Instruction – how to install GMP library (GNU Multiple Precision Arithmetic Library with float numbers) on Ubuntu

Michał Łepek, 05-09-2018

tested on Ubuntu 12.04

Written basing on:

https://en.wikipedia.org/wiki/GNU_Multiple_Precision_Arithmetic_Library

https://gmplib.org/#DOWNLOAD

https://linkevin.me/tutorial-installing-gmp-library-ubuntu/

https://stackoverflow.com/questions/40645433/ld-library-not-found-for-lgmpxx

https://stackoverflow.com/questions/7225990/gmp-shared-libraries-not-found

Installing the GMP Library

1. Download the lastest release of GMP from: https://gmplib.org/#DOWNLOAD. Specifically, I downloaded MPFR version of this library from https://www.mpfr.org/

2. When you have downloaded the file, extract the contents from the tar.lz file to a temporary folder. I created my temporary folder on my desktop for simplicity's sake.

3. Once extracted, open your terminal and wiggle your way to the temporary folder's directory. For example, mine would be at

```
~/Desktop/GMP
```

If you forgot, the "**ls**" command displays the list of items in your current directory, and the "**cd**" command can be used to go to specific locations. "**cd ..**" can be used to go back to the previous directory.

```
cd Desktop/GMP
```

4. Once at the directory, type in the command:

```
./configure
```

To what I understand, "./configure" will check your system to see if all the dependencies are met, and then create the makefile. To enable C++ features use –enable-cxx:

```
./configure --enable-cxx
```

5. (Optional) If you are met with the error: *No usable M4 in $PATH or /usr5bin*, then you need to install the M4 macro processor onto your Ubuntu system. Luckily, doing so is quite easy if you have Internet access. You must have root access as we will be using sudo apt-get:
   ```
   sudo apt-get install m4
   ```

Don't forget to run "./configure" again once you are done!

6. Now you are ready to build the GMP library's files, awesome! Proceed with the command:

```
make
```

Do note that this process may take a few minutes for some computers. For mine (Ubuntu virtualized within Windows), it took me around 3 minutes. The terminal window will start printing out words in supersonic speed. If you have a friend nearby, grab him over so you can tell him you're hacking GMail's servers.

7. When you are ready to install the library, type in this command:

```
sudo make install
```

8. GMP's documentation prompts us to check to see if our installing was correct. To do so:

```
make check
```

If no errors show up, congratulations. You now have GMP installed!

To use the library with a C++ compiler use:

```
g++ test.cpp -o test -lgmp -lgmpxx
```

Issues that I faced:

1. There can be an error like „`gmpxx.h not found`" or „`lgmpxx not found`". It probably means you didn't use –enable-cxx with ./configure.

2. There can be an error „`libgmp.so.4: cannot open shared object file: No such file or directory`". Then simply run `sudo ldconfig` (or `ldconfig` as root) and try again.

Exemplary programs

1. Adding numbers in C-style (no lgmpxx needed)

```
#include<iostream>
#include<gmp.h>

using namespace std;

int main (int argc, char **argv) {

    mpz_t a,b,c;
    mpz_inits(a,b,c,NULL);

    mpz_set_str(a, "1234", 10);
    mpz_set_str(b,"-5678", 10); //Decimal base

    mpz_add(c,a,b);

    cout<<"\nThe exact result is:";
    mpz_out_str(stdout, 10, c); //Stream, numerical base, var
    cout<<endl;

    mpz_abs(c, c);
    cout<<"The absolute value result is:";
    mpz_out_str(stdout, 10, c);
    cout<<endl;

    cin.get();
```

```
  return 0;
}
```

2. Calculating power 2^20000 in C-style (no lgmpxx needed)

```
#include <iostream>
#include <gmp.h>

using namespace std;
int main(void) {
  mpz_t result, base;
  mpz_inits(result,base,NULL);
  mpz_set_str(base, "2", 10);
  mpz_pow_ui(result, base, 20000);
  mpz_out_str(stdout, 10, result);
  return 0;
}
```

3. Multiplying numbers in C++-style (lgmpxx needed)

```
#include <iostream>
#include <gmpxx.h>

int main() {
  mpz_class x("7612058254738945");
  mpz_class y("9263591128439081");

  std::cout << "    " << x << "\n"
            << "*\n"
            << "    " << y << "\n"
            << "--------------------\n"
            << x * y << "\n";

  return 0;
}
```