

Before You Begin ...

(by Han-Shen Yuan, with updates by Mike Clancy)

Overview

This document provides information about computing resources used in the self-paced programming courses. It introduces those unfamiliar with the EECS computing facilities to their location, access, and use, and provides information on how to get started in CS 3S or a CS 9 or 47 series course, how to run and test programs, and how to print work to be evaluated by tutors.

Computer platforms used with CS 3S and the CS 9 and 47 Courses

We recommend that your coursework for CS 3S and the CS 9 and 47 courses be done on the EECS UNIX-based computers. Important course information is provided through these systems as well as supplemental documentation necessary to complete many of the CS 3S, CS 9, and 47 series programming assignments. In addition, all of the software necessary to complete the CS 3S, CS 9, and 47 courses is already installed on these computers, so you don't have to do it yourself.

The UNIX operating system

The computers used in conjunction with the CS 9 and 47 curricula run an operating system (a program that manages user interaction with a computer) named UNIX. Unlike popular operating systems such as MacOS or Windows 95 used on today's personal computers, UNIX is text-based; i.e. all of the commands issued to the operating system are in the form of typed commands as opposed to mouse movements. However, as with other popular operating systems, users may create and edit files, launch programs, and manage the contents of their directories.

For an introduction to basic UNIX commands, see the section entitled "Basic UNIX commands".

Your CS 3S, CS 9, or 47 Account

Each student taking a course in CS 3S or the CS 9 or 47 series is given a computer account with which to access the EECS workstations. A computer account provides a set amount of disk space set aside on a central computer for you to use when working on your coursework. You access this account either directly by physically sitting at a workstation or through the Internet via a personal computer (see the section entitled "Accessing workstations—Points of access"). You get a user name and password so you may verify your identity when using your account. Your account will be active only for the current term.

You should change the password that is used to access your account as soon as possible for security purposes. This procedure is outlined in "Accessing workstations—logging in and changing passwords".

Accessing workstations

Names and locations of workstations

As of Fall 2005, there are three primary facilities where students may access EECS instructional workstations running UNIX:

labs on the second floor of Soda Hall;
199 Cory Hall;
C30 and C50 Hearst Field Annex

Lab sections for CS 3 are run in C30 Hearst Field Annex; lab sections for CS 61A are run in C50 Hearst Field Annex; and lab sections for CS 61B and 61C are run in the Soda rooms. During each lab section, these rooms may not be used by students other than those enrolled in the section. CS 3S students are encouraged to work in the Hearst Field Annex rooms when lab sections are not in progress.

Because of the limited number of machines, it is often necessary to access workstations remotely (see "Points of Access" below). For an up-to-date list of available workstations, type clients at the command prompt (for more information on the command prompt, see "Logging in and changing passwords" below) or check the Web site

<http://inst.eecs.berkeley.edu/labs.html>

A number of computers are especially intended for remote access (add .berkeley.edu to each name):

cory.eecs	c199.eecs	po.eecs	h30.cs	h50.cs
nova.cs	star.cs	pulsar.cs	solar.cs	quasar.cs
torus.cs	rhombus.cs	sphere.cs	pentagon.cs	cube.cs

Points of access

At home using a computer and modem

To access EECS instructional accounts from outside Soda or Cory Hall, you need either a computer attached to the Internet or a modem attached to your computer with which you may connect by phone. In either case, you also need secure software with which to connect to your account; this can be downloaded via the Web site

<http://inst.eecs.berkeley.edu/connecting.html>

This Web site also explains the rationale for secure connection software and provides some information about how the software works.

At a campus microcomputer facility

Microcomputer facilities located on campus outside of Cory and Soda should also provide the ssh program needed for secure connections to EECS computers.

At a workstation

Perhaps the easiest way to access one's CS 3S, CS 9, or 47 series account is to work at one of the EECS workstations. Self-paced students are allowed to use workstations in 199 Cory, Hearst Field Annex, and in all rooms on the second floor of Soda

Hall during daytime hours when lab sections are not in session. (These rooms are sometimes crowded.) Students who wish to use the labs outside regular building hours—7:30am-6:30pm, Monday through Friday—may acquire a card key for evening access at the CS Division main office on the third floor of Soda. Self-paced students may get cardkey access to the labs on the second floor of Soda.

Empty workstations automatically have a “login:” prompt displayed, so no additional steps are required.

Logging in and changing passwords

The “login:” prompt is a request by the computer for your login name as indicated on your account form. After you have entered your account name (typed exactly as indicated on your account form), you are then prompted for your “password.” To complete the login process, you need only type the password indicated on your account form into the computer. If all goes well, a bulletin of recent course-related news should appear followed by a prompt (the machine name you logged into, followed by a %). This prompt is referred to as a *command prompt*, signalling that you may issue commands to the computer.

It is important to recognize that computers can be quite unforgiving when it comes to typos. A password such as “b812*” is not the same as “B8128”. If the computer indicates that a login is “invalid”, one of two things may have occurred: either you made a mistake typing your login name, or you made a mistake typing your password.

To prevent illegal access to class accounts, you should change your passwords frequently. To do this, you will need to perform the following steps:

1. Type `ssh po.eecs.berkeley.edu`
2. Log in again as described in this section.
3. Type `passwd`

After step 3, the system will prompt you for your old password and new password. **IMPORTANT:** It takes a few minutes for your new password to be recorded.

Basic UNIX commands

This section describes a list of commonly used commands that are available on the EECS UNIX systems. Most commands are executed by simply typing the name of the command at the command prompt (*machine_name%*) with the relevant arguments and hitting return. For the purposes of this document the [] notation refers to an argument that the user supplies depending on the situation. Most of the commands that are described in this section are substantially more complicated than the explanation given. See the relevant online manual pages for details (accessing them with the `man` command is covered later in this section).

Directory commands

Because the UNIX system uses a command-line (as opposed to graphical) interface, its conventions for accessing files are different from those of personal computing environments. The following is a list of definitions used in this section's discussion of directory commands:

current directory — This is the directory (a “folder” in Windows or MacOS) that you are currently working in. In many graphical operating systems, a user can “dig” through the directory structure by navigating through a series of windows. The current directory is equivalent to the topmost active window in a graphical operating system that is displaying the contents of a directory.

home directory — When you first log in to the system, the home directory is the first directory that is set as your current directory.

path — In addition to your files, each server stores hundreds of other users' files, each with its own directory within the storage device. Since directories are usually nested in other directories, the UNIX operating system uses a *path*, or series of directories, when referring to files. For example, if relative to one's current directory there exists a directory named “A” and within “A” there existed a directory named “B” then the path would be A/B. Using the `cd` command (as described below), B can be set as the current directory by simply typing

```
cd A/B
```

directory nested above — In UNIX, the directory nested above the current directory is referred to by “..”. Thus, using the `cd` command (as described below), the command

```
cd ..
```

would set the current directory to the directory that is nested above the current directory.

pwd

The `pwd` command displays the path of the current directory starting from the very root level of the hard drive to the current directory's location in its nested hierarchy.

- To display the path of the current working directory, type

```
pwd
```

A path should then appear on the screen.

ls

The `ls` command lists the contents of the current directory or, optionally, the contents of the directory the user chooses to specify. This operation is similar to double-clicking on a Mac or Windows “folder” and viewing its contents.

- To view the contents of the current directory, type

```
ls
```

A listing of files in the current directory should appear.

- To view the contents of a directory that is not the current directory, type

```
ls [path of directory]
```

(Note: don't type the brackets, just the path.) A listing of files in the specified directory should appear.

cd

The `cd` command changes the current directory to another directory specified or simply sets the current directory to the user's home directory.

- To change the current directory to another directory, type

```
cd [path of directory]
```

(Note: don't type the brackets, just the path.) The current directory should now be changed.

- To change the current directory to your home directory, type

```
cd
```

The current directory should now be your home directory.

mkdir

The `mkdir` command creates a new directory. This is similar to selecting "New Folder" from the File menu on Macintosh and Windows systems.

- To create a directory inside the current directory, type

```
mkdir [name of new directory]
```

(Note: don't type the brackets, just the path.) A new directory with the name [name of new directory] should now exist within the current directory.

cp

The `cp` command is used to make a copy of a file. This is similar to highlighting a file on a Macintosh or Windows system, and selecting "Duplicate" from the File menu.

- To create a copy of a file, set the directory of where the file resides in as the current directory, then type

```
cp [name of file] [name of copy]
```

mv

The `mv` command is used to either rename an existing file or folder to something different or to "move" a file from one directory to another (an operation similar to dragging a file from one directory to another on a Macintosh or Windows system).

- To rename an existing file, type

```
mv [name of existing file] [new name for existing file]
```

- To move an existing file, type

```
mv [name of existing file] [path of new location]
```

rm

The `rm` command is used to delete a file. This operation is similar to dragging an item to the Trash (or Recycle Bin) and Emptying it. Files that are “rm’d” cannot easily be recovered (though sometimes the system administrator may have a backup). Use this command with caution.

- To remove an existing file, type

```
rm [name of file]
```

rmdir

The `rmdir` command is a separate command for removing directories. To remove a directory, the directory must be empty.

- To remove an existing directory, type

```
rmdir [name of directory]
```

UNIX online manual

Despite the seemingly user-hostile nature of UNIX, the UNIX system does provide lots of documentation online. As mentioned before, almost all of the commands listed in the previous section are more complicated than the explanation given suggests. There are two situations when you may want to use the online documentation:

- You want to accomplish “something” but you don’t know precisely what the name of the command is.
- You know what the name of a command is but would like to learn more about how to use it.

To find more information about how to accomplish “something”, type

```
man -k [single keyword] | less
```

(Again, don’t type the brackets.) This will return a list of possible commands that the student could investigate further. The term “|less” sends the output of the command `man -k [single keyword]` to the `less` command. The `less` command slows down the display so that the user may scroll up and down the display a page at a time. To scroll up type `b`, to scroll down type the space bar, and to highlight a keyword or phrase type

```
/ [keyword]
```

To find more information about a command, type

```
man [command name] | less
```

A lengthy description of the command should appear. Use the `less` command to scroll up and down the screen as well as to highlight keywords to improve readability.

More general online help files are available in the `/usr/pub` directory on all the instructional UNIX systems.

Other utilities in UNIX

E-mail

E-mail is integrated into most UNIX systems and the EECS instructional systems are no exception. There are a variety of e-mail programs available for use on UNIX; some are easier to use than others. Programs available on the UNIX system include pine, elm, mail, and mailx; you may also set up the netscape Web browser program to enable sending and receiving mail. To find out more about these programs or e-mail in general, use the man command described in the previous section. (They also have online help.) One cautionary note: pine and elm may encode files that you send in such a way as to make it difficult for someone without the proper software to read them. When sending a text file to someone whose software environment you don't know anything about, use mail or mailx.

ssh

The EECS workstations are for the most part connected to each other through a complex network of cables and computer network equipment. The ssh command provides a way to connect from one workstation to another remotely through this network.

The Web page

<http://inst.eecs.berkeley.edu/connecting.html>

contains more information about ssh.

Exiting

Exiting on a UNIX system is trivial for the most part. Simply type exit or logout at the command prompt.

Using a text editor to write programs

A text editor is best described as a basic word processor. It allows the user to compose documents with text. Unlike a word processor however, a text editor lacks formatting and font capabilities. To write a program for CS 3S or a CS 9 or 47 course, you use a text editor to input your program into a file. There are a variety of text editors available for use on the UNIX systems. The easiest text editor to use is covered here. The others are summarized below (to find out more about these editors, use man).

Summary of available text editors

emacs

Emacs is arguably the most powerful text editor available. It combines both relative ease of use and extensibility. With emacs, users can not only edit text, but also read their e-mail and browse news groups among other things. In addition, emacs possesses special "modes" that can be integrated with other programs, such as program debuggers, for additional flexibility and power. From a coding standpoint, emacs also possesses auto-format features in accordance with style guides for the popular programming languages in use today. The section "Emacs Reference" available through

the self-paced course web site includes lists of emacs commands and their descriptions. On the EECS instructional systems, there is also an emacs tutorial available; select Tutorial from the Help menu.

vi

Vi is the most widely available text editor among all UNIX platforms. Despite its relative bare bones appearance, vi also possesses an incredible amount of text editing power. Some people find vi hard to learn. Vi is good to master if you intend on doing UNIX work on other non-EECS machines that may not have some of the other text editors installed.

Pico

Pico is by far the easiest to use of the available text editors because it requires almost no memorization of command-key sequences in order to execute functions within the program. To start pico, simply type

```
pico
```

or

```
pico [name of file]
```

(if the file already exists and you wish to modify its contents) at the command prompt.

To execute a pico command once in pico, simply hold down the ctrl key and press the associated letter. Below is a listing of some of the major commands in pico. These commands are also displayed on screen while pico is running.

ctrl-G	Get help
ctrl-W	find ("Where is") something in the text being edited
ctrl-V	move forward a screenful of text
ctrl-Y	move backward a screenful of text
ctrl-C	report current Cursor position (helpful in debugging)
ctrl-K	to cut ("Kut") a line out of the text
ctrl-U	Uncut (paste) the last section of cut text
ctrl-R	Insert an external file at current cursor position
ctrl-O	Save the current <i>buffer</i> (the text currently being edited) into a file
ctrl-X	Exit pico

Arrow keys may be used to move the cursor right or left in the current line or up to the previous line or down to the next line.

Preparing a program to be executed

Except for CS 3S, CS 9ADE, and CS 47A, every course in the CS 9 and 47 series requires the use of a program called a *compiler*. This section discusses the differences between programming languages that require their source code to be compiled and interpreted languages that do not, along with the relevant commands that need to be issued to compile or interpret a program in a given language.

Compiled versus interpreted programs

For the most part, most programming languages look much like English, if only in some abbreviated form. This allows programmers to readily ascertain a program's logic in a compact yet standardized format. Unfortunately, computers do not have the ability to understand the English-like code of programming languages directly. Thus, there are two types of languages: interpreted and compiled.

An interpreted language is a language that requires a separate program resident inside a computer's local storage capable of processing its commands. Interpreted languages are in some sense very similar to macro languages in many popular packages today (e.g. Excel). In both cases, a program that is capable of executing on a computer is launched and then the instructions of the interpreted language are processed.

On the other hand, a compiled language is a language whose instructions are ultimately translated into a format that the computer can process readily without the aid of another program. The process by which a program written in a compiled language is translated into a form the computer can process and execute is called *compiling*.

Compilers and interpreters available

The following commands are used to compile programs on the EECS UNIX workstations, once the code for them is written:

Pascal	<code>pc [filename]</code>
C	<code>gcc [filename]</code>
C++	<code>g++ [filename]</code>
Fortran	<code>f77 [filename]</code>

Upon issuing the above commands, either the compiler will designate the location of one more errors and their relevant line numbers in the file or the command prompt will reappear. If a file named "a.out" is created, the program has compiled successfully. To launch the program, simply type a.out. Some students may prefer to rename the a.out file to something else. This can be accomplished using the mv command.

The following interpreters are available:

Scheme	<code>stk</code>
Various command shells	<code>(ksh, sh, csh, etc.)</code>

The use of interpreters is sufficiently covered in the context of each course's documentation and needs no further explanation here.

Running java programs

To run a Java program, you use a compiler and an interpreter. The compiler is named `javac`; its argument is a file whose name ends in the characters “.java”. The interpreter to use, once `javac` has been run, depends on whether the program is an application or an applet. Applications are run with the `java` command; applets are run either from a browser or with the `appletviewer` command.

Testing programs and collecting evidence of their correctness

Once a program is compiled (or interpreted), various tests need to be performed to ensure that it is executing properly. It is a student’s responsibility to convince the tutor that his or her program is indeed written correctly and that it executes with the desired behavior.

As a result, the Self-Paced Center requires that all students bring a transcript of their testing session. To create a transcript, simply type `script`. A message saying “Script started, file is typescript” should appear. When you are done testing your program, simply type `exit`. (This will not exit out of the UNIX environment, instead, it should give the message “Script done, file is typescript”.) Between the moment when the command `script` is issued and when `exit` is issued, all other commands and screen output will be copied to a file called `typescript`. It is this script file and accompanying program listing that you are ultimately to present to the tutor in order to justify the correctness of your program.

Some things to beware of: *Do not* type another `script` command without first typing `exit`. *Do not* type another `script` command without first typing `exit`. *Do not* use an editor when you are using `script`. While this does no harm, it does put bizarre characters into the `typescript` file that cannot be printed properly. If you do this your printout will end up being very garbled.

Printing programs

Because programs are created and tested in student accounts, printing can be a rather complicated process. There are two ways to print a document:

- go to an EECS workstation and issue a print command to the local printer, or
- download the file to a personal computer and print the file on a printer attached to the personal computer.

Printing from an EECS workstation:

Type

```
lp -d_____ [filename]
```

where _____ is the name of the printer located at the facility (`lp -dlw105 myfile` prints `myfile` on the printer named “lw105”). If you omit the “-d”, your file will go to the default printer for the system you’re using; type

```
echo $PRINTER
```

to find out where that printer is.

Printing from a personal computer:

Transfer the file to the computer from which you want to print the file. Currently mailing the file to yourself is probably the easiest way to do this. Then retrieve the file with a mail reader on that computer and print it from there.

Sources for additional help

Help sessions:

CSUA offers help sessions every semester. To get a list of help session dates, at the command prompt type

```
finger helper@csua.berkeley.edu
```

or access the URL <http://www.csua.berkeley.edu/help-sessions> via the Web.

The SPC staff may be offering their own UNIX help sessions as well. Consult the self-paced Web pages (inst.eecs.berkeley.edu/~selfpace) for details.

On-line documentation:

UNIX man pages (see man in this document for details)

Documentation available via the Web site

```
http://inst.eecs.berkeley.edu
```

EECS Instructional Computing documentation (Type

```
cd /usr/pub
```

then type ls for a list of available topics.)