Assignment 1

9 September 2012

The purpose of this assignment is to help you set up some tools that we will use for assignments the rest of the semester.

Due Wednesday 12 September at 1 am

Setting up your tools

Install VirtualBox 4

We will use *virtual machine* software for this course. This allows us to have a consistent environment across all operating systems. I have already configured a VM image with all the software we'll need for this course.

1. Download my VM image (about 780 MiB) from `https://s3.amazonaws.com/liucs.net/Debian-Fall-12.ova`
2. Download VirtualBox for your *host OS* (Windows or OS X Mac) from `https://www.virtualbox.org/wiki/Downloads`
3. Install VirtualBox. If the installer gives you any trouble, try rebooting.
4. Run the VirtualBox application. You may just cancel the registration form.

Configure & boot the VM

1. From the **File** menu, select **Import Appliance.** Push the **Choose** button, navigate to your `Downloads` folder, and select the `Debian-Fall-12.ova` file that you downloaded.
2. Click **Next,** then **Finish.** The import process can take some time. Once it has finished, you can delete the `.ova` file, and the VirtualBox installer.
3. Now you should have a Debian "Fall 12" Linux VM in the left panel of VirtualBox. **Before you start it,** select it and click the **Settings** button on the tool bar. In the **System** section, you may have to reduce the base memory of the virtual machine. It should be no more than 75% of the memory of the host machine. For example, if you have 2G RAM, set the VM base memory to 1024MB (1G). If you have only 1G RAM, set the VM base memory to just below 768M.
4. Apply the settings and **start the Linux VM.** You may have to dismiss a dialog about capturing the keyboard or mouse; these things should resolve themselves once the VM is running. You should see the Debian start-up screen on a black background, wait a few seconds and then it will proceed with booting.

Figure 1: Debian boot screen

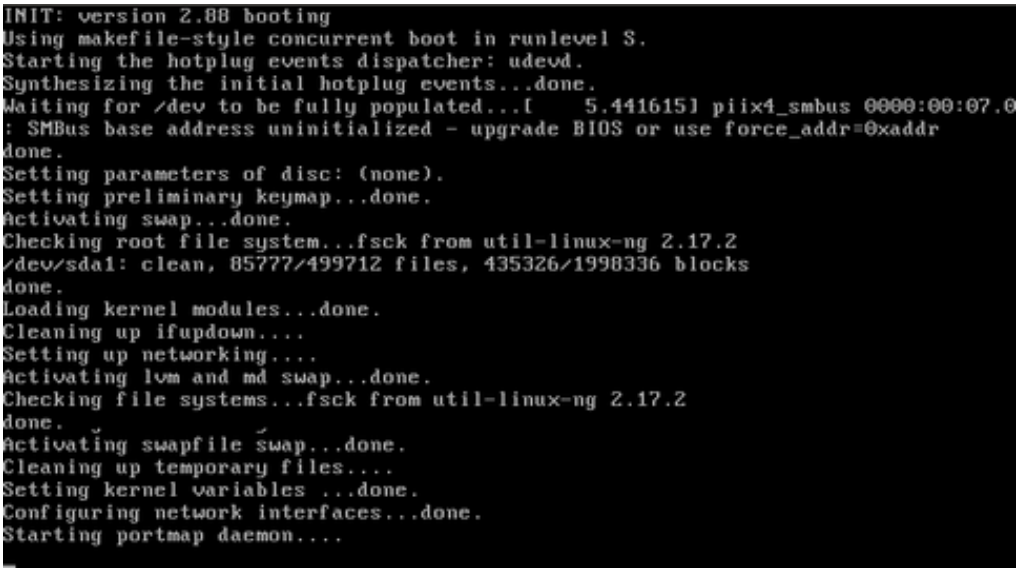

Figure 2: Debian boot process

5. Once the system finishes booting, the login screen should appear. Enter the following user credentials (passwords and user names are case-sensitive):

   ```
   Username: liucs
   Password: LIUcs!@
   ```

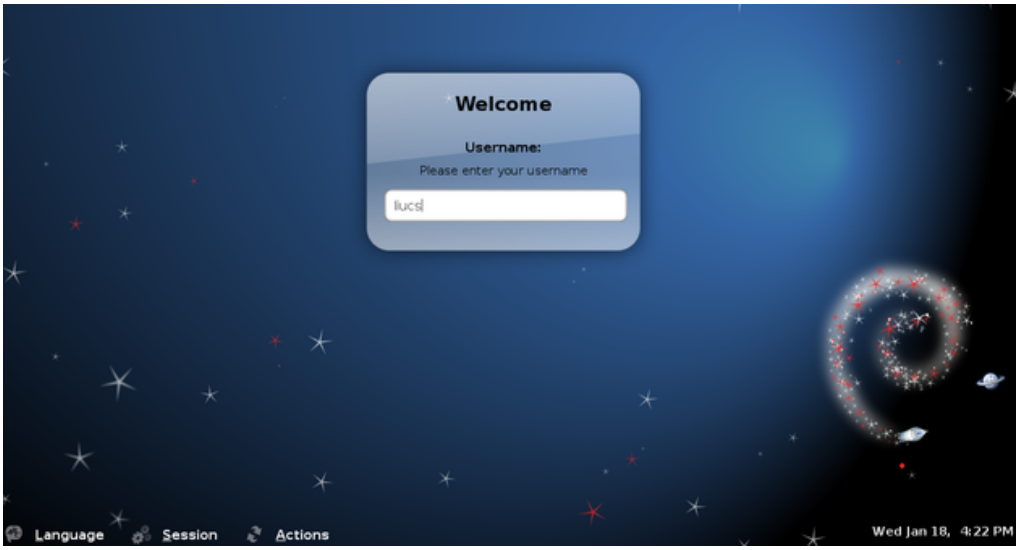

Figure 3: Debian login screen

6. After a successful login, you should see the Debian desktop. The buttons across the lower left (highlighted yellow in the image below) are an application menu, account-setup tool, terminal, editor, file manager, and web browser.
   If your screen has scroll bars or is too small, you can try to select **Auto-resize Guest Display** from the **Machine** menu at the top. Your Debian desktop should match the size of the VM window in which it appears.
7. Make sure that networking works by using the browser (within the VM, not on the host!) to do a Google search. **If the network seems to be down** (but is working on your physical host machine) then you may have to look for the VirtualBox networking setting (right-click on the two-screen icon in the lower right, highlighted green above) and change it from NAT to Bridged. Sometimes it can also help to right-click the Linux network icon (highlighted red above) and reconnect to the *wired* network. (Even if your host machine's connection is wireless, it will show up to the virtual machine as wired.)

Set up your account

1. You should have received an activation code by email to your LIU address. Find that, and then click the "Set up account" button next to the menu in the lower left. It should open up a terminal.
2. Type in your activation code, and follow the other prompts. The entire process should go something like this:

Figure 4: Debian desktop

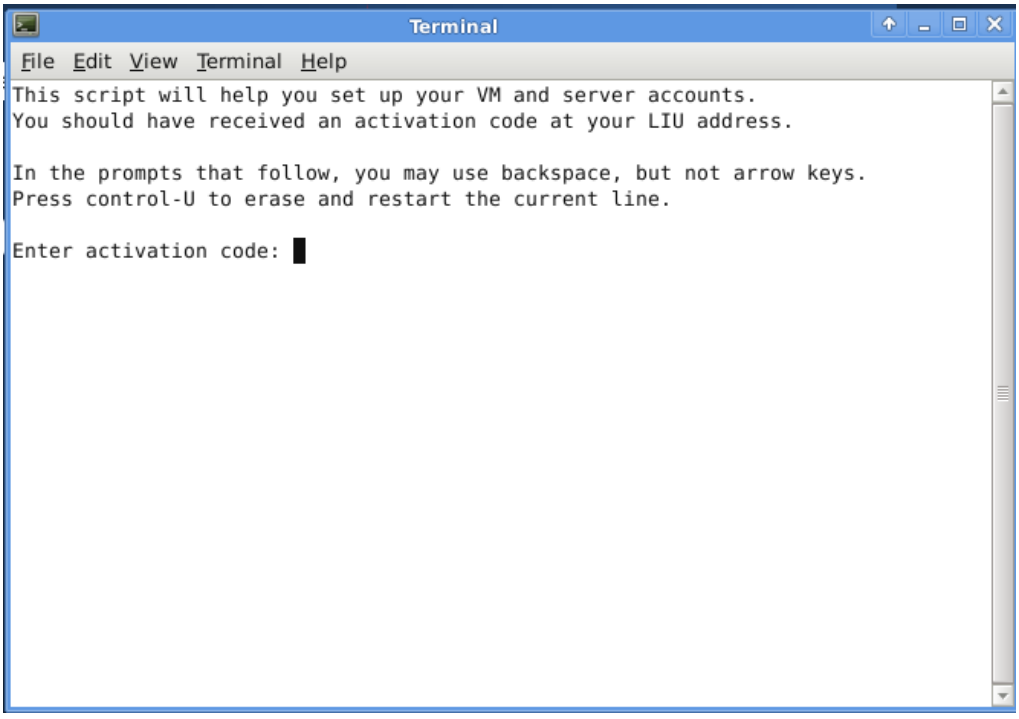

Figure 5: Set up account

```
This script will help you set up your VM and server accounts.
You should have received an activation code at your LIU address.

In the prompts that follow, you may use backspace, but not arrow keys.
Press control-U to erase and restart the current line.

Enter activation code: 0123-4567-89ab-cdef   ## (sample)

Enter your email address (does not need to be LIU): league@acm.org

Fetching key...
OK

Enter your full name: Chris League

1. Storing your SSH private key
2. Writing git config
3. Writing SSH config
4. Cloning your repositories

> git clone liucs.net:cs162-leaguec.git cs162
Cloning into cs162...
remote: Counting objects: 518, done.
remote: Compressing objects: 100% (499/499), done.
remote: Total 518 (delta 218), reused 0 (delta 0)
Receiving objects: 100% (518/518), 5.60 MiB | 704 KiB/s, done.
Resolving deltas: 100% (218/218), done.

Success! Your account name is: leaguec
Press enter to close this window.
```

3. Open up the File Manager application. In your home folder, you should see a new folder called `cs162`. (I have other folders in my account due to other courses I'm teaching.)
4. Descend into `cs162` and find the `README` file there. Double-click it to open it with the editor application called "gedit."
5. Type your name where indicated in the file, and then save it (Control-S). Next we will try to synchronize the file with the server. In the future, this is how you will submit your assignments and receive code and other resources from me.
6. In the gedit Tools menu, under External Tools, select "Sync with git," or press Shift-F9. In the "Shell Output" pane at the bottom of gedit, you should see a transcript of the operation. It will look something like this, although the numbers will vary:

```
Running tool: Sync with git
```

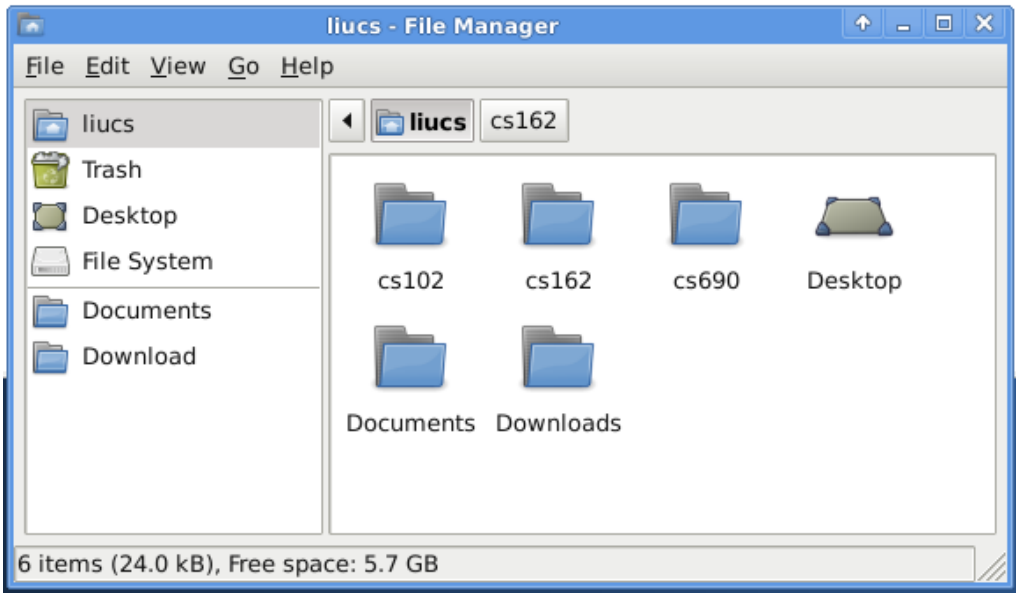

Figure 6: Home folder

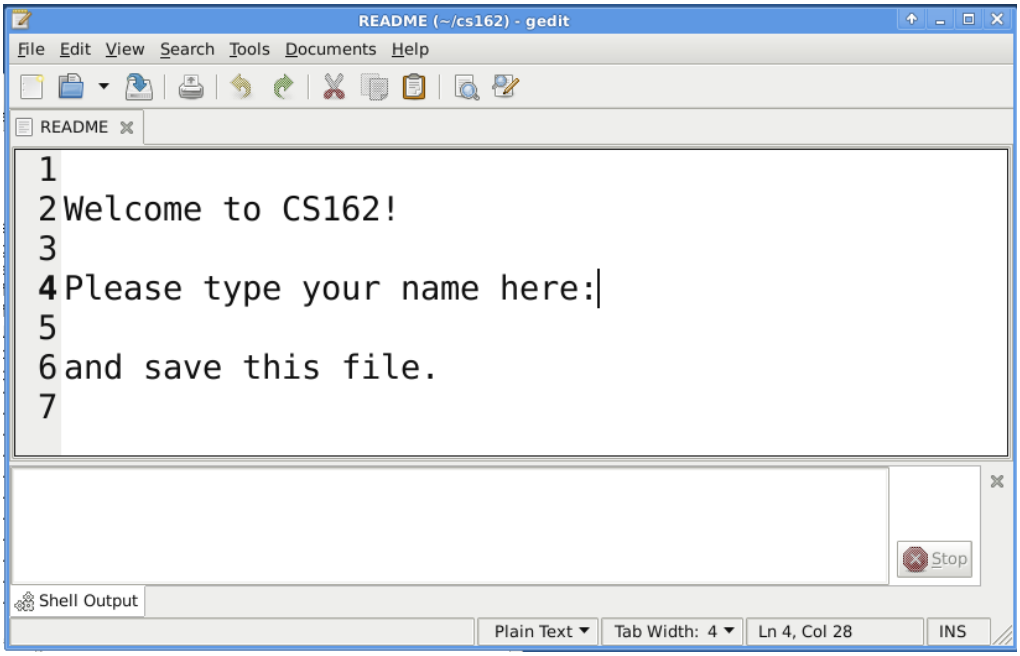

Figure 7: The README file in gedit

```
Using ~/cs162/.git
> git-add
> git-commit
[master 90a5c61] sync
 1 files changed, 0 insertions(+), 2 deletions(-)
> git-pull
Already up-to-date.
> git-push
To liucs.net:cs162-student.git
   65446eb..90a5c61  master -> master

SUCCESS

Done.
```

The important part is that it ends with "SUCCESS." If you see a different message, copy and paste the whole transcript in an email to me, so I can help you troubleshoot.

7. Check your email at the address you gave when setting up your account. You should receive a message from "git version control" containing a summary of the changes you made. Think of this email as your receipt!

Uninformed search

Build and run the code

1. In the file manager, descend into `cs162/a1` and open `dfs.cpp`.
2. From the gedit Tools menu, under External Tools, select "Run," or press Alt-F5. If there are errors building the programs, they will appear in the "Shell Output" pane at the bottom of gedit. If it succeeds, it should open up a new terminal window with the program output. You may have to scroll up or resize the terminal window to see all of it. If you like, you can also select and copy the output to your clipboard using the terminal's Edit menu.

Understand given graph

1. From within gedit, select File » Open, or use the toolbar button. Select `sample.h` and `sample.cpp`, also from the `a1` folder. They should open in additional tabs in gedit.
2. The code in `sample.h` specifies the node number of the root (start) node, the goal node, and the maximum number of nodes. (If there are 10 nodes, they must be numbered 0–9):

```
const int ROOT = 0;
const int GOAL = 7;
const int MAX = 10;
```

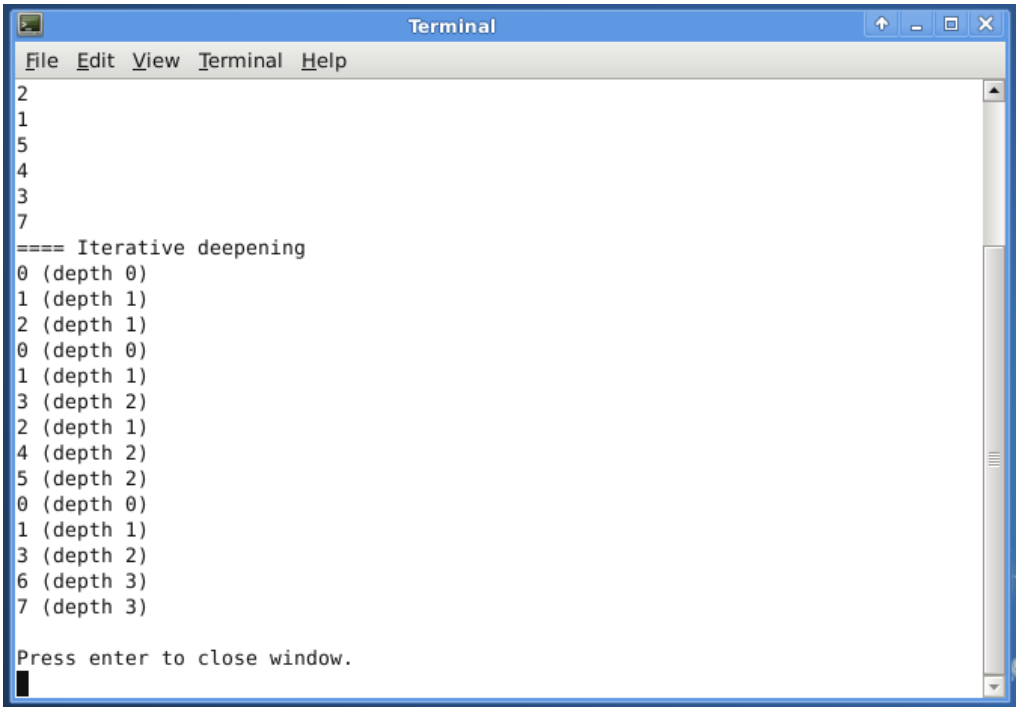

Figure 8: Result of uninformed search programs

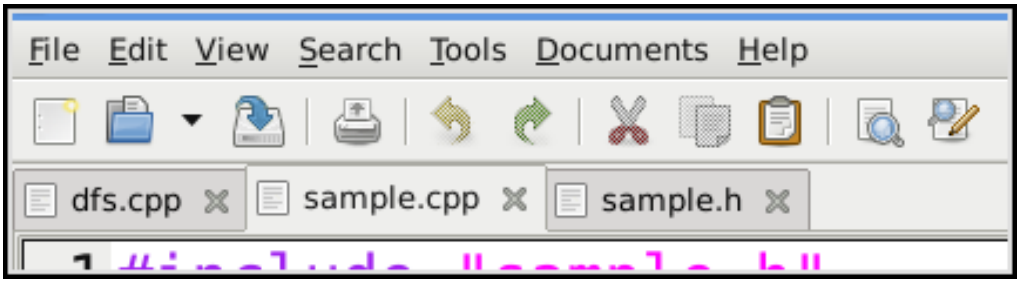

Figure 9: Tabs of open files

3. The code in `sample.cpp` specifies the edges in the graph. Specifically, `addEdge` takes four arguments: the graph, the source node, the destination node, and a 1 to add the edge (or a 0 to remove it):

```
int init_graph( graph_t *g_p )
{
  addEdge( g_p, 0, 1, 1 );
  addEdge( g_p, 0, 2, 1 );
  addEdge( g_p, 1, 3, 1 );
  addEdge( g_p, 2, 4, 1 );
  addEdge( g_p, 2, 5, 1 );
  addEdge( g_p, 3, 6, 1 );
  addEdge( g_p, 3, 7, 1 );
  return 0;
}
```

Below is a diagram representation of this graph. Remember, physical position of nodes don't matter, only the connections between them.

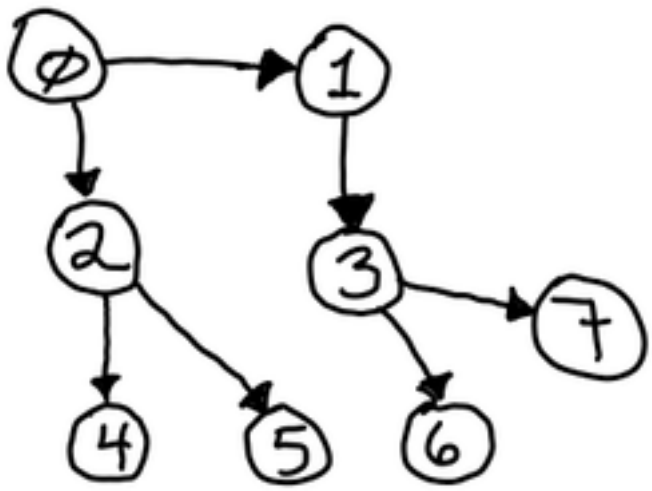

Figure 10: Graph 1

4. Now examine again the output from running the searches. Depth-first search worked like this:

```
visit   stack
  0       2 1
  1       2 3
  3       2 7 6
  6       2 7
  7       2
GOAL.
```

where the stack is built from left to right, but consumed from right to left (LIFO). There is one difference in how these programs work compared to how we did it in class: nodes are inserted in *reverse numerical order.* As you can see, when we visit node 0, we push 2 and *then* 1 onto the stack.

5. Breadth-first search worked like this:

```
visit    queue
  0        2 1
  2        1 5 4
  1        5 4 3
  5        4 3
  4        3
  3        7 6
  7        6
GOAL!
```

where the queue is built from left to right, and consumed from left to right (FIFO). Again, we add elements to the queue in *reverse numerical order.*.

Build a new graph

1. In `sample.cpp`, modify the `init_graph` function so that the `addEdge` statements build the following graph, instead of the original one:

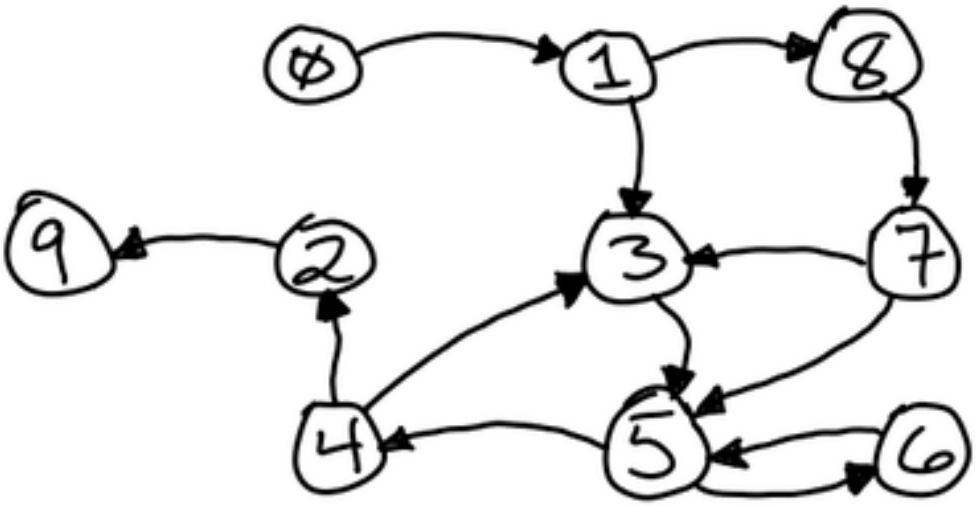

Figure 11: Graph 2

2. Set the `GOAL` in `sample.h` to 9.
3. Open `answers.txt` in the editor.
4. Trace the sequence of visited nodes and stack contents for DFS with this new graph, as I did above. Write your answers in `answers.txt` .

5. Trace the sequence of visited nodes and queue contents for BFS with this graph. Write your answers in `answers.txt`.
6. As part of iterative deepening, we run a *depth-limited search*, which is DFS but we do not push any nodes beyond the depth limit. Trace a depth-limited search up to the maximum depth of 5. Again, write your answers in `answers.txt`.
7. Verify that the sequence of visited nodes matches those produced by the program with your new graph in `sample.cpp`.

Submit everything

Just use "Sync with git" from the Tools » External Tools menu, as before.