

The server core (**dgate.exe** = **dgate** under Linux) compiles and runs on Linux systems and Solaris. I develop primarily under Windows, but currently I test the code and scripts under Linux Ubuntu 18.04. I also had the server compiled on a Raspberry Pi.

The Linux release of the server core works default with SQLite driver built in into the server (no ODBC). The DbaseIII driver is also supported. Piotr Filipczuk has added a PostGresQL driver. The native MySQL interface also can be used. The graphical user interface has not been ported to Linux, but the WEB interface is provided, either using Apache or a built-in mini web server (Ladle). In this version, most options have been well tested – it is a stable release. However, there are often subtle differences between linux distributions, making installation (and writing a manual) difficult. There are several contributions on the forum, and there are text files with specific command orders to be found in the linux subfolder of the server.

To use the server, one needs a valid version of the configuration files and put them in the same directory as the dgate executable. The easiest way to do this is to unpack **dicomserver150.zip** with “unzip dicomserver150.zip”.

INSTALLATION

Prerequisites: 1) a running Linux system. 2) sudo installed and enough rights. If not, the script will not be able to install the server as web service for Apache and you need to copy the files by hand. Note that I only test the scripts on Ubuntu, but the web based installer script linux.sh has a bit of info on Fedora.

These packages needed to be installed in a plain Linux system for a release using SQLite or DbaseIII:

sudo apt update	get compilers
sudo apt install make	may be skipped if using precompiled
(or: sudo apt install build-essential)	
sudo apt install g++	may be skipped if using precompiled
sudo apt install apache2	get webserver
sudo apt install unzip	not standard in Ubuntu server
sudo apt install p7zip-full	parts of the web interface use 7za
sudo apt install lua5.1	since 1.5.0 lua is external
sudo apt install lua5.1-dev	may be skipped if using precompiled
(or sudo apt install liblua5.1-0)	is required when using precompiled
sudo apt install lua-socket	

(or for fedora:

```
dnf install gcc-c++-sh-linux-gnu.x86_64 gcc-c++-x86_64-linux-gnu.x86_64 clang.x86_64
)
```

sudo a2enmod cgi	enable CGI in web server
systemctl restart apache2	
(or for older systems: sudo service apache2 restart)	

The rest of the installation can be performed manually, or by a web based method, explained below.

The following steps illustrate a minimal installation:

First get the server:

```
wget http://ingenium.home.xs4all.nl/dicomserver/dicomserver150.zip
```

get server zip

```
mkdir conquest
```

make folder to store conquest
to there

```
cd conquest
```

```
unzip ../dicomserver150.zip
```

```
rm ../dicomserver150.zip
```

Or:

```
sudo apt install git
```

if git not installed yet

```
git clone https://github.com/marcelvanherk/Conquest-DICOM-Server
```

get latest from GitHub

```
cd Conquest-DICOM-Server
```

Then compile and install it:

```
chmod 777 maklinux
```

```
./maklinux
```

```
choose option 3 or 5
```

```
say 'y' to 'Regenerate the database'
```

compile and install web access

SQLite or SQLite precompiled

Deletes previous database contents

```
say 'y' to 'Install as service' Shows status hit 'q' to return
```

Now the server should be running and <http://localhost/cgi-bin/newweb/dgate> should provide a working web interface.

Note that in [dicomserver150](#) a precompiled dgate (compiled by me on Ubuntu 14.04, using SQLite database) is included, to try that use option 5 in *maklinux*. Tested on Ubuntu 18.04, 19.10. If used the following packages may be *omitted*: **make, g++, lua5.1-dev**; but if you do omit them then the following package must be *added*: **liblua5.1-0**. This option reduces the size of the Linux system by a few hundred MB.

Web based installation

To run the web based installer (after installing prerequisites):

```
wget http://ingenium.home.xs4all.nl/dicomserver/dicomserver150.zip
```

get server zip

```
mkdir conquest
```

make folder to store conquest

```
cd conquest
```

```
unzip ../dicomserver150.zip
```

```
cd install
```

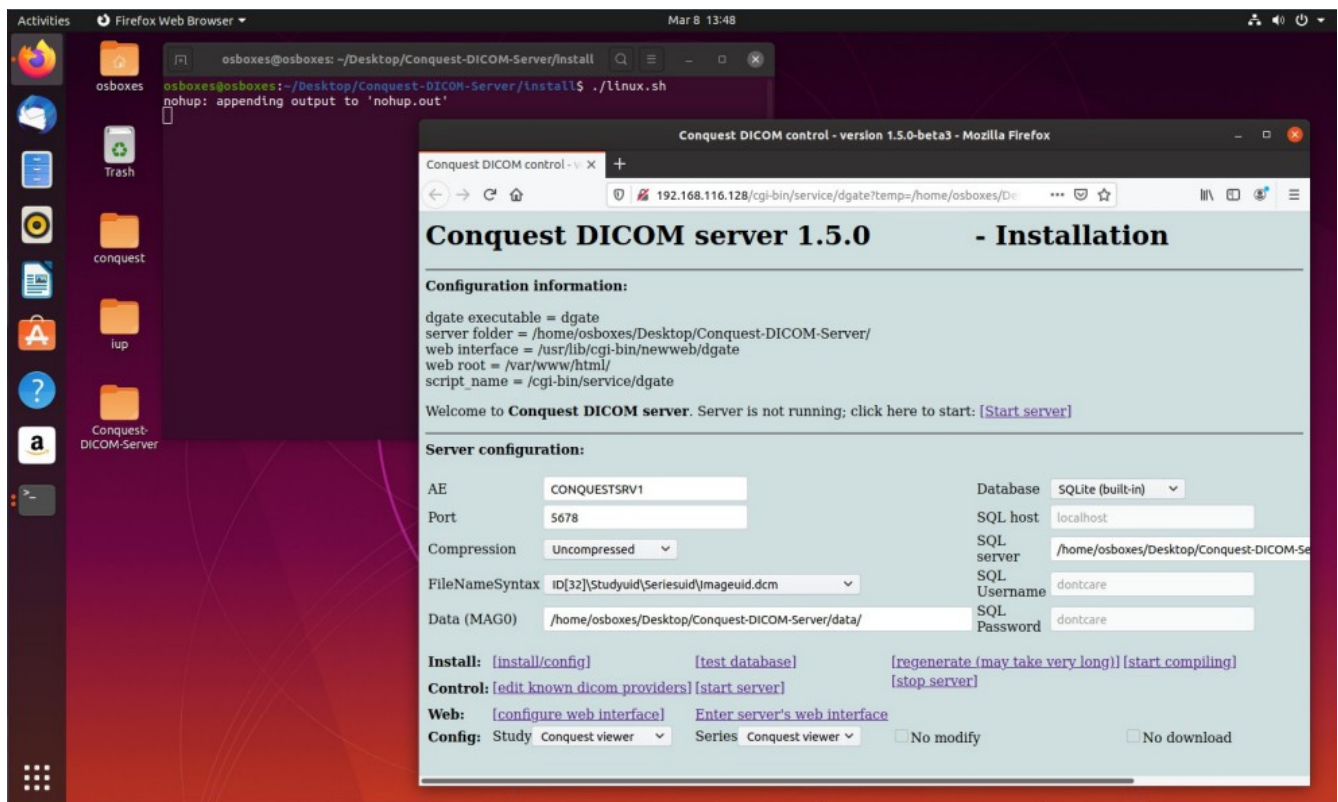
```
chmod 777 linux.sh
```

make run-able

```
./linux.sh
```

This compiles a minimal server binary (dgatesmall) or uses the pre-compiled one, that is run as service

control manager and, if a web server and client (Firefox expected) exist, opens web page <http://127.0.0.1/cgi-bin/service/dgate>. The resulting web page allows and guides the user through compilation, configuration, re-generation of the database if needed, starting the server, setting up the web server and opening the web client. A screen-shot of the install page is shown below:



The required steps (most are shown in the welcome area) are:

- 1) Select required database type (start with SQLite if unsure)
- 2) Start compiling → compile jpeg6c, compile openjpeg, compile charls, compile lua, compile luasocket, compile sqlite, compile dgate; [done].
If any of the compilation steps fails error messages can be found in file nohup.out. If the compilation information disappears click start compiling again.
- 3) Set other parameters (keep defaults if unsure)
- 4) Configure server
- 5) Start server (may need be repeated a few times if does not start)
- 6) Regenerate database
- 7) Configure web interface (select viewers and access rights)
 Note that if you want to use weasis as viewer, the weasis folder from weasis_portable.zip (v3), must be placed in your web servers root folder
- 8) Enter server's web interface

Feedback on this new installation method would be appreciated. After installation, the server runs as part of the control manager. To make it run permanently, stop the server control manager (dgate_small) with ^C, and use the new start-stop-daemon method described above or the old one below. Note that stopping the server using this web page on Linux disables restarting it for a minute or so (due to an IP port being blocked). Be patient when it fails not restart and try again after a while.

Deamon configuration

Both the web install and malinix now create a daemon as follow, changing the file to point to the conquest installation:

```
sudo cp conquest.service /etc/systemd/system/conquest.service
sudo systemctl daemon-reload
```

After installation you can control the conquest service as follows:

```
sudo systemctl start conquest.service
sudo systemctl enable conquest.service
sudo systemctl status conquest.service
```

hit 'Q' to return

```
sudo systemctl stop conquest.service
sudo systemctl disable conquest.service
```

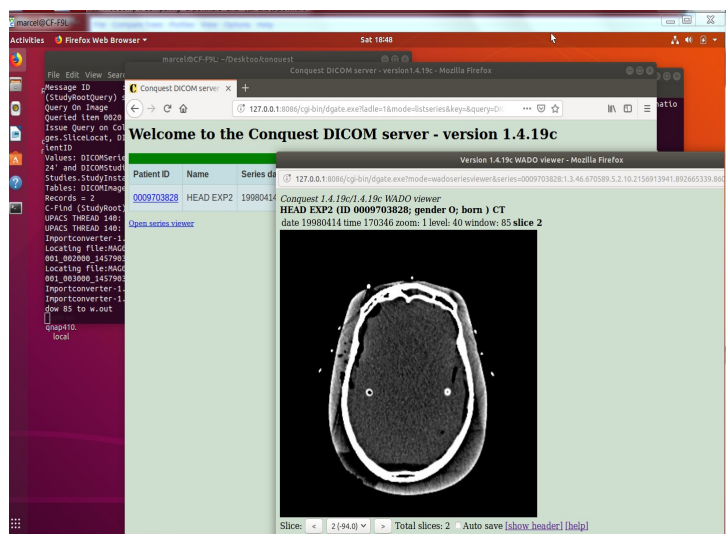
Now the server should be running, also after a system restart, and localhost/cgi-bin/newweb/dgate should provide a working web interface.

Built-in WEB Viewer

A new single user web viewer can be run as follows:

```
chmod 777 webviewer.sh
./webviewer.sh
```

This is the same web viewer as can be accessed from a full featured web server, but instead it runs on 127.0.0.1:8086, using Ladle (single user web server) as mini web server. After stopping the browser, the Ladle function is stopped. It takes a minute or so for the used port (8086) to be released. Until then attempting to start the web viewer fails.



Installing with Postgres

To install with Postgres as database, these commands are needed to install and setup Postgres:

<code>sudo apt-get install libpq-dev</code>	Postgres development tools
<code>sudo apt-get install postgresql</code>	Postgres database
<code>sudo su</code>	become superuser
<code>su - postgres</code>	become postgres user
<code>psql</code>	set the password to postgres
<code>\password</code>	
<code>postgres</code>	(password)
<code>postgres</code>	(repeat password)
<code>\q</code>	
<code>createdb conquest</code>	create database conquest
<code>exit</code>	
<code>exit</code>	
<code>./maklinux</code>	compile and install web access
<code>choose option 2</code>	Postgres

The build process always gives a few error messages that can be ignored:

```
/usr/bin/install: cannot create regular file '/usr/local/man/man1/cjpeg.1': No such file or directory
Makefile:200: recipe for target 'install' failed
mkdir: cannot create directory 'data/dbase': File exists
```

During database creation (dgate -v -r) there can be error messages about non-existing databases, e.g. for postgres:

```
osboxes@osboxes:~/Desktop/distribution$ ./dgate -v -r
Regen Database
Step 1: Re-initialize SQL Tables
*** ERROR: relation "dicomworklist" does not exist
...
***Error: ERROR: table "uidmods" does not exist
WorkList Database
Patient Database
Study Database
Series Database
```

Image Database

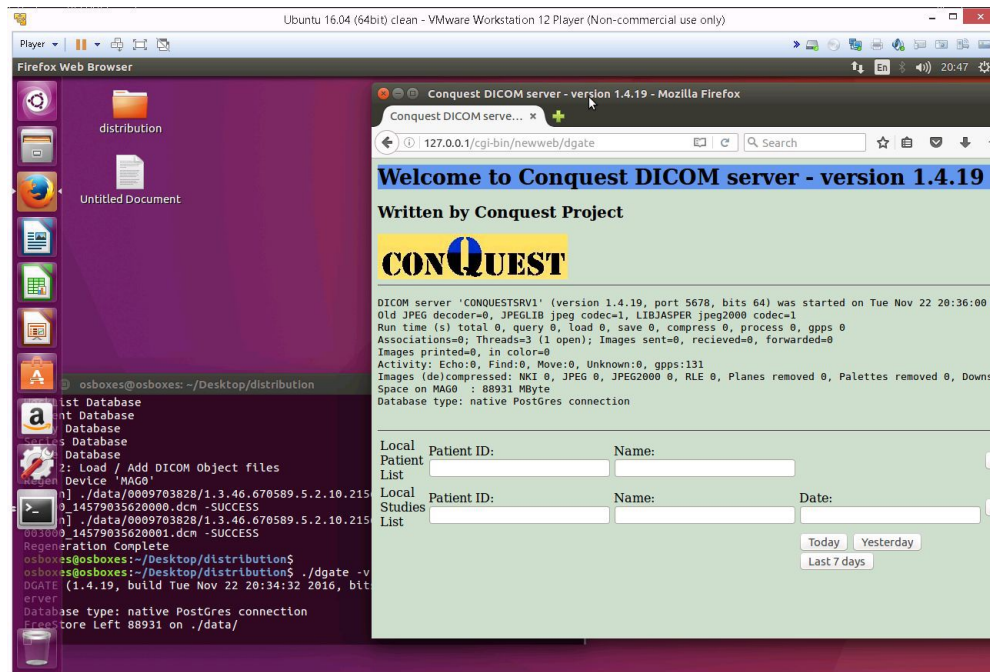
Step 2: Load / Add DICOM Object files

Regen Device 'MAGO'

```
[Regen] ./data/0009703828/1.3.46.670589.5.2.10.2156913941.892665339.860724_0001_003000_14579035620001.dcm -SUCCESS
```

Regeneration Complete

```
osboxes@osboxes:~/Desktop/distribution$ ./dgate -v
```



Conquest in action on Ubuntu16.04, with Postgres database and web interface

Installing with Mariadb

To install with Mariadb as database, these commands are needed to install and setup:

```
sudo apt install mariadb-server
```

Mariadb server

```
sudo apt install libmariadbclient-dev
```

Client code

```
sudo mysql
```

database superuser

```
>create user conquest;
```

create user

```
>grant all privileges on *.* to conquest@localhost identified by 'conquest';
```

create database

```
>create database conquest;
```

```
>flush privileges;
```

```
>\q
```

```
./maklinux
```

compile and install web access

```
choose option 1
```

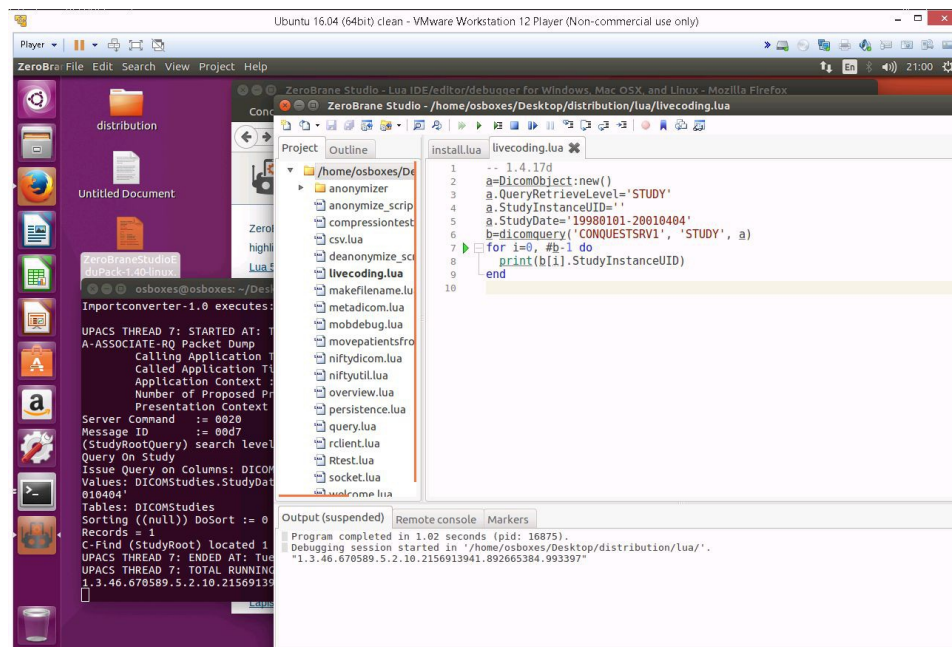
mariadb

ZeroBraneStudio IDE

To install and use ZeroBrane Studio with the conquest DICOM server under Linux, take these steps. First download ZeroBraneStudioEduPack-xxx-linux.sh. Then in a command prompt run:

```
chmod 777 ZeroBraneStudioEduPack-xxx-linux.sh
sudo ./ZeroBraneStudioEduPack-xxx-linux.sh
```

After installation is done run ZeroBrane Studio from the command prompt as “sudo zbstudio” and run the install script /dicomserver/ZeroBraneStudio/install.lua in ZeroBrane Studio as described in this file. After running the conquest install script as root, ZeroBraneStudio can be run as a normal user.



Integration of Conquest with Zerobrane Studio

CONFIGURATION

Configuration files under Windows and Linux are the same except for the use of a forward slash instead of back slash in directory paths. The following essential entries are therefore different for Linux (these are the defaults):

```
SQLServer          = /home/user/conquest/data/dbase/conquest.db3
MAGDevice0         = /home/user/conquest/data/
```

See the Windows manual for more details about the configuration files (you need at least to edit **acrnama.map** to define DICOM systems that will be retrieving information from your server). All configurations options in **dicom.ini** (e.g., for DICOM routing) are listed in **windowsmanual.pdf**. You may also need to edit the web server configuration file **/usr/lib/cgi-bin/newweb/dicom.ini** to set the IP address of the machine. If wrong some 3rd party viewers functions will not function.

After copying the files, if needed, regenerate the database with “conquest/dgate -v -r” then run the server with “conquest/dgate -v 8” or “conquest/dgate -^serverstatus.log”. NOTE: regeneration is only needed after an upgrade if **dicom.sql** is updated. If you want to avoid regeneration do NOT replace **dicom.sql**

The build process for the server was tested with several Linux versions. Both 32 and 64 bit OS's are supported. Warnings are produced but these do not impact server operation.

These are the settings in dicom.ini for MySQL:

```
SQLHost          = localhost
SQLServer        = conquest
Username         = root
Password         =
Mysql            = 1
DoubleBackSlashToDB = 1
```

For Postgres a copy from **dicom.ini.postgres** to **dicom.ini** would set the following values:

```
SQLHost          = localhost
SQLServer        = conquest
Username         = postgres
Password         = postgres
PostGres         = 1
DoubleBackSlashToDB = 1
UseEscapeStringConstants = 1
```

Installation uses a normalized database (as defined in **dicom.sql**) for most database operations, e.g., by copying **dicom.sql.postgres** to **dicom.sql** and a denormalized database for DbaseIII, e.g., by copying **dicom.sql.dbase** to **dicom.sql**.

The following are donated scripts by Mark Pearson for start/stop and rotating logfiles and this information is for expert users only:

To install this script (it is in the distribution as nconquest-pacs.sh) do:

```
sudo cp nconquest-pacs.sh /etc/init.d/  
sudo chmod 755 /etc/init.d/nconquest-pacs.sh  
sudo apt-get install authbind  
sudo /etc/init.d/nconquest-pacs.sh start
```

```
#!/bin/bash  
#  
# conquest-pacs.sh      SysV init script for Conquest PACS.  
#  
#      Written by Miquel van Smoorenburg <miquels>.  
#      Modified for Debian GNU/Linux by Ian Murdock <imurdock>.  
#      Customized for Conquest by Mark Pearson <markp>  
#  
#      HOME and PACSUSER should be the only variables that may need to be modified.  
#  
PATH=/sbin:/bin:/usr/sbin:/usr/bin  
  
# Modify HOME to suit your environment.  
HOME=/usr/local/conquest  
# This is the user to run as. Modify it if you don't use username conquest.  
PACSUSER=conquest  
  
DAEMON=$HOME/dgate  
INI=$HOME/dicom.ini  
NAME=conquest_pacs.sh  
  
# All defaults here will be overridden by values from $HOME/dicom.ini  
STATUSLOG=$HOME/serverstatus.log  
PORT=104  
DESC="Conquest PACS Server"  
  
STOPPACS=$HOME"/dgate --quit:"  
STARTAS=$DAEMON  
  
test -f $DAEMON || echo "Cannot find $DAEMON" exit 0  
test -f $INI || echo "Cannot find $INI" exit 0  
  
set -e  
  
if grep "TCPPort" $INI > /dev/null ; then  
    PORT=`egrep -i '^*TCPPort *= ' $INI | sed 's/\r//' | awk '{ print $3}'`  
fi  
  
if [ $PORT -le 1024 ]; then  
    test -f /usr/bin/authbind || echo "authbind is needed for access to ports < 1024" exit 0  
    STARTAS="/usr/bin/authbind "  
fi  
  
if grep -is "^ *StatusLog" $INI > /dev/null ; then  
    STATUSLOG=`egrep -i '^*StatusLog' $INI | sed 's/\r//' | awk '{ print $3}'`  
fi  
  
PIDFILE=/var/run/$NAME.$PORT.pid  
if [ $STARTAS = $DAEMON ]; then  
    ARGS=" -^$STATUSLOG"  
else  
    ARGS="$DAEMON -^$STATUSLOG"  
fi  
  
case "$1" in  
    start)  
        if [ -f $HOME/disable_autostart ]; then  
            echo "Not starting $DESC: disabled via $HOME/disable_autostart"  
            exit 0  
        fi
```

```

        echo -n "Starting $DESC: "
        start-stop-daemon --start --quiet --pidfile $PIDFILE \
            --chuid $PACSUSER --chdir $HOME --exec $DAEMON \
            --startas $STARTAS --background -- $ARGS
        echo "$NAME."
        ;;
stop)
    echo -n "Stopping $DESC: "
    cd $HOME
    $STOPPACS

    start-stop-daemon --oknodo --stop --quiet --pidfile $PIDFILE \
        --exec $DAEMON -- $ARGS
    echo "$NAME."
    echo
    ;;

restart|force-reload)
    echo -n "Restarting $DESC: "
    start-stop-daemon --stop --oknodo --quiet --pidfile $PIDFILE \
        --exec $DAEMON -- $ARGS
    sleep 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE \
        --chuid conquest --chdir $HOME --exec $DAEMON -- $ARGS
    echo "$NAME."
    ;;
*)
    N=/etc/init.d/$NAME
    echo "Usage: $N {start|stop|restart|force-reload}" >&2
    exit 1
    ;;
esac

exit 0

```

For security reasons I have added a user "conquest" and the package authbind to allow access to priveleged ports. I added the following entries to dicom.ini:

```

HomeDir = /usr/local/conquest
StatusLog = /var/log/conquest/NMPACS.serverstatus.log
TroubleLog = /var/log/conquest/NMPACS.PacsTrouble.log

```

The file /etc/cron.weekly/conquest_rotate does weekly log rotation for me.

```

#!/bin/bash

# conquest_rotate      Cron script to rotate conquest log files.
#   Keep files for 365 days
#   Read filenames from dicom.ini
#
#
#           Written by Mark Pearson 20070711 <markp>.
#
# Modify this line to suit your environment
HOMES=(/usr/local/conquest /usr/local/conquest-icon)
for i in ${HOMES[@]}; do

    INI=${i}/dicom.ini
    STATUSLOG=${i}/serverstatus.log
    TROUBLELOG=${i}/PacsTrouble.log

    set -e

# defaults will be overridden by values from ${i}/dicom.ini
    if grep -is "^ *StatusLog" $INI > /dev/null ; then
        STATUSLOG=`egrep -i '^*StatusLog' $INI | sed 's/\r//' | awk '{ print $3}'`
    fi
    if grep -is "^ *TroubleLog" $INI > /dev/null ; then
        TROUBLELOG=`egrep -i '^*TroubleLog' $INI | sed 's/\r//' | awk '{ print $3}'`
    fi

```

```
if [ -s $TROUBLELOG ]; then
    savelog -p -c 365 -n -q $TROUBLELOG
fi

if [ -s $STATUSLOG ]; then
    savelog -p -c 365 -n -q $STATUSLOG
fi

done
```

This copes with multiple pacs instances on the same host. The advantage of using savelog is that old logfiles are compressed. It should be quite simple to edit the files to have executable or log in /opt. Also, don't forget to set the appropriate file permissions for the user that runs conquest.

Finally, Here are the command lines to compile the server under OS X xcode using 10.4u sdk on a PowerPC (not recently tested):

```
g++ -isysroot /Developer/SDKs/MacOSX10.4u.sdk -arch ppc -Wno-multichar -I/usr/local/mysql/include -L/usr/local/mysql/lib -DDARWIN
-DUSEMYSQL -DHAVE_LIBJASPER -DHAVE_LIBJPEG -DB_DEBUG -o dgate total.cxx -lpthread -lgcc_s.10.4 -lstdc++.6 -lmysqlclient -lz
```

And to compile under SOLARIS 10:

```
/usr/sfw/bin/g++ -DUNIX -DNATIVE_ENDIAN=1 -DHAVE_LIBJASPER -DHAVE_LIBJPEG -DSOLARIS total.cxx -o dgate -lpthread -lsocket -lnsl
-lposix4
```