotedir" with the values
  # appropriate for the remote system on which you're keeping backup copies.
 puts $fd "user your_username your_password\nbinary\nput $bak/$data.gz
your_remotedir/$data.gz\nquit\n"
    close $fd
# "your_remoteserver" should be set to the IP of the remote system which stores your
# backups.
  if [catch {append msg [exec "ftp" "-n" "your_remoteserver" "<$bak/ftp_data.tmp"]}
errmsg] { \
     ns_log Error "ftp data failed: $errmsg"
    ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : ftp data
failed..." "$errmsg"
    ns_db releasehandle $db
    return
  }
  append msg "\n"
  # Replicate the above code to make remote copies to other systems
  ns_log Notice "vacuum beginning..."
  if [catch {append msg [exec "$b/psql" "-q" "-c" "vacuum analyze"]} errmsg] {
     ns_log Error "vacuum failed: $errmsg"
    ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : vacuum
```

```
failed..." "$errmsg"
    ns_db releasehandle $db return
  }
  ns_db releasehandle $db
  ns_log Notice "Backup succeeded."
  append msg "Backups succeeded"
  ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : backup
succeeded" "$msg
}
ns_share -init {set schedule_backup 0} schedule_backup
if {!$schedule_backup} {
  ns_schedule_daily 0 backup
  ns_log Notice "Backup has been scheduled."
}
```

5. Intranet Getting Started Guide

This section was written by Jade Rubick with slight modifications by Roberto Mello. The ACS Intranet module is being used by large corporations such as Siemens to manage their intranets.

Congratulations, you've just installed ACS. You've managed to get through the install guide (or the OpenACS install guide), you've gotten Oracle or Postgres working, you're now a pretty cool person because you got it working.

Well, if you're using the Intranet module, you're going to need some more setup, and currently, I don't know of any good documentation on the module except for what is at ArsDigita. Read that first.

The Intranet is a pretty complicated module because it relies conceptually on so many other modules. Unless you already understand the ACS, it may take a while to get a handle on it.

First of all, if you're using Postgres 7.02, I recommend applying a patch that fixes the http://www.postgresql.org/bugs/bugs.php?4~2. Preferably, you would install this before installing OpenACS, because otherwise you have to export and import your data to get this working. But the directions are pretty good, so just follow them carefully. I'm not sure if this works completely though -- I started from a fresh copy. (*NOTE: This is most likely fixed in PostgreSQL 7.0.3 and later versions.*)

Steps to getting your intranet running:

- Set up your admin account and so on like the directions say.
- Go through your */web/servicename/parameters/servicename.tcl* or .ini file and set the preferences for Intranet and New-ticket modules. Make sure you enable the Intranet module.
- In your web browser, go to */admin/users* and add in the users you want. I assume these are Employees -- they won't be able to use the Intranet unless you then go to /admin/ug and click on Intranet. Add them to both the Authorized Users and Employees category.
- While you're in the Intranet/Employees section of */admin/ug*, click on "*add module*" under "*Modules associated with groups in Employees*". Set up the news module. They will then be able to make company-wide postings for the main page, and be able to see other postings. Otherwise, they will get an error when they click on the post an item link on the main intranet page.
- Go to the /intranet page. Click on "*Offices*", and set up an office.
- Set up your partner types. This is a pain, unless I'm overlooking an easier way to do it. You have to open up a shell, start up postgres, and change the categories in the database. Then, you have to restart AOLserver, because I believe the variables are cached in memory.

```
[postgres@intranet pgsql]$ psql intranet (or your servicename)
Welcome to psql, the PostgreSQL interactive terminal.

Type: copyright for distribution terms
      h for help with SQL commands
      ? for help on internal slash commands
      g or terminate with semicolon to execute query
      q to quit

intranet=# select * from im_partner_types ;

partner_type_id |    partner_type    | display_order
----------------+--------------------+---------------
              1 | Usability          | 1
              2 | Graphics           | 2
              3 | Strategy           | 3
              4 | Supplier           | 4
              5 | Sys-admin          | 5
              6 | Hosting            | 6
              7 | Systems Integrator | 7
(7 rows)

intranet=# update im_partner_types set partner_type = 'Service Provider' intranet
-# where partner_type = 'Usability';
UPDATE 1

intranet=# select * from im_partner_types ;
partner_type_id |     partner_type       | display_order
----------------+------------------------+---------------
              2 | Graphics               | 2
              3 | Strategy               | 3
              4 | Supplier               | 4
              5 | Sys-admin              | 5
              6 | Hosting                | 6
              7 | Systems Integrator     | 7
              1 | Service Provider       | 1
(7 rows)
```

- Don't forget to restart Aolserver
- If you don't have site-wide-search, you might want to disable the site-wide-search box on the main /intranet page.
- Set up the calendar categories in /calendar/admin. Look at the documentation that won't be relevant to this at all :) Think of categories like: Social, Project Meeting, etc...
- Get pictures for everyone in your office and put them in. Or let everyone do it themselves.
- Add in any Discussion Groups you might want.
- Add in teams for the *new-ticket* module. Go to */team* to add in teams. Make sure everyone who will be using the ticket system is part of a team. If you don't do this, tickets won't work.
- Change the project types if the project types don't fit what you need. I haven't done this yet, so no help here. Sorry!

Backups tips from Don Baccus

Acknowledgements

6. Acknowledgements

Several people have contributed to this document in either content or error reporting and I would like to thank them. Please let me know if I forgot to include your name.

Contributors (in no specific order):

Don Baccus, Michael Cleverly, Gregory McMullan, Ben Adida, Jade Rubick.

Simple AOLserver Install Guide

Roberto Mello (rmello@fslc.usu.edu)

A simple AOLserver installation guide. My development box is a marvelous Debian GNU/Linux system, so if this doesn't work on a platform other than GNU/Linux, it's probably due to some specific thing.

This is the April 2001 revision of the documentation.

Table of Contents

Next
Getting the best web/application server up and running

1. Getting the best web/application server up and running

1.1. What you need

- GCC (GNU Compiler Collection)
- GMake (GNU Make - Default in GNU/Linux)
- Development libraries including glibc and libpthread
- GNU tar and Gzip

1.2. Downloading and UN-tarring

Create a user and group for AOLserver, such as nsadmin. In GNU/Linux you would do:

```
addgroup nsadmin
adduser -g nsadmin nsadmin
```

Get the AOLserver source (not the binaries, unless you know what you're doing) distribution from http://www.aolserver.com/download.

Untar it at some temporary directory such as /usr/local/src with the following command:

```
tar xzvf aolserver3_0.tar.gz
```

1.3. Building and Installing

cd into the newly created directory (aolserver3_0 for AOLserver 3.0) and do a:

```
make (or gmake if you are not using GNU/Linux)
```

Once the compilation process is over, it's time to install AOLserver. Choose a directory to install AOLserver at, such as /home/aolserver. Then do a:

```
make install INST=/home/aolserver
```

Now change the ownership of the directory where you installed AOLserver with:

```
chown -R nsadmin:nsadmin /home/aolserver
```

1.4. Testing and Configuring

AOLserver comes with two nsd (the main AOLserver daemon) files: nsd76 and nsd8x. nsd76 uses Tcl 7.6, which is an AOLserver-hacked Tcl version highly optimized for multithreading and performance. nsd8x uses Tcl 8.3 which offers some nice extra features, namely better regular expressions, byte code compiler and other things. As of AOLserver 3.3 (which corrected a memory leak on nsd8x), nsd76 is officially being deprecated. Simply make a symbolic link to the file of your choice (nsd - nsd76 or nsd - nsd8x).

To do a quick-test on your AOLserver installation, CD into the AOLserver directory and (as the AOLserver user - e.g. nsadmin) do a:

```
./bin/nsd -kt nsd.tcl
```

Then with your browser, go to yourdomain.com:8000 and you should see AOLserver's default page.

Find AOLserver's PID (ps aux | grep nsd) and kill it (kill -9 pid). You can then use our nsd.tcl to configure your AOLserver. More information on the nsd.tcl config file can be found in doc/config.txt in the AOLserver source tree.

1.5. Setting it to be restarted

AOLserver must be started as root so it can use the priviledged ports 80 and 443. You need to pass the user and group to which it will switch right after starting.

The most common way to have AOLserver restarted even if the computer needs to be rebooted is by including it in your */etc/inittab* file. Include a line like this in your /etc/inittab:

```
as:2345:respawn:/home/aolserver/bin/nsd -it /home/aolserver/nsd.tcl -u nsadmin -g
nsadmin
```

As of OpenACS 3.2.2, if you need to you can restart AOLserver right from the OpenACS admin pages. To restart it from the command line use the -k flag to nsd (e.g. /home/aolserver/bin/nsd -kt /home/aolserver/nsd.tcl -u nsadmin -g nsadmin).

1.6. Securing Your Installation

It is safer to run AOLserver in a chrooted environment. I am not including instructions here for simplicity's sake. Follow these links to learn how to do it.

http://aolserver.com/docs/admin/sec-ch2.htm#8685

http://www.arsdigita.com/doc/security

Simple PostgreSQL Installation Guide

Roberto Mello (rmello@fslc.usu.edu)

A very simple PostgreSQL installation guide to aid OpenACS users, based on the PostgreSQL Administrators Guide. Not intended to be a full-blown guide to every imaginable architecture on the planet. This was written in a *GNU/Linux* box, so some adaptation may be necessary for other platforms. This guide does not apply for Windows environments. Check the PostgreSQL documentation for more details on installing PostgreSQL on Windows and other platforms (Chapter 5 of the Administrators Guide).

Table of Contents

Next
Before you start

1. Before you start

- You will need GNU Make to build PostgreSQL. GNU Make is the default in GNU/Linux systems, in other systems it may be called gmake. We are simply going to refer to it as make . You can download GNU Make from ftp://ftp.gnu.org.
- You will need a C compiler. GCC (GNU Compiler Collection) is what I am using.
- You will need C/C++ development libraries. On GNU/Linux systems, this usually means glibc and glibc-devel.
- For convenience, install GNU Readline and its development packages (usually readline and readline-devel).
- (Optional) If you are going to use psql from an X Terminal, then install the X development libraries.
- A minimum of 35 Mb of disk space is necessary to build PostgreSQL (including the sources). Of course, you will need more to create databases, so be generous :-)

1.1. Should I compile or use a binary (RPM, DEB) ?

Compiling gives you more flexibility because you can compile PostgreSQL with some extra options (and we will mention some here). It is not a super trivial process, but it is not too hard. Following this guide will help a lot, but some understanding of what's going on is expected of you.

If you are completely new to UNIX systems or if you just want to get it up and running fast, then go with a binary installation, such as the ones provided by RPMs (Red Hat packages, provided in most GNU/Linux distributions) and DEBs (Debian GNU/Linux packages).

2. Compiling and Installing the best free RDBMS

2.1. Preliminaries

As *root*, create a user and group (if you haven't done so before) for PostgreSQL. This is the account that PostgreSQL will run as since it will not run as root. Also give the postgres user a password:

```
groupadd web
useradd -g web postgres -d /usr/local/pgsql
passwd postgres
```

Download PostgreSQL from the mirror closest to you. The list of mirrors is at http://www.postgresql.org/sites.html. You can download it in one big file (Ü 8 Mb) or in smaller files (base, support, doc, test). Put it in a temp directory such as /tmp.

Untar with the command:

```
tar xzvf postgresql-xxxxx.tar.gz
```

Repeat the tar command for each file if you are downloading the multiple-file distribution.

2.2. Optional - How to increase the blocksize in PostgreSQL [1]

This is not required to compile PostgreSQL.

By default PostgreSQL is compiled with a blocksize of 8 Kb. You can compile PostgreSQL to have 16 Kb blocksize instead, which will allow for bigger text and lztext data types. NOTE: If you've built PostgreSQL in that directory before, you need to a) delete the config.cache file (created by autoconf) and b) do a make clean to give it a fresh start.

Here's how you do it:

1. Go into the src/include directory of the PostgreSQL source distribution.
2. Edit the file config.h.in
3. Look for the line *#define BLCKSZ 8192* and change it to *#define BLCKSZ 16384*
4. Save it.

2.3. Compiling and installing

* You will need to run the command ./configure from within the src/ directory of the PostgreSQL distribution. This will create the files with the options that will be used to compile PostgreSQL specifically to your machine (./configure --help will show you all the available options).

* If you are going to access psql (the PostgreSQL's interactive query tool) from within an X terminal (such as xterm, kterm, Eterm,...) and you have the X development libraries installed then do:

```
./configure --with-x
```

Otherwise simply do a *./configure*

* After it's done fire up a:

```
make
```

* The compilation process will take place. Go look at OpenACS and see on what you can help make the toolkit even more butt-kicking. After it's done, you should see a message like this:

```
All of PostgreSQL is successfully made. Ready to install.
```

* If you are upgrading your PostgreSQL, follow the instructions on Chapter 5 of the Administrators Guide.

* Create the directories to house PostgreSQL:

```
mkdir /usr/local/pgsql
chown postgres.web /usr/local/pgsql
```

* Become the postgres user: su - postgres

* Then install the PG executable files and libraries by running:

```
make install
```

* Then you need to initialize the database templates and other things by doing this:

```
# su - postgres
$ cd /usr/local/pgsql
$ mkdir data
$ cd bin
$ ./initdb -D /usr/local/pgsql/data
```

* Tell your system how to find the new shared libraries. This can be accomplished in two ways:

Editing the file /etc/profile and adding the lines:

```
LD_LIBRARY_PATH=/usr/local/pgsql/lib
export LD_LIBRARY_PATH
```

OR editing the file /etc/ld.so.conf and adding the line:

```
/usr/local/pgsql/lib
```

and then running the command /sbin/ldconfig.

* You probably want to install the man and HTML documentation. Type

```
cd /usr/local/src/pgsql/postgresql-7.0/doc (or wherever you untarred it)
make install
```

This will install files under /usr/local/pgsql/doc and /usr/local/pgsql/man.

* Setup some environment variables to make your life easier:

You can do this by editing the /etc/profile file and including these lines:

```
PATH=$PATH:/usr/local/pgsql/bin
MANPATH=$MANPATH:/usr/local/pgsql/man
export MANPATH
```

* As the user *postgres*, create the database installation (the working data files). It can be anywhere that is writable by the postgres user (usually /usr/local/pgsql/data):

```
mkdir /usr/local/pgsql/data
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

* Start the database server with the command:

```
/usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
```

This will start the server in the foreground (CTRL+C to stop it). To make it detach to the background, you can use the -S option, but then you won't see any log messages the server produces. A better way to put the server in the background is :

```
nohup /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data < /dev/null >>
server.log 2>>1 &
```

2.4. Getting it to start with every boot

Unfortunately there's no easy answer here. Every system (even GNU/Linux distributions) is a bit different than the other. In Red Hat / Mandrake systems I modify the postgresql startup script that comes with the RPM distribution to fit a compiled version.

In the *contrib/linux* directory of the PostgreSQL source distribution there's a sample startup script that you can use to include in your /etc/init.d directory.

2.5. Testing

After you've started the database server, try creating a database and connecting to it through psql:

```
createdb mydb
psql mydb
```

If you see a prompt after a few messages, you're set ! Congratulations.

Notes

[1] This is unnecessary in PG 7.1 as it is rid of the blocksize limit.

---

3. Installing Binary Packages

3.1. RPMs

* Download the RPMs form the PostgreSQL mirror closest to you (list at http://www.postgresql.org/sites.html)

The RPMs come split into separate packages so you can install only what you need. Download the packages:

postgresql-7.xxx.i386.rpm

postgresql-devel-7.xxx.i386.rpm

postgresql-server-7.xxx.i386.rpm

postgresql-test-7.xxx.i386.rpm (for the regression tests).

(if you want pgaccess, PostgreSQL graphical tool, also install postgresql-tk-7.xxx.i386.rpm)

* Install them in the sequence above, with the command: (as root)

```
rpm -ivh postgresql-xxx.i386.rpm
```

If it asks for dependency packages, install them from the CD or after downloading them (from rpm.net for example).

The RPM packages will very nicely create the postgres user and group, set the environment variables, install the documentation and other goodies for you. Cozy huh ? Thank Lamar Owens for that.

* Then start the database server with the command:

```
/etc/rc.d/init.d/postgresql start
```

If everything went fine, you should be able to, as user *postgres*, create a database and connect to it through psql:

```
su - postgres
createdb mydb
psql mydb
```

3.2. DEBs

* The DEB packages come split into separate packages so you can install only what you need.

As root, fire up *dselect* and go into *Select* .

Do a / postgresql to search for postgresql packages. Search again by pressing \

Mark for installation (by pressing +) the following packages:

postgresql

postgresql-pl

postgresql-client

postgresql-dev

postgresql-doc

postgresql-test

(if you want pgaccess, PostgreSQL graphical tool also install the package pgaccess)

Then hit Enter to get out of dselect and go into *Install* .

That's it. It should download, install and start the server for you. Hard huh ? Thanks to Oliver Elphick for providing these packages.

* If the setup process doesn't start the server, you can start it with: (as root)

```
/etc/init.d/postgresql start
```

* If everything went fine, you should be able to, as user *postgres*, create a database and connect to it through psql:

```
su - postgres
createdb mydb
psql mydb
```

4. Acknowledgements

Several people have contributed to this document in either content or error reporting and I would like to thank them. Please let me know if I forgot to include your name.

Contributors (in no specific order):

Don Baccus, Mark Rohn, Bob O'Connor.

Oracle to Postgres Conversion

by James Shannon, Ben Adida, and Don Baccus

What you should know before you begin

You should know SQL relatively well. Knowing the details of Oracle SQL and Postgres SQL are obviously tremendous advantages, but the hints in this document should quickly bring you up to speed on what the differences are.

If you're porting Oracle SQL to Postgres SQL for the ACS/pg, you should also be quite familiar with AOLserver Tcl, especially the AOLserver database APIs.

In this document, we're talking about:

- Oracle 8 and 8i
- Postgres 7.0, and sometimes this also works for Postgres 6.5.3

Grammar Differences

There are a handful of grammar differences in Postgres for functionality that is actually the same. ACS/pg attempts to perform these changes automatically, leaving only the major functionality differences to be ported by hand. This is done by `db_sql_prep` which performs a number of regular expression substitutions on a piece of SQL.

Sysdate

Oracle uses the keyword `sysdate` to denote the current date and time. Postgres uses `'now'::datetime`, which ACS/pg has conveniently wrapped in a function named `sysdate()`.

ACS/pg also includes a Tcl procedure named `db_sysdate` which should be used every time the `sysdate` term appears. Thus:

```
set now [database_to_tcl_string $db "select sysdate from dual"]
```

should become

```
set now [database_to_tcl_string $db "select [db_sysdate] from dual"]
```

The Dual Table

Oracle uses the "fake" dual table for many selects. This table was created in postgres as a view to ease porting problems. This allows code to remain somewhat compatible with Oracle SQL without annoying the Postgres parser.

Sequences

Oracle's sequence grammar is `sequence_name.nextval`.
Postgres's sequence grammar is `nextval('sequence_name')`.

In Tcl, getting the next sequence value can be abstracted by calling `[db_sequence_nextval $db sequence_name]`. In case you need to include a sequence's value in a more complex SQL statement, you can use `[db_sequence_nextval_sql sequence_name]` which will return the appropriate grammar.

Decode

Oracle's handy `decode` function works as follows:

```
decode(expr, search, expr[, search, expr...] [, default])
```

To evaluate this expression, Oracle compares expr to each search value one by one. If expr is equal to a search, Oracle returns the corresponding result. If no match is found, Oracle returns default, or, if default is omitted, returns null.

Postgres doesn't have the same construct. It can be replicated with:

```
CASE WHEN expr THEN expr [...] ELSE expr END
```

which returns the expression corresponding to the first true predicate. For example:

```
CASE WHEN c1 = 1 THEN 'match' ELSE 'no match' END
```

NVL

Oracle has another handy function: `NVL`. `NVL` returns its first argument if it is not null, otherwise it returns its second argument.

```
start_date := NVL(hire_date, SYSDATE);
```

The above statement will return `SYSDATE` if `hire_date` is null. Postgres has a function that performs the same thing in a more generalized way: `coalesce(expr1, expr2, expr3,....)` returns the first non-null expression that is passed to it.

Functional Differences

Postgres doesn't have all the functionality of Oracle. ACS/pg is forced to deal with these limitations with specific work-arounds. Almost everything can be done under Postgres, but some features are awaiting new versions of the open-source database.

Outer Joins

Outer Joins work as follows:

```
select a.field1, b.field2
```

```
from a, b
where a.item_id = b.item_id(+)
```

where the `(+)` indicates that, if there is no row in table `b` that matches the correct `item_id`, the match should still happen, with an empty row from table `b`. In this case, for example, for all rows in table `a` where there is no matching row in `b`, a row will still be returned where `a.field1` is correct, but `b.field2` is null.

Depending on the exact operation performed by the outer join, there are different approaches. For outer joins where specific, raw data is extracted from the outer-joined table (e.g. as above), it's best to use a `UNION` operation as follows:

```
select a.field1, b.field2
from a, b
where a.item_id = b.item_id
UNION
select a.field1, NULL as field2
from a
where 0= (select count(*) from b where b.item_id=a.item_id)
```

For queries with quadruple outer-joins, the queries can be quite long! They work quite well, though.

In certain other cases where only aggregate values are pulled out of the outer-joined table, it's possible to not use a join at all. If the original query is:

```
select a.field1, sum(b.field2)
from a, b
where a.item_id = b.item_id (+)
group by a.field1
```

then the Postgres query can look like:

```
select a.field1, b_sum_field2_by_item_id(a.item_id)
from a
```

where you've defined the function:

```
create function b_sum_field2_by_item_id(integer)
returns integer
as '
DECLARE
     v_item_id alias for $1;
BEGIN
     return sum(field2) from b where item_id= v_item_id;
END;
' language 'plpgsql';
```

Connect By

Postgres doesn't have connect by statements. Uggh. No easy way to do this.

CLOBs

Postgres doesn't have decent CLOB support. However, with the lztext extension coming with Postgres 7.0, there is no need for CLOBs in the ACS/pg. We'll be able to do everything using varchars. For now, we're using only varchar(4000).

BLOBs

Binary large object support in Postgres is very poor and unsuitable for use in a 24/7 environment, because you can't dump them with pg_dump. Backing up a database that makes use of Postgres large objects requires one to knock down the RDBMS and dump the files in the database directory.

Don Baccus put together a hack that extends AOLserver's postgres driver with BLOB-like support, by uuencoding/decoding binary files before stuffing them into or extracting them from the database. The resulting objects can be consistently dumped by "pg_dump" while the RDBMS is up and running. There is no need to interrupt service while making your backup.

To get around the one-block limit on the size of a tuple imposed by Postgres, the driver segments the encoded data into 8K chunks.

Postgres large objects are scheduled for a major overhaul in summer 2000. Because of this, only the BLOB functionality used by the ACS was implemented.

To use the BLOB driver extension, you must first create a column of type "integer" with the name "lob" in the table that will store the BLOB, and a trigger on it that calls "on_lob_ref". You **must** use the name "lob". Here's an example:

```
create table my_table (
    my_key                      integer primary key,
        lob                             integer references lobs,
    my_other_data       some_type -- etc
);

create trigger my_table_lob_trig before insert or delete or update
on my_table for each row execute procedure on_lob_ref();
```

To put a binary file into "my_table":

```
set lob [database_to_tcl_string $db "select empty_lob()"]

ns_db dml $db "begin"
ns_db dml $db "update my_table set lob = $lob where my_key = $my_key"
ns_pg blob_dml_file $db $lob $tmp_filename
ns_db dml $db "end"
```

Note that the call to ns_pg to stuff the file into the database MUST be wrapped in a transaction, even if

you're not updating any other tables at the same time. The driver will return an error if you don't.

To return a large object stored in "my_table" to the user:

```
set lob [database_to_tcl_string $db "select lob from my_table
                                     where my_key = $my_key"]
ns_pg blob_write $db $lob
```

Note that you don't need to wrap the call to blob_write in a transaction, as the database isn't being modified.

The large objects are automatically deleted when no longer used. To replace the large object stored in an existing record, just allocate a new one by calling "empty_lob()" and assign the returned key to the "lob" column in your table.

---

james@motifstudios.com / ben@adida.net