Tutorial: the PARI source code; navigation, debugging; git & branches

Karim Belabas

http://pari.math.u-bordeaux.fr/
Introduction

This talk focuses on the current development version of the PARI library \((2-6-*)\), freely available from our GIT repository, see

\[\text{http://pari.math.u-bordeaux.fr/anongit.html}\]

All material displayed during the talk is available at

\[\text{http://pari.math.u-bordeaux.fr/~kb/}\]
Initial configuration

- git clone http://pari.math.u-bordeaux.fr/git/pari.git
- cd pari
- ./Configure --prefix=GP.prf -pg && make -j4 prf
- ./Configure --prefix=GP.dbg -g && make -j4 dbg
- ./Configure --prefix=GP && make -j4 gp
- make doc
- make test-all
- make install
- GP/bin/gp
First, install ctags. On Debian/Ubuntu systems:

```
sudo apt-get install exuberant-ctags
```

1) For vim
   - `make ctags`
   - add to your `.vimrc`: `set tags=./tags,$PARIDIR/src/tags`

2) For emacs
   - `make etags`
   - add to your `.emacs`: `(setq tags-table-list '("$PARIDIR/src"))`
Define special-purpose editor options and macros to help you editing PARI code. This is what I do for vim (Emacs give you analogous possibilities).

Create and customize ~/.vim/ftplugin/c.vim

```
setlocal autoindent
setlocal path+=${PARIDIR}/src/headers
setlocal path+=${PARIDIR}/0linux-i686

map <buffer> <Esc>a :if expand("%:t") == 'paridecl.h'
  <Bar> edit #
  <Bar> else
  <Bar> edit ${PARIDIR}/src/headers/paridecl.h
  <Bar> endif<C-M>
```
Define syntax highlighting conventions valid in GP scripts and PARI C code. This is what I do for vim (Emacs give you analogous possibilities):

Create and customize `~/.vim/after/syntax/c.vim`

N.B. Vim already knows about the GP syntax and keywords; but it does not know about PARI-specific types, like `GEN`.

```plaintext
syntax keyword cType GEN
syntax keyword cNumber NULL avma bot top
syntax keyword cNumber gpi gen_0 gen_1
```
Edit $HOME/.gdbinit to teach gdb a number of useful macros:

Output:

```c
#define i
    call output((GEN)$arg0)
end

#define ilb
    call outmat(lift(lift(lift((GEN)$arg0)))))
end

#define isb
    call outmat(gprec_w((GEN)$arg0,3))
end

#define v
    call dbgGEN((GEN)$arg0,2)
end
```
GDB configuration (2/2)

Advanced uses:

define w1
    shell rm -f /tmp/gp.tmp1
    call gpwritebin("/tmp/gp.tmp1",$arg0)
end

define bb
    break pari_err2GEN
end

define cc
    signal SIGINT
end

define fs
    p fill_stack()
end
GP configuration

Create and customize ~/.gprc.

Make sure you set: histfile, lines, path, and logfile:

logfile = "~/tmp/log.pari/%m.%d-%H.%M.%S" Possibly colors and prompt. You probably want to increase parisize as well.
Local branches: go to ’pari’ (or any subdirectory, including the ones Configure).

> git branch

kb-modsym

* master * = current branch, master is the DEFAULT branch

mpbern

Go to kb-modsym local branch

> git checkout kb-modsym

> git branch

* kb-modsym

master

mpbern

(Configure, make, test things...) git checkout master to get back to main development branch.
Remote branches: by default, we only know the official repository (the one we cloned from, *origin*), but you can add other ones.

```
> git branch -a  # -a = add remote branches
...    many branches omitted!
remotes/origin/bill-qfisom-stage1
...
> git checkout origin/bill-qfisom-stage1 -b qfisom
```

Here we created a local branch *qfisom* out of the *bill-qfisom-stage1*, and switched to it.

(Configure, make, test things...) *git checkout master* to get back to main development branch.