Project 1 Linux Guide

**Part 1: Transferring files from Windows to ICS Linux servers**

1. Create your source code files. This could mean either finishing the project (for example, I used Visual Studio to do all my compiling before testing on Linux) or simply coming up with a Hello World test program. Be careful not to make any Windows system calls. I also found that the C++ compiler installed on the ICS servers, called GCC, does not seem to implement all C++11 features, so try to use as few as possible.  
     
   For the purposes of this tutorial, assume you have implemented Project 1 using only three files, main.cpp, compare.c, and common.h.
2. Download the PuTTY and PSFTP executables from this [website](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). PuTTY is an SSH client that will let you access the ICS Linux servers remotely, and PSFTP is the file transfer program you’ll use to move your code files to Linux.
3. First you will create a directory inside your personal folder on the ICS Linux machines to hold your source code for project 1. In addition, this folder will also contain your makefile and compiled executable. To create this directory, you need to connect to the Linux servers using SSH. Run the putty.exe file you downloaded in Step 2 and type “openlab.ics.uci.edu” in the Host Name field, then click Open.
4. You will be prompted for your login username and then your password. Your username is your UCINetID and your password is you ICS account password, the same as if you were logging into one of the ICS computer lab computers. Note that your password characters are not shown when you type them, not even as placeholder characters such as \*.  
     
   You are now successfully logged into the Linux server. If you scroll up you will find some ICS system announcements and details about which server machine you logged into. The very last line is your command prompt. If you have not [changed your shell](http://www.ics.uci.edu/computing/linux/shell.php) from the default, you will see something similar to [panteater@malory-duchess-archer]~%. The text before the @ sign is your UCINetID, and the part after is the name of the physical server you are logged into, which was malory-duchess-archer when I logged in, but I have also seen other names such as trinette-magoon and pam-poovey.
5. Explore the server’s file system by using the commands ls and cd. The ls command lists the files and subdirectories contained in your current directory, while the cd command changes directories given a directory name. A useful command to know is cd .., which moves you to your parent directory. When you are finished, return to your home directory, which is the directory you were in when you first connected. If you cannot find your way back to that directory, close the PuTTY window and reconnect.
6. In your home directory, create a folder for your Project 1 files by entering the command mkdir project1, then open the newly created folder by entering cd project1.
7. Now that you have a directory to hold your files, you must transfer the files from your Windows computer to the Linux server. Minimize your PuTTY window and run the psftp.exe file you downloaded in Step 2.
8. You will see the PSFTP prompt, psftp>. Connect to the ICS servers by entering open panteater@openlab.ics.uci.edu, replacing Peter the Anteater’s UCINetID with your own. Then enter your ICS account password to continue.
9. As a file transfer program, PSFTP maintains two directory names, the *local directory* and the *remote directory*. The local directory is the directory on your Windows computer where PSFTP will create or locate files sent from or to the Linux server, and the remote directory is the same thing but on the Linux server. You must change the local directory to wherever your Project 1 source code files are found, and the remote directory to the folder you created in your Linux home directory in Step 6. To change your local directory, use the lcd command, such as lcd “C:\Users\PeterAnteater\Documents\CS165\Project1”. To change your remote directory, use the cd command, such as cd project1. You can also use the ls command to list the contents of your current remote directory just as you would in PuTTY. For a full explanation of the commands available in PSFTP, go [here](http://the.earth.li/~sgtatham/putty/0.63/htmldoc/Chapter6.html).
10. With your local and remote directories configured, you can now transfer your files. To do so for the three files named in Step 1, enter mput common.h main.cpp compare.c. If you switch back to your PuTTY window and use the ls command, you will see the files now exist in your Linux project1 folder (you might need to either wait a few seconds to see them or exit and reenter your project1 directory). You have successfully transferred your source files to Linux and are ready to compile them in Part 2.

**Part 2: Compiling and makefile**

1. To prepare to compile your project, connect to the ICS Linux servers using PuTTY as described in Part 1 and navigate to your project1 directory. For the purposes of this tutorial, assume you have the same three source files named in Part 1, Step 1.
2. Enter the command g++ main.cpp compare.c –std=c++0x –o project1. Note that you do not need to specify the common.h header file.  
     
   If you receive compilation errors, you must fix them yourself. One error that I experienced was that the GCC compiler installed on the ICS servers does not understand the C++11 nullptr keyword. I was able to fix this error by adding the following lines in my code before I used the nullptr keyword:  
     
   #ifdef \_\_GNUC\_\_  
   #define nullptr NULL  
   #endif
3. Once compilation finishes without errors, use the ls command to verify that a new file called “project1” has been created. If you are using the default shell, the file will be listed in green to indicate it is an executable file. Run your program by prefixing its name with “./”, for example ./project1. Your program can output to the command line the same as it would on Windows, using functions such as printf or the C++ std::cout object. Once you have finished implementing Project 1, compile and run it on the Linux server to make sure it doesn’t take more than a minute or two to execute.
4. Note down the exact g++ command line that causes your program to compile successfully. For the purposes of this tutorial, assume it is g++ main.cpp compare.c –o project1.
5. Create a new text file on your Windows computer using your favorite text editor. This file will be your makefile. For a good introduction to makefiles, go [here](http://mrbook.org/tutorials/make/). The simplest possible makefile is shown below:  
     
   all:  
    g++ main.cpp compare.c -o project1  
     
   Replace the g++ invocation with your own version, being sure to indent the second line using a tab rather than spaces. Then save the file with the file name “makefile”, without any extension. If your text editor automatically adds a .txt extension, remove it manually.
6. Copy your new makefile from your Windows computer to your project1 folder on the Linux server using PSFTP as described in Part 1.
7. Navigate to your project1 Linux folder using PuTTY. Use the ls command to verify that your makefile is present alongside your source code files, then enter the make command. The make program automatically searches for a file called “makefile” and executes the commands located within. The result of make executing the simple makefile you created is exactly the same as if you had entered the g++ command directly on the command line. Thus, if the g++ command inside your makefile successfully compiles your program when entered directly, it will also successfully compile your program when invoked indirectly through the make utility.

**Further Resources:**

[Bren ICS Linux Server Details](http://www.ics.uci.edu/computing/linux/hosts.php)

[PuTTY: A Free Telnet/SSH Client](http://www.chiark.greenend.org.uk/~sgtatham/putty/)

[Linux Command-Line Cheat Sheet](http://www.computerworld.com/s/article/print/9030259/Linux_Command_Line_Cheat_Sheet)

[GCC Command Options](http://gcc.gnu.org/onlinedocs/gcc/Invoking-GCC.html#Invoking-GCC)