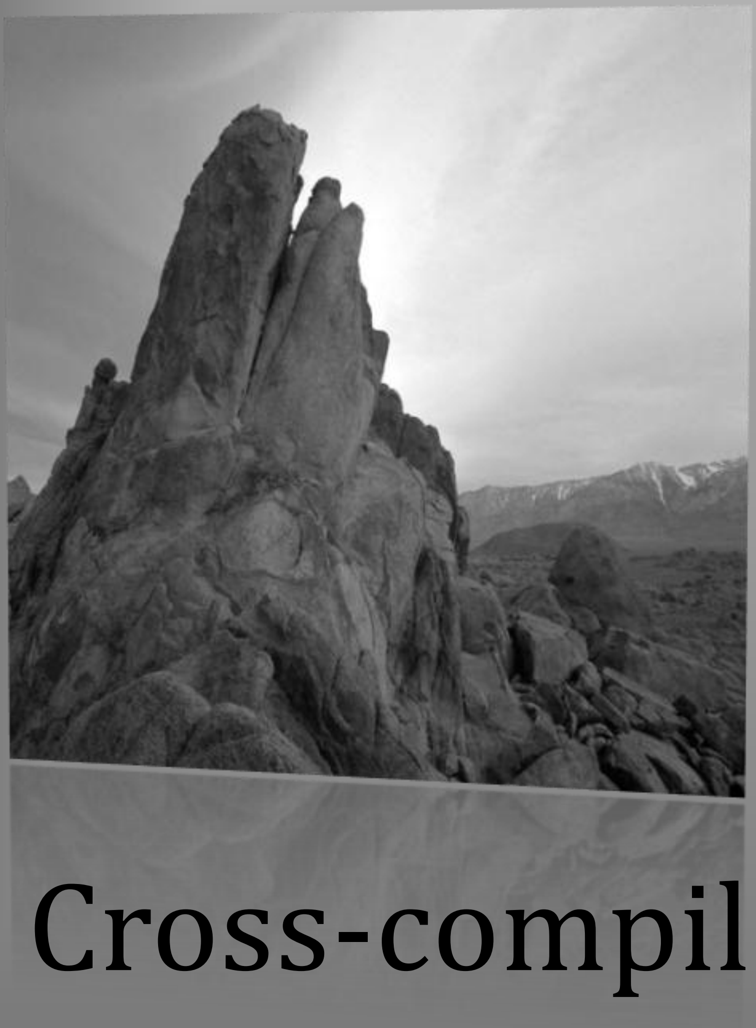


Cross-compilation on Beagle Board-xM

©Neha Girme

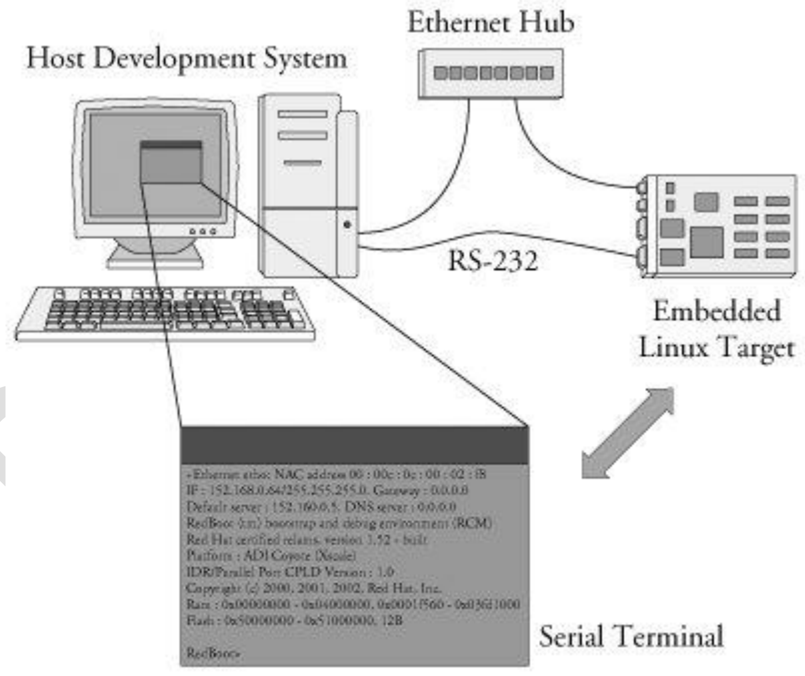
For more details visit http://nehagirme.wordpress.com

CROSS-COMPILATION ON BEAGLE BOARD-xM

System Requirements:

- Host Machine (x86 machine)
- Beagle Board-xM
- Serial Cable
- Ethernet Cable
- 5V Power Supply Cable
- SD card

DEVELOPMENT SETUP

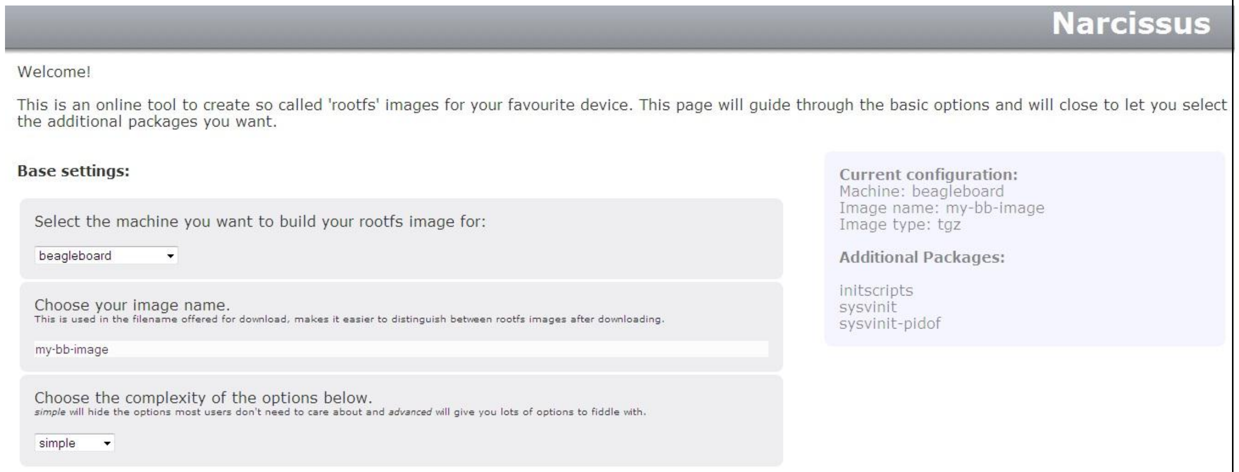


```
+Ethernet eth0: MAC address 00 : 00c : 0c : 00 : 02 : fB
IP : 152.168.0.64/255.255.255.0, Gateway : 0.0.0.0
Default server : 152.160.0.5, DNS server : 0.0.0.0
RedBoot (tm) bootstrap and debug environment (RCM)
Red Hat certified release, version 1.52 - built
Platform : ADI Coyote (Xscale)
IDR/Parallel Port CPLD Version : 1.0
Copyright (c) 2000, 2001, 2002, Red Hat, Inc.
Ram : 0x00000000 - 0x04000000, 0x0001f560 - 0x03fd1000
Flash : 0x50000000 - 0x51000000, 128

RedBoot>
```

Building Angstrom image on SD card:

1. Go to www.narcissus.com website
2. Build an angstrom image for Beagle board according to your requirement and click build me!



Narcissus

Welcome!

This is an online tool to create so called 'rootfs' images for your favourite device. This page will guide through the basic options and will close to let you select the additional packages you want.

Base settings:

Select the machine you want to build your rootfs image for:

beagleboard

Choose your image name.
This is used in the filename offered for download, makes it easier to distinguish between rootfs images after downloading.

my-bb-image

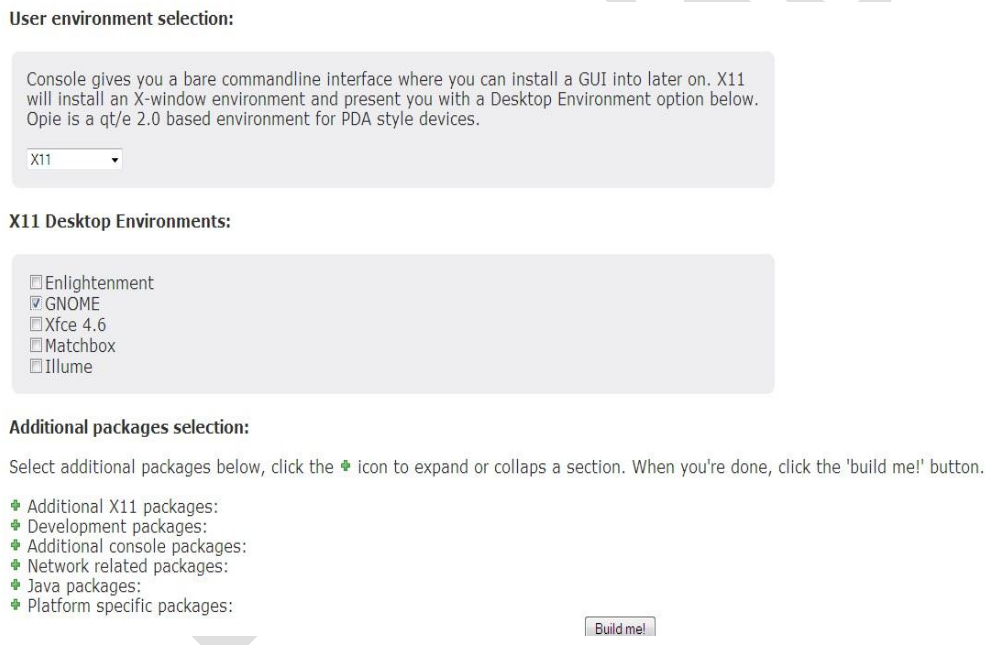
Choose the complexity of the options below.
simple will hide the options most users don't need to care about and *advanced* will give you lots of options to fiddle with.

simple

Current configuration:
Machine: beagleboard
Image name: my-bb-image
Image type: tgz

Additional Packages:

initscripts
sysvinit
sysvinit-pidof



User environment selection:

Console gives you a bare commandline interface where you can install a GUI into later on. X11 will install an X-window environment and present you with a Desktop Environment option below. Opie is a qt/e 2.0 based environment for PDA style devices.

X11

X11 Desktop Environments:

- ☐ Enlightenment
- ☑ GNOME
- ☐ Xfce 4.6
- ☐ Matchbox
- ☐ Illume

Additional packages selection:

Select additional packages below, click the ✚ icon to expand or collaps a section. When you're done, click the 'build me!' button.

- ✚ Additional X11 packages:
- ✚ Development packages:
- ✚ Additional console packages:
- ✚ Network related packages:
- ✚ Java packages:
- ✚ Platform specific packages:

3. Wait for a while (this will take some time)!
4. You will get your build image in the form of tar file

➢ This image consists of :
   - Angstrom-BB-image.tar.bz2
   - MLO
   - README.txt

- U-Boot.bin
- uImage

Prepare the partitions for the SD card

1. FAT partition: this hosts the bootloaders and kernel image
2. Remaining space for ext3 partition or extended file system
3. Insert sd card into your host platform using *fdisk* utility
4. This will create a small bootable FAT partition and then a large ext3 partition
5. For more detailed steps check code.google.com/p/beagleboard/wiki/LinuxBootDiskFormat

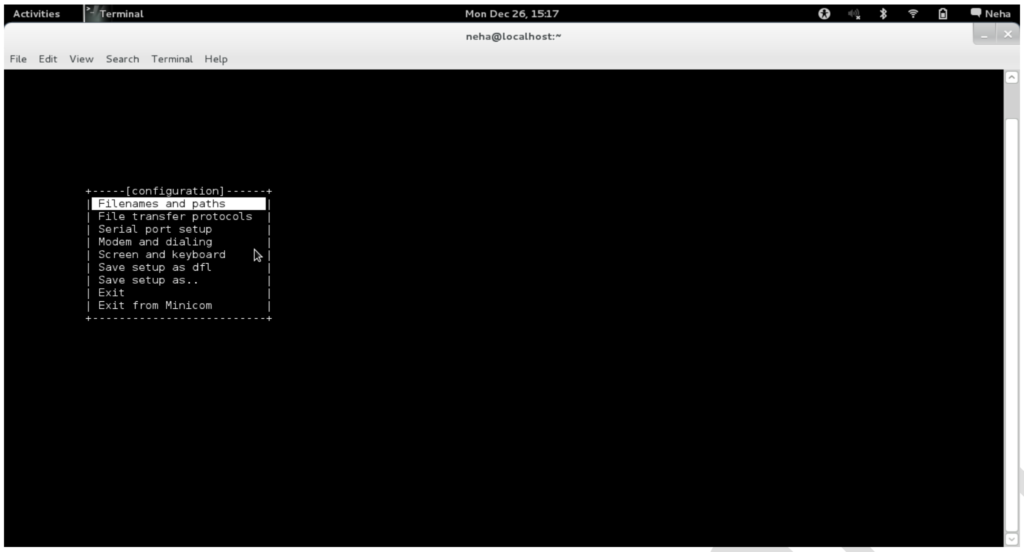
Boot Angstrom on Beagle Board

1. Insert SD card in Beagle Board
2. For serial communication setup,minicom utility is used
3. To install minicom for the first time you can issue

```
neha@localhost: sudo apt-get install minicom
```

4. Issue minicom -s in Host terminal

```
neha@localhost: minicom -s
```



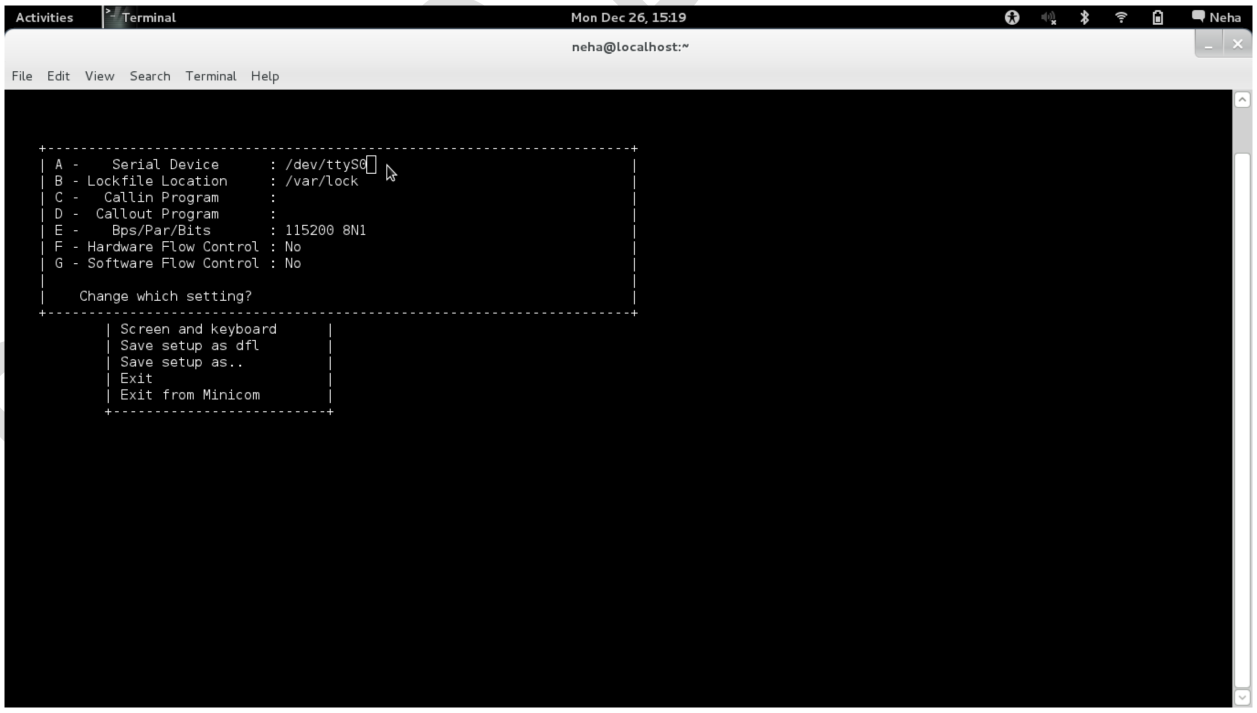
```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

5. Check the serial setup settings:

   Serial device: /dev/ttyS0

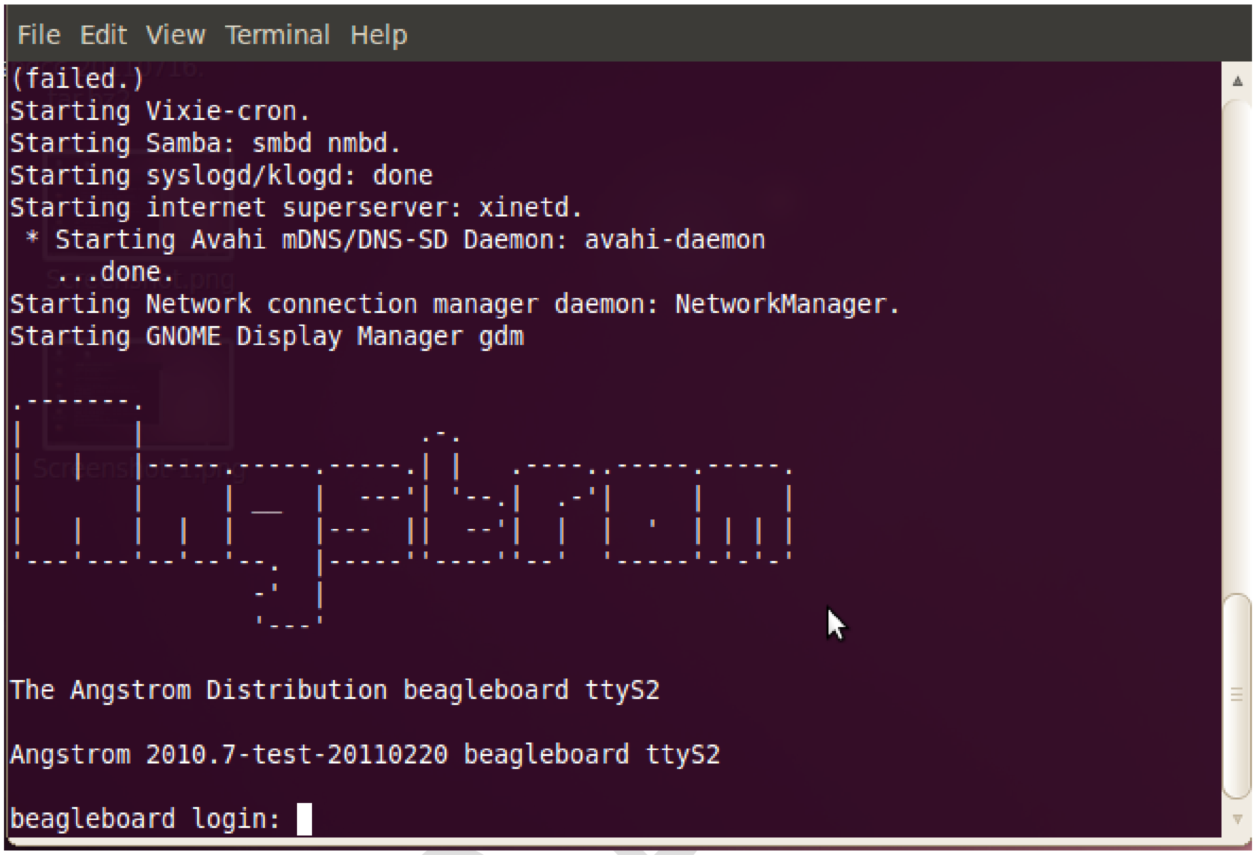
   Bps/Par/Bits: 115200 8N1

   No Hardware Flow Control/Software control



```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyS0                                 |
| B - Lockfile Location     : /var/lock                                  |
| C -   Callin Program      :                                            |
| D -  Callout Program      :                                            |
| E -    Bps/Par/Bits       : 115200 8N1                                 |
| F - Hardware Flow Control : No                                         |
| G - Software Flow Control : No                                         |
|                                                                        |
|    Change which setting?                                               |
+-----------------------------------------------------------------------+
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```

6. Save and exit
7. Angstrom will be loaded on Beagle Board



```
(failed.)
Starting Vixie-cron.
Starting Samba: smbd nmbd.
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
   ...done.
Starting Network connection manager daemon: NetworkManager.
Starting GNOME Display Manager gdm

.-------.
|       |                  .-.
|   |   |-----.-----.-----.| |   .----..-----.-----.
|       |     | __  |  ---'| '--.|  .-'|     |     |
|   |   |  |  |     |---  ||  --'|  |  |  '  | | | |
'---'---'--'--'--.  |-----''----''--'  '-----'-'-'-'
                -'  |
                '---'

The Angstrom Distribution beagleboard ttyS2

Angstrom 2010.7-test-20110220 beagleboard ttyS2

beagleboard login: 
```

8. Login as root

CROSS-COMPILATION:

Creating an executable for a platform (ARM) other than on which it is run (x86)

Required Tools:

Binary tools: Binutils,Ld,as,nm,readelf
Standard C libs: glibc,uClibc or eglibc
C/C^{++} compiler: gcc,g++
Math Libraries
Debugger: gdb

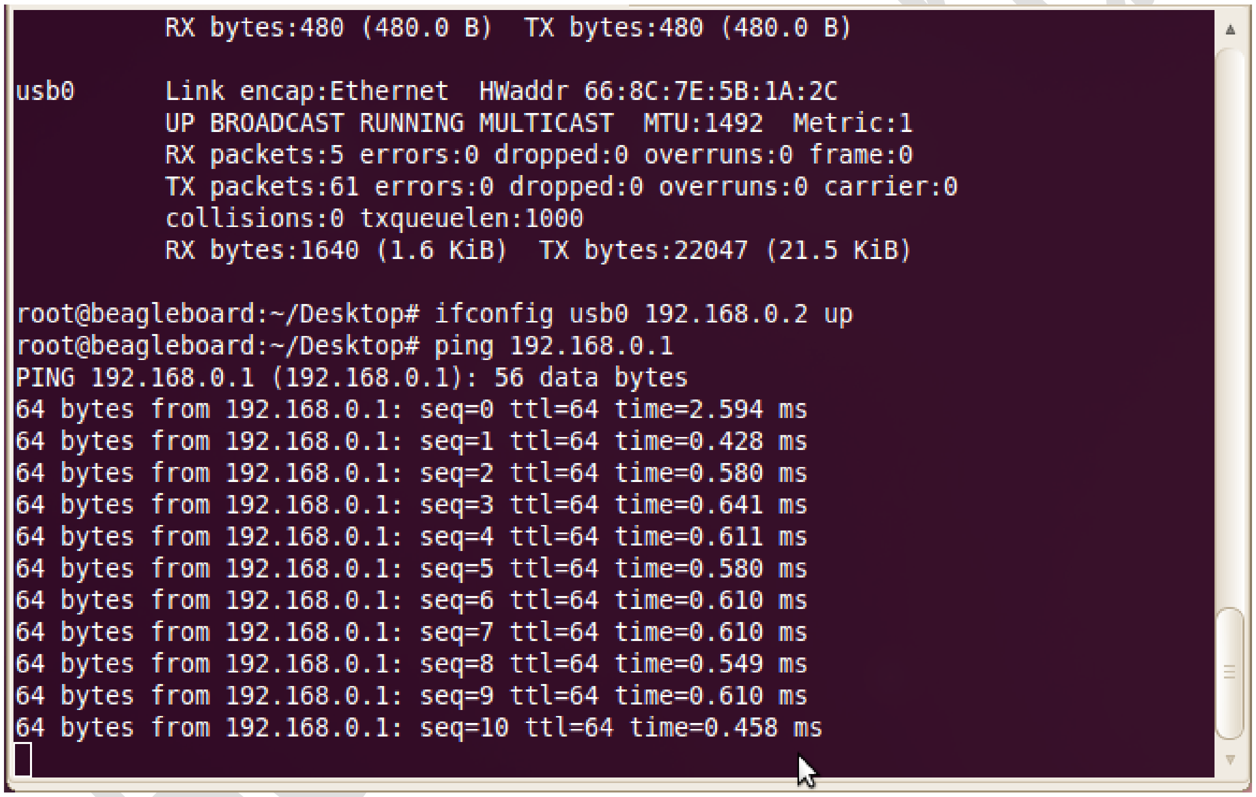
A complete packet consisting of all the necessary tools for cross-compilation is called as toolchain
There are pre-compiled toolchains available like
arm-linux-gcc-4.3.2.tar.bz2

Cross-compilation Steps:

1. Use *ifconfig* command for setting an interface's IP address and enabling a given interface
2. Issue in terminal:

```
host@localhost: ifconfig eth0 192.168.0.1 up
```

3. Similarly do the same in Beagle Board with another IP address say, 192.168.0.2



```
          RX bytes:480 (480.0 B)  TX bytes:480 (480.0 B)

usb0      Link encap:Ethernet  HWaddr 66:8C:7E:5B:1A:2C
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1640 (1.6 KiB)  TX bytes:22047 (21.5 KiB)

root@beagleboard:~/Desktop# ifconfig usb0 192.168.0.2 up
root@beagleboard:~/Desktop# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: seq=0 ttl=64 time=2.594 ms
64 bytes from 192.168.0.1: seq=1 ttl=64 time=0.428 ms
64 bytes from 192.168.0.1: seq=2 ttl=64 time=0.580 ms
64 bytes from 192.168.0.1: seq=3 ttl=64 time=0.641 ms
64 bytes from 192.168.0.1: seq=4 ttl=64 time=0.611 ms
64 bytes from 192.168.0.1: seq=5 ttl=64 time=0.580 ms
64 bytes from 192.168.0.1: seq=6 ttl=64 time=0.610 ms
64 bytes from 192.168.0.1: seq=7 ttl=64 time=0.610 ms
64 bytes from 192.168.0.1: seq=8 ttl=64 time=0.549 ms
64 bytes from 192.168.0.1: seq=9 ttl=64 time=0.610 ms
64 bytes from 192.168.0.1: seq=10 ttl=64 time=0.458 ms
```

4. Check the connectivity using *ping* command as shown above

Create a sample C program on your host system in your /home directory, say ab.c

```
#include<stdio.h>

void main()

int i=20,j=10,k;

{

k=i+j;

printf("%d the addition is:",k);

return 0;

}
```

- Now we need to compile this program and create an ARM executable on Host machine.
- In X-86 we use a gcc compiler to compile a program and create an Intel x-86 executable.

Issue the command:

```
host@localhost: gcc –o ab1 ab.c

host@localhost: file ab
```

Check the type of the file created after the compilation using *'file'* command

Obtain a pre-compiled arm-linux-gcc toolchain,extract the contents.

```
host@localhost: tar –xvf arm-linux-gcc-4.3.2.tar.bz2
```

Modify the PATH Variable:

In order to make the system understand the location of the executable, a PATH variable is defined. So for our arm-linux-gcc compiler we need to give its exact PATH.

Issue the command:

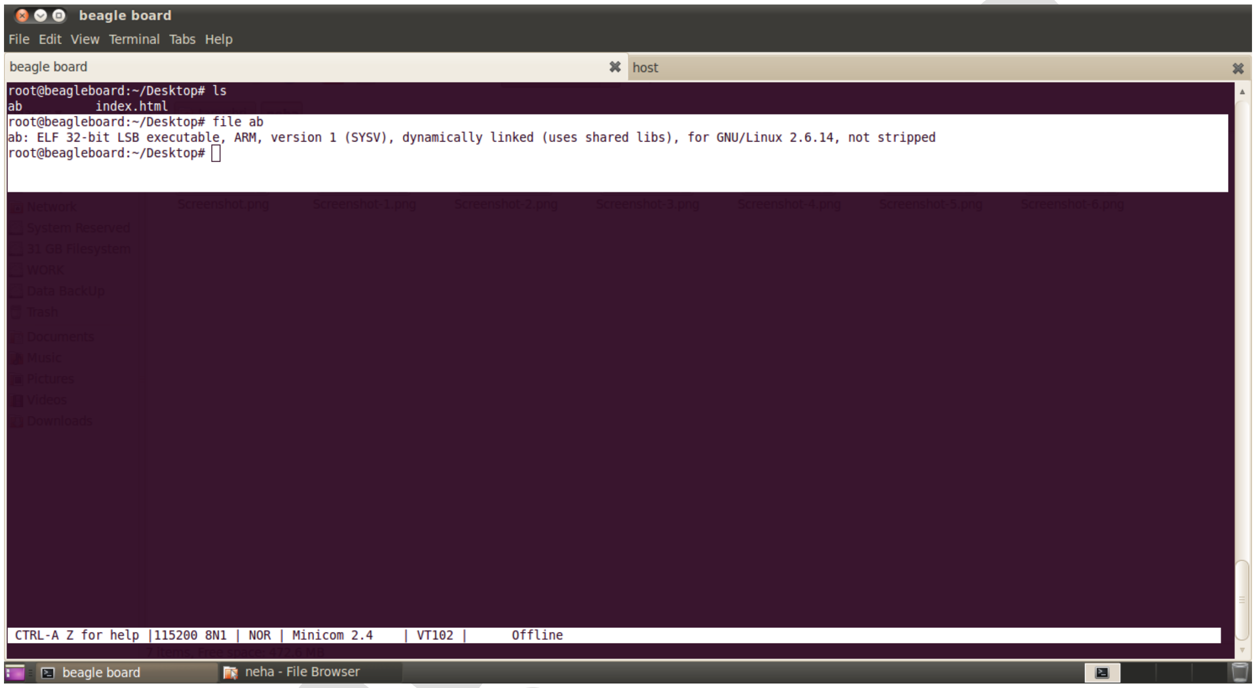
```
host@localhost: export PATH=$PATH:/home/neha/arm/4.3.2/arm-none-linux-gnueabi/bin
```

(Note: PATH will be different in your case, after $PATH:/give/path/to/your/toolchain)

Issue in terminal:

```
host@localhost: arm-linux-gcc  -o  ab ab.c
```

This command will compile the code for ARM and create an ARM executable. U can check by 'file' command



```
root@beagleboard:~/Desktop# ls
ab          index.html
root@beagleboard:~/Desktop# file ab
ab: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.14, not stripped
root@beagleboard:~/Desktop# 

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 |       Offline
```

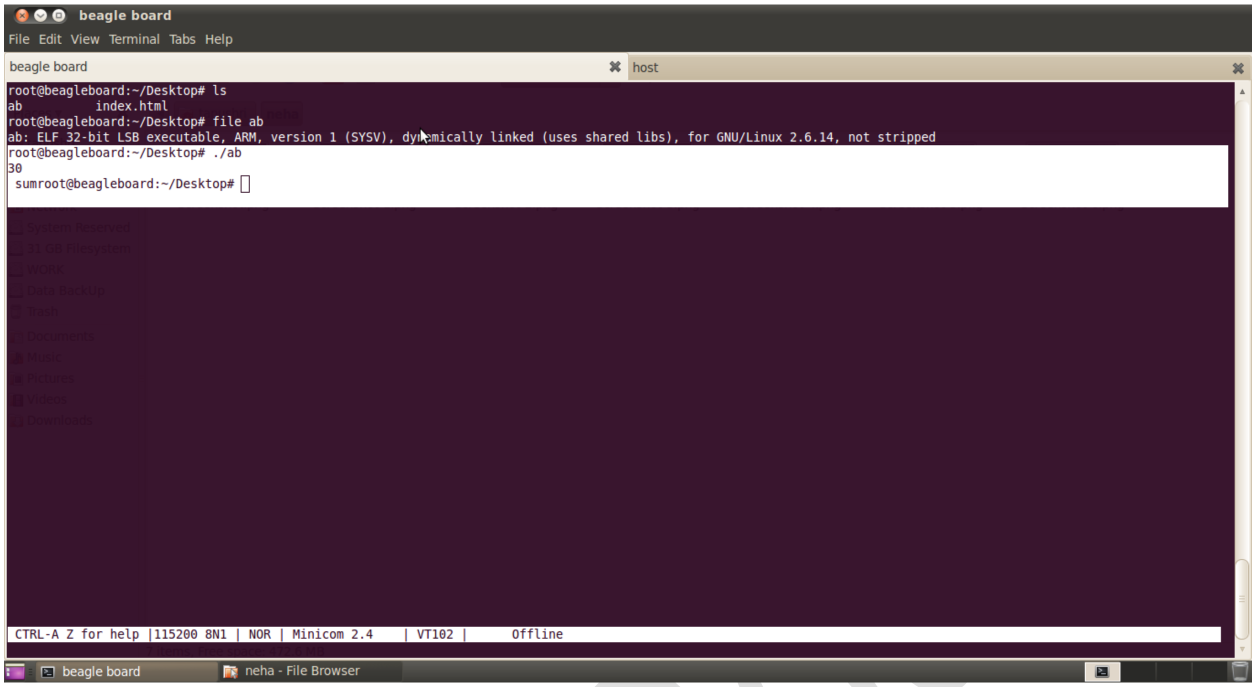
- ➢ Now that we have an ARM executable with us, we need to execute it on ARM board(Beagle Board)
- ➢ Now let us transfer the ARM executable to the Beagle Board through Ethernet

Issue the command in BeagleBoard shell:

```
root@beagleboard: wget http://192.168.0.1 /ab
```

- ➢ This will fetch the file 'ab' (ARM executable) from the IP address 192.168.0.1 (x86) through *'wget'* command . See the previous steps where we had configured the Host and Target with their own IP address's.
- ➢ Execute the ARM executable file on Beagle Board

```
root@beagleboard: ./ab
```



```
root@beagleboard:~/Desktop# ls
ab          index.html
root@beagleboard:~/Desktop# file ab
ab: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.14, not stripped
root@beagleboard:~/Desktop# ./ab
30
 sumroot@beagleboard:~/Desktop# 
```

```
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 |      Offline
```

This is how we have successfully cross-compiled a program on Beagle Board

[Note:The procedure for the cross-compilation on any ARM platform remains more or less similar]