Setting up GEM5 from Source Files in RHEL7.6

By Rishabh Chitoor | 17th April 2021

Before proceeding to build GEM5, we need to install the following;

1) Libffi-devel
2) Openssl
3) Python3.6+
4) SCons & required packages

Many times, we might be in a situation where we do not have root access to a server. The major challenge then is we need to install all of the above through source files, which is not easy work.

After having gone through this tedious process myself, filled with hours of "googling" and decoding error messages, I have made this document to hopefully make it much easier to all of you. All the best!

NOTE: Hereafter in this document, I will call your working directory (where you want to install gem5 for your project) as $HOME.

Using "Screen"

When you connect to another server (like the Aqua server) using 'ssh', you might get logged out due to internet issues or server timeout settings. In these cases, if you get logged out, the instruction currently running, and the outputs displayed get deleted when you log back in. Thus, when you are running time-intensive commands, such as "make", there is a high risk of getting logged out and losing your progress.

For this purpose, we use the **"screen"** command. Just type the command "screen" and press enter. You will enter a screen. This is just like your terminal, and can run your regular commands normally.

In case you get logged out of the server, log back in, type "**screen -d -r"** and you will **r**econnect to the previous screen, and the process will still be running.

If you want to run a process, and then come back to your regular server terminal, just use the keyboard shortcut; **"ctrl + a" and then "d".** You will be "detached" from your screen. If you want to reconnect, type **"screen -d -r"** as mentioned before.

Through this, you can ensure you are not at the risk of losing your progress. I suggest you run all the "configure", "make", "make install" and "build" commands through a screen.

Installing Libffi-devel

1) Go to the $HOME directory
2) “wget https://centos.pkgs.org/7/centos-x86_64/libffi-devel-3.0.13-19.el7.x86_64.rpm.html”
3) “rpm2cpio ./libffi-devel-3.0.13-18.el7.x86_64.rpm | cpio -idmv”
4) A “usr” directory would be automatically formed inside the $HOME directory
5) Go to $HOME/usr/share.
6) Create a file “config.site” and add the following two lines to the file;
    a) CPPFLAGS=-I$HOME/usr/include
    b) LDFLAGS=-L$HOME/usr/lib64

Installing Openssl

Run the following commands;

1) Cd $HOME/usr
2) wget https://www.openssl.org/source/openssl-1.0.2o.tar.gz
3) Tar xvf https://www.openssl.org/source/openssl-1.0.2o.tar.gz
4) cd openssl-1.0.2o
5) ./config shared --prefix=$HOME/usr
6) Make
7) Make install

Setting the Library, Package config and Binary paths

The following lines need to be added to the ~/.bash_profile file. You can use ‘vim’ to modify the file.

1) export PATH=$HOME/usr/bin:$PATH
2) export PATH_SCREEN=$PATH
3) export LDFLAGS="-rdynamic"
4) export LDFLAGS_SCREEN=$LDFLAGS
5) export SETUPTOOLS_USE_DISTUTILS=stdlib
6) export SETUPTOOLS_USE_DISTUTILS_SCREEN=$SETUPTOOLS_USE_DISTUTILS
7) export LD_LIBRARY_PATH=$HOME/usr/lib64$HOME/usr/lib:$LD_LIBRARY_PATH
8) export PKG_CONFIG_PATH=$HOME/usr/lib64/pkgconfig:$PKG_CONFIG_PATH
9) export LD_LIBRARY_PATH_SCREEN=$LD_LIBRARY_PATH
10) export PKG_CONFIG_PATH_SCREEN=$PKG_CONFIG_PATH

After adding the above lines, add the following lines to the ~/.screenrc file (if not present, create one). This is to ensure the environment variables are also passed to whichever screen you might be using.

1) setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH_SCREEN
2) setenv PATH $PATH_SCREEN
3) setenv SETUPTOOLS_USE_DISTUTILS $SETUPTOOLS_USE_DISTUTILS_SCREEN
4) setenv LDFLAGS $LDFLAGS_SCREEN
5) setenv PKG_CONFIG_PATH $PKG_CONFIG_PATH_SCREEN

After adding the above lines; If it is your own machine, restart the machine. If you are connecting to some other server through ssh, logout and log back in.

Check if the above environment variables have been updated successfully by doing "echo $(name)" and checking the output. For example, **echo $LDFLAGS** should print **-rdynamic.**
Check the same in the Screen too by running "echo" after entering a screen.

If it has not been updated, terminate the screen, logout from the server, login again and check. If it is still not working, check if you have entered the values in the **.bash_profile and .screenrc** files correctly.

Installing Python3.7

GEM5 needs Python3.6+. The following instructions are for installing Python3.7, but will also work for the later versions.

1) Come to the $HOME directory.
2) wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz
3) Tar xvf https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz
4) ./configure --prefix=$HOME/usr/ --enable-shared --enable-optimizations
5) Make
6) Make altinstall

The Python3.7 binary would now be installed in the $HOME/usr/bin folder. However, GEM5 will search for the name "Python" and "Python3".
Thus, go to $HOME/usr/bin and create the following symbolic links;

1) python → python3.7
2) python3 → python3.7
3) python-config → python3.7m-config
4) python3-config → python3.7m-config

You can check whether it is indeed the latest version of Python, and whether the links are working properly by the following methods;

- The commands “which python”, “which python3” and “which python3.7” should be the same, ie, $HOME/usr/bin
- Run all “python --version”, “python3 --version” and “python3.7 --version” and ensure it is that which you installed.

Similarly, we need to create the links for the Python libraries too. Go to $HOME/usr/lib/ folder, and you will find **libpython3.7m.a** and **libpython3.7m.so.1.0** files. We need to create the following symbolic links in the $HOME/usr/lib folder;

1) libpython3.7m.so -> libpython3.7m.so.1.0
2) libpython.a -> libpython3.7m.a
3) libpython.so.1.0 -> libpython3.7m.so.1.0

That concludes the installation of Python3.7. For one final installation verification, do the following;

1) Run the command “ldd $HOME/usr/bin/python”
2) Check if there is the following line in the output; “libpython3.7m.so.1.0 => $HOME/usr/lib/libpython3.7m.so.1.0”
3) If so, then **SUCCESS!**

Getting PIP

Pip is a very powerful Python tool which will help us install the subsequent required packages.

- Cd ~
- curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
- python3 get-pip.py

You can check the installation with the following command;

- “Which pip” ; if the output is $HOME/usr/bin, then it has been installed successfully.

Installing Protoc

Protoc is a dependency for GEM5, install through the following steps;

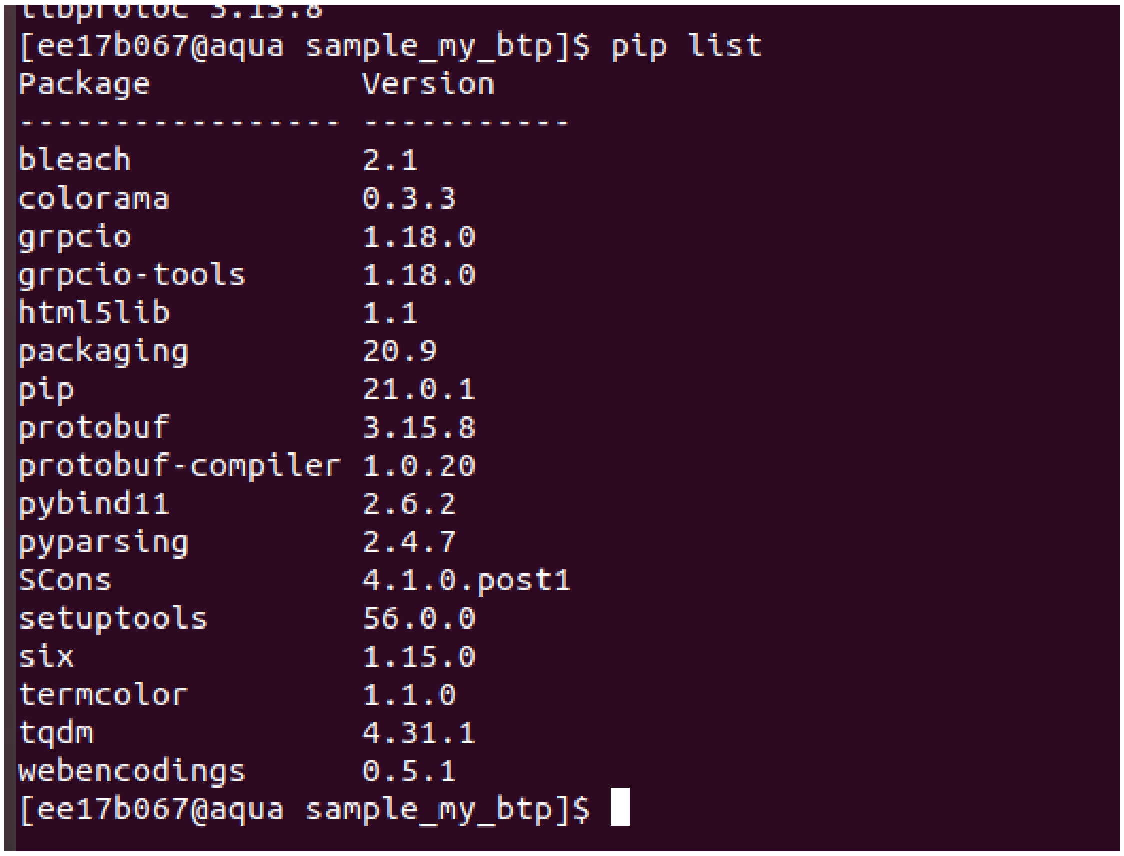
1) Cd $HOME
2) wget https://github.com/protocolbuffers/protobuf/releases/download/v3.15.8/protoc-3.15.8-linux-x86_64.zip
3) Unzip protoc-3.15.8-linux-x86_64.zip

4) After running the unzip command, two folders, “bin” and “include” would have been created in the $HOME directory.
5) Move the files in this $HOME/bin directory to $HOME/usr/bin and the files in this $HOME/include directory to $HOME/usr/include.
6) You can now check installation of protoc by running the “**protoc --version”** command. You should get the output as “**libprotoc 3.15.8**”.

Installing SCons & other packages through Pip

We can install packages through pip using the command “**pip install <package_name>”** and uninstall using the command “**pip uninstall <package_name>”.**
We can check the list of all packages currently installed in the system using “**pip list”.**

We need the following packages;



```
[ee17b067@aqua sample_my_btp]$ pip list
Package           Version
----------------- -----------
bleach            2.1
colorama          0.3.3
grpcio            1.18.0
grpcio-tools      1.18.0
html5lib          1.1
packaging         20.9
pip               21.0.1
protobuf          3.15.8
protobuf-compiler 1.0.20
pybind11          2.6.2
pyparsing         2.4.7
SCons             4.1.0.post1
setuptools        56.0.0
six               1.15.0
termcolor         1.1.0
tqdm              4.31.1
webencodings      0.5.1
[ee17b067@aqua sample_my_btp]$
```

If you run “**pip list”** in your terminal, you might have most of the above packages already installed. However, they would have been installed by the System Python, and not the Python3.7 that we installed. Thus, we need to uninstall all the packages, and install again.

Follow the steps below;
1) Run “**pip list”**

2) Uninstall every package except pip using “**pip uninstall <package_name>”.**
3) Run “**pip list”** now, and only one package should be there, “pip” itself.
4) From the above image, install every package except “SCons” using **“pip install <package_name>”**
5) Once every package (except SCons) has been installed, then only can we install SCons. Run “**pip install scons”**; notice “scons” is in all small letters unlike the actual package name.

To verify if the packages have been installed correctly, just check their path, and if they are somewhere in $HOME, it is correct.

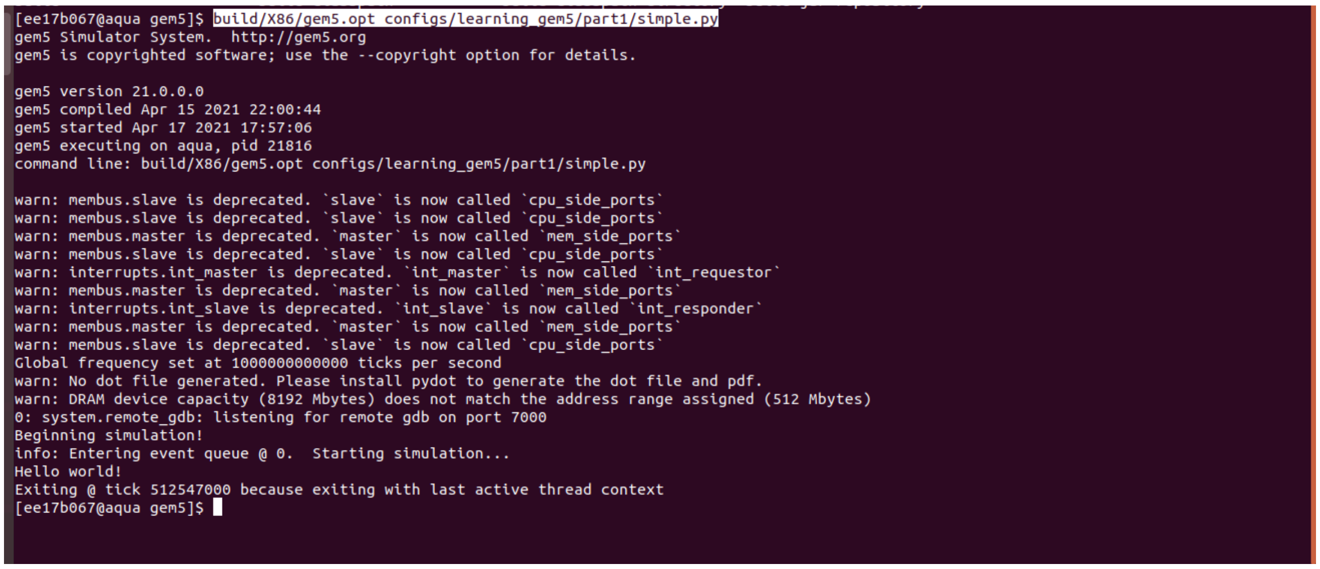
Building GEM5!

Now that we have installed all the required dependencies, we can proceed to build gem5. Do the following steps;

1) cd $HOME/usr
2) git clone https://gem5.googlesource.com/public/gem5
3) Cd gem5
    a) basically $HOME/usr/gem5
4) scons build/X86/gem5.opt -j 16
    a) 16 denotes the number of cores gem5 will build with. I used 16, number doesn’t matter much.
    b) This step will take 20-40mins to run.

GEM5 has now been built on your system. To do a basic check and see if it’s working, run the following command from the $HOME/usr/gem5 folder;
build/X86/gem5.opt configs/learning_gem5/part1/simple.py

You should get the following output;



```
[ee17b067@aqua gem5]$ build/X86/gem5.opt configs/learning_gem5/part1/simple.py
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 21.0.0.0
gem5 compiled Apr 15 2021 22:00:44
gem5 started Apr 17 2021 17:57:06
gem5 executing on aqua, pid 21816
command line: build/X86/gem5.opt configs/learning_gem5/part1/simple.py

warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.master is deprecated. `master` is now called `mem_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: interrupts.int_master is deprecated. `int_master` is now called `int_requestor`
warn: membus.master is deprecated. `master` is now called `mem_side_ports`
warn: interrupts.int_slave is deprecated. `int_slave` is now called `int_responder`
warn: membus.master is deprecated. `master` is now called `mem_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 512547000 because exiting with last active thread context
[ee17b067@aqua gem5]$
```

Congrats!

GEM5 has now been successfully installed in your system!

You can learn to use GEM5 through this link;
https://www.gem5.org/documentation/learning_gem5/gem5_101/