

PAR4ALL Installation Guide

—
HPC Project

PAR4ALL TEAM

April 23, 2012

This installation guide is for Par4All version 1.4

This document can be found in PDF format on http://download.par4all.org/doc/installation/par4all_installation_guide.pdf and in HTML on http://download.par4all.org/doc/installation/par4all_installation_guide.htdoc.

1 Introduction

You can install PAR4ALL in different ways, more or less automatic. The easiest way to install PAR4ALL is to use update manager or tarball files. If you are interested in a more advanced installation from the sources, you can clone the PAR4ALL GIT repository.

This document describes these different ways of installing PAR4ALL as well as the prerequisites of the installation and the way to verify that your installation is correctly done.

2 Requirements

Current version of PAR4ALL should work on any GNU/Linux distribution, especially Debian, Ubuntu and Fedora.

2.1 Packages needed to build and run Par4All

To run PAR4ALL with Debian or Ubuntu, the following packages are needed:

```
cproto, gfortran, gnuplot, ipython, libmpfr-dev, libncurses5, libpython,
libreadline5, python, python-ply.
```

If you get a precompiled version of PAR4ALL, you may select a more specific Python version.

To run PAR4ALL with Fedora Core 15, 14 or 13, the following packages are needed:

```
cproto, gnuplot, ncurses-5, readline, python, python-ply, python3,
gcc-gfortran, ipython.
```

The following packages need to be installed especially if you use a less automatic way of installation such as a tarball file, or if you install from the sources.

To compile PAR4ALL with Debian or Ubuntu, the following additional packages are needed:

```
cproto, indent, flex, bison, automake, libtool, autoconf, libreadline6-dev,
python-dev, swig, python-ply, libgmp3-dev, libmpfr-dev, gfortran, subversion,
git, wget
```

In Ubuntu 11.10 there is a bug in the `libtinfo-dev` and `libtinfo5` packages that prevents `tpips` from being compiled. If you are really brave, you can try to download manually the Debian packages and force their installation with:

```
wget -nd http://ftp.us.debian.org/debian/pool/main/n/ncurses/libtinfo-dev_5.9-4_amd64.deb
sudo dpkg --force-depends -i libtinfo-dev_5.9-4_amd64.deb
wget -nd http://http.us.debian.org/debian/pool/main/n/ncurses/libtinfo5_5.9-4_amd64.deb
sudo dpkg -i libtinfo5_5.9-4_amd64.deb
```

The links from Debian/unstable are expected to change. So look at <http://packages.debian.org/sid/amd64/libtinfo5/download> and <http://packages.debian.org/sid/amd64/libtinfo-dev/download>. See <https://bugs.launchpad.net/ubuntu/+source/ncurses/+bug/900635> for the bug report.

To compile PAR4ALL with Fedora Core 15, 14 or 13, the following additional packages are needed:

```
cproto, indent, flex, bison, automake, libtool, autoconf, readline-devel,
python-devel, swig, python-ply, gmp-devel, mpfr-devel, gcc-gfortran,
subversion, git, wget
```

To compile the documentation with Debian or Ubuntu, the following additional packages are needed:

```
python-docutils, texlive-full, tex4ht
```

To compile the documentation with Fedora Core 15, 14 or 13, the following additional packages are needed:

```
texlive, tex4ht
```

2.2 CUDA environment to compile and execute on NVIDIA GPU

To compile and to run the CUDA and OPENCL output, you should have the following environment variables set:

- `CUDA_DIR` to the directory where CUDA has been installed (default to `/usr/local/cuda`)
- `LD_LIBRARY_PATH` should contains at least `$CUDA_DIR/lib64`

3 Installations

This section represents different ways to install PAR4ALL.

The installation is done into `/usr/local/par4all`. If you want PAR4ALL installed in another location, have a look at the section 3.6 for a binary installation and at the section 3.8 for the installation from the sources.

3.1 Installation using Par4All package repository

The best way if you are on GNU/Linux Debian or Ubuntu is to use our package repository. This way, when a new version is out, your classical package manager can automatically install it.

To use our package repository, pick one of the following lines, and add it graphically with the Update Manager with Settings/Third-Party Software or append it with a text editor to your `/etc/apt/sources.list`, if you are using Ubuntu:

```
deb http://download.par4all.org/apt/ubuntu releases main
# --OR--
deb http://download.par4all.org/apt/ubuntu development main
```

or if you are running Debian::

```
deb http://download.par4all.org/apt/debian releases main
# --OR--
deb http://download.par4all.org/apt/debian development main
```

So you need to choose between `releases` or `development` versions. Development packages are generated often, may be unstable, and are best suited if you want to track more closely the PAR4ALL development.

Once this is done, run your favorite graphics package tool (synaptic...) or:

```
sudo aptitude update
sudo aptitude install par4all
```

To set your environment up and test your PAR4ALL installation, please refer to the section 4.

3.2 Installation using a package

A less automatic way on Debian or Ubuntu is to install a Par4All `.deb` package manually.

3.2.1 Getting the package

For release versions, according to your OS and architecture, download a package from:

- <http://download.par4all.org/releases/debian/i686>
- http://download.par4all.org/releases/debian/x86_64
- <http://download.par4all.org/releases/ubuntu/i686>
- http://download.par4all.org/releases/ubuntu/x86_64

For development versions, according to your OS and architecture, download a package from:

- <http://download.par4all.org/development/debian/i686>
- http://download.par4all.org/development/debian/x86_64
- <http://download.par4all.org/development/ubuntu/i686>
- http://download.par4all.org/development/ubuntu/x86_64

3.2.2 Installing the package

You can then install the package with:

```
sudo gdebi <the_package>.deb
```

or

```
sudo dpkg -i <the_package>.deb
```

The second one would also work but does not automatically install dependencies you should install later.

3.3 Installation using tar.gz binary distribution

An even less automatic way is to use a tarball `.tar.gz` file. It contains the binaries as built on a stable Ubuntu or unstable Debian distribution.

It should work also on any GNU/Linux distribution with the following libraries installed: (a fairly recent) `libc.so.6`, `libncurses.so.5`, `libreadline.so.6`, etc. and Python 2.7. We chose this Python version because it is recent enough to provide nice features for Par4All and not too recent to be absent from most Linux distributions. See the section 2 to have the list of some needed packages.

If you are not on Debian on Ubuntu, try one of the following, it may work.

3.3.1 Getting the distribution

For the 64-bit x86 architecture, according to the OS and version you want, download a `...x86_64.tar.gz` file from

- http://download.par4all.org/releases/debian/x86_64
- http://download.par4all.org/releases/ubuntu/x86_64
- http://download.par4all.org/development/ubuntu/x86_64
- http://download.par4all.org/development/debian/x86_64

For the 32-bit x86 architecture, according to the OS and version you want, download a `...i686.tar.gz` file from

- <http://download.par4all.org/releases/debian/i686>
- <http://download.par4all.org/releases/ubuntu/i686>
- <http://download.par4all.org/development/debian/i686>
- <http://download.par4all.org/development/ubuntu/i686>

3.3.2 Installing the distribution

Once you have downloaded one of these `.tar.gz` packages from <http://download.par4all.org>, extract it with the following command::

```
tar xvzf <the_package>.tar.gz
```

It will create a directory named `par4all`. Move this directory to its final location:

```
sudo mv par4all /usr/local
```

Then please go to the section 4 to get information on how to set up your environment and to test your installation.

3.4 Installation using the sources

This is not the preferred way to work, but it can be useful for people who cannot use a precompiled version.

3.4.1 Get the source distribution

First get a source tar ball in the following directories (Ubuntu or Debian do not matter here) :

- <http://download.par4all.org/releases>
- <http://download.par4all.org/development>

Pick up a file with a name ending with `_src.tar.gz`. You can decompress it with a `tar zxvf`.

3.4.2 Get a source snapshot from the git repository

On <https://git.hpc-project.com/cgit/par4all> you can find the snapshots in the Download section in `.zip`, `.tar.gz` or `.tar.bz2`.

By tweaking the URL of a tag snapshot you can even get a snapshot from any branch or commit, not only tag. For example the last snapshot of the `p4a` integrated branch is in:

```
https://git.hpc-project.com/cgit/par4all/snapshot/p4a.tar.bz2
```

3.4.3 Installing from the sources

Please refer to the section 3.5 to get to know how to install PAR4ALL from the sources.

3.5 Installation using git

From the PAR4ALL source directory, PAR4ALL is compiled and configured by running `src/simple_tools/p4a_setup.py`. See http://www.par4all.org/documentation/par4all_organization for details.

To download and compile PAR4ALL from the GIT, do the following:

```
# Get a working copy of the Par4All public read-only git repository:
git clone --branch p4a git://git.hpc-project.com/git/par4all.git
# Go into the working copy:
cd par4all
# Build everything using 8 processes to speed up things:
src/simple_tools/p4a_setup.py [--prefix=/install/dir] [-v[v[v]]] --jobs=8 [...]
```

PAR4ALL will be installed into `/usr/local/par4all` by default. The target directory must be writable by the installer, either by running as root with `sudo` or by creating first a writable target directory as follows:

```
mkdir /usr/local/par4all
chown your_login_name /usr/local/par4all
```

In general, it is less dangerous to limit the number of commands executed as root, therefore, the latter approach to PAR4ALL installation is preferable.

To install in another location, the `--prefix` option can be used, for example to choose `/opt/par4all`.

```
src/simple_tools/p4a_setup.py --prefix=/opt/par4all
```

Warning: do not use plain system directory names such as `/usr` or `/usr/local` because some system files such as `/usr/include/assert.h` may be overwritten and havoc may happen...

To see more installation options (including specifying the locations for other packages), run `p4a_setup.py -h` or see section 3.8.

To pull a new version, do:

```
git pull origin p4a
```

and run `p4a_setup.py` again. The `--rebuild` and `--clean` parameters should be used to ensure that all sources are recompiled (since PIPS is a frequently-updated project, incremental build is not always guaranteed to succeed). Removing the `build` directory when the `--prefix` directory changes is also recommended in order to remove obsolete information about the installation directory that may remain in the `build` directory and cause the compilation to fail.

3.6 Relocating binary installation

If needed a PAR4ALL binary installation can be relocated. There are a few ways to relocate it:

1. Copy or move all PAR4ALL binary installation to another directory, for example to copy from `/usr/local/par4all` to `/opt`:

```
cp -av /usr/local/par4all /opt
```

2. Extract from a Debian package, for example to install PAR4ALL in `/opt` directory:

```
sudo dpkg -x par4all-1.3.1-xxx.deb /opt
```

After having installed PAR4ALL in a new installation directory (not the default one), `par4all-rc.sh` should be updated. `P4A_DIST` and `P4A_ROOT` variables in the `par4all-rc.sh` should be changed according to this new installation directory. The following examples are for the cases 1 and 2.

Relocation has been done by copying all the binary installation (case 1):

```
P4A_DIST='/opt/par4all'  
P4A_ROOT='/opt/par4all'
```

Relocation has been done using `dpkg -x` command (case 2) :

```
P4A_DIST='/opt/usr/local/par4all'  
P4A_ROOT='/opt/usr/local/par4all'
```

3.7 Advanced installation

To get information on a more advanced installation please see http://www.par4all.org/documentation/par4all_developer_guide

3.8 Installation options

This section concerns the installation with compilation from the PAR4ALL sources. The compilation and installation of PAR4ALL is controlled by the `p4a_setup.py` script, with the usage and options described in this section.

Usage: `p4a_setup.py [options]`; run `p4a_setup.py --help` for options

3.8.1 Options

`-h, --help`: show this help message and exit

3.8.2 Setup Options

`-R, --rebuild`: Rebuild the packages completely.

`-C, --clean`: Wipe out the installation directory before proceeding. Implies `-R` and not skipping any package.

`--skip-polylib, --sp`: Skip building and installing of the polylib library.

`--skip-newgen, --sn`: Skip building and installing of the newgen library.

`--skip-linear, --sl`: Skip building and installing of the linear library.

`--skip-pips, --sP`: Skip building and installing of PIPS.

`--skip-examples, --sE`: Skip installing examples.

`-s PACKAGE, --skip=PACKAGE`: Alias for being able to say `-s pips` (besides `--skip pips`), for example. `-sall` or `--skip all` are also available and means 'skip all', in which case only final installation stages will be performed.

`-o PACKAGE, --only=PACKAGE`: Build only the selected package. Overrides any other option.

`-r PACKAGE, --reconf=PACKAGE`: Always run autoreconf and configure selected packages. By default, only packages which lack a Makefile will be reconfigured. If `--rebuild` is specified, all packages will be reconfigured.

`--root=DIR`: Specify the directory for the Par4All source tree. The default is to use the source tree from which this script comes.

- P DIR, --packages-dir=DIR, --package-dir=DIR: Specify the packages location. By default it is <root>/packages.
- p DIR, --prefix=DIR: Specify the prefix used to configure the packages. Default is /usr/local/par4all.
- b DIR, --build-dir=DIR: Specify the build directory to be used relatively to the root directory as specify the --root option. Default to build
- polylib-src=DIR: Specify polylib source directory.
- newgen-src=DIR: Specify newgen source directory.
- linear-src=DIR: Specify linear source directory.
- pips-src=DIR: Specify PIPS source directory. When changing the directory, do not forget to reconfigure since this is when the source location is taken into account.
- nlpmake-src=DIR: Specify nlpmake source directory.
- c OPTS, --configure-options=OPTS, --configure-flags=OPTS: Specify global configure options. Default is '--disable-static CFLAGS='-O2 -std=c99' OR '--disable-static CFLAGS='-ggdb -g3 -O0 -Wall -std=c99' if --debug is specified.
- g, --debug: Set debug CFLAGS in configure options (see --configure-options). Please note that this option has NO EFFECT if --configure-options is manually set.
- polylib-conf-options=OPTS, --polylib-conf-flags=OPTS: Specify polylib configure opts (appended to --configure-options).
- newgen-conf-options=OPTS, --newgen-conf-flags=OPTS: Specify newgen configure options (appended to --configure-options).
- linear-conf-options=OPTS, --linear-conf-flags=OPTS: Specify linear configure options (appended to --configure-options).
- pips-conf-options=OPTS, --pips-conf-flags=OPTS: Specify PIPS configure options (appended to --configure-options). Defaults to --enable-tpips --enable-pyps --enable-hpfc --enable-fortran95. Setting this option will reset the default value. Note that several flags can be set like this : --pips-conf-options "--enable-tpips --enable-pyps --enable-doc"
- m OPTS, --make-options=OPTS, --make-flags=OPTS: Specify global make options.
- polylib-make-options=OPTS, --polylib-make-flags=OPTS: Specify polylib make opts (appended to --make-options).
- newgen-make-options=OPTS, --newgen-make-flags=OPTS: Specify newgen make options (appended to --make-options).
- linear-make-options=OPTS, --linear-make-flags=OPTS: Specify linear make options (appended to --make-options).
- pips-make-options=OPTS, --pips-make-flags=OPTS: Specify PIPS make options (appended to --make-options).
- j COUNT, --jobs=COUNT: Make packages concurrently using COUNT jobs.
- I, --no-install: Do not install any package (do not run make install for any package). NB: this might break the compilation of packages depending on the binaries of uninstalled previous packages.
- F, --no-final: Skip final installations steps in install directory (installation of various files). NB: never running the final installation step will not give you a functional Par4All build.

3.8.3 General options

- v, --verbose:** Run in verbose mode: each -v increases verbosity mode and display more information, -vvv will display most information.
- log:** Enable logging in current directory.
- report=YOUR-EMAIL-ADDRESS:** Send a report email to the Par4All support email address in case of error. This implies --log (it will log to a distinct file every time). The report will contain the full log for the failed command, as well as the runtime environment of the script like arguments and environment variables.
- report-files:** If --report is specified, and if there were files specified as arguments to the script, they will be attached to the generated report email. **WARNING:** This might be a privacy/legal concern for your organization, so please check twice you are allowed and willing to do so. The Par4All team cannot be held responsible for a misuse/unintended specification of the --report-files option.
- report-dont-send:** If --report is specified, generate an .eml file with the email which would have been send to the Par4All team, but do not actually send it.
- z, --plain, --no-color, --no-fancy:** Disable coloring of terminal output and disable all fancy tickers and spinners and this kind of eye-candy things :-)
- no-spawn:** Do not spawn a child process to run processing (this child process is normally used to post-process the PIPS output and reporting simpler error message for example).
- execute=PYTHON-CODE:** Execute the given Python code in order to change the behaviour of this script. It is useful to extend dynamically Par4All. The execution is done at the end of the common option processing
- V, --script-version, --version:** Display script version and exit.

4 Setting up environment and testing Par4All

4.1 Setting up environment variables

In any case, you will then need to source one of the following shell scripts which set up the environment variables for proper PAR4ALL execution:

- if you use `bash`, `sh`, `dash`, etc.:

```
source /usr/local/par4all/etc/par4all-rc.sh
```

- if you use `csh`, `tcsh`, etc.:

```
source /usr/local/par4all/etc/par4all-rc.csh
```

4.2 Testing and verification

Once you set your environment up, you can, for example, run `p4a -h` to get help about the usage of the `p4a` frontend script.

4.2.1 Examples and demos

In the examples directory there are few examples or demos showing some `p4a`, `tpips`, `pyps` use cases. You can use these examples to test your installation. For example, to test `p4a` using one of the examples in the `P4A` directory go to one of the directories in `P4A` and run:

```
make demo
```

if you got all needed environments installed, included `CUDA`. This will chain all available demos included:

- sequential execution;
- automatic parallelization with `p4a` for parallel execution on multi-cores with `OPENMP`;
- automatic parallelization with `p4a --cuda` for parallel execution on `nVidia GPU`;
- automatic parallelization with `p4a --accel` for an `OPENMP` parallel emulation of a GPU-like accelerator;
- automatic parallelization with `p4a --opencl` for the `OPENCL` parallel execution on `nVidia GPU`.

If you do not have `CUDA` environment in your machine, you can test your installation with `OPENMP`, as the following:

```
make display_openmp
```

which will transform the sequential codes to the `OPENMP`, compile the generated codes, run and display the results. To get more information on the examples, have a look at `examples/README.txt`.

Further you can look at the manual of `p4a` on <http://www.par4all.org/documentation> for information on how to use `Par4All`.