A BEGINNER'S GUIDE TO INSTALLING PARI ON WINDOWS COMPUTERS

JAMES RICKARDS

Using GP with Windows is easy, as the developers make a Windows binary available for download. However, if you want to work with the PARI library, this is not enough, and you have to work with a Linux (sub)system. If you are a Linux beginner (like I was), then this can be a daunting task! The intent of this manual is to make it straightforward, and it is aimed at Windows users who are familiar with programming in GP but only have very basic notions of Linux.

1. Windows Subsystem for Linux

Windows Subsystem for Linux (WSL) imitates a Linux system, and allows you to take advantage of many features and capabilities without having to install a whole new operating system. Directions for the install can be found here: https://docs.microsoft.com/en-us/windows/wsl/install-win10. You can choose to install either WSL1 or WSL2, but I highly recommend WSL1. The issue is that if you store files in your typical Windows directories, you have to access them using the "/mnt/" folder in WSL. For some reason, this is much slower on WSL2 than WSL1, so your programs run about 50% longer. You can instead host your files in the Linux system, and this makes it better, but it still remains slower than WSL1. On WSL1, I don't find any noticeable difference between storing your files on the Windows or Linux systems.

You also need to choose a Linux distribution; I recommend the latest version of Ubuntu (currently 20.04 LTS), or Debian (stable). The first time you launch WSL, it will install and configure. You need to choose a username and password, and type the following command:

```
sudo apt update
```

You will need your password to execute this command ("sudo" is akin to administrator privileges), as well as internet access. This command updates your system on the most recent versions of available packages. Your are now all set up! Call "wsl.exe" from any directory in a command prompt terminal to launch WSL, and it will open up quickly.

One downside is these distributions are not up to date. The version of PARI/GP (as well as the supporting packages) that can be installed will not be the most recent! In general, Debian typically has slightly newer packages. Besides changing your installation to Debian testing (much more up to date), you can either use the old package (very easy), or install PARI/GP from source (slightly harder, but still straightforward). If you just want to test it out and have a look around, I would recommend the first method (next section). Otherwise, you should go for the up to date installation (the section after that).

Date: September 28, 2021.

2. Installing an old version of PARI/GP in one line

Open up WSL, and type `sudo apt install pari-gp`. In Ubuntu 20.04, this installs PARI/GP version 2.11. If you later decide to use the updated version, you can either go to "Add and Remove Programs" and reset Ubuntu, or you can remove the package with the command `sudo apt remove pari-gp`.

3. Installing the current version of PARI/GP

Installing the current version will take about 5 minutes. Open up WSL, and follow the following commands (don't copy the comments, which come after #):

```
sudo apt-get install build-essential #C compiler and related items
sudo apt-get install libreadline-dev #Enables readline
sudo apt-get install libgmp3-dev #GMP library
sudo apt-get install bison #Needed to use PARI with git
git clone https://pari.math.u-bordeaux.fr/git/pari.git #Clone PARI from the source
cd pari
./Configure
sudo make install
```

PARI/GP is now ready to go! You can call "gp" from anywhere in WSL, and it will open up a session. Note that this installs the development branch of PARI, which is revised almost daily. Using git makes it easy to update to newer versions when they are available, or perhaps switch to the most recent stable version. For example, navigate to the installation folder, and run:

```
git branch -r #List all available branches; press enter to see more, q to end
git branch pari-2.13 origin/pari-2.13 #Track the remote branch locally
git checkout pari-2.13 #Switch to that branch
```

The switch only updates the source code used to generate the installation, so you are still running the old branch until you re-configure and re-compile. This will take a couple of minutes, since you start from scratch.

On the other hand, if you stay in one branch, updating is easy and quick. Call

```
git pull #Update your local branch
sudo make install
```

Since most files are likely unchanged, this runs quite quickly.

4. Advanced installation options

You can configure PARI with `./Configure --tune` instead. This tunes the values of certain constants to your specific setup, and will make PARI/GP faster! This tuning will take about 30 minutes.

For even more speed, you can download the updated version of GMP from source, tune the configuration (before tuning PARI/GP), and install it. Unfortunately, I have run into issues with this tuning on WSL that make it fail at a certain point. It seems to be an issue related to the clock and synchronization, but I don't know how to fix it. If you can make it work, then let me know!

5. Using PARI/GP

Using GP in WSL is much nicer than the Windows port, since you have more control over colours and options. To set them, you need to create a gprc preferences file. Navigate to your home folder in WSL (opening Ubuntu from the start menu shortcut should take you there; for me, it is "home/james"), and call `sudo vim .gprc` to create a .gprc file. You can then insert your preferences, as laid out in the file "pari/misc/gprc.dft" (press i to be able to type, and "ESC :wq" to save and quit). Personally, I choose the following options:

```
lines  = 40
colors = "brightfg"
prompt = "(%H:%M) gp > "
timer = 1
histfile = "gp_history.txt"
breakloop = 1
```

Now that everything is setup, it's time to write programs! The background structure behind a basic program is the following:

- A .h header file containing structure definitions and non-static method declarations.
- Several .c files containing your code. In addition to your .h declaration file, you must include the PARI library with the line `#include <pari/pari.h>`.
- A .gp file where you use "install(...)" to install any methods you want accessible to GP. Adding "addhelp(...)" methods to each installed function makes the package more user-friendly.
- A Makefile to compile the .c files into .o, and assemble these into your .so library file.

Note that if you put all of your code in one .c file, you can just do all the structure and method declarations at the top of the file, saving the need for a .h file (though it is better practice to make a separate .h file). A sample Makefile taking in two files of code ("code1.c" and "code2.c") and generating the library "libcode.so" is as follows:

```
#Where the PARI files are located
PARI_LIB = /usr/local/lib
PARI_INCLUDE = /usr/local/include

#OBJS is the set source files, and target names the library
OBJS = code1.o code2.o
TARGET = code

#Don't change these unless you know what you are doing
CFLAGS     = -O3 -Wall -fno-strict-aliasing
CC         = cc
CPPFLAGS   = -I. -I$(PARI_INCLUDE)
MODLD      = cc
MODLDFLAGS = -shared  $(CFLAGS) $(DLCFLAGS) -Wl,-shared
EXTRAMODLDFLAGS = -lc -lm -L$(PARI_LIB) -lpari
```

```
DLCFLAGS   = -fPIC

#Naming of the library
DYN = lib$(TARGET).so
ALL = $(DYN)

#Rule to make the object files from the .c files
%.o: %.c
    $(CC) -c $(CFLAGS) $(CPPFLAGS) $(DLCFLAGS) $<

#Compiles the library
all: $(DYN)

#Rule to build the library
$(DYN): $(OBJS)
    $(MODLD) -o $@ $(MODLDFLAGS) $(OBJS) $(EXTRAMODLDFLAGS)

#Removes the .o and .so files
clean:
    -$(RM) *.o $(ALL)
```

Ensure that any tabbed line is actually a tab, and not spaces, as the make will fail otherwise. If your code contains a function f that takes two GENs as inputs and outputs a GEN (GEN is the standard data type of PARI objects, and can be an integer, real number, vector, polynomial, etc.), then it can be installed in GP with the command `install(f, GG, f, "./libcode.so")`.

To learn about PARI syntax, go to https://pari.math.u-bordeaux.fr/doc.html and look at the tutorials and manuals there, especially "Users' Guide to the PARI library". There are also some example programs in the "examples" sub-folder of the PARI installation.

Another good tool to consider is GP2C, which can be download with git: "git clone https://pari.math.u-bordeaux.fr/git/gp2c.git". This tool allows you to take GP programs and convert them to C code automatically! The generated code will not be an optimal translation, but it will typically run faster than the corresponding GP program.

6. Tips

- If you use Notepad++ as your text editor, you can download text highlighting packages for various languages that are not already in the program. Someone has created one for GP, and can be found on the master list here: "https://github.com/notepad-plus-plus/userDefinedLanguages/blob/master/udl-list.md" (listed as PARI/GP). I don't know how often the list is updated, so it may be missing some functions in newer versions of GP.
- There is not yet a public syntax highlighting package for PARI in Notepad++, but using the built-in C highlighting works well enough.

- The online documentation for a GP function includes declaration(s) of the PARI function(s) used. This can also be accessed directly from gp by calling "??METHOD".
- A list of all non-static functions can be found in "pari/src/headers/paridecl.h". By going to the corresponding C file, you can examine the source code for any function.
- You can view your Linux files in Windows File Explorer by navigating to the path "\\wsl$". You can modify the files, move them, etc.; behind-the-scenes magic makes sure that nothing gets broken.

CU Boulder, Boulder, Colorado, USA
Email address: james.rickards@colorado.edu
URL: https://math.colorado.edu/~jari2770/