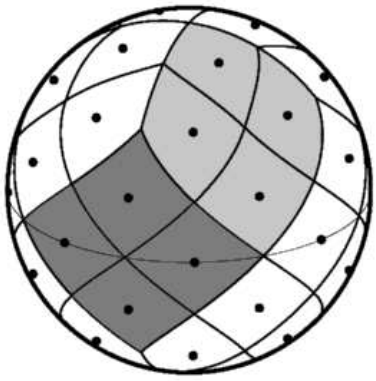
HEALPix Facility Installation Guidelines



Revision:	Version 3.30; October 8, 2015
Prepared by:	Eric Hivon, Anthony J. Banday, Matthias Bartelmann, Benjamin D. Wandelt, Frode K. Hansen and Krzysztof M. Górski
Abstract:	This document describes the installation for the **HEALPix** facilities.

http://healpix.sf.net

TABLE OF CONTENTS

1 Introduction

In this document the installation procedure for the **HEALPix** distribution is outlined. **HEALPix** comprises a suite of Fortran 90, C++, IDL, Java and Python routines providing both stand-alone facilities and callable subroutines as an alternative for those users who wish to build their own tools. A set of C subroutines and functions is also provided.

The distribution can be downloaded as a gzipped tarred file, *or* as a zipped file, which can respectively be unpacked by executing the commands[1]

```
% gunzip Healpix_3.30.tar.gz
% tar -xpf Healpix_3.30.tar
```

or

```
% tar -xzpf Healpix_3.30.tar.gz
```

or

```
% unzip Healpix_3.30.zip
```

creating a directory named `Healpix_3.30` whose structure is shown in Figure 1.

As with most freely available software, the distribution comes with caveats, the major one being that although we have attempted to automate the installation as much as possible, not all eventualities can ever be foreseen. We have tested the installation on the following platforms:
AIX, IRIX, IRIX64, Linux, SunOS, ALPHA and Darwin (MacOS)

There may be problems in the facility build due to the local system configuration which is beyond our control.

2 Installation Requirements

The major part of the **HEALPix** distribution is written in both **Fortran 90** and **C++** and so the appropriate compiler(s) must be present (Linux and Darwin users should look at Section 7.1 about free F90 compilers. Microsoft Windows users should look at Section 7.2). Many visualisation tools and map manipulation routines are provided in **IDL** (please note that at least version 6.4 is required), **Java** and **Python**. Some of the **HEALPix** routines are also available in **C**.
Starting with version 3.0, the healpy (HEALPix in Python) library has been integrated into **HEALPix** releases. Since it is, to a large extent, a wrapper to the C++ routines, installing it also requires a C++ compiler (on top of `python` and a few supporting Python libraries) but it will perform its own compilation of the current **HEALPix** C++ library.

*This section and the next focus on the compilation and installation of the **C**, **C++**, **Fortran 90**, **IDL** and **Python** routines. For more information on the **Java** routines see*

[1] Microsoft Windows users can have a look at
http://sourceforge.net/p/healpix/wiki/Windows%20and%20peazip/

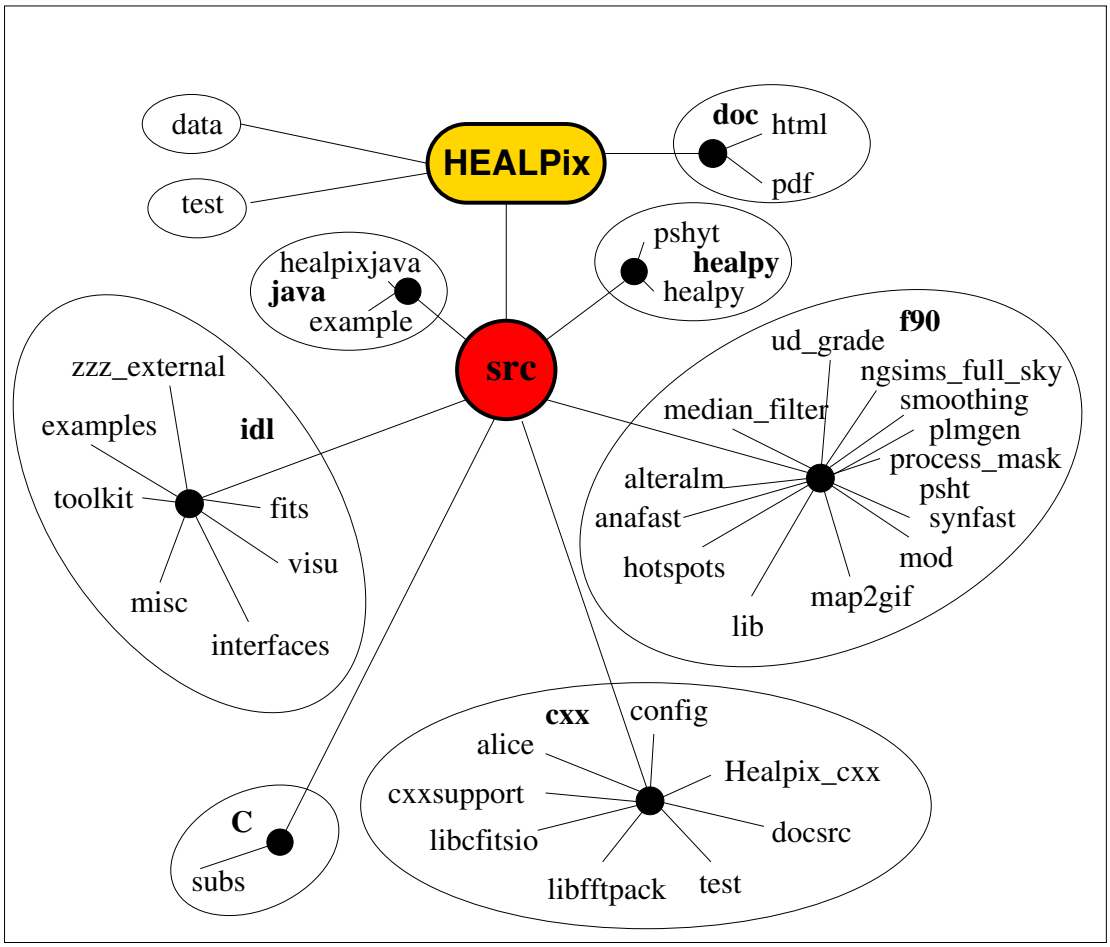


Figure 1: The directory structure for the **HEALPix** distribution.

table 1

The configure script is written in the Bourne shell. The script attempts to generate a `Makefile` which is tailored to one of the above Operating Systems (OS's) and using `Makefile.in` as a template for non system-specific statements. Only the basic UNIX make facility is required to build the software, although we do still recommend the GNU make facility (`ftp://ftp.gnu.org/gnu/make/`). In addition, several environment configuration files and an IDL/GDL startup file are generated. These automatically establish various environment variables and aliases to make the use of the **HEALPix** package simpler.

The **HEALPix Fortran 90**, **C++**, **C** and **Python** distributions also require the publicly available CFITSIO library. Note that the **Fortran 90** routines require version 3.20 or more (post August 2009) of CFITSIO

Healpix Package	**Information on installation**	**Information on routines**
Fortran 90	This document	"Fortran Facilities" and "Fortran Subroutines" documents
IDL/GDL	This document	"IDL Facilities"
C++	This document, or `src/cxx/README.compilation`	"C++ Facilities and Subroutines" (HTML only)
C	This document, or `src/C/README`	"C Subroutines Overview"
Java	`src/java/README`	"Java Overview" (HTML only)
Python	This document, or `src/healpy/INSTALL`	"Healpy Documentation" (HTML only)

Table 1: Documentation on the installation and usage of the different packages

Software Package	**Source**
CFITSIO V 3.20 or more	`http://heasarc.gsfc.nasa.gov/fitsio/`

The **IDL** visualization software is commercially available at

Software Package	**Source**
IDL V 6.4 or more	`http://www.exelisvis.com/`

while the GNU Data Language **GDL**, a *free* clone of IDL 6.0, can also be used (with some caveats, see §7.9) and can be downloaded for free from

Software Package	**Source**
GDL 0.9.3 or more	`http://sourceforge.net/projects/gnudatalanguage`

As it was already the case in version 1.20, users no longer need to acquire the **I**DL Astronomy User's Library (`http://idlastro.gsfc.nasa.gov/homepage.html`) or the COBE

(IDL) Analysis Software (http://lambda.gsfc.nasa.gov/product/cobe/cgis.cfm), although we do recommend these packages to the user. The 100-odd routines required for version 3.30 are contained in the subdirectory `Healpix_3.30/src/idl/zzz_external`. These procedures are included in the **HEALPix** package unchanged and solely for the purpose of making it self contained. In this way, we remove the burden of installation of additional libraries from the end user.

The **Python** `healpy` package requires

Software Package	**Source**
Python 2.6, 2.7, or 3.2–3.5	http://www.python.org
Numpy 1.0.1 or more	http://numpy.scipy.org
Matplotlib 0.91.2 or more	http://matplotlib.sourceforge.net
PyFITS	http://www.stsci.edu/institute/software_-hardware/pyfits

While not required, the `IPython` (http://ipython.org) and `Cython` (http://cython.org) softwares can also be useful.

A parallel implementation (based on OpenMP, for shared memory architectures) of the Spherical Harmonics Transforms involved in **F90** `synfast, anafast, smoothing, plmgen, alteralm` and **C++** `synalm_cxx, alm2map_cxx, anafast_cxx, smoothing_cxx, rotalm_cxx` ... is now available by default and can be readily compiled and used with the standard installation script.

A set of routines with MPI parallelization (for distributed memory architectures) is also available for Spherical Harmonics Transform, thanks to the work of H.K. Eriksen (UIO) and Snorre Boasson (ITEA, NTNU). See the **F90** subroutines documentation for more information on how to use those routines in your code.

We found that it was remarkably difficult to find random number generators in the public domain which are simple yet powerful and easy to use. We are providing one (both in **C++** and **F90**) which is an adaptation of an xorshift generator described in Marsaglia (Journal of Statistical Software 2003, vol 8). It has a theoretical period of $2^{128} - 1 \approx 3.4\ 10^{38}$.

3 `healpix_doc`: an easy access to HEALPix documentation

The shell script `healpix_doc` now is available to provide easy access to the HTML and/or PDF documentation of *all* Healpix packages. It will automatically open a web browser or PDF viewer (among those found on the system) on the documentation available locally (at `$HEALPIX/doc`) or on remote web sites. To use it, simply type

```
$HEALPIX/healpix_doc
```
or
```
$HEALPIX/healpix_doc -p
```
to access respectively the HTML and PDF documentation. The default browser and viewer used by `healpix_doc` can *optionally* be set with the environment variables `$HEALPIX_HTML_BROWSER` and `$HEALPIX_PDF_VIEWER`.

4 The Installation Procedure

If the user has one of the supported OS's, then installation proceeds utilizing the following commands. If your OS is not supported, the configuration step should be omitted, `Makefile.in` should be copied as `Makefile` and explicitly tailored to the user environment.

`% ./configure` *[-L]*	uses `Makefile.in` as a template to build the correct Makefile (from user inputs as required), it will also configure the IDL/GDL routines
`% make`	builds all the facilities
`% make test`	tests all the facility previously compiled
`% make clean`	removes object (`*.o`) files
`% make tidy`	removes object files, module files (`*.mod`), executables and libraries
`% make distclean`	same as above and restores the directories to the state of the original distribution

These different steps are detailled below.

4.1 ./configure [-L]

The `./configure` script manages the configuration of the C, C++, Fortran90, IDL and Python suites of routines and facilities.

Since v2.11, it accepts the `-L` option to write the **HEALPix** specific configuration files into the **HEALPix** directory itself rather than in installer's home directory (see § 4.1.1). Using the `-L` option is recommended when doing a *project* or *system wide* installation of **HEALPix** to be accessed by several different users.

An online help is available with `./configure -h`, while `./configure -v` will return the **HEALPix** release number (currently 3.30) and exit.

4.1.1 Configuration profile

A feature introduced in previous releases and enhanced since v2.10, is that the configure script creates a shell configuration file

(located in `${HOME}/.healpix/3_30_`⟨*OS_TYPE*⟩`/config` or in `${HEALPIX}/confdir/3_30_`⟨*OS_TYPE*⟩`/config` if `./configure -L` was used) according to shell type in which various environment variables and aliases are defined for your convenience. If you agree upon prompting, it will also change your default system profile during installation to automatically source this profile. If you do not agree to this change, you will need to explicitly source the configuration file above for any session in which you intend to run **HEALPix** facilities. **In particular, you will have to make sure that the `HEALPIX` system variable is correctly defined (as the full path to the HEALPix directory) before running the package**.

4.1.2 C configuration

The `./configure` script will ask for the C compiler and options to be used, and for the full path of an installed `cfitsio` library to link to. By default, only a static library is created, but the user can also ask for a shared (Unix/Linux systems) or dynamic (Darwin) library. After compilation (see `make` section) and linking, all libraries will be in `${HEALPIX}/lib/chealpix.*` .

4.1.3 C++ configuration

Two options are available:

- The interactive `./configure` script will ask for the full path to an installed `cfitsio` library to link to, and then provide a choice of predefined targets corresponding to different combinations of C++ compilers and options. Each of those targets is defined in a configuration file located in `Healpix_3.30/src/cxx/config/config.`*target*. The user can therefore add new targets or edit existing ones, and the `./configure` script will update its menu accordingly. If a fairly recent version (4.2 or higher) of gcc and g++ is installed on the system, the target "generic_gcc" should always work, except under MacOSX, where "osx" target is required.
  The environment variables EXTERNAL_CFITSIO, CFITSIO_EXT_LIB, CFITSIO_EXT_INC and HEALPIX_TARGET will be set according to the choices made above.
  If the HEALPIX configuration file is sourced as described in § 4.1.1, the full path to the C++ executables will be added to the environment PATH variable.

- Starting with HEALPix 3.0, an automated (`autoconf` generated) `configure` script is available in `Healpix_3.30/src/cxx/configure`. To use it (and also make the compilation, and test it) do
  ```
  cd src/cxx
  ./configure
  make
  ```

```
make test
```

which will put the compilation products in `Healpix_3.30/src/cxx/auto`.

4.1.4 Fortran 90 configuration

When you run `./configure` on a supported system you will be prompted to enter compiler optimisation flags. We have not attempted to provide the best optimisation flags for all operating systems. The configure script will have a guess at optimisation options for some systems, but it is up to the user to figure out an optimal set[2]. From our experience, we have not found significant accumulation of numerical error even when using the most aggressive optimisation level available.
If the HEALPIX configuration file is sourced as described in § 4.1.1, the full path to the F90 executables will be added to the environment PATH variable.

4.1.5 IDL/GDL configuration

You will be asked for the external applications you want to use to visualize the Postscript and PNG files created by IDL (or GDL).
If the HEALPix configuration file is sourced as described in § 4.1.1, the aliases `hidl`, `hidlde`, `hgdl` and `hgdlde` are also defined to give you access to HEALPIX routines from IDL (and GDL).
See the **HEALPix** IDL Document for more information on using **HEALPix** IDL/GDL together with other IDL libraries.

4.1.6 Java installation

The configuration and installation of the Healpix Java package is currently handled separately. See table 1 for more information.

4.1.7 Python (healpy) installation

The `./configure` script will ask for the Python command you want to use (in case many coexist on your computer) and will check that its version number meets the `healpy` requirements (see above). Note that during the compilation with `make` (see below), the `src/healpy/setup.py` Python script will be invoked to automatically prompt a *fresh* compilation of the `src/cxx/*` libraries, with all the options necessary to Python linkage, and can be done independently of the C++ installation described above.

[2]In particular, the Intel Fortran Compiler, available for free for Linux systems with Intel-like processors, have specific tuning options for each Intel processor family and instructions set. Please consult the online help (`ifort -help`) or PDF documentation (`/opt/intel/composer*/Documentation/en_US/Release_NotesF.pdf`) or HTML documentation (`/opt/intel/composer*/Documentation/en_US/documentation_f.htm`) for further information.

4.2 Compilation and installation

The
```
make
```
command will compile one or several of the C, C++, F90 and Python packages depending on what was configured with the `./configure` script. Specific packages can be compiled with the respective commands

```
make c-all
make cpp-all
make f90-all
make healpy-all
```

To perform several compilation jobs simultaneously, the command `make -j [jobs]` can be used.

Please neglect any possible warnings at compile time. If you run into trouble please refer to the section **Troubleshooting and further information**.

After running `make`, the user must re-login to ensure that the new profiles built by the installation procedure are correctly sourced. Only then will the user have full access to the specific **HEALPix** environment variables etc.

4.3 Testing the installation

All installed libraries and executables can be tested with

```
make test
```

while specific tests of the C, C++ and Fortran products can be performed with, respectively

```
make c-test
make cpp-test
make f90-test
```

For the latter, Table 2 lists the codes tested with the parameter files used, as well as the data files produced and the respective reference files.

Notes:

- the input power spectrum (in `Healpix_3.30/test/cl.fits`) used to generate the Fortran90 test maps is currently the WMAP 1yr best fit, in $(\mu\mathrm{K})^2$, and is therefore different from the one included in releases 1.* (that can still be found in `cl_old.fits`).

code & parameter file	output data	reference data	output image	reference image
synfast syn.par	test_map.fits	map.fits	test_map.gif	map.gif
	test_alm.fits	alm.fits	*NA*	*NA*
smoothing smo.par	test_sm.fits	map_sm.fits	test_sm.gif	map_sm.gif
ud_grade udg.par	test_LOres.fits	map_LOres.fits	test_LOres.gif	map_LOres.gif
hotspot hot.par	test_ext.fits	map_ext.fits	test_ext.gif	map_ext.gif
	test_max.asc	max.asc	*NA*	*NA*
	test_min.asc	min.asc	*NA*	*NA*
anafast ana.par	test_cl.fits	cl_out.fits	*NA*	*NA*
alteralm alt.par	test_almdec.fits	almdec.fits	*NA*	*NA*
median_filter med.par	test_mf.fits	map_mf.fits	test_mf.gif	map_mf.gif
sky_ng_sim ngfs.par	test_ngfs.fits	map_ngfs.fits	test_ngfs.gif	map_ngfs.gif
process_mask prmask.par	test_distmask.fits	distmask.fits	test_distmask.gif	distmask.gif

Table 2: Data files and images produced by the Fortran codes during the tests, and the respective reference files to which they can be compared. All the files listed are located or produced in the `Healpix_3.30/test` directory. The GIF images of full sky maps were produced using `map2gif`. *NA*: No image available, because the data set is not a sky map

- Other input fiducial $C(\ell)$, all in $(\mu\mathrm{K})^2$ and defined on the multipole range $[0, \ell_{\max}]$, have been included for convenience in `Healpix_3.30/test/` in a **HEALPix** compatible format.

File name	**alias**	**Origin**	$\ell_{\max}$
wmap_lcdm_pl_model_yr1_v1.fits	cl.fits	WMAP-1yr (2005)	3000
wmap_lcdm_sz_lens_wmap5_cl_v3.fits	cl_wmap5.fits	WMAP-5yr (2008)	2000
wmap_lcdm_sz_lens_wmap7_cl_v4.fits	cl_wmap7.fits	WMAP-7yr (2011)	3726
planck2013ext_lcdm_cl_v1.fits	cl_planck1.fits	Planck 2013	4500
planck2015_lcdm_cl_v2.fits	cl_planck2.fits	Planck 2015	4900

  For more information on their respective origin and underlying model see their FITS header, or `Healpix_3.30/test/README`

In order to test the new **HEALPix** profile set-up one can then attempt to run any C++ or F90 facility from any directory on your system. Similarly, IDL (and/or GDL) can be tested by invoking `hidl` or `hidlde` (resp. `hgdl` or `hgdlde`).

4.4 Cleaning up

Three levels of cleaning are available:

```
make clean
```

will remove the intermediate files created during compilation, such as object files, (Fortran) modules files, ... found in the source or build directories;

```
make tidy
```

same as above, and will also remove the **HEALPix** executables, libraries and module and/or include files;

```
make distclean
```

will return the **HEALPix** directory to its original 'distribution' state by discarding the same files as above, as well as the executable and library directories and the top level Makefile.

As a consequence, `make clean` can be used after a successful compilation and installation in order to remove now useless intermediate files while keeping the codes functional, while `make tidy` should be used between consecutive (failed) attempts with different compilers, compiler versions or compiler options, to avoid any conflict between new and pre-existing files.

5 A Note on *Re*-installation

As a result of the line added to your shell profile which explicitly sources the **HEALPix** profile, care must be taken if the package is reinstalled in a different directory. If such reinstallation is desired, the included line must be removed from your system profile, allowing the corrected version to be added.

6 Pkg-config files

Starting with **HEALPix** 3.12, `pkg-config` (`.pc`) files are generated during the configuration of the C, C++ and F90 packages, and are initially located respectively in `${HEALPIX}/lib/chealpix.pc`, *install_lib_directory*`/pkgconfig/healpix_cxx.pc` and `${HEALPIX}/lib`*suffix*`/healpix.pc`. If the `pkg-config` software is available on your system (see http://www.freedesktop.org/wiki/Software/pkg-config/ to download, install and use it) and if the location of the **HEALPix** `pkg-config` files above are known to it (either by moving/copying them to one of the standard locations returned by
`pkg-config --variable pc_path pkg-config`
or by customizing the environment variable `PKG_CONFIG_PATH`[3]) then linking your own code with the C, C++, F90 **HEALPix** library simply becomes

```
cc `pkg-config --cflags --libs chealpix` mycode.c -o mycode
c++ `pkg-config --cflags --libs healpix_cxx` mycode.cpp -o mycode
FC `pkg-config --cflags --libs healpix` mycode.f90 -o mycode
```

(where *FC* has to be replaced by the Fortran compiler used to generate the **HEALPix** library).

[3] a third option is provide the location of the `.pc` file in full at each `pkg-config` invocation : *eg* `pkg-config --cflags --libs` *full_path*`/healpix.pc`

7 Troubleshooting and further information

This section contains a list of difficulties which we have dealt with. It is by no means exhaustive. In case of problems, see http://healpix.sourceforge.net/support.php or contact *healpix-support* `at` *lists.sourceforge.net*

7.1 Free Fortran90/95 Compilers

The **free** Fortran90/95 compilers that can be used do compile **HEALPix** include:

- **Intel Fortran** Compiler for Linux based computers (versions 11.* to 15.*[4])
  http://software.intel.com/en-us/intel-compilers
- **GNU Fortran 95** compiler (gfortran) included in GNU Compiler Collection *GCC* version 4.0.0 and up and available for Linux, Mac OSX, Windows, Sun ... platforms
  http://www.gnu.org/software/gcc/fortran/.
  GFortran binaries for all platforms can also be downloaded from
  http://gcc.gnu.org/wiki/GFortranBinaries.
  Please note that only the most recent versions of gfortran (Aug 2005 and later) compile HEALPix correctly, and v4.2.1 has given satisfying results so far, including native OpenMP support.
- **G95** compiler available for Linux, Mac OSX, Windows, Sun and HP platforms with 32 and 64 bit architectures (eg, x86 and x86-64). In the latter case, the '32bit default integer' (32bit DI) version of `g95` *must* be used.
  http://www.g95.org

7.2 Installation under Microsoft Windows

The installation and usage of HEALPix require many standard Unix/Linux tools (such as `sh, make, awk, grep, sed, ls, wc, cat, more, nm, ar`) as well as C, C++ and Fortran compilers. To install it under Windows, you will need to

- Install Cygwin on your machine (see http://cygwin.com/). In addition to the default packages, you need at least the `binutils, coreutils, util-linux, bash, gawk, grep, make` and `sed` packages, as well as `gcc` and gcc-g++ packages, all available at http://cygwin.com/packages/.
- Install the latest gfortran binaries for Cygwin from http://quatramaran.ens.fr/coudert/gfortran/, following the instructions at http://gcc.gnu.org/wiki/GFortranBinaries.

[4] problems have been reported with one of the code (`sky_ng_sim`) when compiled with `ifort 14.0.1.106`

- Unpack the HEALPix software package
- Run configure as you would on other platforms
- The C++ code can be compiled using HEALPIX_TARGET=generic_gcc

7.3 Problems with CFITSIO

Compilation of CFITSIO Fortran wrappers

The most common problem with the Fortran **HEALPix** compilation will produce messages like:

```
ld: Undefined symbols:
 _ftbnfm_
 _ftclos_
 _ftcrhd_
 _ftdkey_
 ...
```

or

```
fitstools.f90: undefined reference to ‘ftdkey_’
fitstools.f90: undefined reference to ‘ftbnfm_’
fitstools.f90: undefined reference to ‘ftclos_’
...
```

or

```
Undefined symbols:
 "_ftghbn_", referenced from:
     ___fitstools_MOD_read_fits_cut4.clone.2 in libhealpix.a(fitstools.o)
     ___fitstools_MOD_getsize_fits.clone.1 in libhealpix.a(fitstools.o)
     ___fitstools_MOD_getsize_fits in libhealpix.a(fitstools.o)
  ...
ld: symbol(s) not found
collect2: ld returned 1 exit status
```

and occurs when the CFITSIO installation script could not find a valid fortran compiler. To solve this problem

1. Go into the CFITSIO directory.
   Assuming that **ifort** is available on your system (it can be replaced below by **gfortran**, **g95**, **f77**, **f2c**, ...) type:

```
./configure FC=ifort
make
make install                (optional).
```

2. Then go back into the **HEALPix** directory and do
```
./configure                 (making sure that you are using the newly created
```
`libcfitsio.a` library)
```
make
make test
```

See also the note below on 64 bit architectures.

CFITSIO problems on systems with 64 bit architecture

1. **Linux, Mac OS X**

   If the **HEALPix** codes are compiled in 64 bits, and the GNU C Compiler (`gcc`) is used to compiled CFITSIO, then issue the following commands in the CFITSIO directory:

```
./configure FC='gcc -m64'
make
```

   You can then force compilation to the same binary format by entering `-m64` when asked for the optimisation options in the **HEALPix** configure script.

2. **IRIX64**

   On a 64-bit architecture such as IRIX64, CFITSIO will have to be compiled in the same binary format as the **HEALPix** codes. This can be achieved by typing the following on the command line in the CFITSIO directory:

```
rm config.cache
setenv CC 'cc -n32'
./configure
make
```

   Alternatively you can replace the `-n32` with `-64`. You can then force compilation to the same binary format by entering either `-n32` or `-64` when asked for the optimisation options in the **HEALPix** configure script.

CFITSIO linking problems

A particular problem encountered with the CFITSIO Version 2.0 release relates to the inclusion of various libraries within the system release for a given machine. This led to some modifications to the Makefile to include the specific library links `-lm -lnsl -lsocket` on SunOS, but only `-lm` for IRIX64. If your OS is not completely supported by the distribution, you may find this as one source of errors. The CFITSIO developers recommend compilation of the `testprog` routine. Inspection of the libraries linked after executing the `make testprog` statement will reveal those you need to include in the Makefile.

CFITSIO and Debian/Linux

Some problems have been reported on Debian/Linux systems during the linking to the CFITSIO library shipped with Linux. If these problems occur, try to recompile the CFITSIO library from scratch before linking to **HEALPix** .

7.4 `diff` shows that the test files are different from the supplied files

This by itself is no cause for concern. When comparing using a `diff` on the test files will most likely report a difference even when the installation has been successful. This may be due to the fact that different installations have different floating point representations. Also, the FITS files carry date information.

7.5 Try `unlimit`

If you have unforeseen problems at runtime, try `unlimit` (under csh or tcsh) or `ulimit` (under sh or bash), in order to increase the heap and stack memory size. It sometimes helps.

7.6 `hidl` usage

We have found that in very rare cases the alias `hidl` is not recognised by the user's system. Usually, this is related to the local system's IDL script. A quick-fix is achieved by setting the environment variable `IDL_STARTUP` to be equal to the **HEALPix** startup file `HEALPix_startup` **including** the directory path to the file. This enables the user to access the **HEALPix** IDL procedures simply by invoking IDL. For example, in the typical installation documented above for a user running the tcsh shell, the command
`setenv IDL_STARTUP /disk1/user1/HEALPix_3.30/src/idl/HEALPix_startup`
should be issued (or added to the user's shell profile).

If the user already has an IDL startup file, then this should be merged with `HEALPix_startup`. This temporary solution does mean that the **HEALPix** IDL procedures are available in the `IDL_PATH` at all times, which may lead to conflicts with user-defined procedures. The `hidl` invocation was intended to circumvent these issues, allowing **HEALPix** IDL procedures to be available only when desired.

A proper fix requires the user to ask the local system administrator to adjust the local IDL script.

7.7 Using HEALPix IDL together with other IDL libraries

See the eponymous section in the "IDL Facilities Overview"

7.8 Mac OS X, X11 and IDL cursor

If the IDL cursor does not work correctly on X11 windows under Mac OS X, and the 2nd and 3rd button clicks are ineffective, type

- with Apple's X11:
  - under Tiger (10.4.*):
    `defaults write com.apple.x11 wm_click_through -bool true`
  - under Leopard (10.5.*), Snow Leopard (10.6.*) and Lion (10.7.*):
    `defaults write org.x.x11 wm_click_through -bool true`
- with Xquartz (default under Moutain Lion (10.8.*), Mavericks (10.9.*) and Yosemite (10.10.*)):
  `defaults write org.macosforge.xquartz.X11 wm_click_through -bool true`
- with MacPort's X11 (package xorg-server):
  `defaults write org.macports.X11 wm_click_through -bool true`

at your X11 prompt and restart X11.
Note that the command `ls -lrt $HOME/Library/Preferences/*[xX]11*plist` can be used to determine the X window system installed on your Mac. See also http://www.idlcoyote.com/misc_tips/maccursor.html and `mollcursor` documentation in "IDL Facilities").

7.9 Using GDL instead of IDL

GNU Data Language (GDL), is a *free* clone of IDL 6.0 (for more information see http://gnudatalanguage.sourceforge.net). Both the source code and precompiled executables for various platforms are available.

When used to run IDL-Healpix routines, GDL 0.9.3 or more gives satisfactory results[5]. The calculations agree with those done under IDL, with comparable computation times, but a few features are missing in the production of Postscript, GIF and PNG files, as described below.

GDL Specific requirements

1. By default, GDL uses the value of the environment variable `$GDL_DIR`, or the location of the `gdl` executable, as temporary storage disc space location, which may create problems in many situations. It is therefore recommended to set the environment variable `IDL_TMPDIR` to a more suitable location with unrestricted access (such as `/tmp`, `/usr/tmp` or `/var/tmp`) before starting GDL.
   Ie, if your shell is bash, sh, ksh, or zsh:
   ```
   % export IDL_TMPDIR=/tmp
   % hgdl
   ```
   If your shell is csh or tcsh:
   ```
   % setenv IDL_TMPDIR /tmp
   % hgdl
   ```

2. Please note that GDL must be linked with `ImageMagick` (or `GraphicsMagick`) during installation to produce PNG and JPEG output files.

Impact of GDL limitations on HEALPix

- `Ximview` won't work under GDL 0.9.3 (because it requires the IDL native routine `WIDGET_DRAW`)
- When run under GDL, the `cartview`, `gnomview`, `mollview` and `orthview` routines won't produce Postscript outputs (because GDL's `TV` routine does not accept the `NORMAL` keyword) nor GIF outputs (because GDL's `WRITE_GIF` is not yet functional). But PNG and JPEG files are OK, see above.
- In those same routines, the `TRANSPARENT` keyword will be ignored in the production of PNG files under GDL. For the same reasons, `hpx2gs` won't mark missing pixels as transparent in the output PNG file.

HEALPix with GDL status update (Oct 2015)

Some of the caveats and limitations detailed above, and other not listed here, are still present in GDL 0.9.4 (Sept 2013) and 0.9.5 (Oct 2014). However, they have been clearly

[5] All the caveats listed below have been noticed in GDL v0.9.3 and may be solved in subsequent versions. Please send all your questions *on* GDL directly to GDL developpers.

identified and actively looked at by the GDL development team and most, if not all, of them are expected to be fixed in the forthcoming GDL 0.9.6 (scheduled for late 2015).

8 Appendix I: Recent Changes and New Features

8.1 Bug corrections and Improvements in Version 3.30

8.1.1 C++

- support for multi-order coverages (MOC);
- allow generation of $a_{\ell m}$ from 6-component power spectra;
- moved from `alice2` to `alice3`, which produces FITS **HEALPix** maps as output. These can be visualized more flexibly with external tools.
- switch from custom `xcomplex` class to `std::complex`;
- `rangeset` class has been redesigned.

8.1.2 Fortran 90 facilities and subroutines

- `anafast` facility now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when analyzing two polarized maps;
- `alm2cl` subroutine can now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when called with two sets of polarized $a_{\ell m}$ and can also symmetrize the output $C(\ell)$ if requested;
- the $a_{\ell m}$ generated by `create_alm` subroutine can now take into account non-zero (exotic) TB and EB cross-spectra (option `polar=2`) if the input FITS file contains the relevant information
- new routines `nest2uniq` and `uniq2nest` for conversion of standard pixel index to/from Unique ID number. See "The Unique Identifier scheme" section in "**HEALPix** Introduction Document" for more details.
- improved `repeat` behavior in `write_bintabh` routine
- edited `map2alm_iterative` routine to avoid a bug specific to Intel's Ifort 15.0.2
- CFITSIO version 3.20 (August 2009) or more now required;

8.1.3 IDL

- `azeqview, cartview, gnomview, mollview, orthview` visualization routines:
  - addition of `PDF` keyword for production of Adobe PDF outputs;

  - addition of LATEX keyword for genuine or emulated LaTeX processing of character strings;
  - addition of PFONTS keyword to select origin and type of character font;
  - the CROP keyword now has the same behavior for all output media (GIF, JPEG, PDF, PNG, PS, ... and X); the NOBAR keyword now removes the color bar *or* the polarization color wheel, as applicable; correct EQUINOX date in header of output FITS map; the double precision maps and those with constant value are now correctly handled.

- `fits2cl`: addition of /PLANCK2 keyword to read best fit $C(\ell)$ model to Planck 2015 data.

- new routines `nest2uniq` and `uniq2nest` for conversion of standard pixel index to/from Unique ID number. See "The Unique Identifier scheme" section in "**HEALPix** Introduction Document" for more details.

- **HEALPix** enabled GDL commands (`hgdl` and `hgdlde`) are defined during the configuration process.

- update of the required `IDL-astron` library routines, and `Coyote` library routines (2015-09-23).

8.1.4 Java

- deprecated parts of the library have been removed;
- MOC support (see http://ivoa.net/documents/MOC/ for high-level description);
- queries for arbitrary polygons (using MOC);
- new targets in `build.xml` which allow compilation without external JARs.

8.1.5 Python

- switch to `healpy 1.9.0`
  - same C++ source code as **HEALPix** 3.30
  - drop support for Python 2.6
  - support for `astropy.fits`
  - improvements to `read_map` and `write_map`
  - renamed `get_neighbours` to `get_interp_weights`
  - several bug fixes in build and installation processes

9 Appendix II: Older changes (versions 3.00 to 3.20

Bug corrections and Improvements in Version 3.20 (2014-12)

General

- Generation of pkg-config files during the configuration of the C, C++ and F90 packages. See Section 6 of "**HEALPix** Installation" for details.

C

- Top `configure` script now proposes compilation with *or* without CFITSIO-related functions
- Improved `autotools` support

C++

- automatic workaround for bugs in older versions of GNU `g++` compiler (bug reports 37, 45, 48, 51)
- workaround for possible bug in Intel `icc` 14.0 compiler
- bug fix in Mollweide projection in `map2tga` when not looking at (0,0)
- `autotools` updates
- deprecation warnings in `alice2`, soon to be replaced

Fortran 90 facilities and subroutines

- **HEALPix**-F90 routines and facilities can now also be compiled with the free Fortran95 compiler **g95** (http://www.g95.org/). See Section 7.1 of "**HEALPix** Installation" for details.
- A separate `build` directory is used to store the objects, modules, ... produced during the compilation of the source codes
- improved handling of long FITS keywords, now producing FITS files fully compatible with the PyFITS and `Astropy` (www.astropy.org) `Python` libraries
- improved FITS file parsing in generate_beam, affecting the external $B(l)$ reading in the F90 facilities alteralm, synfast, sky_ng_sim, smoothing.

IDL

- addition of ialteralm to modify Spherical Harmonics coefficients (a_{lm}).
- addition of planck_colors to modify current color table to one used in Planck 2013 publications.
- cartview, gnomview, mollview, orthview:
  - addition of BAD_COLOR, BG_COLOR and FG_COLOR keywords to change the color of the missing pixels, background and foreground labels and lines.
  - support for COLT='planck1' and COLT='planck2' to use the Planck color tables defined in planck_colors
- Bugs correction in bin_llcl, query_disc.
- update of the required IDL-astron library routines, and their supporting Coyote routines (2014-11-10).

Java

- explicit deprecation warnings in the source codes

Python

- switch to healpy 1.8.1
  - fixes bugs in monopole removal,
  - adds orthographic projection,
  - easier install on MacOSX

Bug corrections and Improvements in Version 3.11 (2013-04)

General

- libsharp C library used for Spherical Harmonics Transforms in Fortran and C++ since **HEALPix** 3.10 can now be compiled with *any* gcc version.

C++

- See General section above

Fortran 90 facilities and subroutines

- bug correction in query_disc routine in inclusive mode
- bug correction in alm2map_spin routine, which had its spin value set to 2
- See General section above

IDL

- ang2pix_ring and pix2ang_nest routines now accept scalar arguments

Bug corrections and Improvements in Version 3.10 (2013-03)

General

N/A

C

- experimental GNU autotools support (undocumented); the standard configuration script remains available

C++

- Spherical Harmonics Transform library libpsht replaced by libsharp (Reinecke & Seljebotn, 2013).
  Note that some gcc versions (4.4.1 to 4.4.6) crash with an internal compiler error during compilation of libsharp. The problem has been fixed in gcc 4.4.7, 4.5.*, 4.6.*, 4.7.* and newer versions and was not present in versions 4.2.* and 4.3.*.
- added boundaries() method to T_Healpix_Base
- experimental GNU autotools support (undocumented); the standard configuration script remains available

Fortran 90 facilities and subroutines

- all Fortran facilities now support most of cfitsio's "Extended File Name Syntax" features, allowing the reading and processing of an arbitrary HDU and table column out of remote, compressed FITS files. For example, setting
  infile = ftp://*url/file.fits*.gz[*extn*][col *colname*]
  in anafast will download the FITS file *file.fits.gz* from *url*, uncompress it, open the HDU (extension) featuring keyword EXTNAME=*extn*, or the one with 1-based rank number *extn*, read the table column with TTYPE*=*colname* out of it and will analyze it.
  It is also possible to perform a remote anafast analysis of a Planck Legacy Archive (PLA) sky map named *map.fits* via the PLA AIO Subsystem by simply setting infile=http://pla.esac.esa.int/pla/aio/product-action?MAP.MAP_ID=*map.fits* as input map file.
- yet faster synfast, anafast, smoothing thanks to libsharp routines (see warning on gcc releases above).

IDL

- bug corrections: query_disc: correct handling of empty disc; bin_llcl: correct handling of optional argument.
- double precision of input now preserved in gaussbeam and euler_matrix_new.
- fits2cl: addition of /PLANCK1 keyword to read best fit $C(l)$ model to Planck 2013 + external data.
- it is now possible to read a specific FITS file extension identified by its (0-based) number or its case-insensitive EXTNAME value with the Extension keyword added to fits2cl, getsize_fits, read_fits_map, read_fits_s and read_tqu.
- update of the required IDL-astron library routines, and their supporting Coyote routines (2013-02-08).

Java

N/A

Python

- switch to healpy 1.5.0: addition of gauss_beam to generate Gaussian beam window function.

Bug corrections and Improvements in Version 3.0 (2012-11)

General

Introduction of the script `healpix_doc` for easy access to the **HEALPix** PDF and HTML documentation.

C

- Interface has remained unchanged, but the code has been replaced by a C port of the relevant Healpix C++ functions, resulting in significant speedups.
- Additional functions are provided which support Nside values up to 2^{29}. They have the same name as the traditional functions, with a "64" suffix appended.

C++

- Query routines: query_polygon() and query_polygon_inclusive() added. Query routines now return lists of pixel ranges instead of lists of pixels, which is much more economic. Inclusive query routines: tradeoff between performance and number of false positives is tuneable. Queries now work natively in both NESTED and RING schemes. Operations on the NESTED scheme are typically slower than in RING, but still much faster than computing the query in RING and converting all pixel numbers to NESTED afterwards.
- Healpix_Base: Healpix_Base and Healpix_Base2 have been merged into the templated class T_Healpix_Base; functionality is still available under the old names. Various performance improvements to T_Healpix_Base functionality
- User-friendliness: module parameters can now optionally be passed on the command line instead of using a parameter file. For example: `anafast_cxx nlmax=500 infile=test.fits iter_order=3` ⟨...⟩
  Facilities now check input maps for undefined pixels before calling map2alm(). If undefined pixels are found, a warning is printed, and the pixels are set to zero. udgrade_cxx refuses downgrading of polarised maps (which would produce unphysical results)
- Bug fixes: accuracy of pix2ang near the poles at high resolutions has been improved.
- Configuration: optional autoconf support
- Interface changes:
  - Healpix_Base::query_*(): new interface
  - cxxutils.h has been split up into announce.h (dealing with module banners), share_utils.h (dealing with subdividing tasks between multiple workers) and string_utils.h (dealing with string manipulation and file parsing)
  - psht.h: interface to alm_info changed in order to add MPI support
  - ylmgen_c.h: Ylmgen_init() interface has changed
  - bluestein.h: bluestein_i() interface changed

Fortran 90 facilities and subroutines

- Compressed and/or remote (ftp or http) FITS files can now be read. CFITSIO 3.14 or more is now required;
- introduction of the process_mask facility to compute the angular distance of valid pixels to the closest invalid pixels for a input binary mask, and of the supporting routines dist2holes_nest, fill_holes_nest, maskborder_nest, size_holes_nest;
- the pixel query routine query_disc has been improved and will return fewer false positive pixels in the inclusive mode;
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in nest2ring, ring2nest, pix2ang_*, pix2vec_*, ...;
- sky_ng_sim now allows the computation of the spatial derivatives of the non Gaussian map being produced, and the output of the a_{lm} coefficients of that map;
- anafast now allows the pro/down-grading of the input mask to match the resolution of the map(s) being analyzed;
- the median filter routine medfiltmap, used by the facility median_filter is now parallelized.

IDL

- New routines to go from circular beam profile to transfer function (beam2bl), and back (bl2beam); to go from indexed list of a_{lm} to a(l,m) 2D table (alm_i2t), and back (alm_t2i); and to compute the angular distance between pairs of vectors (angulardistance).
- addition of iprocess_mask interface to F90 `process_mask` facility to compute the angular distance of valid pixels to the closest invalid pixels for a input binary mask.
- creation of hpx2dm routine to generate DomeMaster images of **HEALPix** maps that can be projected on planetariums.
- the pixel query routines query_triangle, query_polygon, and in particular query_disc, have been improved and will return fewer false positive pixels in the *inclusive* mode
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in nest2ring, ring2nest, pix2ang_*, pix2vec_*, ...
- cartview, gnomview, mollview, orthview: the length and spacing of the headless vectors used to represent polarization is now user-controlled via `POLARIZATION` keyword. The `COLT` keyword now allows the use of an interactively modified color table.
- orthview now accepts `STAGGER` keyword to overplot staggered spheres (with a twist) in order to detect periodic boundary conditions on the sky
- fits2cl: addition of `WMAP7` keyword to read best fit $C(l)$ model to WMAP 7yr data.
- read_fits_map can now read $N_{\rm side}$=8192 **HEALPix** maps and is generally faster than previously for smaller maps
- update of `astron` library routines (01-Feb-2012).

Java

- Core functionality has been reimplemented from scratch in the form of the "healpix.essentials" package. It is strongly recommended to use this package directly in future projects making use of Java HEALPix. "healpix.essentials" is a port of the Healpix C++ library and presents a very similar interface.

  The "healpix.core" package is still provided. It uses "healpix.essentials" internally, and its interface has been kept stable as much as possible. Some adaptations in user code will still be necessary, however. Please note that using "healpix.core" will result in slightly lower performance than calling "healpix.essentials" methods directly, because of the necessary data conversion.

- New features and improvements introduced with the HealpixBase class, compared to the HealpixIndex, Healpix and PixTools classes:

  - close similarities with Healpix_Base_T class from Healpix C++, which allows simultaneous development and bug fixes for both.
  - support for arbitrary positive Nside values in RING scheme; no longer limited to powers of 2
  - maximum supported Nside value: 2^{29}
  - significant performance improvements: most methods have been accelerated by integral factors, some by more than an order of magnitude.
  - re-implementation of queryDisc and queryPolygon, with same new features as the C++ implementation (see above).
  - the HealpixProc class offers a procedural (instead of object-oriented) interface to the HealpixBase functionality, which simplifies transition for users of the "Healpix" and "PixTools" classes. NOTE: this only works for Nside parameters which are powers of 2
  - many bug fixes
  - no external library dependencies, except for "nom.tam.fits" if FITS I/O is required

Python

- the healpy package (C. Rosset, A. Zonca et al.) is now part of **HEALPix**