Lab Assignment 1
Zedboard Linux Getting Started

Patricia Gavelek, Anders Dahl
Gavelek.p@husky.neu.edu
dahl.a@husky.neu.edu

Submit date: 1/21/16
Due Date: 1/20/16

Abstract
The goal of this first lab was to get started and familiarized with the Zedboard, Linux command line, and the C programming language. Connections were formed to the board through the use of SSH. Working with the Zedboard requires familiarity with the Linux command line. For the final part of the lab, various C programs were written on the Zedboard. We used Vim as our text editor of choice.

Introduction

The goal of this first lab was to get started and familiarized with the Zedboard, Linux command line, and the C programming language. From the Linux command line, one is able to securely connect to the Zedboard through the use of SSH. The command line also allows for general navigation once on the system. Getting comfortable with the C programming language was a major part of the lab. We looked at some simple programs – hello world, sort 10 integers, and sort 10 strings. We used Vim as our text editor of choice.

Lab Discussion

In this lab, a computer, a Zedboard and all its' corresponding cables are all the parts that are needed. MobaXterm is the program on the computer used to run a secure shell which connects to the Zedboard. An IP address and username were used to SSH into the Zedboard. STFP was used to transfer files from the Zedboard back to our computers. SSH and SFTP are used because they encrypt information, whereas Telnet and FTP use plain text, which can be more easily read by third parties. [1]

Results and Analysis

Lab 0.1 Connecting to ZedBoard

The Zedboard is connected to the host via Ethernet to the RJ45 connector. The host connects from the USB to Ethernet adapter. Once the Zedboard is powered on, a secure connection can be made with the use of SSH. The IP address is 192.168.1.10, and the user is a combination of the section and group number. A password also needs to be entered.

```
ssh user406@192.168.1.10
```

Lab 0.2 User Accounts and ZedBoard Login

The root user has access to all files and commands. Because the root user has access to everything, files that are not supposed to be accessed may be tampered with and in general being the root user can cause security issues. This is why we are logged in as user406 instead.

Lab 0.3 Hello World on ZedBoard

"Hello world" is the most basic of codes and is used to introduce almost every single programming language.

```c
#include <stdio.h>      //Declare standard ouput and input function

int main () {   //Start function main
        printf("Hello World");  //Print "Hello World" on screen
        return 0;       //Complete function main
}
```

```
user406@localhost:~/lab0$ vi hello.c
user406@localhost:~/lab0$ gcc hello.c -o hello
user406@localhost:~/lab0$ ./hello
Hello Worlduser406@localhost:~/lab0$
```

The gcc hello.c –o hello compiles the text file into an executable file that the computer can read and carry out. Running this newly produced file displays the “Hello World” in the terminal. SFTP was used to transfer the files back to our other computers.

Lab 0.4 Sorting 10 integers

```c
// Import libraries to use
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Reference:
// http://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm

// An integer comparative function that works with qsort. The arguments need to
// be in the format given for it to work with qsort. Takes in a void * a and a
// void * b which can basically be anything. We will make it less generic later.
int cmpfunc (const void * a, const void * b)
{
    // Cast the void*'s to be integer pointers and then dereference them.
    // Lastly, it follows how a comparative function should work with qsort.
    // A negative values means >, 0 is =, and a positive value is <.
    return ( *(int*)a - *(int*)b );
}

// A main function that runs the entire program. We take in an argc and argv,
// but they are not used in the program. It is standard practice to include
// them.
int main( int argc, char *argv[] )
{
    // Declare the size of the array and initalize an array that can hold 10
    // integers.
    int size_of_array = 10;
    int values[10];
```

```
    // Enter values for the 10 integers using scanf.
    int i;
    for(i = 0; i < size_of_array; i++) {
        printf("Enter the %d integer: ", i+1);

        // Scanf needs to take in a pointer.
        scanf("%d", &values[i]);
    }

    // Sort the values using qsort and our custom cmp function. It needs the
    // sizeof(int) because we are comparing and sorting integers in this case.
    qsort(values, size_of_array, sizeof(int), cmpfunc);

    // Print the sorted values
    int n;
    printf("Here are the sorted values: ");
    for( n = 0 ; n < size_of_array; n++ )
    {
        printf("%d ", values[n]);
    }
    printf("\n");

    // Exit the program and return.
    return 0;
```

```
Linok-2 :: ~ » gcc sort_10_integers.c -o sort_10_integers
```

```
Linok-2 :: ~ » ./sort_10_integers
```

```
Enter the 1 integer: 43
Enter the 2 integer: 1
Enter the 3 integer: 34
Enter the 4 integer: 2
Enter the 5 integer: 421
Enter the 6 integer: 43
Enter the 7 integer: 1
Enter the 8 integer: 234
Enter the 9 integer: 234
Enter the 10 integer: 11
Here are the sorted values: 1 1 2 11 34 43 43 234 234 421
```

Extra credit. Sorting 10 strings

```
 0 // Import libraries to use
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4
 5 // Reference:
 6 // http://www.anyexample.com/programming/c/qsort__sorting_array_of_strings__integers_and_structs.xml
 7
 8 // An integer comparative function that works with qsort. The arguments need to
 9 // be in the format given for it to work with qsort. Takes in a void * a and a
10 // void * b which can basically be anything. We will make it less generic later.
11 int strcmpfunc (const void * a, const void * b)
12 {
13     // Cast the void*'s to be character pointer pointers and then dereference them.
14     // This is done in order to avoid issues we were having with the consts.
15     // Lastly, it follows how a comparative function that should work with qsort.
16     // A negative values means >, 0 is =, and a positive value is <. This is
17     // what strcmp returns.
18     char const **ia = (const char **)a;
19     char const **ib = (const char **)b;
20     return strcmp(*ia, *ib);
21 }
22
23 // A main function that runs the entire program. We take in an argc and argv,
24 // but they are not used in the program. It is standard practice to include
25 // them.
26 int main( int argc, char *argv[] )
27 {
28     // Declare the size of the array and initalize an array that can hold 10
29     // strings. We need to allocate space for each string individually later.
30     int size_of_array = 10;
31     char * strings[10];
32
33     // Enter values for the 10 strings using scanf.
34     int i;
35     for(i = 0; i < size_of_array; i++) {
36         // We need to malloc some space for the strings we are entering from
37         // command line
38         strings[i] = (char *) malloc(50);
39         printf("Enter the %d string: ", i+1);
40
41         // Scanf needs to take in a pointer.
42         scanf("%s", strings[i]);
:wa
```

```
    // Sort the values using qsort and our custom cmp function. It needs the
    // sizeof(char *) because we are comparing and sorting strings in this case.
    qsort(strings, size_of_array, sizeof(char *), strcmpfunc);

    // Print the sorted strings
    int n;
    printf("Here are the sorted values: ");
    for( n = 0 ; n < size_of_array; n++ )
    {
        printf("%s ", strings[n]);
    }
    printf("\n");

    // Exit the program and return.
    return 0;
}
```

```
gcc sort_10_strings.c -o sort_10_strings
```

```
Linok-2 :: ~ » ./sort_10_strings
Enter the 1 string:
```

```
Enter the 1 string: sdf
Enter the 2 string: fgs
Enter the 3 string: dsf
Enter the 4 string: asf
Enter the 5 string: gd
Enter the 6 string: v
Enter the 7 string: s
Enter the 8 string: g
Enter the 9 string: x
Enter the 10 string: z
Here are the sorted values: asf dsf fgs g gd s sdf v x z
```

Conclusion

After having completed this lab, one should be able to program on the Zedboard using mobaxterm and write a simple sort in C. There are significant differences between C++ and C, which doing this lab made abundantly clear.

References

[1] http://www.itcs.umich.edu/ssh/