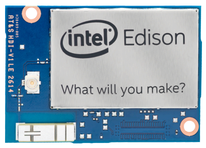


IoT Platform: Tutorial 2: Linux Platform

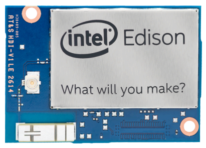


Table of Contents

Revision history		
Version	**Date**	**Comment**
1.0	21/9/2015	Initial release

Introduction

In this tutorial, you will:

1. Be introduced to the Linux image running the Intel Edison,

2. Learn to connect to the Edison via serial connection,

3. Be introduced to the Linux command line, and

4. Learn about the software development tools on the Linux image.

Things Needed

• An Intel Edison with Arduino-compatible breakout,

• Two micro USB cables, and

• a PC or Mac

Yocto Embedded Linux

Figure 1 Yocto From www.yoctoproject.org

After updating the firmware, the Edison runs the latest official Yocto Embedded Linux image, which is a light-weight yet fully functional Linux image that is packed with tools for IoT development. This custom Linux image is created with Yocto project, which is an open source project that provides templates, tools, metadata, and documentation to assist development of custom Linux-based systems for any embedded systems [1].

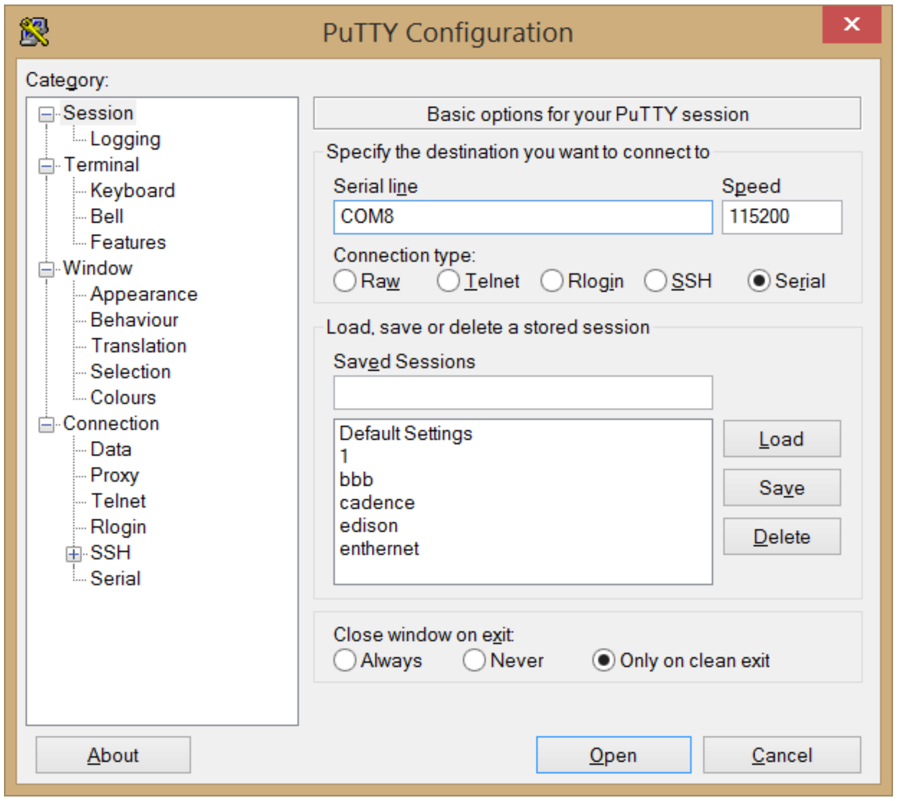
Serial Connection to the Intel Edison

Let's explore the Yocto Embedded Linux. First, we need to connect to the Linux side of the Edison. This can be achieved with a Universal Asynchronous Receiver/Transmitter (UART) serial connection. A UART is a piece of hardware that is usually used for serial communications over serial ports. The Edison with Arduino breakout board comes with a UART serial USB port. Now, let's make a serial connection to the Edison by following the steps below.

1. Toggle the switch to enable "device mode" (lower position) and connect a micro USB cable to the multi-gadget USB port and to your computer.
2. Connect another micro USB cable to the UART serial USB port (the bottom micro USB port) and to your computer.
3. Follow the steps below for your computer's OS.

Windows

4. Open a SSH client such as PuTTY (Install if you haven't already). Download PuTTY at http://www.putty.org.
5. Select "**Session**" from the category on the left.



Specify the destination you want to connect to
Serial line: COM8
Speed: 115200
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Load, save or delete a stored session
Saved Sessions:
Default Settings
1
bbb
cadence
edison
enthernet
Close window on exit: Always Never [✓ Only on clean exit]

Figure 2 PuTTY Configuration

6. Enter COM# in the Serial line field. You can find the COM# from Windows Device Manager (In this example, COM8). The Edison should be listed as USB Serial Port.

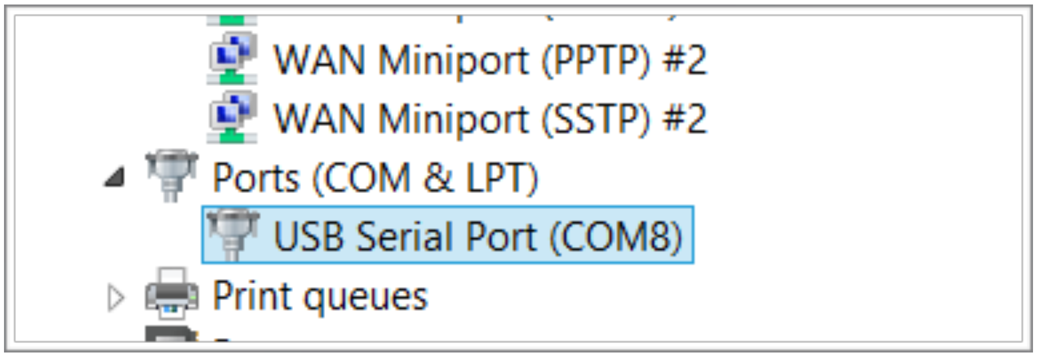

WAN Miniport (PPTP) #2
WAN Miniport (SSTP) #2
Ports (COM & LPT)
USB Serial Port (COM8)
Print queues

Figure 3 COM Port

7. Enter 115200 in the Speed field.

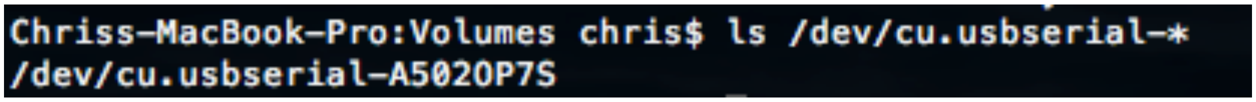
8. Make sure Serial is selected in the Connection type field.

9. Click Open then a blank screen will appear.

10. Proceed to step 11 below.

Mac

4. Open Terminal.
5. Enter “ls /dev/cu.usbserial-*” to list connected devices.



```
Chriss-MacBook-Pro:Volumes chris$ ls /dev/cu.usbserial-*
/dev/cu.usbserial-A5020P7S
```

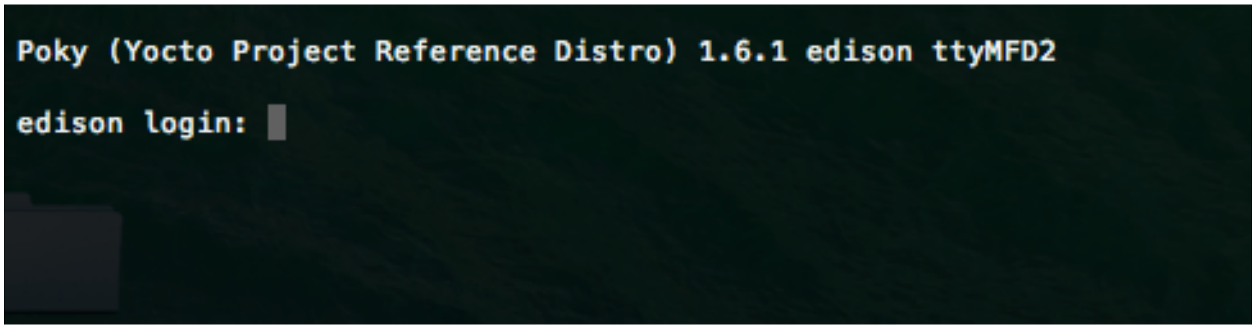
Figure 4 Command Line

6. Then, unplug the micro USB cable from the UART serial USB port and re-enter “ls /dev/cu.usbserial-*”. The disappeared device in the list is the Edison.
7. Replug the micro USB cable.
8. Enter “screen /dev/cu.usbserial-XXXXXXXX 115200 -L”. Replace XXXXXXXX with what you found in the previous steps (in this example, A502OP7S).
9. A blank screen will appear.
10. Proceed to step 11 below.

Linux

Refer to https://software.intel.com/en-us/setting-up-serial-terminal-on-system-with-linux.

11. At the blank screen, press the Enter key.
12. The login screen is displayed. If not, press the Enter key again. If you are having a problem with this, unplug the USB cables and repeat the steps above.

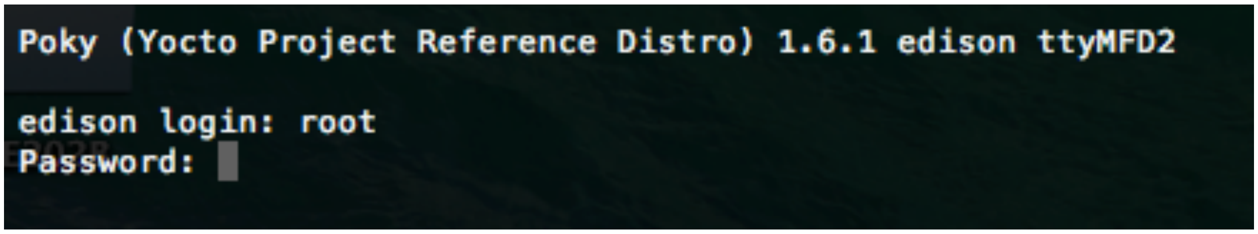


```
Poky (Yocto Project Reference Distro) 1.6.1 edison ttyMFD2

edison login:
```

Figure 5 Login

13. Type "root" and press the Enter key.
14. It will ask for the password, which is not set up at default. Just press the Enter key.



```
Poky (Yocto Project Reference Distro) 1.6.1 edison ttyMFD2

edison login: root
Password:
```

Figure 6 User Name and Password

15. You are now logged into the Linux side of your Edison board!!

Linux Command Line

As soon as you log into the Edison, you see a command line interface (CLI) rather than a graphic user interface (GUI) most people are more familiar with. GUIs are very convenient, but CLIs have their own advantages. For instance, let's imagine that you are copying all of the current directory's files whose names start with "edison" and pasting into a directory. In most GUIs, you need to search for the files that contain "edison" in their names. Then, you need to select the files and copy. Then, you need to select the directory and paste. In the Linux command line, this can be done in a single line of command, "cp ./edison* ../directory".

The Yocto Embedded Linux image includes Bash (Bourne Again SHell), which is a command shell by GNU Project [3]. A shell is a command-line interpreter that provides a user interface to send commands to the operating system. This section covers some shell commands, which are used frequently throughout this tutorial sequence. The steps below demonstrate how to use the commonly used shell commands.

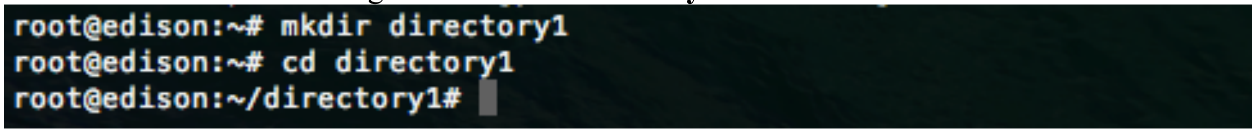
Note: The dollar sign '$' means it is a shell prompt and it is not a part of the command. For instance, enter only "ls" when you see "$ ls".

1. $ **mkdir directory1**

   mkdir command makes a directory. The usage of this command is "**mkdir [option] directory_name**". You can ignore the **[option]** for now. directory_name is the directory name of your choice. In this example, this creates a directory named "**directory1**".

2. $ **cd directory1**

   cd command changes the current location to the specified directory. In this example, this lets you navigate to directory1. In the figure below, you see that the location changed from ~ to **~/directory1**.

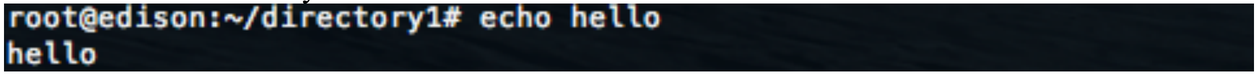


```
root@edison:~# mkdir directory1
root@edison:~# cd directory1
root@edison:~/directory1#
```

Figure 7 Command Line

3. **$ echo hello**

   echo command prints a string to the standard output, which is displayed on the screen by default.



```
root@edison:~/directory1# echo hello
hello
```

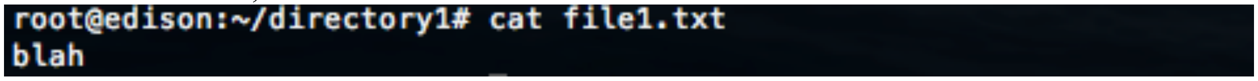
Figure 8 Command Line

4. **$ echo blah > file1.txt**

   The standard output can be redirected. In this case, > indicates that the standard output of the echo command is redirected to a text file named file1.txt.

5. **$ cat file1.txt**

   cat command reads the content of a file and prints it to the standard output. In this case, it reads the content of file1.txt.



```
root@edison:~/directory1# cat file1.txt
blah
```

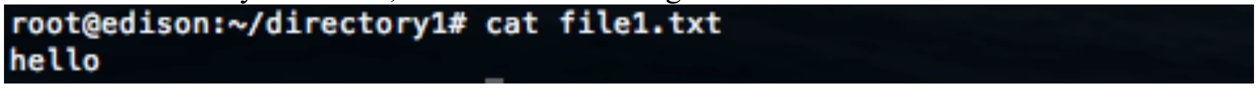
Figure 9 Command Line

6. **$ echo hello > file1.txt**

   Redirection with > overwrites the previous content.

7. **$ cat file1.txt**

   As you can see, the content has changed from "blah" to "hello".



```
root@edison:~/directory1# cat file1.txt
hello
```

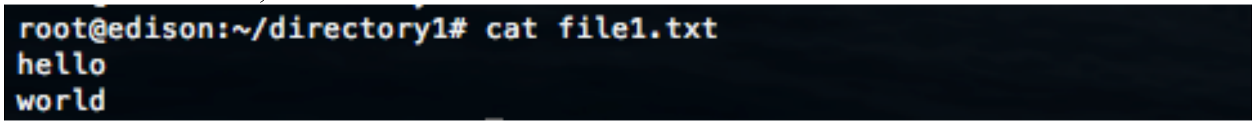
Figure 10 Command Line

8. **$ echo world >> file1.txt**

   >> redirects the standard output and append to a file.

9. **$ cat file1.txt**

Now, the content of the file includes both “hello” and “world”.



```
root@edison:~/directory1# cat file1.txt
hello
world
```

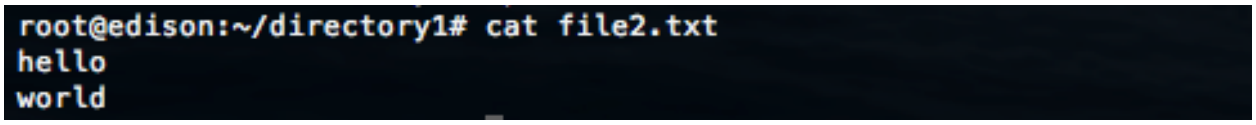
Figure 11 Command Line

10. $ cp file1.txt file2.txt

cp command copies files and directories. In this case, file1.txt in the current directory is copied as file2.txt in the current directory.

11. $ cat file2.txt

You can see that the content of file2.txt is the same as that of file1.txt.



```
root@edison:~/directory1# cat file2.txt
hello
world
```

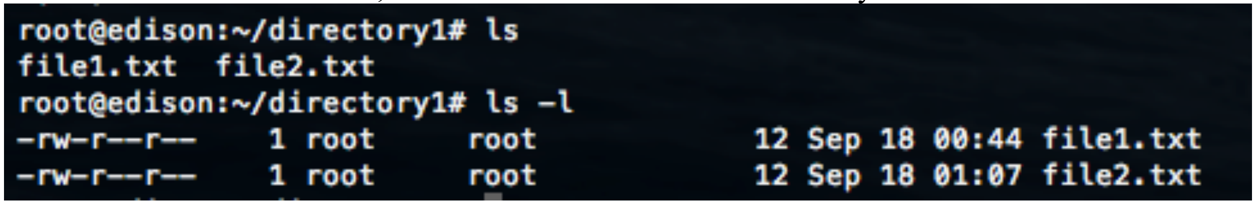
Figure 12 Command Line

12. $ ls

ls command lists the directory contents. In this case, it lists the current directory’s contents, which are file1.txt and file2.txt.

13. $ ls –l

“-l” is an option for ls command. With this option, ls command uses a long listing format. The displayed output includes the permission, the owner, the size, the modified time, and the name of each file or directory.



```
root@edison:~/directory1# ls
file1.txt  file2.txt
root@edison:~/directory1# ls -l
-rw-r--r--    1 root     root              12 Sep 18 00:44 file1.txt
-rw-r--r--    1 root     root              12 Sep 18 01:07 file2.txt
```

Figure 13 Command Line

14. Press the up-arrow key once.

The previous command entered, “**ls -l**”, appears.

15. Press the up-arrow key twice.

“**cat file2.txt**” command appears.

16. Press the down-arrow key once.

“ls” command appears. As you saw, you can press on the up-arrow key and the down- arrow key to search through the command history.

17. Type up to “cat file1” and press the Tab key.

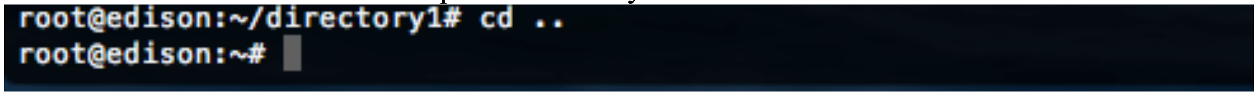
You can see that the command automatically changed to “cat file1.txt”. The TAB key is used for auto-complete.

18. Type up to “cat file” and press the Tab key twice.

This displays file1.txt and file2.txt. Pressing the Tab key twice will list the all possible completions that start with "file".

19. **$ cd ..**

    This command navigate to the parent directory. "." means the current directory and ".." means the parent directory.



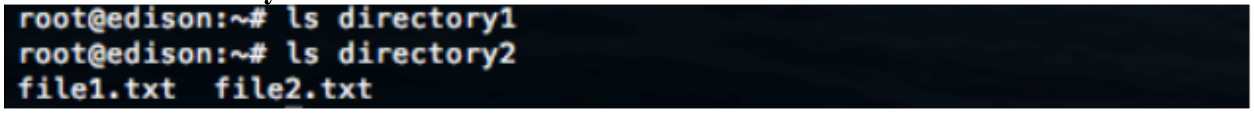
```
root@edison:~/directory1# cd ..
root@edison:~#
```

Figure 14 Command Line

20. **$ mkdir directory2**
21. **$ mv directory1/* directory2**

    mv command moves files to the specified destination. * is a wildcard character that represents zero or more characters. Thus, directory1/* means everything in ./**directory1**/. Another example is ./directory2/file*. This means everything whose name starts with "file" in ./directory2/.
22. **$ ls directory1**

    This command lists the contents of the directory directory1. As you can see, nothing is listed because the contents of this directory were moved to directory2.!
23. **$ ls directory2**



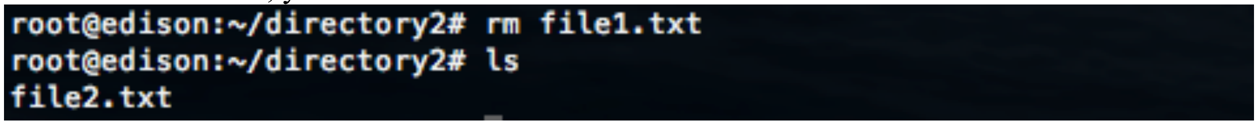
```
root@edison:~# ls directory1
root@edison:~# ls directory2
file1.txt  file2.txt
```

Figure 15 Command Line

24. **$ cd directory2**
25. **$ rm file1.txt**

    rm command removes files and directories.
26. **$ ls**

    Now, you see that file1.txt is removed.

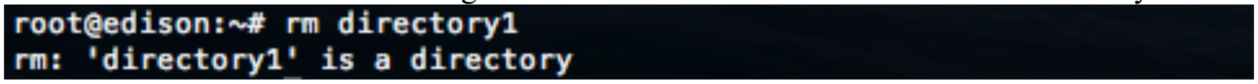


```
root@edison:~/directory2# rm file1.txt
root@edison:~/directory2# ls
file2.txt
```

Figure 16 Command Line

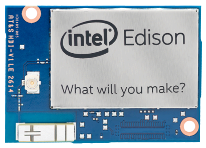
27. **$ cd ..**
28. **$ rm directory1**

    This command will give an error. Directories cannot be removed this way.



```
root@edison:~# rm directory1
rm: 'directory1' is a directory
```

Figure 17 Command Line



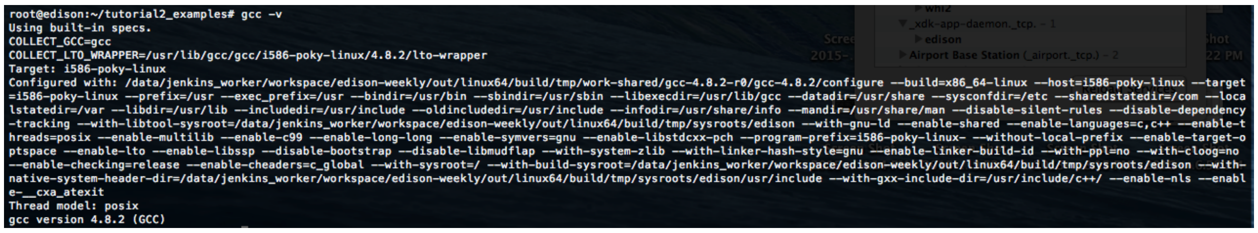
29. **$ rm -r directory1**
    With "-r" option, we can remove the contents of the directory recursively.
30. **$ rm -r directory2**
31. **$ reboot**
    reboot command literally reboots the system. Please keep all cables connected while rebooting. You will need to log in again once the reboot is complete.
32. **$ shutdown -h now**
    This command is used to shutdown the system. Once the system completely turns off, you will need to unplug the micro USB cable from the multi-gadget USB port and replug in order to turn on the system.

Software Development Tools

The Linux image has many preinstalled tools such as GCC, G++, Vim, Python, Node.js, etc. In this tutorial sequence, the development is done mostly in C programming language. In this section, we will write a simple "Hello World" program in C. This example will demonstrate not only how to write the C code but also how to use Vim and GCC.

You can write programs in C and compile with GCC (GNU Compiler Collection). Try the following command to check the version of GCC installed on the Linux image.

$ gcc –v

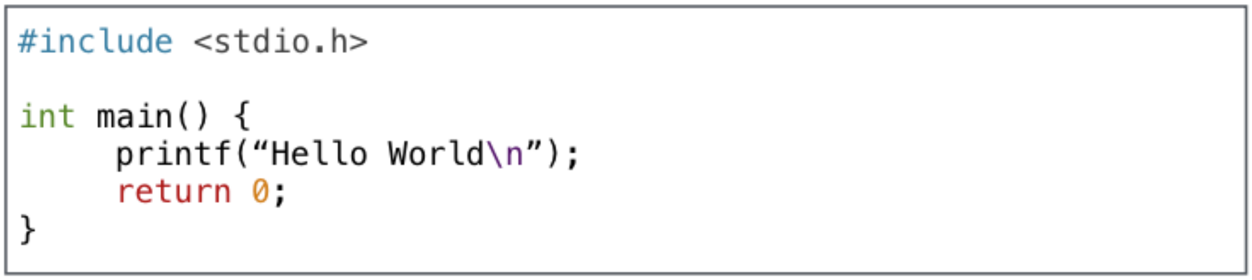


```
root@edison:~/tutorial2_examples# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/gcc/i586-poky-linux/4.8.2/lto-wrapper
Target: i586-poky-linux
Configured with: /data/jenkins_worker/workspace/edison-weekly/out/linux64/build/tmp/work-shared/gcc-4.8.2-r0/gcc-4.8.2/configure --build=x86_64-linux --host=i586-poky-linux --target
=i586-poky-linux --prefix=/usr --exec_prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --libexecdir=/usr/lib/gcc --datadir=/usr/share --sysconfdir=/etc --sharedstatedir=/com --loca
lstatedir=/var --libdir=/usr/lib --includedir=/usr/include --oldincludedir=/usr/include --infodir=/usr/share/info --mandir=/usr/share/man --disable-silent-rules --disable-dependency
-tracking --with-libtool-sysroot=/data/jenkins_worker/workspace/edison-weekly/out/linux64/build/tmp/sysroots/edison --with-gnu-ld --enable-shared --enable-languages=c,c++ --enable-t
hreads=posix --enable-multilib --enable-c99 --enable-long-long --enable-symvers=gnu --enable-libstdcxx-pch --program-prefix=i586-poky-linux- --without-local-prefix --enable-target-o
ptspace --enable-lto --enable-libssp --disable-bootstrap --disable-libmudflap --with-system-zlib --with-linker-hash-style=gnu --enable-linker-build-id --with-ppl=no --with-cloog=no
--enable-checking=release --enable-cheaders=c_global --with-sysroot=/ --with-build-sysroot=/data/jenkins_worker/workspace/edison-weekly/out/linux64/build/tmp/sysroots/edison --with-
native-system-header-dir=/data/jenkins_worker/workspace/edison-weekly/out/linux64/build/tmp/sysroots/edison/usr/include --with-gxx-include-dir=/usr/include/c++/ --enable-nls --enabl
e-__cxa_atexit
Thread model: posix
gcc version 4.8.2 (GCC)
```

Figure 18 GCC

Now, follow the steps below to write and compile your first program on the Edison.

1. Enter "**mkdir tutorial2_examples**" to make a new directory.
2. Enter "**cd tutorial2_examples**" to navigate to the directory created.
3. **$ vi hello_world.c**
   This creates a file named hello_world.c if it does not exist already and then opens it in Vim editor, which is a commonly used text editor in Linux.
4. Press "**i**" to enable insert mode. Then, type the following code.

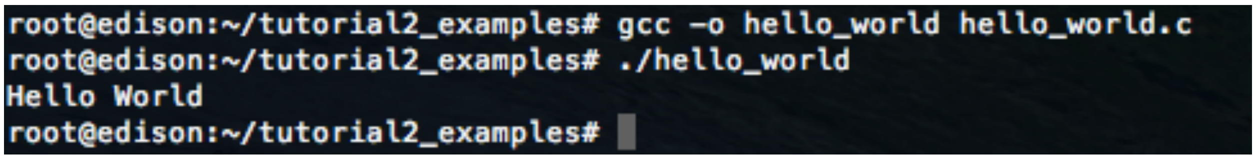


```
#include <stdio.h>

int main() {
     printf(“Hello World\n”);
     return 0;
}
```

Figure 19 Hello World

5. Once you finished writing the code, press the ESC key and enter “**:wq**”.
   By pressing the ESC key, you can go back to the normal mode. “**:wq**” means save and quit. Alternatively, you can enter “**:w**” to save and then enter “**:q**” to quit.! If you are not familiar with Vim editor, you can find documents at http:// vimdoc.sourceforge.net.
6. **$ gcc hello_world.c**
   This command compiles the C code in **hello_world.c** and generate a program called **a.out.**
7. **$ ./a.out**
   In order to run the program, we need to specify the path to the program. Since the program is in the current directory, the path is specified as “**./**”. The program outputs “**Hello World**” to the screen.
8. **$ gcc -o hello_world hello_world.c**
   You can specify the name of the program with “**-o**” option. This command compiles the C code and generates a program named “**hello_world**”. What follows “**-o**” is the specified program name, so “**gcc hello_world.c -o hello_world**” is equivalent. However, “**gcc -o filename sourcefile.c**” is the conventional way.
9. **$ ./hello_world**



```
root@edison:~/tutorial2_examples# gcc -o hello_world hello_world.c
root@edison:~/tutorial2_examples# ./hello_world
Hello World
root@edison:~/tutorial2_examples# 
```

Figure 20 Command Line

References

1. https://www.yoctoproject.org/about
2. https://software.intel.com/en-us/setting-up-serial-terminal-intel-edison-board
3. https://www.gnu.org/software/bash/
4. http://linuxcommand.org/lts0010.php