CS 211 Quick Start Guide

This document is designed to get students in CS 211 up and going as quickly as possible. It will explain how to securely access bertvm.cs.uic.edu, and how to use bertvm to write, compile, and run a basic C program, along with some simple UNIX commands. Last revised May 2022 by John Bell.

Connecting to a remote server

In this course we will use a UNIX command-line interface on *bertvm.cs.uic.edu* during lab sessions, and as a platform for grading assignments. (Outside of lab you may develop your code wherever you wish, but you should test it on bertvm before submitting it, as that is where it will be graded.) Bertvm is owned by the CS department, but it is accessed using standard UIC NetIDs and passwords. Bertvm requires the use of a Secure Shell (SSH) for secure access. (Remote login, not a web site.)

Acquiring a Secure Shell and Secure File Transfer Program

UIC offers a lot of free or discounted software through its WebStore, located at https://webstore.illinois.edu . (Or just start at uic.edu and search for "WebStore".) One of those free products, which we recommend for this class is "Secure CRT & Secure FX". To obtain, install, and employ this software, follow these steps:

1. Go to the WebStore web site https://webstore.illinois.edu , select "Personal Purchases", and login using your NetID and password.
2. Select "Windows Products" (regardless of your real OS), find Secure CRT & Secure FX on the list, add the package to your cart, and proceed to the checkout.
3. At checkout you will be able to choose your preferred download package. At this time you can choose between Windows, Mac, Red Hat, or Ubuntu OSes, and you can also choose to download just the Secure CRT, or just the Secure FX, or **both items bundled together**.
4. Below the download selection is license information that will be needed to license the product after it is installed. You will need to cut-and-paste about 7 lines of information to complete the license process, not just the license key, from "Name" through "Features".
5. There is also a link to full installation instructions on the check out page. It is worth looking over those instructions, particularly if you have not installed a lot of software before.
6. Launch SecureCRT, and specify bertvm.cs.uic.edu as a new session. The first time you connect to any new site using Secure CRT you will get a "New Host Key" warning about the key not being found in the existing database. This is normal for new sites. Just select "Accept and Save" and continue. You will need to provide your UIC NetID and password.
7. Once SecureCRT is connected to bertvm, find and click the button to launch SecureFX. Secure CRT will provide a secure terminal window to bertvm, and Secure FX will provide secure file transfer between local and remote machines. The latter will be needed in CS 211 to transfer starting shell programs to bertvm and to transfer completed work back to a local machine for submission to Gradescope.

Built-in ssh connection on Mac/Linux

If you are using a Mac/Linux system, you can directly use *ssh* service through any terminal window. Simply open your terminal and type the following command:

```
ssh netid@bertvm.cs.uic.edu
```

Then proceed by entering your password and you will be connected to your *bertvm* account. For securely transferring files, you can either use something like *Filezilla* or the built-in *sftp* command.

Editing, Compiling, and Running Programs

This tutorial will help you get familiar with the basics of creating a C program, editing in Vim/Vi, compiling and running it from the terminal. After completing these tasks you should be comfortable in running your programs on *bertvm* or your personal machine's terminal.

First connect to your *bertvm* server. After successful connection, you need to navigate to a proper directory and create a folder for your C program. Navigate to your *Documents* directory and create a new folder:

```
[i211@bertvm ~]$ cd Documents

[i211@bertvm ~/Documents]$ mkdir Lab1
```

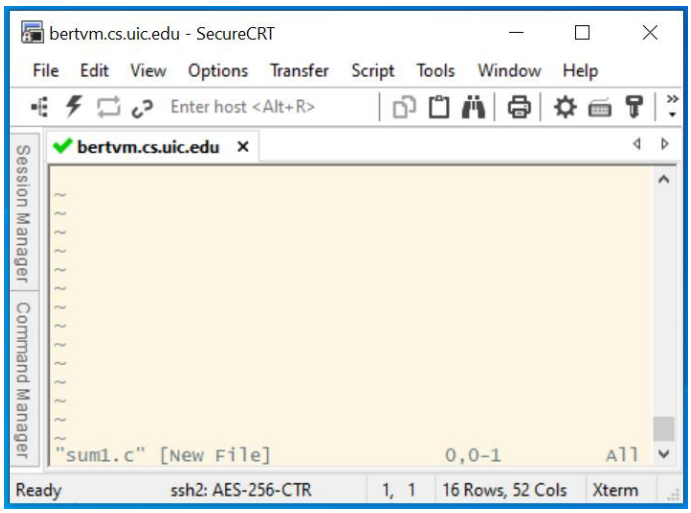
You can check the Linux command section of this write-up to get familiar with Linux commands and basics of navigation in a Linux environment.

Now navigate to your recently created folder titled "Lab1" using *cd* command (*cd Lab1*). Here you are going to create a new file and write your C program. We will use Vim/Vi for creating and editing files, but you can use any other editors like nano or emacs if you want.

```
[i211@bertvm Lab1 ]$ vim sum1.c
```

Since your folder is empty, the command above will create a new empty file *sum1.c* and open it in Vim editor. You can use the same command for opening an already existing file.

(NOTE: sum1.c is used as an example in this handout, but it is NOT a program that will be used in CS 211 this term, either in lab or for homework.)



Vim has two modes: **Command Mode** and **Insert Mode**. Command Mode is for executing commands. Operations like custom Vim commands, saving files, etc. Insert Mode is for editing text freely.

To enter Insert Mode simply type ”i”

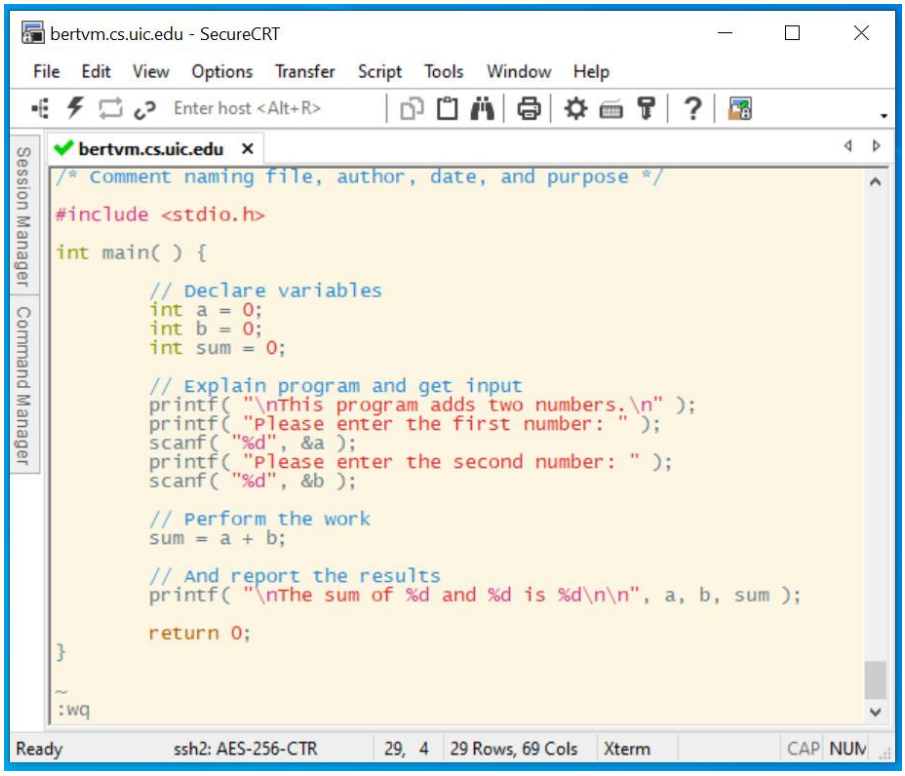
Now you can edit your file and write your program.

Your first program in this lab is a simple program that reads in two integers and prints their sum:

```
#include < stdio.h > int main( void ) {

        int a = 0 ;
        int b = 0 ;
        int sum = 0 ;

        printf( ”Please enter the  first number: “ ) ;
         scanf( ”%d”, &a );

        printf( ”\nPlease enter the second number: ” );
        scanf( ”%d”, &b);

        sum = a +b ;
        printf(”\nThe sum of %d and %d is : %d \n ”, a, b, sum );

        return 0;
}
```

For saving your program you need to first exit Insert Mode and enter Command Mode by hitting *ESC*.

Once you're back into command mode, you'll need to save the file and then quit Vim. To enter a command, you need to hit the colon key ":". The command to save the edits (write, quit) is *:wq*.



Alternatively, if you want to quit Vim without saving changes, just type *:q!*. This is "quit and discard changes" command as the exclamation mark means discard changes.

Now you need to compile and run your *sum.c* program. A computer processor only runs programs written in machine language. Machine language files are written in binary and do not contain text that you can read. Compiler is a piece of software that acts as a translator between the source code, such as C or C++, and the target language, such as machine language. In other words, the compiler converts our source code to executable instruction file for computers.

We use **gcc** compiler to compile our C program.

GCC stands for "GNU Compiler Collection". GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada.

1. Run the compiler (gcc) to convert your source code into an executable file you can run.

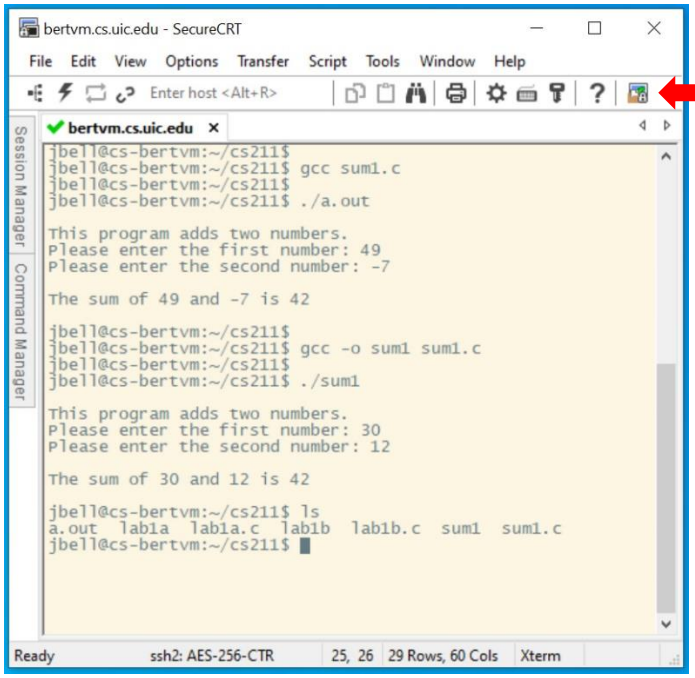
```
[i211@bertvm Lab1 ]$ gcc sum1.c
```

2. If your program contains errors, the compiler will let you know. If the compilation is successful, it creates an executable called a.out which you can run:

```
[i211@bertvm Lab1 ]$ ./a.out
```

3. Alternatively, the "-o" flag to gcc lets you specify an executable program name other than a.out:

```
[i211@bertvm Lab1 ]$ gcc -o sum1 sum1.c
[i211@bertvm Lab1 ]$ ./sum1
```



Launch SecureFX Button

Transferring Files

Launch SecureFX by selecting the "Launch SecureFX" button in SecureCRT (see above.) That will open a linked pair of side-by-side directory windows for the remote (bertvm) and local computers. Then you can simply drag files from one side to the other to transfer them over. **CAUTION:** Be careful not to drag files the wrong direction, or you may overwrite all your hard work.

Linux Basics

Here is a list of useful Linux command. For learning more about Linux please check the resources page on course website.

Navigation

Linux filesystems are based on a directory tree. You can create directories (or “folders”) inside other directories, and files can exist in any directory.

- **pwd**

  Stands for ”print working directory”, will print the path to current working directory

- **ls**

  List the names of the files and directories in the current directory. Useful Options:

  - *ls -l*: list files and directories with detailed long format
  - *ls -a*:

    list all files including hidden file starting with ’.’

- **cd** < *nameofdirectory* >

  Change your new current working directory to the directory you specified. Any files that you mention are assumed to be in the current working directory.

  Additionally, you can specify **..** to change to the directory one level up in your path.

- **mkdir dir**

  Create new directories in current working directory.

- **rm PATH**

  Remove file path. If you want to remove a directory and everything in it, use *rm -r dir*

  The rm command does a permanent removal. You cannot get the file back.

File Manipulation

- **cat** < *FileName* >

  Print out the entire contents of a file to the terminal.

- **touch** < *FileName* >

  Create an empty file with specified name in current working directory.

- **mv old new**

  Move file or directory *old* to *new*. (Renames if same source and destination directories.)

- **cp old new**

  Copy file or directory *old* to *new*.

You can always check manual pages for detailed and insightful documentation for nearly every single command. Just type **man** < *command* > to access the manual page for that command.

Typing **apropos** followed by a keyword will list all known commands containing the keyword in their brief description. (May be a long list in some cases.)

Going Further

Visit the "Resources" section of the CS 211 web site to find links to additional tutorials and other references, particularly for UNIX commands, the vim editor, and C/C++ Programming.