Foswiki > FLAO Web > SoftwareDevelopment > OsIssues > OsConfiguration (25 Sep 2015, AoAdmin)

OS installation, configuration and setup for servers supporting FLAO software

Installation sequence on CentOS 6.x/7.x (64 bit)

See CentOS67Installation

Installing Linux OS distribution (as of today: CentOS 5.4)

- Install CentOS 5 from DVD distribution and fully update.
- Install RPMForge package repository (but see updated instructions at: http://rpmrepo.org/RPMforge/Using)

```
wget http://packages.sw.be/rpmforge-release/rpmforge-release-0.5.2-
2.el5.rf.i386.rpm
rpm -Uhv rpmforge-release-0.5.2-2.el5.rf.i386.rpm
```

- Install the EPEL repository:

```
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

Note: Verify the CentOS version on /etc/redhat-releasei (In the example above it is 5.3. If needed, modify the package name).

- Install the priorities package:

```
yum install yum-priorities
```

- Make sure that yum-priorities is enabled by editing the **/etc/yum/pluginconf.d/priorities.conf** file, and ensuring that it contains the following lines:

```
[main]
enabled=1
```

- Edit the **.repo files in /etc/yum.repos.d/** and set up priorities by adding the line:

```
priority=N
```

to every repository entry. In particular in the file **CentOS-Base.repo**set "priority=1" for the packages:

```
[base] [addons] [updates] [extras]
```

and set "priority=2" for the packages

```
[centosplus] [contrib]
```

and in the file **rpmforge.repo**set "priority=11" for the package

```
[rpmforge]
```

and in the file **epel.repo**

```
[epel] [epel-debuginfo] [epel-source]
```

- Now simply install everything using the file yumpack.txt:

```
yum -y install `cat yumpack.txt`
```

```
 Add the two following lines in the /etc/sudoers file with the command
/usr/sbin/visudo:
```

```
Defaults    env_reset
Defaults    env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR
LS_COLORS hagen@jauu.net PS1 PS2 QTDIR USERNAME PWD LBTCONF_PATH LANG L
C_ADDRESS LC_CTYPE LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES
LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE LC_TIME LC_ALL
LANGUAGE LINGUA  _XKB_CHARSET XAUTHORITYi ADOPT_ROOT ADOPT_SOURCE ADOPT_LOG
ADOPT_IDL_SOURCE"
```

Prerequisite packages

- **pyhton** development files
- **qt3** development files
- **qwt** library (installed from source, see below)
- **pyqt** bindings
- **pyfits** library
- **python-uuid** Support for uuid generation (used by prepare/deploy procedures)
- **libXpm-devel**
- **libmotif-devel**
- **numpy** python package
- **boost** libraries (modules: base, serialzation, thread)
- **gmp** library development files
- **lrzsz** `sxmodem` support program
- **libreadline-devel**: development files for readline
- **IDL** with proper license (tested version 6.3)
- **dpu** documentation package
- **kde-devel**

Installation of required packages on CentOS 4

The list of all the packages installed on a i386 PC with a working Supervisor is in the attached file yumlist.txt

How to protect CentOS 4 from 3rd part repositories

```
yum install yum-plugin-priorities
```

How to install RPMForge repository on CentOS 4

Download the rpmforge-release package.

Choose one of the two links below, depending on your architecture. If you are unsure of which one to use you can check your architecture with the command **uname -i**

i386i: http://dag.wieers.com/packages/rpmforge-release/rpmforge-release-0.3.6-1.el4.rf.i386.rpm

Install DAG's GPG key

```
rpm --import http://dag.wieers.com/rpm/packages/RPM-GPG-KEY.dag.txt
```

Verify the package you have downloaded

```
rpm -K rpmforge-release-0.3.6-1.el4.rf.*.rpm
```

Install the package

```
rpm -i rpmforge-release-0.3.6-1.el4.rf.*.rpm
```

This will add a yum repository config file and import the appropriate GPG keys. At this point, you can set the priority of the RPMForge repository, and also of the CentOS repositories if you have not done so yet.

Add a line containing

```
priority=10
```

to /etc/yum.repos.d/rpmforge.repo

Test with this command: `yum check-update`

It should output these two lines:

```
 Loading "priorities" plugin ...
```

```
285 packages excluded due to repository priority protections
```

An example command is as follows:

```
sudo yum install build-essential python-devel qt3 lqt3-devel python-qt-dev
python-qt3 boost boost-devel libreadline5-dev libgmp3-dev python-numarray
python-pyfits libXpm-dev libmotif-dev
sudo ln -s /usr/include/python2.5 /usr/include/python
cvs checkout Supervisor
cd Supervisor
make
sudo make install
```

Installing numarray

```
yum install python-numarray
```

Installation of required packages on ubuntu

These packages are enough for the compilation (maybe they are more than the necessary ones):

```
sudo apt-get install build-essential python-dev libqt3-mt libqt3-mt-dev
python-qt-dev python-qt3 python-uuid libboost-dev libboost-serialization-dev
libboost-thread-dev libreadline5-dev libgmp3-dev python-numarray python-pyfits
libXpm-dev libmotif-dev
sudo ln -s /usr/include/python2.5 /usr/include/python
cvs checkout Supervisor
cd Supervisor
make
sudo make install
```

Installing QWT

QWT is installed from the source package stored in the contribdirectory of the CVS repository. Installation is made with the usual procedure for QT based programs:

```
qmake
make
install
```

Network configuration

- Put in `/etc/hosts` the addresses of the devices as specified in IpNumbers.
- **Firewall**
  The firewall configuration must allow network packets from the BCU's. The easiest way is to declare the ethernet interface of the BCU subnet as trusted.

Non standard issues

Kernel parameters

In order to full configure the Adaptive Secondary it's needed to expand (if not) the shared memory size provided by the Operating System.

To check if there is enough shared memory you can write (from root):

```
# sysctl -a | grep shm
```

and look for the lines:

```
kernel.shmall = 131072
kernel.shmmax = 536870912
```

If you have values for kernel.shmall and kernel.shmmax have lower values, please change /etc/rc.d/rc.localfile adding the two lines:

```
sysctl -w kernel.shmall=131072
sysctl -w kernel.shmmax=536870912
```

This will set the values at reboot. You may also give the commands at a command propmpt for immediate effect.

Partitioning

- It is important to reserve a partition for log files: so that even though the log partition is filled up, the OS can go on working.

IDL related issues.

Installing IDL

IDL installation is done in the usual way. All the IDL code is tested against IDL 7.1.

Note 1: the IDL code is **not** compatible with IDL 8.0

Note 2: It is recommended to perform a standard local installation (so thet IDL71 is installed on directory: `/usr/local/itt`. This avoids many pitfalls in the following installation of required libraries and in the PATH used at run time.

4D and IDL

The 4D Interferometer control program uses Pyro to sends command. Pyro version 3.6 has been tested. To install it the package must be untarred on any directory and directions found in the manual must be followed (the usual `python setup.py install` procedure is all you need to do).

To use the 4D PhaseCam 4020 with the IDL wrapper:

- Add the **PYRO_CONFIG_FILE** environment variable (i.e equals to $(ADOPT_ROOT)/conf/left/Pyro_Client.conf)
- Set the variable **PY_VER** in Makefile.gen

Setup for compilation of IDL wrappers

- A file name `idl.conf` must be put in `/etc/ld.so.conf.d`, with the following line:

```
/usr/local/itt/idl71/bin/bin.linux.x86/
```

Then issue (as root) the command **ldconfig** (it will be reissued at each boot)

Environment variables

- **ADOPT_ROOT**: root of installation directories (e.g.: `/usr/local/adopt`)
- **ADOPT_SOURCE**: root of source directories (e.g.: `/home/labot/AO2.0/Supervisor`)
- **ADOPT_LOG**: logging directory (/tmp if not specified)
- **QTDIR**: QT development directory (e.g.: `/usr/share/qt3`)
- **PYTHONPATH**: not needed, clear it.
- **IDLDIR**: IDL basedir (e.g.: `/usr/local/itt/idl`)
- **LD_LIBRARY_PATH:** must contain the path to idl libraries (typically: `/usr/local/itt/idl70/bin/bin.linux.x86`)
- In addition `/usr/include/python` directory must point to the current Python development includes (usually a link will be made).

Daemons

See AdOptProcesses.

Miscellaneous

How to get rid of annoying warning in boost libraries

Compiler warns:

```
/usr/include/boost/serialization/shared_ptr.hpp:201: warning: unused parameter "file_version"
```

Solution: comment out the variable name, without changing the interface

```
inline void load( Archive & ar, boost::shared_ptr &t, const unsigned int /*file_version*/ ){
```

Note on `sudo`

Because you need `sudo` when doing any `make install` command, remember that on some distributions it is configured so that it will cancel most environmet variables, including `ADOPT_ROOT` and `ADOPT_SOURCE`, so that the make install command will fail.

You must edit /etc/sudoersand add the two above variables to the line:

```
Defaults env_keep = " ........   ADOPT_ROOT ADOPT_SOURCE ADOPT_LOG"
```

Attachments (4)

Attach files Show options

Pyro-3.6.tar.gz (237.91K)
Version 1 uploaded by Luca Fini on 17 Aug 2010 - 12:07
Pyro install package

qwt-5.1.2.tgz (5444.43K)
Version 1 uploaded by Luca Fini on 13 Jan 2010 - 09:30
QWT tarball

yumlist.txt (476.78K)
Version 1 uploaded by Marco Xompero on 08 Jul 2008 - 19:07
list of packages in a working

yumpack.txt (0.44K)
Version 3 uploaded by Alfio Puglisi on 04 Nov 2011 - 15:48 ... more
list of packages in a working

Select all

Edit | Attach | Print version | History: r21 < r20 < r19 < r18 | Backlinks | View wiki text | Edit wiki text | More topic actions

Topic revision: r21 - 25 Sep 2015, AoAdmin