

# ENGRECE 12: Assignment-1

January 8, 2004

Due Wednesday 21 Jan 2004 at 12:00pm

## 1 Login to your Unix account

For this class, you will be doing your assignments by *logging on* to a shared machine (server) running the Unix operating system. Even though you may be using a personal computer or a workstation that is capable of computation locally, you will mainly be using them as *terminals* (clients), whose job is to pass keystrokes to the server and display outputs from the server.

To use a shared machine, first you need an *account* on the machine. You need to get your username and password for the ECE account from a TA if you don't already have them. DO NOT contact NACS directly. We are NOT using NACS unix machines for ECE 12. You must go through the ECE 12 head TA (ece12@ece.uci.edu) if you have not received your user name / password by now.

The name of the instructional server is east.ece.uci.edu. You can log in to your account with your user name and password. Your account also comes with a certain amount of disk space. You can use this space to store homework assignment files, and you don't need to bring your own disks or other storage media.

If east.ece.uci.edu is down, then you can try another machine, such as malibu.ece.uci.edu, newport.ece.uci.edu, etc. You can use the same user name and password, and your files will be the same. However, these are not designated to be instructional servers, and you should avoid using them unless absolutely necessary.

### 1.1 Software and commands for remote login

You can connect to east.ece.uci.edu from virtually any computer anywhere that has internet access. What you need is a client program for *remote login*.

Previously, people used **rlogin** or **telnet** to connect to the server, and **ftp** or **rcp** to transfer files. However, they are insecure, because your keystrokes or output are in clear text and can be *snooped* by others. This means your account name and password can be stolen this way. So, for security reasons, do not use either of these programs.

Instead, use **ssh** as the primary way to connect to the server. **ssh** stands for *secure shell*, and it encrypts your network communication, so that your data looks like garbage to snoopers. For file transfers, use **sftp** or **scp**, which are secure. You could also set up an *ssh-tunnel* so that previously unencrypted communications can be encrypted.

Depending on what computer you use, it may have a different *implementation* of **ssh**, but the basic function underneath are all the same. Check out NACS's page on SSH:

[http://www.nacs.uci.edu/support/sysadmin/ssh\\_info.html](http://www.nacs.uci.edu/support/sysadmin/ssh_info.html)

- If you are logging in from a Win32 machine, you can use **PuTTY**.
- MacOS X already has this built-in (use Terminal or X11 to run a unix shell). Most Linux distributions also bundle ssh.
- If you are logging in from an X terminal, you can use the command  
% **ssh** east.ece.uci.edu -X -l *yourUserName*  
(note: % is the prompt, not part of your command) It will prompt you for your password. Note that the -X option allows you to run programs that open X windows on your screen (e.g, Netscape, IDLE, etc)

## 1.2 Unix Shell

By now you should be logged in, and you should be looking at the prompt  
east% \_

Note: in the following writeup, we will show just  
%  
for the prompt, instead of  
east%

If you login to another machine, such as malibu, then the prompt would instead look like  
malibu%

You should change your password using the **passwd** command. The password will be changed on all the ECE Sun machines, not just on east.ece.uci.edu.

Try out the following commands at the shell prompt.

<b>ls</b>	list files
<b>cd</b>	(change working directory)
<b>pwd</b>	(print working directory)
<b>mkdir</b>	(make directory)
<b>mv</b>	(rename/move files)
<b>cp</b>	(copy files)
<b>rm</b>	(remove files)
<b>rmdir</b>	(remove directory)
<b>cat</b>	(print the content of a file)
<b>more</b>	(print the content of a file, one screen at a time)
<b>echo</b>	(print the arguments on the rest of the command line)

Most commands take one or more file names as parameters. When referring to files, you may need to qualify the file name with directory references, absolute vs. relative paths:

<b>.</b>	(current directory)
<b>..</b>	(one level higher)
<b>~</b>	(home directory)
<b>/</b>	the root (top level) directory

## 1.3 Follow the Unix Guide

Follow the unix guide at:

<http://www.nacs.uci.edu/help/manuals/uci.unix.guide/>

Learn basic shell commands: list files, change directory, rename files, move files, copy files, show file content.

There is nothing to turn in for this part.

## 2 Learn the vi text editor

Follow the **vi** guide at:

[http://www.nacs.uci.edu/doc/uci.unix.guide/the\\_vi\\_editor.html](http://www.nacs.uci.edu/doc/uci.unix.guide/the_vi_editor.html).

Learn how to edit a file, move the cursor, insert text, insert text from file, delete words, delete lines, cut/paste, save changes, save to another file, quit without saving.

There is nothing to turn in for this part. However, it is critical that you get enough practice with **vi**. It is fast and powerful and will save you a lot of time later on, compared to using a WYSIWYG text editor.

## 3 Exercise 2.3 (page 66) [10 points]

First create a subdirectory named hw1 (for homework one). Then, use **vi** to create a text file named 2.3.txt to answer the question. Do not use a word processor and transfer or paste the content. The file should state your name

and exercise number. Check your answers by typing the statements into the python prompt.

## 4 Exercise 2.5 (page 66) [10 points]

Use **vi** to create a file named `2.5.py`. Run your program by typing

```
% python 2.5.py
```

You also need to show that it works with your own test cases by turning in a typescript named `2.5.script`. For instructions on how to create a typescript, see Scripting Guide Section at the end of this notes.

## 5 Submit your work

Here is a checklist of the files you should have:

In the `hw1` directory, you should have the following files in your unix account:

- `2.3.txt`
- `2.5.py`
- `2.5.script`

You should **cd** into a level above `hw1` and type the command

```
% python ~ece12/tools/submit.pyc
```

which will guide you through the submission process.

You will be asked if you want to submit the script file. Type `yes` or `no`. If you type `"n"` or `"y"` or just plain return, they will be ignored and be taken as a `no`.

Below is an example of how you would submit your homework:

```
% ls #This step is just to make sure that you are in the correct directory that contains hw1/
hw1/
% python ~ece12/tools/submit.pyc
=====
ECE12 hw1 submission process for ece12
Due date: Wed, 21 Jan 2003 12:00:00
=====
Submit 2.3.txt[yes, no]? yes
File 2.3.txt has been submitted
Submit 2.5.py[yes, no]? yes
File 2.5.py has been submitted
Submit 2.5.script[yes, no]? yes
File 2.5.script has been submitted
=====
Summary:
=====
You just submitted file(s):
2.3.txt
2.5.py
2.5.script
east% _
```

## 6 Typescript

A typescript is a text file that captures an interactive session with the unix shell. Very often you are required to turn in a typescript to show that your program runs correctly. To create a typescript, use the **script** command. Here is an example:

- Type the command **script** into the shell. It should say  
Script started, file is typescript  
% \_  
This means it is recording every key stroke and every output character into a file named “typescript”, until you hit **^D** or type **exit**.
- Type some shell commands; you can also enter the python prompt. But don’t start a text editor!
- Stop recording the typescript by typing **exit**.  
% **exit**  
Script done, file is typescript  
% \_
- Now you should have a text file named `typescript`. Make sure it looks correct.  
% **more** typescript  
Script started on Thu Sep 25 23:43:56 2003  
...  
...

You should immediately rename the typescript to another file name. Otherwise, if you run **script** again, it will overwrite the typescript file.

Note: If you backspace while in script, it will show the **^H** (control-H) character in your typescript. This is normal. If you use **more** to view the typescript, then it should look normal.