# EECS 10: Assignment 1

September 26, 2008

Due Monday 6 Oct 2008 at 12:00pm (noon)

## 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. EECS support has created an *account* for each student. To retrieve the username and password go to the following website:
`https://newport.eecs.uci.edu/account.py`.
The website asks for your UCInetID and the according password before giving you the account information of your new EECS account. Note that your browser may also ask you to accept a certificate to open the secure website. DO NOT contact NACS directly. We are NOT using NACS unix machines for EECS 10. If you have a problem please contact your EECS 10 TA, (eecs10@eecs.uci.edu).

The name of the instructional server is `malibu.eecs.uci.edu.` You can log into your account with your EECS 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 `malibu.eecs.uci.edu` is down, then you can try another machine, such as `vivian.eecs.uci.edu`, `newport.eecs.uci.edu`, or `east.eecs.uci.edu`. You can use the same user name and password regardless of the machine, and your files will be the same.

### 1.1 Software and commands for remote login

You can connect to `malibu.eecs.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, these protocols 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 cannot be understood by 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`
or check the course web site:
`http://eee.uci.edu/08f/18010/resources.html`

- If you are logging in from a Windows 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 malibu.eecs.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.

## 1.2 Unix Shell

By now you should be logged in, and you should be looking at the prompt
`malibu% _`

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

If you login to another machine, such as `vivian`, then the prompt would instead look like
`vivian%`
You should change your password using the **`passwd`** command. The password will be changed on all the EECS Sun machines, not just on `malibu.eecs.uci.edu`.

Try out the following commands at the shell prompt (See reference to the Unix Guide in section 1.3 for more details about these commands.).

| | |
|---|---|
| **`ls`** | list files |
| **`cd`** | (change working directory) |
| **`pwd`** | (print working directory) |
| **`mkdir`** | (make directory) |
| **`mv`** | (rename/move files) |
| **`cp`** | (copy files) |
| **`rm`** | (remove files) |
| **`rmdir`** | (remove directory) |
| **`cat`** | (print the content of a file) |
| **`more`** | (print the content of a file, one screen at a time) |
| **`echo`** | (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:

| | |
|---|---|
| `.` | (current directory) |
| `..` | (one level higher) |
| `~` | (home directory) |
| `/` | 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 to use a text editor

There are three editors that are available on nearly all unix systems that you may choose from.
**`pico`** is the easiest to get started with. A guide for **`pico`** can be found at:
`http://www.dur.ac.uk/resources/its/info/guides/17Pico.pdf`.
**`vi`** is a very powerful editor, but is arguably a bit more difficult to learn. Follow the **`vi`** guide at:
`http://www.nacs.uci.edu/help/manuals/uci.unix.guide/the_vi_editor.html`.

Finally, **emacs** is another editor that you may use. **emacs** is also a powerful editor, but is a bit easier to learn than **vi**. Follow the **emacs** guide at:
`http://www.nacs.uci.edu/help/manuals/uci.unix.guide/editing_with_gnu_emacs.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 your editor, so that you can do the homeworks for this class.

## 3 Exercise 2.25 (text book, page 60) [20 points]

First create a subdirectory named `hw1` (for homework one). Change into the created directory `hw1`. Then, use your editor to create a C file named `initials.c`. Do not use a word processor and transfer or paste the content. The C file should state your name and exercise number as a comment at the top of the file. Write the C program according to exercise 2.25 in the text book.

### 3.1 Compiling your code

To test your program, it must be compiled with the **gcc** command. This command will report any errors in your code. To call **gcc**, use the following template:

```
% gcc -o targetfile sourcefile
```

Then, simply execute the compiled file by typing the following:

```
% ./targetfile
```

Below is an example of how you would compile and execute the excersise 2.25:

```
% gcc -o initials initials.c
% ./initials
program executes
malibu% _
```

A brief text file, initials.txt, must be submitted as well that explains what the program does and why you chose your method of implementation. For this homework a single sentence should be sufficient.

You also need to show that it works with your own test cases by turning in a typescript named `initials.script`. For instructions on how to create a typescript, see Section 5 Typescript at the end of this document.

## 4 Submit your work

To submit your work, you have to be logged in to one of the following servers `malibu`, `vivian`, or `east`. The submission will not work on the server `newport`.

Here is a checklist of the files you should have:

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

- `initials.c`
- `initials.txt`
- `initials.script`

We do require these *exact* file names (i.e. do *not* replace initials with your actual initials). If you use different file names, we will not see your files for grading. Now, you should change the current directory to the directory containing the `hw1` directory. Then type the command:

```
%  /ecelib/bin/turnin
```

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. You can use the same command to update your submitted files until the submission deadline.

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/
% /ecelib/bin/turnin
=========================================
EECS 10 Fall 2008:
Assignment "hw1" submission for hschirne
Due date:  Mon Oct.  6 11:59:59 2008
=========================================
Submit initials.c [yes, no]?  yes
File initials.c has been submitted
Submit initials.txt [yes, no]?  yes
File initials.txt has been submitted
Submit initials.script [yes, no]?  yes
File initials.script has been submitted
=========================================
Summary:
=========================================
You just submitted file(s):
initials.c
initials.txt
initials.script
malibu% _
```

### 4.1 Verify your submission

This step is optional, but recommended. If you want to confirm which files you have submitted, call the following command:

```
% /users/ugrad/2004/fall/eecs10/bin/listfiles.py
```

This command lists your submitted files. Don't worry if you submitted too many files. We will only look at the files with defined names (here: `initials.c`, `initials.txt` and `initials.script`) and ignore other files.

## 5 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. 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 Fri Sep 26 14:32:00 2008
...
...
```

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.