

ENVIRONMENT SETUP

We have already given a VM for each of you which is a Windows 10 machine. Log in to your VM and install a new Ubuntu-20.04 in the VirtualBox we already installed on your machine. Although Ubuntu-20.04 is installed there you will install another one by your own and work on the machine you built inside the VirtualBox.

Here is the download link for Ubuntu-20.04 <https://releases.ubuntu.com/focal/> . Go to the link and select the following one:

Select an image

Ubuntu is distributed on three types of images described below.

Desktop image

The desktop image allows you to try Ubuntu without changing your computer at all, and at your option to install it permanently later. This type of image is what most people will want to use. You will need at least 1024MiB of RAM to install from this image.

64-bit PC (AMD64) desktop image

Choose this if you have a computer based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2). Choose this if you are at all unsure.

Here is the tutorial for installing Ubuntu in VirtualBox <https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview>

Step 1: Build required toolchain

1.1 Base dependencies

The basic build package on Ubuntu is the build-essential package. To install run:

```
$ sudo apt-get update
```

```
$ sudo apt-get install build-essential
```

1.2 Additional base dependencies for building seL4 projects on Ubuntu include installing:

```
$ sudo apt-get install cmake ccache ninja-build cmake-curses-gui
```

```
$ sudo apt-get install libxml2-utils ncurses-dev
```

```
$ sudo apt-get install curl git doxygen device-tree-compiler
```

```
$ sudo apt-get install u-boot-tools
```

1.3 For Ubuntu20.04, install cross-compiling for ARM targets:

```
$ sudo apt-get install python3-dev python3-pip python-is-python3
```

```
$ sudo apt-get install protobuf-compiler python3-protobuf
```

1.4 Simulating with QEMU:

To run seL4 projects on a simulator you need to install QEMU on VM:

```
$ sudo apt-get install qemu-system-arm qemu-system-x86 qemu-system-misc
```

```
zhlliao@Ubuntu2004:~$ sudo apt-get install qemu-system-arm qemu-system-x86 qemu-s
ystem-misc
[sudo] password for zhlliao:
Reading package lists... Done
Building dependency tree
Reading state information... Done
qemu-system-arm is already the newest version (1:4.2-3ubuntu6.24).
qemu-system-misc is already the newest version (1:4.2-3ubuntu6.24).
qemu-system-x86 is already the newest version (1:4.2-3ubuntu6.24).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
```

Build seL4 manual (optional):

If you would like to build the seL4 manual, you will need the following LaTeX packages:

```
$ sudo apt-get install texlive texlive-latex-extra texlive-fonts-extra
```

1.5 Python dependencies:

Python dependencies are required to build seL4, the manual and its proofs. To install you can run:

```
$ pip3 install --user setuptools
```

```
$ pip3 install --user sel4-deps
```

Currently seL4 foundation duplicate dependencies for python2 and python3 as a python3 upgrade is in process

```
$ pip install --user setuptools
```

```
$ pip install --user sel4-deps
```

Step 2: CAMkES Build Dependencies

To build a CAMkES based project on seL4, additional dependencies need to be installed on your host machine. Projects using CAMkES (the seL4 component system) need Haskell and some extra Python libraries in addition to the standard build tools. The following instructions cover the CAMkES build dependencies for Ubuntu.

2.1 Install Google's Repo tool

```
$ sudo apt-get update
```

```
$ sudo apt-get install repo
```

You might see an error while installing the repo. If the above commands do not work, please install it manually as follows:

```
$ mkdir -p ~/.bin
```

```
$ PATH="${HOME}/.bin:${PATH}"
```

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/.bin/repo
```

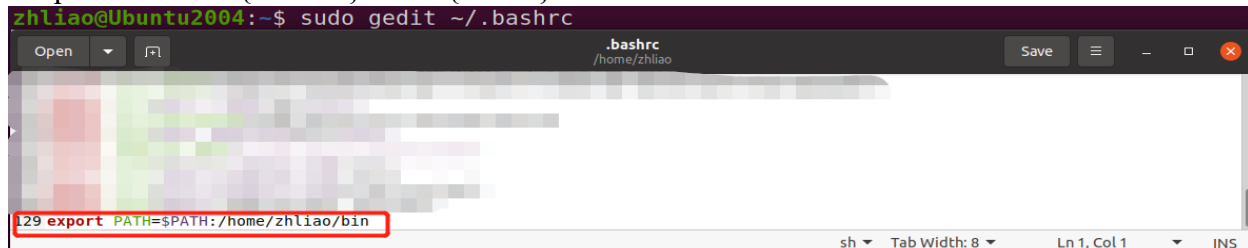
```
$ chmod a+rx ~/.bin/repo
```

#Then add the ~/.bin directory to your PATH variable to make it work:

```
$ nano ~/.bashrc (Please learn how to save and exit in nano)
```

Go to the end of the file and add the following line

```
$export PATH="${HOME}/.bin:${PATH}"
```



```
$ source ~/.bashrc
```

\$repo version

2.2 Install Python dependencies required by the CAMkES build toolchain:

```
# Currently seL4 duplicate dependencies for python2 and python3 as a python3 upgrade is in process
```

```
$ pip install --user camkes-deps
```

```
$ sudo apt-get install haskell-stack
```

```
Activities Terminal Mar 25 19:17  
ryan@ubuntu: ~/Desktop/se4_tutorials_manifest/ipc_build  
$ sudo apt install ninja-build  
Installing collected packages: ninja  
WARNING: The script ninja is installed in '/home/ryan/.local/bin' which is not on PATH.  
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.  
Successfully installed ninja-1.11.1  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest$ ls  
hello-world hello-world.knit ipc_ipc_build ipcvaep8bjh ipcvaep8bjh_build kernel projects README-cankes.md README.nd tools  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest$ cd ipc_build  
bash: cd: ipc_build: no such file or directory  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest$ cd ipc_build  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest/ipc_build$ ninja  
[21/33] Generating capDL-tool/parse-capDL.c  
capDL-tool/parse-capDL.c  
cd /home/ryan/Desktop/se4_tutorials_manifest/ipc_build/capDL-tool && printf capDL-tool/parse-capDL.c > /home/ryan/Desktop/se4_tutorials_manifest/ipc_build/capDL-tool  
cd /home/ryan/Desktop/se4_tutorials_manifest/projects/capdl/capDL-tool -type f -printf %p >> /home/ryan/Desktop/se4_tutorials_manifest/ipc_b  
uild/capDL-tool/parse-capDL.d && cp -a /home/ryan/Desktop/se4_tutorials_manifest/projects/capdl/capDL-tool/*.* /usr/bin/cnake -E env make  
stack build --fast  
Downloading lts-19.12 build plan ...  
RedomloadInvalidResponse Request {  
  host = "raw.githubusercontent.com"  
  port = 443  
  secure = True  
  requestHeaders = {}  
  path = "/fpc/lts-haskell/master//lts-19.12.yaml"  
  queryString = ""  
  method = "GET"  
  proxy = Nothing  
  rawBody = False  
  redirectCount = 10  
  responseTimeout = ResponseTimeoutDefault  
  requestVersion = HTTP/1.1  
}  
/home/ryan/stack/build-plan/lts-19.12.yaml: (Response {responseStatus = Status {statusCode = 404, statusMessage = "Not Found"}, responseVersion = HTTP/1.1, responseH  
eaderFields = [{"Content-Type": "application/javascript"}, {"X-Frame-Options": "deny"}, {"X-XSS-Protection": "1; mode=block"}, {"Content-type": "text/plain; charset=utf  
-8"}, {"X-GitHub-Request-Id": "5050:0B86:CDBF8A:FEB6D9:641F8B01"}, {"Accept-Ranges": "bytes"}, {"Date": "Sun, 26 Mar 2023 00:12:49 GMT"}, {"Via": "1.1 varnish"}, {"X-Served-By  
": "cache-chi-kig898091-zrh"}, {"X-Cache": "MISS"}, {"X-Cache-Hits": "0"}, {"X-Timer": "S1679789569.449664,V50,V85"}, {"Vary": "Authorization,Accept-Encoding,Origin"}, {"Access  
Control-Allow-Origin": ""}, {"X-Fastly-Request-ID": "2788d7ad3ed0617b429d02c58afca4bc07493cb"}, {"Expires": "Sun, 26 Mar 2023 00:17:49 GMT"}, {"Source-Age": "0"}], response  
Body = (), responseCookieJar = C[] (expose = []), responseClose = ResponseClose)  
make: *** [Makefile:51: parse-capDL] Error 1  
[22/33] Linking C static library libse4tutorials/libse4tutorials.a  
ninja: build stopped: subcommand failed.  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest/ipc_build$ pip install capDL-tool  
ERROR: Could not find a version that satisfies the requirement capDL-tool (from versions: none)  
ERROR: No matching distribution found for capDL-tool  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest/ipc_build$ ls  
build.ninja check cnake_install.cnake cspace_server.c input_files lib nushlib se4_libs uttl_libs  
capdl CNakeCache.txt cspace_client.1.c elfloader kernel libse4 output_files  
capDL-tool cspaceClient.c cspace_client.2.c gcc.cnake launch_gdb libse4tutorials rules.ninja s4druntime  
ryan@ubuntulaptop:~/Desktop/se4_tutorials_manifest/ipc_build$
```

For the above figure: It seems that there is an issue with downloading the Haskell build plan for the stack tool. The error message indicates that the stack tool is trying to download a specific version of the build plan file (lts-19.12.yaml), but it cannot find it.

One possible reason for this could be that the build plan file has been moved or deleted from the source repository. You can try updating the stack tool and retrying the build process to see if that resolves the issue: **\$ sudo stack upgrade**

2.4 Install build dependencies:

```
$ sudo apt-get install clang gdb
```

```
$ sudo apt-get install libssl-dev libclang-dev libcunit1-dev libsqlite3-dev
```

```
$ sudo apt-get install qemu-kvm
```

Step 3: Fetching, Configuring and Building seL4test on QEMU

This section presents a case study, by the end of which you can run seL4test on a simulator (QEMU). To build a project, you need to:

- check out the sources using Repo,
- configure a target build using CMake,
- build the project using Ninja.

3.1 Use repo to check sel4test out from GitHub. Its manifest is located in the sel4test-manifest repository:

```
$ mkdir seL4test
```

```
$ cd seL4test
```

```
$ repo init -u https://github.com/seL4/sel4test-manifest.git
```

```
shafiqul@shafiqul-VirtualBox:~/seL4test$ repo init -u https://github.com/seL4/sel4test-manifest.git
Downloading Repo source from https://gerrit.googlesource.com/git-repo
repo: Updating release signing keys to keyset ver 2.3

Traceback (most recent call last):
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 874, in <module>
    _Main(sys.argv[1:])
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 850, in _Main
    result = repo._Run(name, gopts, argv) or 0
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 294, in _Run
    result = run()
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 275, in <lambda>
    lambda: self._RunLong(name, gopts, argv, git_trace2_event_log) or 0
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 442, in _RunLong
    execute_command()
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 408, in execute_command
    execute_command_helper()
  File "/home/shafiqul/seL4test/.repo/repo/main.py", line 374, in execute_command_helper
    result = cmd.Execute(copts, cargs)
  File "/home/shafiqul/seL4test/.repo/repo/subcmds/init.py", line 395, in Execute
    self._ConfigureUser(opt)
  File "/home/shafiqul/seL4test/.repo/repo/subcmds/init.py", line 212, in _ConfigureUser
    name = self._Prompt("Your Name", mp.UserName)
  File "/home/shafiqul/seL4test/.repo/repo/project.py", line 778, in UserName
    self._LoadUserIdentity()
  File "/home/shafiqul/seL4test/.repo/repo/project.py", line 791, in _LoadUserIdentity
    u = self.bare_git.var("GIT_COMMITTER_IDENT")
  File "/home/shafiqul/seL4test/.repo/repo/project.py", line 3803, in runner
    p.Wait()
  File "/home/shafiqul/seL4test/.repo/repo/git_command.py", line 553, in Wait
    self.VerifyCommand()
  File "/home/shafiqul/seL4test/.repo/repo/git_command.py", line 543, in VerifyCommand
    raise GitCommandError(
git_command.GitCommandError: GitCommandError: 'var GIT_COMMITTER_IDENT' on manifests failed
stderr:
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
```

You might see this error while running this command:

```
$repo init -u https://github.com/seL4/sel4test-manifest.git
```

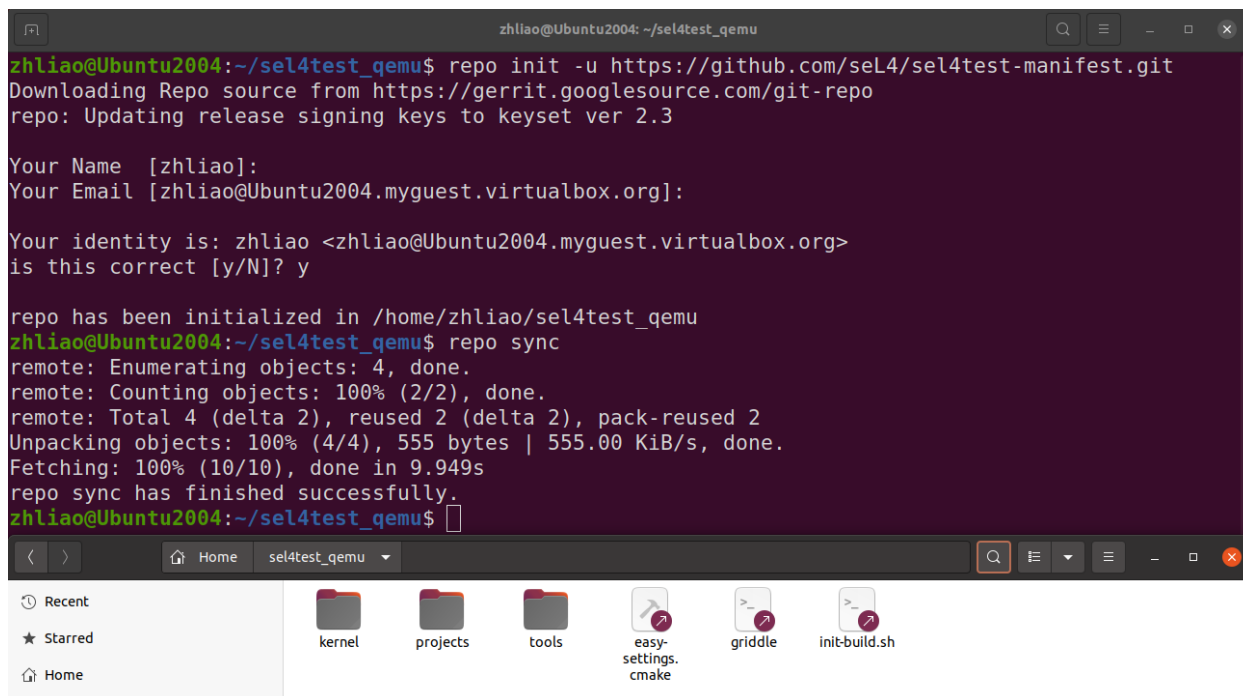
To resolve this issue run the commands with your email and name.

```
Run
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Then again run this command

```
$repo init -u https://github.com/seL4/sel4test-manifest.git
```

```
$ repo sync
```

A screenshot of a terminal window titled 'zhlliao@Ubuntu2004: ~/sel4test_qemu'. The terminal shows the following commands and output:

```
zhlliao@Ubuntu2004:~/sel4test_qemu$ repo init -u https://github.com/seL4/sel4test-manifest.git
Downloading Repo source from https://gerrit.googlesource.com/git-repo
repo: Updating release signing keys to keyset ver 2.3

Your Name [zhlliao]:
Your Email [zhlliao@Ubuntu2004.myguest.virtualbox.org]:

Your identity is: zhlliao <zhlliao@Ubuntu2004.myguest.virtualbox.org>
is this correct [y/N]? y

repo has been initialized in /home/zhlliao/sel4test_qemu
zhlliao@Ubuntu2004:~/sel4test_qemu$ repo sync
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 2 (delta 2), pack-reused 2
Unpacking objects: 100% (4/4), 555 bytes | 555.00 KiB/s, done.
Fetching: 100% (10/10), done in 9.949s
repo sync has finished successfully.
zhlliao@Ubuntu2004:~/sel4test_qemu$
```

The terminal window is overlaid on a desktop environment. The desktop has a dark theme and a sidebar on the left with 'Recent', 'Starred', and 'Home' sections. The main area shows several icons: 'kernel', 'projects', 'tools', 'easy-settings.cmake', 'griddle', and 'init-build.sh'. The terminal window's title bar includes standard Ubuntu window controls (minimize, maximize, close) and a search icon.

3.2 Configure an x86_64 build directory, with a simulation target to be run by QEMU. QEMU is a generic and open-source machine emulator and virtualizer, and can emulate different architectures on different systems.

```
$ mkdir build-x86
```

```
$ cd build-x86
```

```
$ ../init-build.sh -DPLATFORM=x86_64 -DSIMULATION=TRUE
```

```
$ ninja
```



```

zhlliao@Ubuntu2004: ~/sel4test_qemu/build-x86
zhlliao@Ubuntu2004:~/sel4test_qemu$ ls
easy-settings.cmake griddle init-build.sh kernel projects tools
zhlliao@Ubuntu2004:~/sel4test_qemu$ mkdir build-x86
zhlliao@Ubuntu2004:~/sel4test_qemu$ cd build-x86/
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ ls -ll
total 0
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ ../init-build.sh -DPLATFORM=x86_64 -DSIMULATION=TRUE
loading initial cache file /home/zhlliao/sel4test_qemu/projects/sel4test/settings.cmake
-- Set platform details from PLATFORM=x86_64
-- KernelPlatform: pc99
-- KernelSel4Arch: x86_64
-- Found sel4: /home/zhlliao/sel4test_qemu/kernel
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- The ASM compiler identification is GNU
-- Found assembler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/g++
-- Check for working CXX compiler: /usr/bin/g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found elfloader-tool: /home/zhlliao/sel4test_qemu/tools/sel4/elfloader-tool
-- Found musllibc: /home/zhlliao/sel4test_qemu/projects/musllibc
-- Found util_libs: /home/zhlliao/sel4test_qemu/projects/util_libs
-- Found sel4_libs: /home/zhlliao/sel4test_qemu/projects/sel4_libs
-- Found sel4_projects_libs: /home/zhlliao/sel4test_qemu/projects/sel4_projects_libs
-- Found sel4runtime: /home/zhlliao/sel4test_qemu/projects/sel4runtime
-- Performing Test compiler_arch_test
-- Performing Test compiler_arch_test - Success
-- libmusl architecture: 'x86_64' (from KernelSel4Arch 'x86_64')
-- Detecting cached version of: musllibc
-- Found Git: /usr/bin/git (found version "2.25.1")
-- Not found cache entry for musllibc - will build from source
-- Found Nanopb: /home/zhlliao/sel4test_qemu/tools/nanopb
-- CPIO test cpio_reproducible_flag PASSED
-- Configuring done
-- Generating done
-- Build files have been written to: /home/zhlliao/sel4test_qemu/build-x86
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$

```

```

zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ ninja
[263/263] objcopy kernel into bootable elf
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ ls
apps          CMakeFiles      gcc.cmake       launch_gdb      nanopb
build.ninja    cmake_install.cmake  images          lib              rules.ninja
CMakeCache.txt  elfloader        kernel          libsel4          simulate
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$

```

3.3 The build images are available in build-x86/images, and a script build-x86/simulate that will run Qemu with the correct arguments to run sel4test.

```

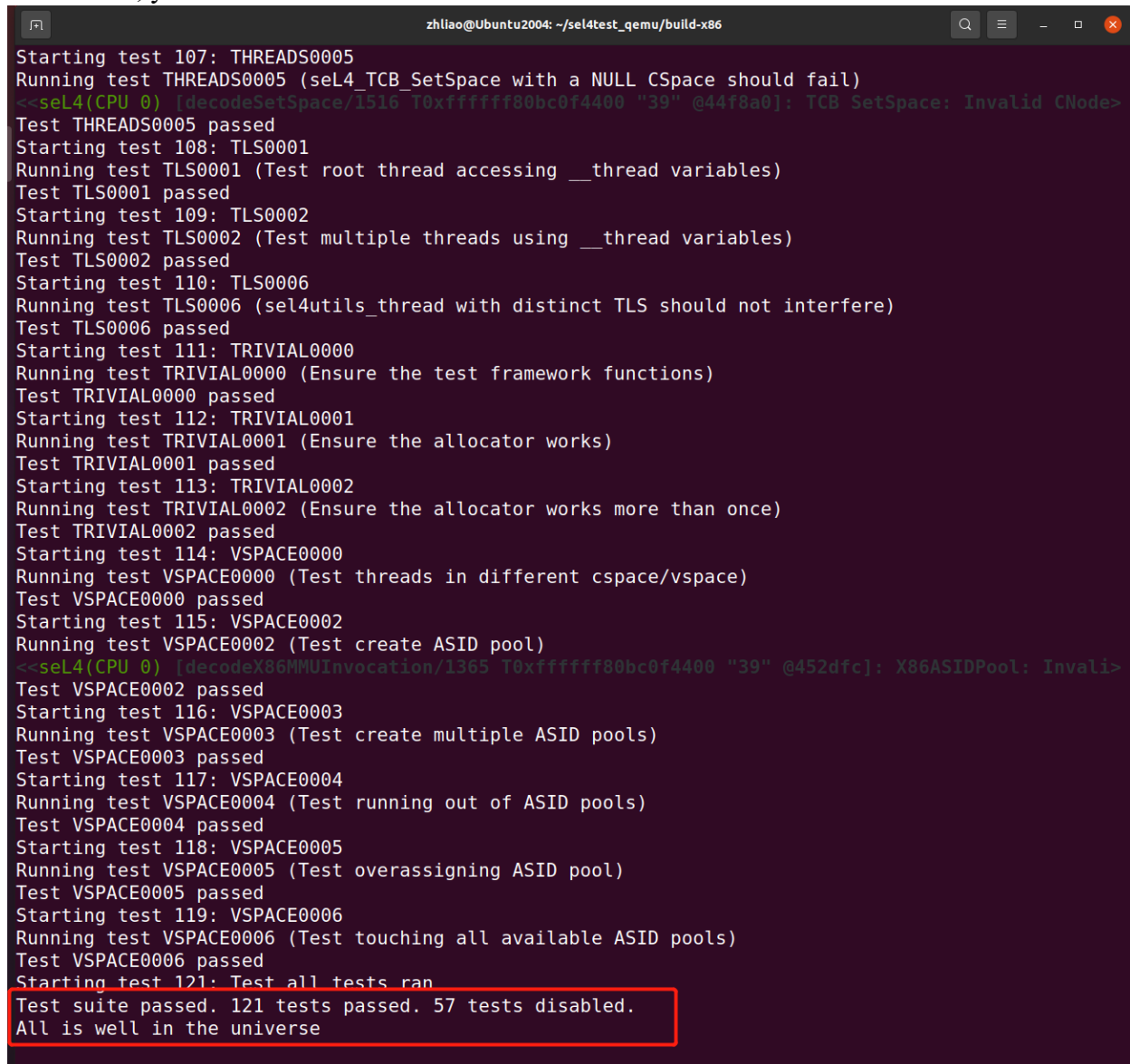
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ ls
apps          CMakeFiles      gcc.cmake       launch_gdb      nanopb
build.ninja    cmake_install.cmake  images          lib              rules.ninja
CMakeCache.txt  elfloader        kernel          libsel4          simulate
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$ cd images/
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86/images$ ls
kernel-x86_64-pc99  sel4test-driver-image-x86_64-pc99
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86/images$ cd ..
zhlliao@Ubuntu2004:~/sel4test_qemu/build-x86$

```

Run simulate as follows:

\$./simulate

On success, you should see:



```
zhllao@Ubuntu2004: ~/sel4test_qemu/build-x86
Starting test 107: THREADS0005
Running test THREADS0005 (seL4_TCB_SetSpace with a NULL CSpace should fail)
<<seL4(CPU 0) [decodeSetSpace/1516 T0xffffffff80bc0f4400 "39" @44f8a0]: TCB SetSpace: Invalid CNode>
Test THREADS0005 passed
Starting test 108: TLS0001
Running test TLS0001 (Test root thread accessing __thread variables)
Test TLS0001 passed
Starting test 109: TLS0002
Running test TLS0002 (Test multiple threads using __thread variables)
Test TLS0002 passed
Starting test 110: TLS0006
Running test TLS0006 (sel4utils_thread with distinct TLS should not interfere)
Test TLS0006 passed
Starting test 111: TRIVIAL0000
Running test TRIVIAL0000 (Ensure the test framework functions)
Test TRIVIAL0000 passed
Starting test 112: TRIVIAL0001
Running test TRIVIAL0001 (Ensure the allocator works)
Test TRIVIAL0001 passed
Starting test 113: TRIVIAL0002
Running test TRIVIAL0002 (Ensure the allocator works more than once)
Test TRIVIAL0002 passed
Starting test 114: VSPACE0000
Running test VSPACE0000 (Test threads in different cspace/vspace)
Test VSPACE0000 passed
Starting test 115: VSPACE0002
Running test VSPACE0002 (Test create ASID pool)
<<seL4(CPU 0) [decodeX86MMUInvocation/1365 T0xffffffff80bc0f4400 "39" @452dfc]: X86ASIDPool: Invalid>
Test VSPACE0002 passed
Starting test 116: VSPACE0003
Running test VSPACE0003 (Test create multiple ASID pools)
Test VSPACE0003 passed
Starting test 117: VSPACE0004
Running test VSPACE0004 (Test running out of ASID pools)
Test VSPACE0004 passed
Starting test 118: VSPACE0005
Running test VSPACE0005 (Test overassigning ASID pool)
Test VSPACE0005 passed
Starting test 119: VSPACE0006
Running test VSPACE0006 (Test touching all available ASID pools)
Test VSPACE0006 passed
Starting test 121: Test all tests ran
Test suite passed. 121 tests passed. 57 tests disabled.
All is well in the universe
```

Then, press “Ctrl” + “A” and then press “X” in your terminal, you will exit QEMU.

Congratulations! You have successfully ported seL4 to QEMU! Next, you can develop applications on it and test them.

References:

1. seL4 Building for the BeagleBone Black <https://docs.sel4.systems/Hardware/Beaglebone.html>
2. seL4 Docs: Getting Started <https://docs.sel4.systems/GettingStarted#getting-cross-compilers>
3. seL4 Docs: Host Dependencies <https://docs.sel4.systems/projects/buildsystem/host-dependencies.html>
4. seL4 Docs: seL4Test <https://docs.sel4.systems/projects/sel4test/>
5. gtkterm <https://postimg.cc/VSHvWJQj>
6. seL4 and RefOS on BBB <https://mokshasoft.com/seL4/seL4-beaglebone-black.html>
7. Booting sequence of Beaglebone Black hardware <https://fastbitlab.com/linux-device-driver-programming-lecture-6-booting-sequence-beaglebone-black-hardware/>
8. <http://qdosmsq.dunbar-it.co.uk/blog/2015/09/getting-arduino-working-from-a-windows-7-virtualbox-guest/>
9. Cp2102 Usb-to-Serial Driver Installation https://exploreembedded.com/wiki/Cp2102_Usb-to-Serial_Driver_Installation
10. Linux Device Driver Programming Lecture 6- Booting sequence of Beaglebone Black hardware <https://fastbitlab.com/linux-device-driver-programming-lecture-6-booting-sequence-beaglebone-black-hardware/>
11. Linux device driver lecture 3 : Beaglebone black boot sequence <https://www.youtube.com/watch?v=uJWWzdsL4tA>
12. Minicom no cmd on start up <https://stackoverflow.com/questions/16672829/minicom-no-cmd-on-start-up>
13. fatload command <https://u-boot.readthedocs.io/en/v2022.04/usage/cmd/fatload.html>
14. Linux / UNIX minicom Serial Communication Program <https://www.cyberciti.biz/tips/connect-soekris-single-board-computer-using-minicom.html>