Appendix A:
Installation Guide for the UNIX Versions

1. Required tools.

We assume that you have either an `ANSI C` or a `C++` compiler available. If your machine does not have one (for example if you still use `/bin/cc` in `SunOS 4.1.x`), we strongly suggest that you obtain the `gcc/g++` compiler from the Free Software Foundation or by anonymous `ftp`. As for all GNU software mentioned afterwards, you can find the most convenient site to fetch `gcc` at the address

`http://www.gnu.ai.mit.edu/order/ftp.html`

You can certainly compile PARI with a different compiler, but the PARI kernel takes advantage of some optimizations provided by `gcc` if it is available. This results in about 20% speedup on most architectures*.

1.1. Optional packages: The following programs and libraries are useful in conjunction with GP, but not mandatory. They're probably already installed somewhere on your system (with the possible exception of `readline`, which we think is really worth a try). In any case, get them before proceeding if you want the functionalities they provide. All of them are free (though you ought to make a small donation to the FSF if you use (and like) GNU wares).

• GNU `readline` library. This provides line editing under GP, an automatic context-dependent completion, and an editable history of commands. Note that it is incompatible with SUN command-tools (yet another reason to dump Suntools for X Windows). A recent readline (version number at least 2.2) is preferred, but older versions should be usable.

• GNU `gzip/gunzip/gzcat` package enables GP to read compressed data.

• GNU `emacs`. GP can be run in an Emacs buffer, with all the obvious advantages if you are familiar with this editor. Note that `readline` is still useful in this case since it provides a much better automatic completion than is provided by Emacs GP-mode.

• `perl` provides extended online help (full text from this manual) about functions and concepts, which can be used under GP or independently (`http://www.perl.com` will direct you to the nearest `CPAN` archive site).

• A colour-capable `xterm`, which enables GP to use different (user configurable) colours for its output. All `xterm` programs which come with current X11R6.3 distributions will satisfy this requirement. Under X11R6, you can for instance use `color_xterm` (get the latest version at `http://www.clark.net/pub/dickey/xterm`).

* One notable exception is the native AIX C compiler on IBM RS/6000 workstations, which generates fast code even without any special help from the PARI kernel sources.

2. Compiling the library and the GP calculator.

2.1. Basic configuration: First, have a look at the `MACHINES` file to see if anything funny applies to your architecture or operating system. Then, type

```
./Configure
```

in the toplevel directory. This will attempt to configure GP/PARI without outside help. Note that if you want to install the end product in some nonstandard place, you can use the `--prefix` option, as in

```
./Configure --prefix=/an/exotic/directory
```

(the default prefix is `/usr/local`). This phase extracts some files and creates a directory `O`*xxx* where the object files and executables will be built. The *xxx* part depends on your architecture and operating system, thus you can build GP for several different machines from the same source tree (the builds are completely independent, so can be done simultaneously).

`Configure` will let the following environment variable override the defaults if set:

`AS`: Assembler.

`CC`: C compiler.

`DLLD`: Dynamic library linker.

For instance, `Configure` avoids `gcc` on some architectures due to various problems which may have been fixed in your version of the compiler. You can try

```
env CC=gcc Configure
```

and compare the benches. Also, if you insist on using a `C++` compiler and run into trouble with a recent `g++`, try to use `g++ -fpermissive`.

2.2. Troubleshooting and fine tuning: Decide whether you agree with what `Configure` printed on your screen (in particular the architecture, compiler and optimization flags). If anything should have been found and was not, consider that `Configure` failed and follow the instructions below. Look especially for the `readline` and `X11` libraries, and the `perl` and `gunzip` (or `zcat`) binaries.

In case the default `Configure` run fails miserably, try

```
./Configure -a
```

(interactive mode) and answer all the questions (there aren't that many). Of course, `Configure` will still provide defaults for each answer but if you accept them all, it will fail just the same, so be wary. In any case, we would appreciate a bug report including the complete output from `Configure` and the file `O`*xxx*`/dft.Config.in` that was produced in the process.

Note that even in interactive mode, you can't directly tell `Configure` where the `readline` library and include files are. If they are not in a standard place, it won't find them. Nonetheless, it first searches the distribution toplevel for a `readline` directory. Thus, if you just want to give `readline` a try (as you probably should), you can get the source and compile it there (you don't need to install it). You can also use this feature together with a symbolic link, named `readline`, in the PARI toplevel directory if you have compiled the readline library somewhere else, without installing it to one of its standard locations.

Technical note: Configure can build GP on different architectures simultaneously from the same toplevel sources. Instead of the `readline` link alluded above, you can create `readline-`*osname-arch*, using the same naming conventions as for the `O`*xxx* directory, e.g `readline-linux-i686`.

2.3. Debugging/profiling: If you also want to debug the PARI library,

```
Configure -g
```

will create a directory `O`*xxx*`.dbg` containing a special `Makefile` ensuring that the GP and PARI library built there will be suitable for debugging (if your compiler doesn't use standard flags, e.g. `-g` you may have to tweak that `Makefile`). If you want to profile GP or the library (using `gprof` for instance),

```
Configure -pg
```

will create an `O`*xxx*`.prf` directory where a suitable version of PARI can be built.

2.4. Compilation and tests: To compile the GP binary, simply type

```
make gp
```

in the distribution directory. If your `make` program supports parallel make, you can speed up the process by going to the `O`*xxx* directory that `Configure` created and doing a parallel make here (for instance `make -j4` with GNU make).

2.4.1. Testing

To test the binary, type `make bench`. This will build a static executable (the default, built by `make gp` is probably dynamic) and run a series of comparative tests on those two. To test only the default binary, use `make dobench` which starts the bench immediately.

The static binary should be slightly faster. In any case, this should not take more than one minute (user time) on modern machines. See the file `MACHINES` to get an idea of how much time comparable systems need (we would appreciate a short note in the same format in case your system is not listed and you nevertheless have a working GP executable).

If a `[BUG]` message shows up, something went wrong. Probably with the installation procedure, but it may be a bug in the Pari system, in which case we would appreciate a report (including the relevant `*.dif` file in the `O`*xxx* directory and the file `dft.Config.in`).

Known problems:

• `elliptic`: the test `cmcurve=ellinit([0,-3/4,0,-2,-1])` may give results which differ slightly from the template (last decimal in a few entries). This ultimately depends on the output of

```
polroots(x^3-3/4*x^2-2*x-1)[1]
```

at `\p38`, which may be 2.0 or 1.999... depending on your hardware, libraries, compiler... Intel Pentiums running Linux often trigger this `BUG` (unrelated to the infamous `fdiv` bug), which can safely be ignored in any case: both results are correct given the requested precision.

• `program`: the GP function `install` may not be available on your platform, triggering an error message ("not yet available for this architecture"). Have a look at the `MACHINES` files (the `dl` column) to check if your system is known not to support it, or has never been tested yet.

• If when running `gp-dyn`, you get a message of the form

`ld.so: warning: libpari.so.`*xxx* `has older revision than expected` *xxx*

(possibly followed by more errors), you already have a dynamic PARI library installed *and* a broken local configuration. Either remove the old library or unset the `LD_LIBRARY_PATH` environment variable. Try to disable this variable in any case if anything *very* wrong occurs with the `gp-dyn` binary (e.g Illegal Instruction on startup). It doesn't affect `gp-sta`.

2.4.2. Some more testing [*Optional*]

You can test GP in compatibility mode with `make test-compat`. If you want to test the graphic routines, use `make test-graphic`. You will have to click on the mouse button after seeing each image (under X11). There will be eight of them, probably shown twice (under X11, try to resize at least one of them as a further test).

The `make bench` and `make test-compat` runs produce a Postscript file `pari.ps` in `O`*xxx* which you can send to a Postscript printer. The output should bear some similarity to the screen images.

3. Installation.

When everything looks fine, type

```
make install
```

(You may have to do this with superuser privileges, depending on the target directories.) Beware that, if you chose the same installation directory as before in the `Configure` process, this will wipe out any files from version 1.39.15 and below that might already be there. Libraries and executable files from newer versions (starting with version 1.900) are not removed since they are only links to files bearing the version number (beware of that as well: if you're an avid GP fan, don't forget to delete the old pari libraries once in a while).

This installs in the directories chosen at `Configure` time the default GP executable (probably `gp-dyn`) under the name `gp`, the default PARI library (probably `libpari.so`), the necessary include files, the manual pages, the documentation and help scripts and emacs macros.

By default, if a dynamic library `libpari.so` could be built, the static library `libpari.a` will not be created. If you want it as well, you can use the target `make install-lib-sta`. You can install a statically linked `gp` with the target `make install-bin-sta`. As a rule, programs linked statically (with `libpari.a`) may be slightly faster (about 5% gain), but use much more disk space and take more time to compile. They are also harder to upgrade: you will have to recompile them all instead of just installing the new dynamic library. On the other hand, there's no risk of breaking them by installing a new pari library.

3.1. The Galois package: The default `polgalois` function can only compute Galois groups of polynomials of degree less or equal to 7. If you want to handle polynomials of degree bigger than 7 (and less than 11), you need to fetch a separate archive: `galdata.tgz` which can probably be found at the same place where you got the main PARI archive, and on the `megrez` ftp server in any case. Untar the archive in the `datadir` directory which was chosen at `Configure` time (it's one of the last messages on the screen if you did not run `Configure -a`). You can then test the `polgalois` function with your favourite polynomials.

3.2. The `GPRC` file: Copy the file `misc/gprc.dft` (or `gprc.dos` if you're using `GP.EXE`) to `$HOME/.gprc`. Modify it to your liking. For instance, if you're not using an ANSI terminal, remove control characters from the `prompt` variable. You can also enable colors.

If desired, also copy/modify `misc/gpalias` somewhere and call it from the `gprc` file (this provides some common shortcuts to lengthy names). Finally, if you have superuser privileges and want to provide system-wide defaults, you can copy your customized `.gprc` file to `/etc/gprc`.

In older versions, `gphelp` was hidden in pari lib directory and wasn't meant to be used from the shell prompt, but not anymore. If gp complains it can't find `gphelp`, check whether your `.gprc` (or the system-wide `gprc`) does contain explicit paths. If so, correct them according to the current `misc/gprc.dft`.

4. Getting Started.

4.1. Printable Documentation: To print the user's guide, for which you'll need a working (plain) TEX installation; type

```
make doc
```

This will create, in two passes, a file `doc/users.dvi` containing the manual with a table of contents and an index. You must then send the `users.dvi` file to your favourite printer in the usual way, probably via `dvips`. Also included are a short tutorial (`doc/tutorial.dvi`) and a reference card (`doc/refcard.dvi` and `doc/refcard.ps`) for GP.

If the `pdftex` package is part of your TEX setup, you can produce these documents in PDF format, which may be more convenient for online browsing (the manual is complete with hyperlinks); type

```
make docpdf
```

All these documents are available online from PARI home page and on the `megrez` ftp server.

4.2. C programming: Once all libraries and include files are installed, you can link your C programs to the PARI library. A sample makefile `examples/Makefile` is provided to illustrate the use of the various libraries. Type `make all` in the `examples` directory to see how they perform on the `mattrans.c` program, which is commented in the manual.

4.3. GP scripts: Several complete sample GP programs are also given in the `examples` directory, for example Shanks's SQUFOF factoring method, the Pollard rho factoring method, the Lucas-Lehmer primality test for Mersenne numbers and a simple general class group and fundamental unit algorithm (much worse than the built-in `bnfinit`!). See the file `examples/EXPLAIN` for some explanations.

4.4. EMACS: If you want to use `gp` under GNU Emacs, read the file `emacs/pariemacs.txt`. If you are familiar with Emacs, we suggest that you do so.

4.5. The PARI Community: There are three mailing lists devoted to the PARI/GP package (run courtesy of Dan Bernstein), and most feedback should be directed to those. They are:

• `pari-announce`: to announce major version changes. You can't write to this one, but you should probably subscribe.

• `pari-dev`: for everything related to the development of PARI, including suggestions, technical questions, bug reports or patch submissions.

• `pari-users`: for everything else.

To subscribe, send empty messages respectively to

`pari-announce-subscribe@list.cr.yp.to`
`pari-users-subscribe@list.cr.yp.to`
`pari-dev-subscribe@list.cr.yp.to`

The PARI home page (maintained by Gerhard Niklasch) at the address

`http://www.parigp-home.de/`

maintains an archive of all discussions as well as a download area. If don't want to subscribe to those lists, you can write to us at the address

`pari@math.u-bordeaux.fr`

At the very least, we will forward you mail to the lists above and correct faulty behaviour, if necessary. But we cannot promise you will get an individual answer.

If you have used PARI in the preparation of a paper, please cite it in the following form (BibTeX format):

```
@manual{PARI2,
    organization = "{The PARI~Group}",
    title        = "{PARI/GP, Version 2.1.5}",
    year         = 2000,
    address      = "Bordeaux",
    note         = "available from {\tt http://www.parigp-home.de/}"
}
```

In any case, if you like this software, we would be indebted if you could send us an email message giving us some information about yourself and what you use PARI for.

Good luck and enjoy!