**[EE414] Embedded Systems**

**Lab1 Report**

Woojin Lee

20180480

School of Computing, KAIST

March 20, 2021

1. **Purpose**
   1. **Purpose**

Get acquainted with the cross development system:Target board: Beaglebone Black with Linux (in microSD).Host computer: PC with Linux, cross compiler, NFS, and minicom

* 1. **Problem Statement**

An Embedded Application Program

Construct a cross-development system on a Lab (or your own) PC.

Edit and cross compile an example embedded application program:

Compute a sine function using Taylor series.

Download to the embedded system, debug, and run.

1. **Experiment Sequence**
   1. **Overall Experiment Sequence**

|  |  |  |
| --- | --- | --- |
| **Step 1** | Test Ubuntu on PC | PC |
| **Step 2** | Install Debian on Beaglebone | PC, SD Writer |
| **Step 3** | Configure Beaglebone Ubuntu | PC, BeagleBone, Router |
| **Step 4** | Install cross compiler on PC and test | PC, BeagleBone, Router |
| **Step 5** | NFS in PC and Beaglebone | PC, BeagleBone, Router |
| **Step 6** | Test Makefile | PC, BeagleBone, Router |
| **Step 7** | Test application with gdb | PC, BeagleBone, Router |

Below in the appendix, I attached my terminal output step by step.

1. **Experimental Results**

I attached a video in which beaglebone runs a cross-compiled ```taylor\_ce``` file.

Also, here comes ```taylor\_re.c``` file and ```taylor\_ok.c``` code.

1. **taylor\_re.c**

|  |
| --- |
| // Program Taylor\_re.c // Compute sine() using the Taylor series with one to five terms.   #include <stdio.h> #include <math.h>  // Define PI #define PI 3.14 #define TERMS 5  int factorial(int m) {  int facto = 1;  for (int i=2; i<=m; ++i)  facto = facto\*i;   return facto; }  int main() {  float s[TERMS], e[TERMS];  int indeg;  int pwr, facto;  int n, i, j; // Loop index  double angpwr;   // 1. Get input angle  printf("Enter angle in degrees: ");  scanf("%d", &indeg);   // 2. Compute sine with Math library  double pi = atan(1.) \* 4.;  double dtr = pi/180;  double inrad;  inrad = indeg \* dtr;  float s0 = sin(inrad);    printf("sin(%d)= %g using Math library.\n", indeg, s0);   // 3. Compute sine using the Taylor series up to n terms  for (n=1; n<=TERMS; ++n) {  // Loop to sum term i  for (i=1; j<=n; ++i) {  // Compute pwr  pwr = 2\*i - 1;  // Compute angle^pwr  angpwr = indeg;  for (j=2; j<=i; ++j) angpwr \*= indeg\*indeg;  // Compute factorial pwr!  facto = factorial(pwr);  // Compute sine by Taylor series  s[i-1] += angpwr/facto;  }  }   // 4. Compute error  for (n=1; n<=TERMS; ++n)  e[i-1] = s[i-1] - s0;   // 5. Print result  printf("sin(%d) using Taylor series:\n", indeg);  for (int i=1; i<=TERMS; ++i) {  printf("%d terms: sin(%d)= %g, error= %g\n", i, indeg, s[i-1], e[i-1]);  }   return 0; } |

1. **taylor\_ok.c**

|  |
| --- |
| // Program Taylor\_ok.c // Compute sine() using the Taylor series with one to five terms.   #include <stdio.h> #include <math.h>  // Define PI #define PI 3.14 #define TERMS 5  int factorial(int m) {  int facto = 1;  for (int i=2; i<=m; ++i)  facto = facto\*i;   return facto; }  int main() {  double s[TERMS], e[TERMS];  int indeg;  int pwr, facto;  int n, i, j; // Loop index  double angpwr;   // 1. Get input angle  printf("Enter angle in degrees: ");  scanf("%d", &indeg);   // 2. Compute sine with Math library  double pi = atan(1.) \* 4.;  double dtr = pi/180;  double inrad;   inrad = indeg \* dtr;  float s0 = sin(inrad);    printf("sin(%d)= %g using Math library.\n", indeg, s0);   // 3. Compute sine using the Taylor series up to n terms  for (n=1; n<=TERMS; ++n) {  s[n-1] = 0.0;  // Loop to sum term i  for (i=1; i<=n; ++i) {  // Compute pwr  pwr = 2\*i - 1;  // Compute angle^pwr  angpwr = inrad;  for (j=2; j<=pwr; ++j) angpwr \*= inrad;  // Compute factorial pwr!  facto = factorial(pwr);  // Compute sine by Taylor series  if(i%2 == 1)  s[n-1] += angpwr/facto;  else  s[n-1] -= angpwr/facto;  }  }   // 4. Compute error  for (n=1; n<=TERMS; ++n)  e[n-1] = s[n-1] - s0;   // 5. Print result  printf("sin(%d) using Taylor series:\n", indeg);  for (int i=1; i<=TERMS; ++i) {  printf("%d terms: sin(%d)= %g, error= %g\n", i, indeg, s[i-1], e[i-1]);  }   return 0; } |

1. **Result**

Example result of taylorok.c file when input is 30:

|  |
| --- |
| Enter angle in degrees: 30  sin(30)= 0.5 using Math library.  sin(30) using Taylor series:  1 terms: sin(30)= 0.523599, error= 0.0235988  2 terms: sin(30)= 0.499674, error= -0.000325821  3 terms: sin(30)= 0.500002, error= 2.13259e-06  4 terms: sin(30)= 0.5, error= -8.13098e-09  5 terms: sin(30)= 0.5, error= 2.02799e-11 |

We can find this result is identical to the given reference. Also, this is the result that ran on the beaglebone-black board.

1. **Discussion**

* **Troubleshooting with Virtual Machine**

I first used Virtual Machine on Windows for Linux. However, when I use Virtual Machine, it requires some more effort to work properly. For instance, when I use LAN, it requires network settings like NAT, Bridge Network, etc… I used USB to make my environment work in dual boot mode.

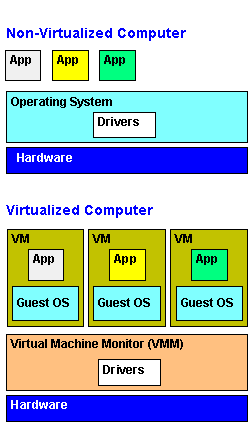
****

Image source : <https://www.pcmag.com/encyclopedia/term/virtual-machine>

* **Code that I modified.**
  + **taylor\_re.c**

In taylor\_ce.c line 31: missed semicolon

In taylor\_ce.c line 42: missed semicolon, 2\*i instead of 2i

In taylor\_ce.c line 63: missed space

In taylor\_ce.c, we have to modify indeg to inrad since Math library interprets input as radian scale.

* + **taylor\_ok.c**

I added code that initialize s[n-1] to 0.0 in every loop step.

I modified for loop index.

When I compute angle^pwr, I have to multiply inrad, not indeg\*indeg.

When I add terms for taylor series, I have to change the sign alternatively according to odd and even index.

1. **References**

[1] Provided Problem Statement

[2] Getting Started with Beaglebone and Beaglebone Black, <http://www.beagleboard.org/Getting%20Started>

[3] Beaglebone Rev. A5. System Reference Manual,

<http://circuitco.com/support/index.php?title=BeagleboneRevA5>.

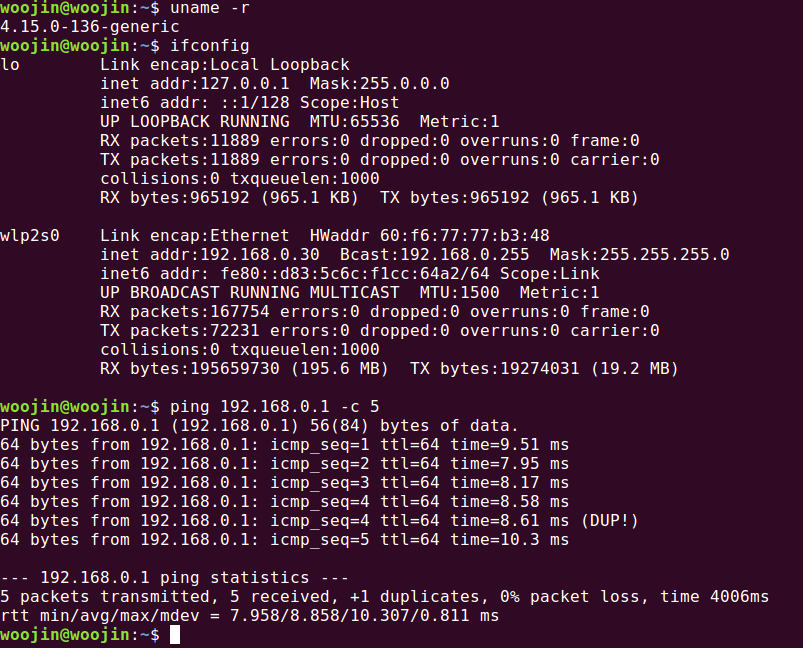
NOTE. The version of Beaglebone boards in the Lab is Rev. A5.

Be sure to download the correct version.

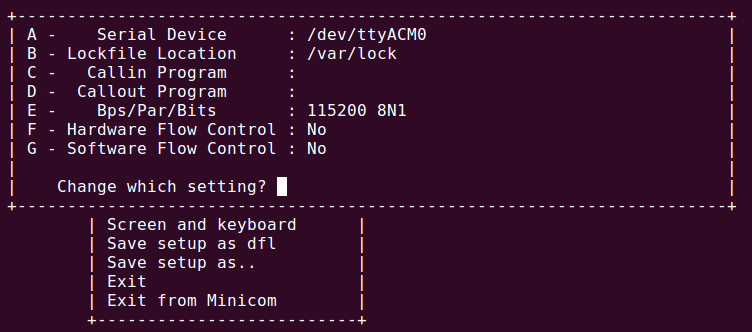
[4] “Embedded Linux Primer”, C. Hallinan, Prentice Hall

**Appendix: Step by Step procedures**

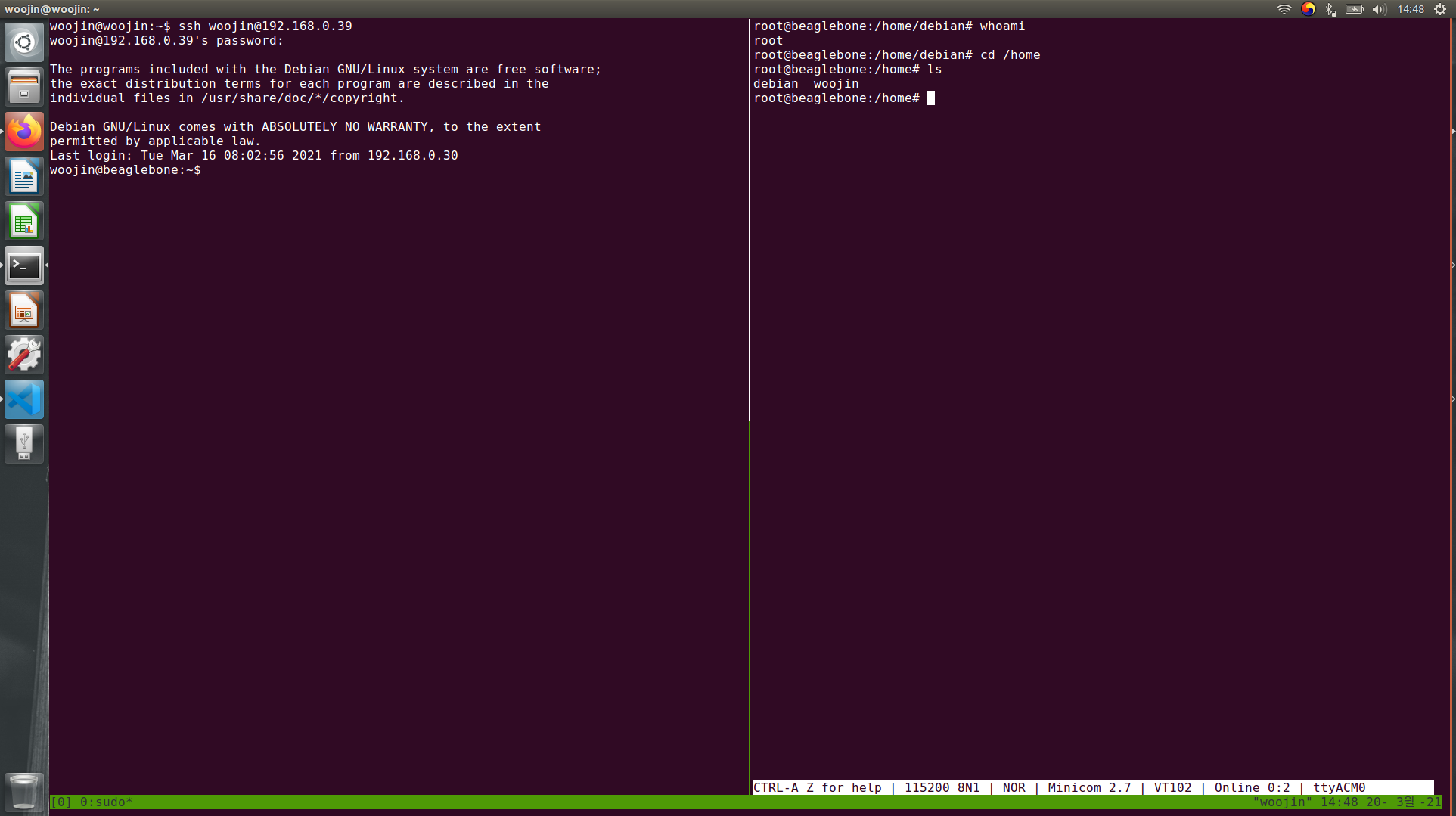
**Step 1. Test Ubuntu on PC**



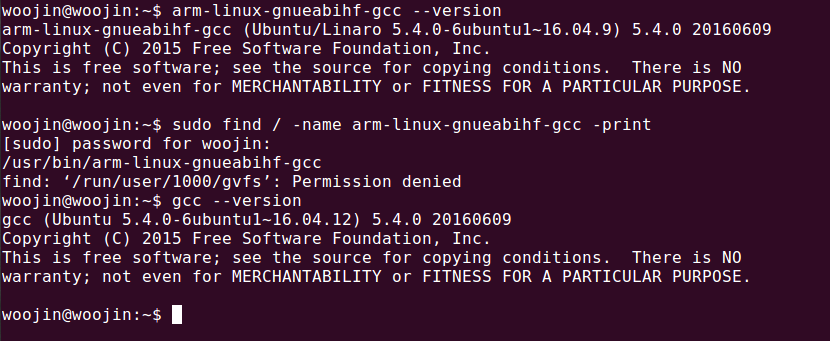
**Step 2. Install Ubuntu on Beaglebone microSD**

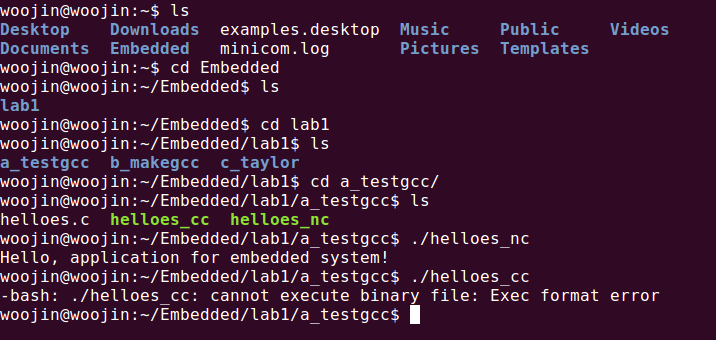


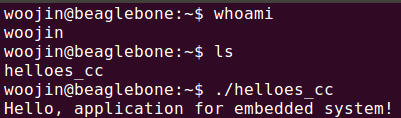
**Step 3. Configure Beaglebone Debian**



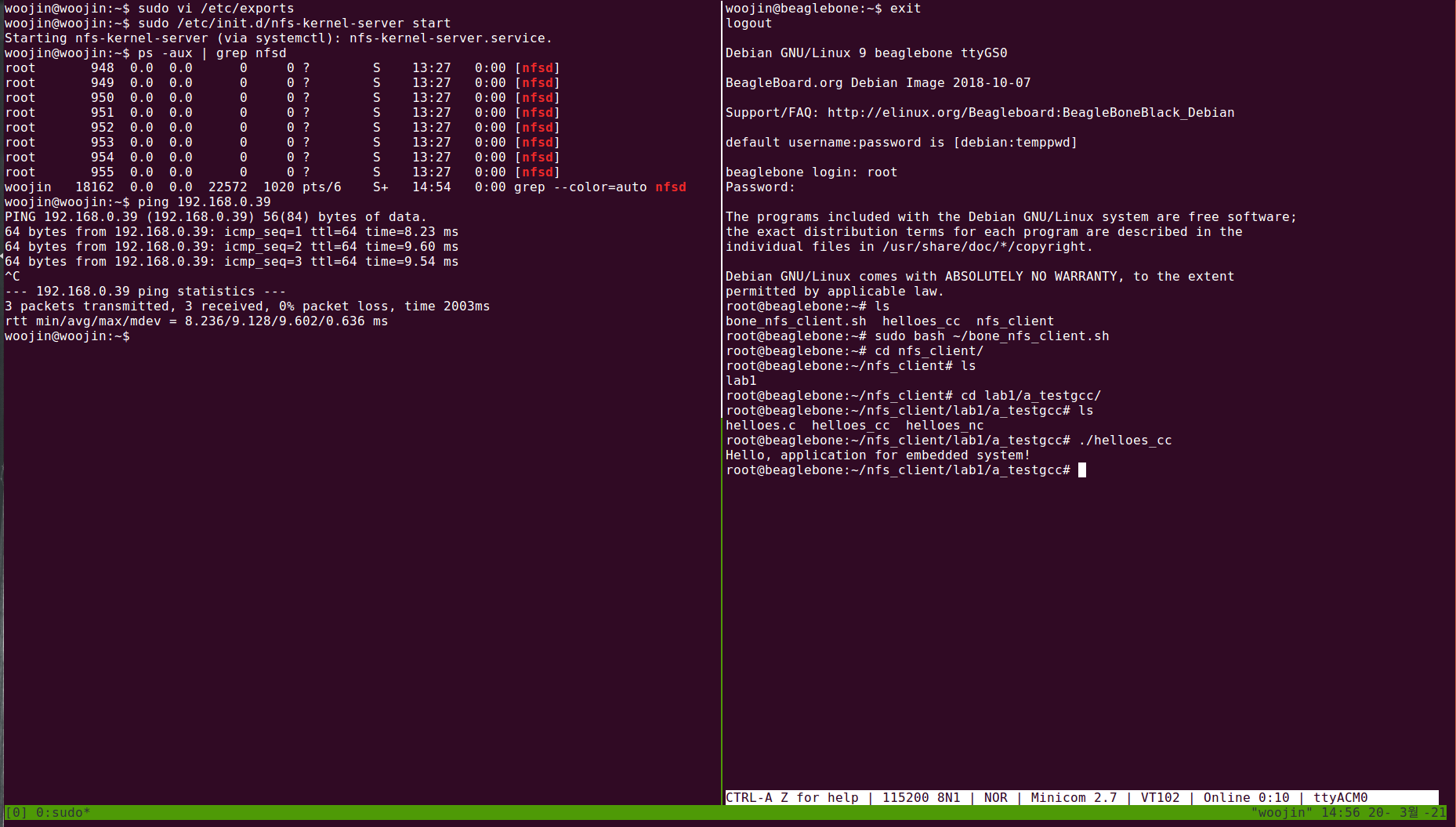
**Step 4. Install Cross-Compiler and Test**



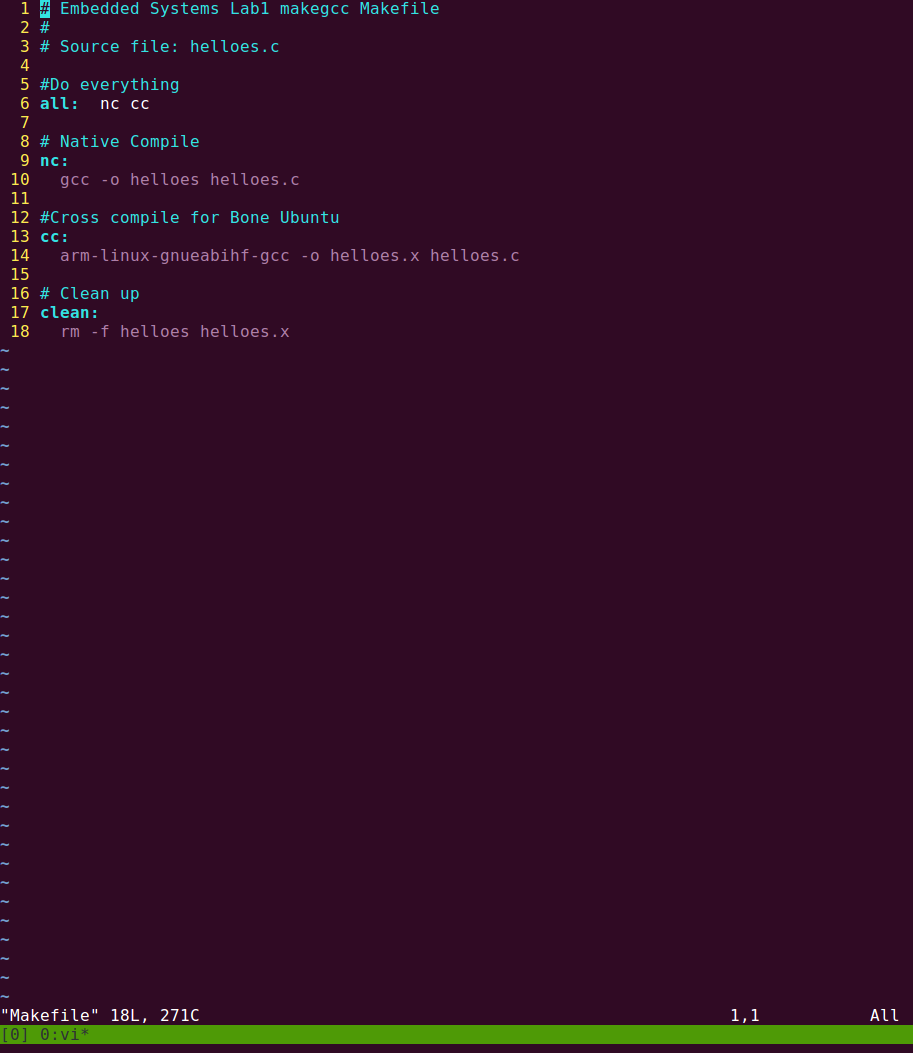


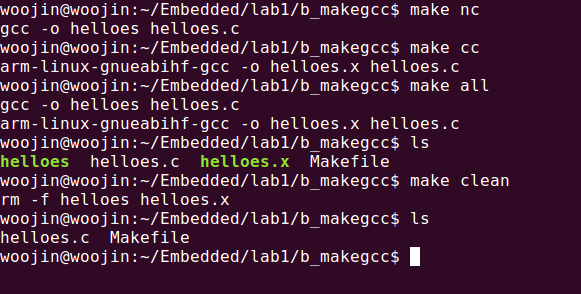


**Step 5. NFS in Bone-Ubuntu & PC-Ubuntu**



Step 6. Test Makefile





**Step 7. Debug Application for PC**

Since I’ve done all debug procedures, I would replace this part with the attached video.