PACT/BASIS/CORSICA Builds

4-June-2004

R. A. Jong

1. Building and Installing PACT

PACT is updated on an irregular time schedule, generally only when an updated version of BASIS requires new or updated features from PACT. Since we do not have privileges to access the PACT cvs archive, we must get a self-extracting archive containing the source code from one of the PACT developers. I usually get it from Stewart Brown or Lee Minner. Have them do a "give" on one of the Livermore Computing Center (LC) platforms and you can retrieve the file with an appropriate "take" command. Then secure copy the file to the other platforms where PACT is to be built.

I put the source file in my PACT build directory on each platform. My build directory is named something like:

$HOME/build_pact/pactxxyyzz

where xx is the year designator (e.g. 04)
yy is the month designator (e.g. 05)
zz is the day designator (e.g. 11)

The date designators signify the version of PACT that is being built and installed. The self-extracting archive (e.g. pact04_05_11-src) that contains the source for PACT must be stored in this build directory during the build. The date designator, 04_05_11 in the example, is key, since the install script described below, uses this designator to name the target install directory as well as construct the command to build the code.

To build and install PACT, use the sh shell script setup_pact as in:

setup_pact <arg1> <arg2>

For the Solaris build and installation, you must run the script on engy.llnl.gov.

The installation process puts the installed version in a directory named something like:

<PACTBASE>/pact_xx_yy_zz or
<PACTBASE>/pactxx_yy_zz

where <PACTBASE> is different on each platform. You must have write permissions in the target installation directory to complete the installation.

On some platforms (MFE Solaris, GA, MFE Linux), the installation directory names have the extra "_" in the name because the directories on the different machines were set up at different times, and I was inconsistent in setting up the directories. You must have write permissions in the target directory where PACT is to be installed.

See the Appendix below for the location of the setup_pact script and other scripts that are used to build and configure PACT, BASIS and CORSICA on the platforms we normally use on the MFE Solaris and Linux platforms, at General Atomics (GA), and at the Livermore Computing Center (LC).

The setup_pact script must be executed from the build directory that contains the PACT sources. The script takes two arguments, for example:

```
cd  $HOME/build_pact/pact04_05_11
setup_pact 04_05_11 config
```

The first argument (e.g. 04_05_11) must be the version designator for the PACT source file. The second argument is the configuration file name to be used for the build. The usual options are:

Platform Type	Configuration File Name
Solaris	solaris
Linux	linux
HPUX	hp
ALPHA	osf
AIX (frost)	aix-gcc

The first argument string is used to create and prepare the installation target directory name. The configuration files are extracted from the PACT source archive and placed in the $HOME/build_pact/pact04_05_11/pact/manager/configs.std sub-directory.

The script does some initial checking with the user before actually doing anything. If you are on a Solaris system, it checks to see if you are running on engy.llnl.gov. This is required because the partition where PACT is installed can only be written from engy. If your PACT environment variable is set to the usual public version of PACT, the script will use this information to create the base directory for the target installation directory.

For example, if your current PACT environment is something like:

```
/usr/local/pact/pact04_04_23
```

then in the example above, the script will set PACT_BASE to:

```
/use/local/pact
```

and the new default installation target would become:

```
/usr/local/pact/pact04_05_11
```

You are given the option of over-riding the default and selecting another PACT_BASE before the actual installation. If you do not have your PACT environment set, the script will ask you to choose a PACT_BASE location.

Because the script must be run from the directory that contains the PACT source file, the script will ask you to confirm that you are located there.

After all the checks, the script starts the build, test, and install process using the standard PACT build and install command. It also spawns an xterm process window where you can monitor the build, test, and install process. In the xterm window, type the command:

tail –f Build.log

and you can monitor the build, test and install progress. Problems, if they occur, or a successful install message can only be seen in real time in this window. If you choose not to monitor the process here, you can check the build log after the setup_pact process has completed. The log file will be have been re-named to something like <host>.BUILD, where <host> is the name of the machine you are on.

The setup_pact script performs a clean extraction and install from the PACT source archive. The build process is pretty robust, so this is all you normally have to do. If there are problems encountered in the build, test, or install process, then you need to manually fix them. This script will not help in this case, because you need to know more about the build procedure to do it manually. Traditionally, we have only seen problems on platforms that are not one of the standard build and test platforms used by the PACT developers (e.g. HPUX like at GA or the AIX platforms at NERSC).

This script has been successfully tested on the MFE Linux cluster (hrothgar), on the MFE Solaris network (engy), on the Alpha (cardea) and HPUX (hydra) network at General atomics (GA), and on the Livermore Computing Center (LC) Alpha (gps) and Linux (ilx) platforms.

For further general information about building PACT, see the documentation at

http://pact.llnl.gov

and for additional information about the build experience on our various platforms, see (note: https):

https://skibuff.llnl.gov/PACT_build.html

2. Building and Installing BASIS

Our version of BASIS is updated whenever an obvious bug has been reported and fixed by the BASIS developers. Someone needs to be on the BASIS developers list to monitor these emails that are sent out whenever their cvs repository is updated. Even when there are no obvious bug reasons to update BASIS, the MFE vbasis version should be updated about every two weeks to ensure that the BASIS development path maintains compatibility with our build of CALTRANS. This is not a big a problem now as it was earlier when BASIS major version 12.1 was introduced, but might become a problem again when BASIS version 12.2 is released.

We maintain multiple versions of BASIS (v, n, and p) that are synchronized with the corresponding versions of CALTRANS. I also keep placeholders (zbasis and zcaltrans, although these are often empty) for testing purposes.

There is now also a version of BASIS that we maintain only on the MFE Linux and MFE Solaris networks called sbasis. This version was intended as a fallback in case a bug was found in the latest volatile version of vbasis. The sbasis version is meant to contain a prior more stable version of vbasis that can be used to build the volatile vcaltrans version of the CORSICA code. The intention was to build a CORSICA version designated as scaltrans using the same source code as in the vcaltrans version, but to use sbasis for the build instead of vbasis.

To access the BASIS cvs repository, you must have an account on the A/X Division unclassified network. If you do not have such access, you will need to get a copy of the BASIS source archive from one of the BASIS developers. Note that because cvs access to the BASIS repository is restricted to platforms in the llnl.gov domain, the getbasis script described below will not work from sites such as General Atomics. To get the BASIS source code to GA, you must check out the code at LLNL and secure copy the results to a GA platform.

If you have A/X network access, then you can use the sh shell script called

```
getbasis
```

to access the cvs repository and download a copy of the sources. This script sets the environment variables necessary to make the ssh connection with the BASIS cvs repository and check out the source to your working directory.

The getbasis script can be used to get a fresh checkout of the BASIS repository, or to do an update in an existing BASIS sandbox. The getbasis script takes zero or more arguments. If no arguments are given, a new checkout of basis is done in the working directory. If the first command-line argument is "-h" or "-help", then a help message is displayed. If you are running the script in a working directory that contains an existing BASIS source structure and the first command-line argument is "-u", then a complete update is done in the working directory. If the first command-line argument is not "-u" or "-h" or "-help", and one or more arguments are given, then an update is done in the working directory for the module(s) in the argument list.

Use the command

```
getbasis -h
```

to see the options and use instructions.

I run the getbasis script in a build directory with the name:

```
$HOME/build_basis
```

to start the build process. If you are building a fresh/new version, then the following commands will get the source:

```
cd $HOME/build_basis
getbasis
```

For a fresh checkout, the script will determine the BASIS version number and put the sources in a directory name based on the version number. For example, if the BASIS version number is 12.1.1134, then the sources will be in the directory named basis_12.1.1134.

First create the target installation directory, for example, on hrothgar:

```
mkdir /usr/local/basis/vbasis
```

Recall that for the Solaris build, the target file creation must be done on engy.llnl.gov. The installation directory you create here must have the same name as the InstRoot variable in the corresponding BASIS configuration file.

To build the code on hrothgar, for example, then do the following:

```
cd $HOME/build_basis/basis_12.1.1134/builder
dsys config hothgar
dsys build
de-archive
dsys test
dsys install
```

The “dsys config” command configures the build according to the specifications in the desired configuration file (hothgar in the above example). The “dsys build” command compiles the sources and creates the BASIS library files for the packages requested in the configuration file. The “de-archive” command regenerates all the object files from the library archive files since they are often used in the builds of some of our BASIS codes. The “dsys test” command runs the BASIS test suite.

The “dsys install” command installs the BASIS binaries, scripts, libraries (.a), objects (.o), and include files in the location specified by the InstRoot variable in the configuration file. For the Solaris build, the “dsys install” step must be performed on engy.llnl.gov, but the config, build, de-archive, and test steps can be performed on any Solaris platform. You must have write permissions in the target installation directory in order to complete the installation.

The configuration files for our usual build locations are stored in the BASIS cvs repository as part of the BASIS distribution. Hence for the example build on hrothgar, the required configuration file is located in:

```
$HOME/build_basis/basis_12.1.1134/builder/local/hothgar
```

Note that the file is named hothgar, rather than hrothgar, because the machine name changed after the file was put into the cvs repository. Other configuration files we commonly use are named skibuff, racer, wormhole, cardea, hydra, gps-mfe and ilx-mfe.

You should look in the configuration files to see which items need to be changed or set for the build. In particular, you should set:

PACTRoot=<PACT installation directory>
InstRoot=<BASIS installation directory>

PACTRoot should point to the PACT installation as created in Part 1 of this document. InstRoot should be set to the location where you want to install BASIS. Typical example values for the build on hrothgar are:

PACTRoot =/usr/local/pact/pact04_05_11
InstRoot=/usr/local/basis/vbasis

The PACTRoot value is used for the build only if the installer's PACT environment is unset. If the user's PACT environment is set, then it, rather than the PACTRoot value will be used by the BASIS build process.

The various BASIS versions are installed in the following BASIS directory locations on each of our supported platforms:

Site	BASIS Directory
MFE Solaris	/mfe/theory/Basis
MFE Linux	/usr/local/basis
General Atomics	/d/osf/basis and /d/hp/basis
LC	/usr/gapps/mfe/<sys_type> where <sys_type> is redhat_7a_ia32 for the ilx cluster and is tru64_5 for the gps cluster

For additional information about the build experience on our various platforms, see (note: https):

https://skibuff.llnl.gov/Basis_build.html

3. Installing IMAKE into the BASIS Installation

After BASIS has been installed (for example, on hrothgar) in /usr/local/basis/vbasis, it is still not ready to use in building most codes such as UEDGE or CORSICA. To use this BASIS installation to build these codes, we must configure the BASIS installation to make the necessary links to the PACT installation. In addition, to build CORSICA, we must increate and modify an

imake sub-directory under the BASIS directory. All this is accomplished by running the script called mklink (with two arguments). An example of the command sequence is:

```
cd /usr/local/basis/vbasis
mklink . $PACT
```

or the single command line:

```
mklink /usr/local/basis/vbasis $PACT
```

On the Solaris platform, this mklink script must be run on engy.llnl.gov. Follow the prompts from the script and select the vbasis option for the imake modifications.

This script modifies the vbasis/lib directory with the necessary links to the PACT installation. It copies the imake executable and a script to the vbasis/bin directory. It creates and modifies the imake directory contents to be consistent with the BASIS installation (vbasis in this example but [n|p|s|z]basis are other options).

A copy of the template for the contents of the imake subdirectory that will be installed in BASIS is kept on each of the platforms we support. The primary version is kept on the Solaris cluster in /mfe/theory/Basis/imake_dist. Copies of this imake_dist directory are kept on the other supported platforms in the BASIS directory (see the table in Section 2 above for the location). The imake_dist location is important because it must be accessible from the mklink script on all platforms. There is one minor site-dependent modification to the files in the imake_dist/config directory. The GA_Site, MFE_Site, or LC_Site variable is defined here in the architecture configuration files such as osf1.cf, hp.cf and sun.cf.

The mklink script also makes links in the vbasis/bin directory to two (seldom) used executables and scripts in a directory called utils. These executables and scripts convert namelist to basis input or output. This utils directory is also copied to the BASIS directory on all the relevant platforms.

4. BASIS Version Rotation

We normally maintain three versions of BASIS on each platform. The version named vbasis is the most volatile and contains the latest BASIS build. This is the BASIS version that is used to build vcaltrans. The version named nbasis is a former vbasis that is considered to be more stable. It is used to build the CORSICA version kept as ncaltrans. The most stable (and oldest) BASIS version is named pbasis and is used to build the CORSICA version kept as pcaltrans.

If the sandbox that you used to build a certain version of BASIS still exists, then it is simple to create another [v|n|p|s]basis version. For example, suppose you have

built basis_12.1.1134 in a sandbox and installed it into vbasis on hrothgar. Some time later you decide this BASIS candidate should be called nbasis. You can then do the following to reinstall into the nbasis target. First move aside any existing nbasis installation and then create the new target directory location:

```
cd /usr/local/basis
mv nbasis nbasis_old
mkdir nbasis
```

and remove any existing soft links for basis_12.1.1134 that exist.

Now relocate to the existing sandbox where the compiled BASIS code exists.

```
cd $HOME/build_basis/basis_12.1.1134/builder/config
```

Edit the hothgar configuration file to change the InstRoot variable to point to /usr/local/basis/nbasis

Now reconfigure and re-do the install command

```
cd $HOME/build_basis/basis_12.1.1134/builder
dsys config hothgar
dsys install
```

Next re-run the mklink script as in

```
mklink /usr/local/basis/nbasis $PACT
```

and select the nbasis option for the imake modifications. Here I assume the PACT environment points to the version used to build this version of BASIS.

If you keep the above sandbox long enough and eventually want to make this version pbasis, then the above procedure can be repeated except that references to nbasis are replaced by pbasis. However, it is more likely that the sandbox has long ago disappeared and what you want to do is convert an existing nbasis installation (that is linked, for example, to basis_12.1.1134) to a new installation of pbasis. In this more likely scenario, the steps to take (for the example build on hothgar) are:

```
cd /usr/local/basis
```

Save the existing pbasis version if it exists

```
mv pbasis pbasis_old
```

Move the existing nbasis installation to pbasis

```
mv nbasis pbasis
```

Now you must manually edit several files in the new pbasis/imake/config/Basis and pbasis/bin sub-directories. In the pbasis/imake/config/Basis directory change the soft links for Version.defs and Project.defs to point to the p versions rather than the n versions of the files. In the pbasis/bin directory you must edit the basis and install-basis scripts to change the nbasis references to pbasis in a total of three places.

The final step is to remake the soft link:

```
cd /usr/local/basis
ln –s pbasis basis_12.1.1134
```

Of course, this latter manual method can be used to change a vbasis installation to a new nbasis or sbasis installation in the event that the build sandbox no longer exists.

5. Building CORSICA

The code can be built using the original build command sequence at the top-level directory or using the newer "dsys" system. In the example below, we shall build the code on hrothgar. I assume that the reader can access the CALTRANS cvs repository on hydra.gat.com and check out the CORSICA sources. In the example below, we assume that the sources have been extracted to the file

```
$HOME/build_caltrans/vcaltrans
```

The original build command sequence to compile the code in the directory called vcaltrans is:

```
cd $HOME/build_caltrans/vcaltrans
MMF
gmake Makefiles
gmake
```

The last two gmake commands in the above sequence can be replaced with a single command:

```
gmake World
```

This sequence assumes that the user has set the appropriate environment variables for BASIS_ROOT to point to an existing BASIS installation like vbasis (or some other value like [n|p|s|z]basis).

An alternative build procedure uses the dsys command system developed in conjunction with the test suite. The command sequence for the example build on hrothgar (LINUX) is:

```
cd $HOME/build_caltrans/vcaltrans
manager/dsys config hothgar.llnl.gov
manager/dsys build Makefiles
manager/dsys build
```

The last two commands could be replaced by the single command

```
manager/dsys build World
```

Both build procedures produce an executable file called caltrans in
    $HOME/build_caltrans/vcaltrans/corsica/<arch>/caltrans
where <arch> for the hrothgar build is LINUX.

The command to run the complete CORSICA test suite from the top-level vcaltrans directory is:

```
manager/dsys test –s 0
```

If you had built the code from the top-level directory using the original build command sequence, then the test sequence is started by:

```
cd $HOME/build_caltrans/vcaltrans
manager/dsys config hothgar.llnl.gov
manager/dsys test –s 0
```

6. Installing CORSICA as vcaltrans

To install the code as built in the $HOME/build_caltrans/vcaltrans sub-directory described in Section 3 above, you can use the install_caltrans script. This script makes certain assumptions about the build directory structure as well as the install directory structure. By convention, we designate a vcaltrans version by a date string such as 040519 for the version built on May 19, 2004. This date string is used to rename the caltrans executable created above.

```
cd $HOME/build_caltrans/vcaltrans/corsica/<arch>/
mv caltrans caltrans.040519
ln –s caltrans.040519 caltrans
```

where <arch> is LINUX as before.

An additional link containing the date string is also necessary to designate the top-level build directory. A link must be made to point to the vcaltrans build directory as in:

```
cd $HOME/build_caltrans
ln –s vcaltrans vcaltrans.040519
```

The install_caltrans script checks to ensure that the caltrans.040519 as well as the vcaltrans.040519 files or links both exist.

Assumptions are also made regarding the target installation directory. The default installation directory name for the script is vcaltrans, but can be over-ridden by the user when the install_caltrans script is run. This (vcaltrans) installation target

directory is created in a base directory that is platform dependent. The different values for this base directory (designated BASE) are summarized in the following table:

Site	BASE Directory
MFE Solaris	/mfe/theory/Caltrans
MFE Linux	/usr/local/Caltrans
General Atomics	/d/Caltrans
LC	/usr/gapps/mfe/Caltrans

In this example on hrothgar, the vcaltrans directory would be created as
    /usr/local/Caltrans/vcaltrans

As usual, the installer must have write permission in the target installation directory in order to complete the CORSICA installation. In addition, for installation on the Solaris platforms, the install_caltrans script must be run on engy.llnl.gov.

The typical command line to run the installer script and install into the default (/usr/local/Caltrans/vcaltrans) directory is thus:
    install_caltrans 040519

To install into another directory named dirname, the command is:
    install_caltrans 040519 dirname

To make it easier to determine which version of vcaltrans is installed in /usr/local/Caltrans/vcaltrans, the script also makes a link in the BASE directory pointing to a directory named with the string designator. In our example build on hrothgar, this is equivalent to doing:
    cd /usr/local/Caltrans
    ln –s vcaltrans vcaltrans.040519

An alternative installation procedure uses the install option for the dsys script that was used for the code build. For example on hrothgar,
    cd $HOME/build_caltrans/vcaltrans
    manager/dsys install /usr/local/Caltrans/vcaltrans
would install the code into vcaltrans. As of June 3, 2004, the details of the actual sub-directory contents and the soft links are different from those installed using the install_caltrans script. However, executing the command:
    /usr/local/Caltrans/vcaltrans/bin/caltrans
would successfully run the CORSICA code for either installation procedure.

7. Installing or Patching a Tagged Public Version of CORSICA

   The instructions for installing or patching a tagged public version of CORSICA designated as ncaltrans or pcaltrans are described in laborious detail in the README_TAGS file that is checked out from cvs in the top-level directory of the caltrans distribution. There is no single, simple script to do this installation or patching procedure. You must read and follow the instructions exactly. You must have write permissions in the target directory where CORSICA is to be installed.

8. Appendix

   The installer and configuration scripts described or used above are all stored in public locations on the various platforms that we support. The scripts are:

   cpu_id
   setup_pact
   getbasis
   mklink
   install_caltrans

   The table below summarizes the public locations on the relevant platforms.

Site	Script Location
MFE Solaris	/mfe/local/tools
MFE Linux	/usr/local/tools
General Atomics	/link/tools/LLNL
LC	/usr/gapps/mfe/tools

   To more easily use the scripts, the above directories should be placed in your path or links to the five included scripts should be made to a location that is already in your path.