Lab 0: Setting up the Environment

In order to ensure a uniform programming environment for each student in this class, you are required to test and run your solutions for each programming lab on `eceprog` machines. This is important because we will test your solutions only on `eceprog` machines. Hence if your code does not run on `eceprog`, you will not receive any credit, regardless of whether your code runs on your personal machine or any other machine.

1 Accessing `eceprog` Machines

You can access `eceprog` machines in two ways:

1.1 ThinLinc client application

1. Download and install ThinLinc client software on your computer.
2. Start ThinLinc.
3. Enter `desktop.eceprog.ecn.purdue.edu` as the server name.
4. Log in with your Purdue username and password. Note that these clusters might be using DUO multi-factor authentication. If so, use **password,push** as your password to receive an authentication request on your DUO app.
5. When using a computer off-campus, you may first need to connect to Purdue's VPN. The instructions to install and establish the VPN connection can be found here:

   MacOS: https://engineering.purdue.edu/ECN/Support/KB/Docs/WebVPNmac

   Windows: https://engineering.purdue.edu/ECN/Support/KB/Docs/WebVPNforWindows

1.2 ThinLinc web interface

1. Go to https://desktop.eceprog.ecn.purdue.edu
2. Log in with your Purdue username and password (password,push if DUO multi-factor authentication is enabled).

2 `gcc` and `python3` versions

Please make sure that the `gcc` version on your `eceprog` machine is **11.4.0** and the `python3` version is **3.10.12**. You can check the versions of these software by running the following commands in a terminal in your `eceprog` machine:

```
$ gcc --version

$ python3 --version
```

3 Software Development Environment (SDE)

While you are required to run and test your code on `eceprog` machines, you are free to use any machine and any SDE of your choice to write your code. Below we suggest some popular options.

3.1 SSH-capable terminal

Alternatively, you can log into the `eceprog` machine using a SSH-capable terminal from any remote machine (including your personal machine). Mac and linux terminals are SSH-capable by default. For Windows, you can get a SSH-capable terminal by installing Git (https://git-scm.com/downloads). This will install the SSH-capable terminal at the location "`C:\Program Files\Git\bin\bash.exe`".

```
$ ssh [username]@eceprog.ecn.purdue.edu
```

Use your Purdue username and password (password,push if DUO multi-factor authentication is enabled) to log in. Once logged in, you can use terminal editors such as **Vim** and **Emacs** to write code.

3.2 `eceprog` GUI

You can write your code directly inside the `eceprog` machine using its default text editor. However, you might be limited in terms of the SDE choices on these machines.

3.3 SDE on your personal machine

Alternatively, you can code on your personal machine using the SDE of your choice. However, in that case you will need to synchronize the code on your personal machine with the code on `eceprog`, every time you want to run and test your code. You can do it using one of the following ways:

3.3.1 Synchronize using version control

You can use a version control system, such as **git**, to synchronize your code.

1. Create a git repository and copy the lab files into the repository. **Make the repository "Private"**.
2. Clone the repository on both your personal machine and `eceprog` machine, using `git clone` command.
3. `git push` on your personal machine every time you make a code change. And `git pull` on the `eceprog` machine to pull the latest code changes.

3.3.2 Synchronize using `rsync`

1. Create a directory on `eceprog` where you will store all your lab files. On a terminal in `eceprog`,

   ```
   $ cd ~
   ```

   ```
   $ mkdir workdir
   ```

   (we use workdir in example, choose the name you feel appropriate).

   Move all files and directories you need to synchronize into `workdir`.

2. Configure key-based ssh authentication (to not have to enter your password every time you synchronize): make ssh key pair on your personal machine, and copy the public key into the file `~/.ssh/authorized_keys` on `eceprog`. If the above file does not exist, first create one and then copy the public key to the file.

3. Now you can synchronize the content of your `workdir` from `eceprog` to your personal machine. On your personal machine, open a SSH-capable terminal and execute `rsync` command:

   ```
   $ rsync -avuz [username]@eceprog.ecn.purdue.edu:~/workdir .
   ```

Note the dot "." - it is the second operand which specifies where on your personal machine to rsync. dot stands for current directory. This will create the directory `workdir` in the current directory of your personal machine.

4. After adding, deleting, or making changes to some files on your personal machine, you need to synchronize these to `eceprog` as follows:

   ```
   $ cd /directory/that/contains/workdir
   ```

   ```
   $ rsync -avuz workdir [username]@eceprog.ecn.purdue.edu:~/
   ```

   Note, that only newly added and changed files will be transferred to `eceprog` by `rsync` command. If you change the same file on your personal machine and on `eceprog` and do not synchronize between changes, the file with latest timestamp will win. If you want to keep older versions of files you can add "-b" (for "backup") option to rsync command:

   ```
   $ rsync -avuzb ......
   ```

   Older versions of files will be preserved with appended "~" symbols to filenames.

3.3.3 Synchronize using shared folder

Perhaps the most convenient option to synchronize files between your personal machine and `eceprog` is to mount your home directory in `eceprog` to your local machine. This way you will have access to all the files on your `eceprog` machine locally. You can add/delete/modify any of the files inside the mounted directory locally, and it will be automatically synchronized with `eceprog` over the network.

NOTE: If you are outside the Purdue's network, you will need a VPN connection to mount the directory. The instructions to install and establish the VPN connection are described in Section 1.

To mount your home directory in `eceprog` to your local machine, follow the following instructions:

1. If your local machine is running **MacOS**, please try to follow the article:
   https://engineering.purdue.edu/ECN/Support/KB/Docs/MacOSXConnectingToSMB

   For the server address field, please use the following address:

   ```
   smb://[servername].ecn.purdue.edu/[username]
   ```

   Here, `username` is your Purdue username.
   `servername` can be found as follows:

   On `eceprog`, open a terminal and execute the following command:

   ```
   cat /etc/passwd | grep [username]
   ```

   e.g. for username `vshriva`, the command will be:

   ```
   cat /etc/passwd | grep vshriva
   ```

   The command returns the line:

   ```
   vshriva:x:733637:5000:Vishal Shrivastav:/home/dynamo/a/vshriva:/bin/bash
   ```

   Here, "`dynamo`" is the server name.
   For undergrads, the server name is typically "`shay`".

2. If your local machine is running **Windows**, please try to follow the article:
   https://engineering.purdue.edu/ECN/Support/KB/Docs/MappingECNDriveWin8

   For the Folder field, please use the following address:

   ```
   \\[servername].ecn.purdue.edu\[username]
   ```

Here, `username` is your Purdue username and `servername` can be found in the same manner as described in Point 1 above.

From non-domain machine then there will be step 4c, in which when you are asked in authentication pop-up username/password, please use username:

`boilerad\[username]` (here again, the `username` is your Purdue username)

3. If your local machine is running **Linux (Ubuntu)**, please follow the below steps:
   (a) Open the "Files" app and click on "Other Locations"
   (b) In the Connect to Server field, enter the following address and click Connect
   `smb://[servername].ecn.purdue.edu/[username]`
   Here, `username` is your Purdue username and `servername` can be found in the same manner as described in Point 1 above.
   (c) Enter the following options for password security and click Connect
   Connect As: Registered User
   Username: Purdue username
   Domain: ECN
   Password: Purdue password, not boilerkey

4 Task

Once you have set up your `eceprog` environment, you can test it using the following task. Download the Lab 0 files from Brightspace either directly on your `eceprog` machine, or download it on your personal machine and synchronize it with `eceprog` using one of the options from Section 3.3.

4.1 Testing the socket interface

Socket interface is the most popular interface for writing network applications. You will use it extensively in Labs 1 and 2. But for now, you are only required to test that the socket interface is working on your `eceprog` machine. For that, we have provided you with two sample socket programs to test the two common socket interface protocols, namely TCP and UDP. The code can be found in the folder "`Lab0/sample-socket-code/`".

4.1.1 Testing UDP socket

The first application is a DNS application using UDP — client sends the domain name to server and server returns back IP(s).

To run the program, open two terminals on `eceprog`. On the first terminal, run:

```
$ cd Lab0/sample-socket-code
```

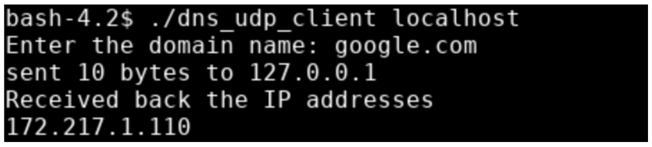
`$ ./compile.sh` (first run `$ chmod 755 compile.sh` if you are getting "Permission denied" error)

```
$ ./dns_udp_server
```

On the second terminal, run:

```
$ cd Lab0/sample-socket-code
$ ./dns_udp_client localhost
```

This will result in a prompt on the second terminal, asking for domain name. Assuming you enter `google.com`, the output on the second terminal would look something like below:



```
bash-4.2$ ./dns_udp_client localhost
Enter the domain name: google.com
sent 10 bytes to 127.0.0.1
Received back the IP addresses
172.217.1.110
```

4.1.2 Testing TCP socket

The second application is a file transfer application over TCP — server transfers an image file to client, which is displayed on the terminal.

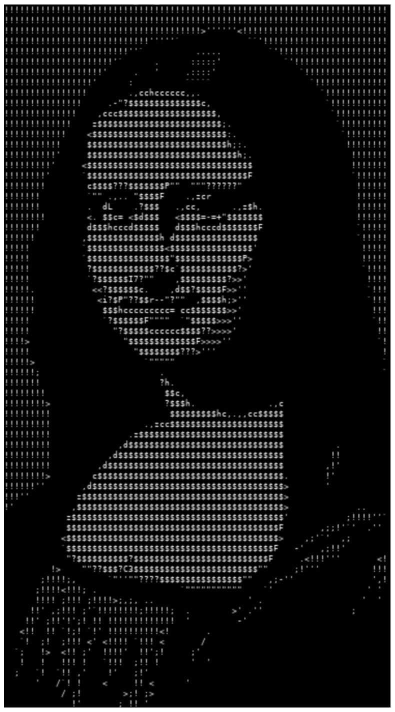
To run the program, open two terminals on `eceprog`. On the first terminal, run:

```
$ cd Lab0/sample-socket-code
$ ./compile.sh
$ ./file_transfer_tcp_server
```

On the second terminal, run:

```
$ cd Lab0/sample-socket-code
$ ./file_transfer_tcp_client localhost
```

This should produce the following output on the second terminal.



5 Submission

No submission is required for this lab.