CERN – European Laboratory for Particle Physics

CERNLIB Unix build procedures

The **CERNLIB** build procedure uses `Imake`, `makedepend`, `gmake` and `cpp` to define and make the different versions. It is not necessary to use these tools but some knowledge of `cpp` will be useful to understand the code or to recompile it.

Typically, the top of the source tree is the directory /cern/LEVEL/src or, from the `ftp` server `asisftp`, `cernlib/share/LEVEL/src`. `LEVEL` is the release version, e.g. `pro`, `old`, `new`, `99`, etc. The file `Imakefile` is the `Imake` description of the build procedure, which also gives the other important directories at this level, e.g.

```
Imakefile config include packlib pawlib graflib mathlib geant321 mclibs phtools
```

These are the different major parts which together make up **CERNLIB**.

Config contains the `Imake` rules and is somewhat complicated if you are not familiar with `Imake`. It also contains the system configuration files (like linux.cf) which contain the description of the system components and the compiler options. These are the files that will need to be modified when starting with a new system or compilers.

In every directory, there is an `Imakefile` which defines the directories and/or files to make the next level. Even if you are not using `Imake`, they contain useful information about the different versions.

The `include` subdirectory is needed at this level to allow cross referencing of certain include files between packages; the simplest way is to make a copy of the complete include directory from the sources or a symbolic link to its normal location (e.g. /cern/new/include).

Beneath this level comes the individual packages and libraries. E.g for `mathlib`:

```
 Imakefile  bvsl       gen
```

where `gen is the general section containing directories of source files.`

Organization of packages and include files

`In general, each package has an Imakefile and a directory with the include files, which usually has the same name as the package, e.g. gen, hbook,`

minuit, etc. These include files are also copied into the release include directory, e.g. /cern/pro/include. The Fortran routines have the extension ".F" which most Unix Fortran compilers preprocess automatically with the C preprocessor or something similar. Thus, the Fortran and C routines address their include files with a cpp #include statement, e.g.

```
#include "minuit/d506dp.inc"
```

For the selection of different code versions, cpp flags starting with CERNLIB_ are used. For each system, the compiler is called with system specific cpp flags which are described in the appropriate tar/README file. Typically (but geant321 has also -DCERNLIB_BLDLIB),

```
 g77 -DCERNLIB_UNIX -DCERNLIB_LINUX -I/cern/new/include file.F
```

Each package has its own steering header file pilot.h which expands these cpp flags into package specific flags. For a new system it will probably be necessary to understand what functionality is required and modify these header files.

Starting on a new system

First make sure that you have all the tools in your path, i.e imake, makedepend, gmake, cpp, the Fortran and C compilers, and the archiver ar.

All the source compressed tar files for the version you want should be downloaded into any suitable directory, e.g. /tmp/cernlib, $HOME/cernlib or /cern. Typically, this would be the latest version. Only files with names like src_*.tar.gz are used in the build. By running this shell script the files will be unpacked, the directory structure created and, provided that the system options are compatible with those in the distribution, the libraries will be installed. This also defines the environment variables which will need to be set to run the tests, use the cernlib command, rebuild modified libraries or binaries like pawX11.

Variable	Meaning	Example
CERN	Cernlib base directory	/tmp/cernlib
CERN_LEVEL	Cernlib version	2004
CERN_ROOT	Cernlib build directory	/tmp/cernlib/2004
CVSCOSRC	Cernlib SOURCE directory	/tmp/cernlib/2004/src
PATH	Search path	/tmp/cernlib/2004/src:$PATH

If successful, the libraries will be in $CERN_ROOT/lib. Failure is usually indicated in the end of the log files in $CERN_ROOT/build/log. Pawlib and some parts of mathlib depend on Lapack3, which can now be built as described here.

However, if the build is to be done for a new system or compiler options, then the relevant configuration file (e.g. config/linux.cf) and the Lapack make files will need to be modified. Then, at least the build directory should have all its makefiles removed and the gmake commands repeated. This simple script cleans up the previous build, builds the libraries, the Paw and packlib executables, and runs some tests.

Tests

With the structure and and environment variables set as above, tests can be run for the individual packages by going into each directory and typing gmake test.

It can sometimes be useful to run the tests without building the libraries or to specify a particular library for testing. One technique is to change the PACKAGE_LIB gmake variable and use the *cernlib* command in your search path to locate other required libraries. Try the following procedure to test the production libraries in /cern/pro/lib:

```
mkdir test
cd test
setenv CVSCOSRC /cern/pro/src
$CVSCOSRC/config/imake_boot
gmake packlib/Makefile
cd packlib
gmake test PACKAGE_LIB=/cern/pro/lib/libpacklib.a
```

Executables in the bin directory

With the structure and and environment variables set as above, the executables for the packlib and pawlib packages can now be built and installed in the $CERN_ROOT/bin directory. Remember that paw requires the lapack3 and blas libraries. Note that paw++ requires kxterm, while pawX11 is self contained.

```
cd $CERN_ROOT/build/pawlib
gmake install.bin
... this should install pawX11 and paw++.

cd $CERN_ROOT/build/packlib
gmake install.bin
... this should install cdbackup, cdmake, cdmove, cdserv, fatback,
fatmen, fatnew, fatsend, fatsrv, hepdb, kuesvr, kxterm, pawserv,
zftp and zserv.
```

Other executables

There are a number of products, mainly in "bin", which are not part of the standard build procedures. These include old stable products which are simply

copied from previous releases as long as they keep working. There are the Patchy 4 programs (ypatchy etc.) and the Patchy 5 suite (nypatchy etc.). Other special programs may also be added as a convenience to certain users or as a backup. There may also be packages (e.g. Stdhep) which are not compiled at CERN and Cernlib is then only a distributor.

Installing Individual packages

Some Imakefiles are somewhat inconsistent in their treatment of the default Make target, especially when multiple libraries are built at different levels. The best technique is to use "install.lib" for libraries and "install.bin" for executables (e.g. PAW and the scripts). So, when the build environment is set up as above, try a test build with phtools which is the simplest:

- gmake phtools/Makefile
  cd phtools
  gmake install.lib

- To build kernlib (e.g. when there are problems building packlib):

  gmake packlib/Makefile
  cd packlib
  gmake kernlib/Makefile
  cd kernlib
  gmake install.lib

  This procedure can be repeated for geant, mathlib, graflib, etc.

- And, to remake pawlib, pawX11 and paw++ (making sure you have kuipc and cernlib in your search path (e.g. in $CERN/new/bin):

  gmake pawlib/Makefile
  cd pawlib
  gmake install.lib
  gmake install.bin
