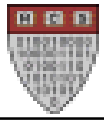
USING YOUR FAS UNIX ACCOUNT

HCS

HARVARD COMPUTER SOCIETY

Table of Contents



COMPUTING AT HARVARD

Harvard provides a superb computing environment to meet the needs of all its students. The campus-wide FAS Network connects together student's personal computers, public lab computers, and large central servers, allowing access to nearly any computing resource from any point on campus.

The heart of the FAS Network is the **fas.harvard.edu** cluster, a group of top-of-the-line servers providing email and login services to nearly 20,000 users. These systems run the powerful and robust Unix operating system. Each student at Harvard is given their own account on fas.harvard.edu. The most important thing about your Unix account is that it is also your email account, but it is useful for many things beyond that. This seminar will provide an introduction to the resources available to you as part of your FAS Unix account, the basics of using the Unix operating system, and an introduction to some of the many things you can do with it.

An Introduction to Unix

The Unix operating system provides a stable multi-user, networked computing environment. An operating system is the basic software the runs on a computer and controls the computer's fundamental operations. Typical operating system functions are file and directory maintenance, networking, program management, and security. The various flavors of Windows and the MacOS are operating systems you may already be familiar with. Unix has both similarities and differences to those OSes, some of them quite major.

One of the most important characteristics of Unix is that it is fundamentally a multi-user operating system. Unlike a Windows PC or a Macintosh, which is typically only used by one person at a time, a Unix server is capable of supporting hundreds of users at the same time. Now, obviously, those hundreds of users cannot all physically sit at the server and type on its own keyboard. Instead, Unix allows users to connect to and access the resources of the Unix server from a remote computer. By using a **telnet** or **ssh** program to connect to the Unix server, you can then access your files, run programs, print documents, and generally do anything which you could do if you were sitting at the console of the computer. Telnet and ssh provide a primarily text-based interface to Unix, but there also exists a graphical user interface known as the X Windows System.

The FAS Unix Systems

The central FAS Unix system is a group of six machines known collectively as **fas.harvard.edu (fas)**. When you log into fas, you are automatically routed to one of a group of identical machines. These several machines all work together, acting (mostly) as if they were a single machine. **fas.harvard.edu** is the primary Unix system for Harvard College, and for many students, it is the only Unix machine they ever need to use.

The Instructional Computing Environment (ICE) cluster is a second group of systems intended primarily for use by students for coursework in computer science classes. **ice.fas.harvard.edu (ice)** is very similar to fas, but the computers are configured somewhat differently, to better handle the more strenuous needs of computer science students. The individual computers that make up the **fas** and **ice** clusters are interchangable between those two clusters, and are referred to as the **is*** computers, where ***** is a different number for each of the computers.

In room B-14 of the Science Center, by the Help Desk, there are 33 Unix workstations, named **ws1.fas.harvard.edu** through **ws33.fas.harvard.edu**. These computers run the X Windows graphical user interface, and are also intended for use primarily by those doing academic coursework, although anyone should feel free to use them during non-peak hours.

Finally, a number of other FAS Unix systems provide specific services. Yet another cluster of machines, collectively known as **www.fas.harvard.edu,** serves as FAS's web servers. **news.fas.harvard.edu** is FAS's USENET news server, providing access to thousands of discussion groups on all kinds of topics. The **smtp.fas.harvard.edu** cluster handles the delivery of email for all 20,000 users of fas, while **pop.fas** and **imap.fas** pass that email along

to mail programs running on personal computers.

Connecting To Your FAS Unix account

Your computer user account is one of your most important possessions during your time at Harvard. Your same FAS username and password are also used to access PC and Macintosh computers throughout campus, in addition to your Unix account. Remember to keep your password safe by never sharing it with anyone else and changing it at least twice a year.

To connect to fas, you must use a **telnet** or **ssh** program. Telnet is a standard Internet protocol for accessing another computer remotely; ssh (which stands for "Secure SHell") is an improved version of telnet with better security features, and is preferred over Telnet if you have a choice. If you own a PC, the FAS Network Installer disk will allow you to install an ssh program called **SecureCRT** onto your computer. If you own a Mac, the FAS Network Installer disk will allow you to install a telnet program called **BetterTelnet** onto your computer. Both of these programs are available on public PCs and Macs throughout campus.

When you run SecureCRT on a PC, a **Connect** window should appear. Select fas.harvard.edu from the window's **Session List** or type "fas.harvard.edu" into the **Hostname or IP** text field near the top of the box, and then click the OK button.

When you run BetterTelnet on a Mac, an **Open Connection…** window should appear. Make sure that fas.harvard.edu is entered into the **Host Name** text field, and then click the Connect button.

Once you have connected to fas, enter your username and password when prompted. When you have logged in successfully, you will see a **fas%** prompt. (The first time you log into fas, you may be asked to take a brief quiz on computer rules and regulations, but that is a one-time occurrence) The **fas%** prompt is a powerful command line interface to Unix that allows you to run programs, access files, and connect to network resources.

Security & Privacy on FAS Unix Systems

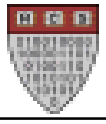
Unix systems are public computing environments. Just as in any public space such as a library, it is possible to see who else is using them and what they are doing. Unix has a powerful security model which enables you to control which of your files others can see (i.e. your web page) and which they cannot (your email). However, other types of information, such as how often you connect to fas, where you connect from, and the names of all the programs you run, are considered inherently public pieces of information and can be seen by all users. If you have any concerns about the availability of information about you, or about other security-related issues, do not hesitate to contact the Help Desk in the basement of the Science Center or email **security@fas**.

As with any other resource, there are both appropriate and inappropriate uses of the FAS Unix Systems. You should carefully read the FAS Network Acceptable Use Policy and abide by its rules. Most importantly, please note that *any* form of attempting to circumvent security restrictions on *any* computer over the FAS network can result in disciplinary action, and that transmission of copyright-protected material such as software or music is not allowed, and is indeed illegal.

Finally, it is important to use common sense when sending email. It is very much *not* appropriate to send out unsolicited mass emails to hundreds of people. Chain letters are *always* a bad idea. Do not 'spam' people; it's prohibited and it will result in disciplinary action against you.

Your FAS Unix Account's Password

To change your password, type **passwd** at the fas% prompt and follow the on-screen instructions. If you ever forget your password, you can bring your Harvard ID to the Help Desk in room B-14 of the Science Center and speak with one of the User Assistants.



USING THE UNIX COMMAND SHELL

You interact with Unix systems primarily through the use of a text-based command-line prompt. By default, on fas this prompt looks like this:

```
fas%
```

Running commands on Unix is as simple as typing the command at the prompt and pressing Enter. Unix commands are case-sensitive, so typing **pine** is not the same as typing **Pine**.

While the command-line interface may seem clumsy or old-fashioned at first, it is actually a tremendously powerful and flexible way of interacting with and controlling a computer. The command prompt is actually part of a program called your "shell". The shell is the program which is responsible for reading the commands that you type and following the instructions there; For instance, when you type "pine", the shell reads in your command, and then goes about the business of starting the pine program and displaying it on the screen. There are actually various different kinds of shells available; by default all users on fas use one called "tcsh", the TC Shell.

A note about entering commands: Whenever you enter a command at the prompt, the shell treats the first word you enter as the name of a command to be run. If you enter more than one word, then all the other words are treated as arguments or options which are passed to the command that is run. Often, different options for a program will be specified using a "-" character followed by one or more letters. For instance, compare the results of typing "**pine**" and "**pine -i**". When you start pine with the i option, it begins immediately with the index screen.

The tcsh shell provides the ability to scroll up and down through the 'history' of your commands. Simply press the up arrow key while at a command prompt to scroll back up through the commands you have most recently entered. The down key scrolls back down the same list. This feature enables you to repeat commands easily, or to quickly edit a command with an error in it. Another useful feature is command line tab completion: If you start typing a command or the name of a file and press the tab key, the shell will attempt to guess the rest of what you are typing and fill it in. If it can uniquely guess what you are typing, then it will fill it in automatically; otherwise, it will display its best guesses and allow you to continue typing. Tab completion is very useful when working with files or directories, especially those with long names.

Managing Multiple Programs under Unix.

The command shell provides the ability to switch between multiple running programs by 'backgrounding' a program. In effect, this is like minimizing an application under Windows or hiding it under the MacOS: The backgrounded program is still open, but it is hidden from the user temporarily. To put an application into the background, just press **Control-Z**. For example, if you are in pine and press **Control-Z**, you will see the message "suspended (signal)", which means that pine received the **Control-Z** signal and has suspended itself, and will then be back at the command prompt, where you can enter any other commands you wish. To return to pine, just type **fg**, which stands for "foreground". This will bring the suspended pine program back to the front, placing you exactly where you were when you suspended it.

To see a list of all programs currently in the background, use the **jobs** command. Typical output of jobs would be something like the following:

```
[1]  - Suspended (signal)           pine
[2]  + Suspended                    tin
```

This means that there are two programs currently suspended, pine and the newsreader program **tin**. If you have multiple suspended programs, then each one is given its own job number, which is displayed in brackets. Type **fg <job number>** to bring a specified job to the foreground. This enables you to switch between many programs running at the same time, just as under Windows or the MacOS.

For more detailed information on Unix commands and the command shell, we recommend *A Student's Guide to Unix* by Harley Hahn and *Unix in a Nutshell* by O'Reilly and Associates.

Files and Directories in Unix

Unix structures data into a hierarchy of directories and files, exactly as do Windows and the MacOS, although Unix has security features in its file system which most other operating systems lack. In a multi-user operating system like Unix, each user controls his or her own files. By default, these files are private and no other user can access them. However, it is possible to make some of your filespublic so that other users can see them; for example, you would do this with your web page.

Your Home Directory

Every user has their own personal directory in which to store their own files, called their **home directory**. When you log into fas, you are automatically placed in your own home directory. To find out the name of your home directory, type **pwd** at the command prompt. (This stands for "Print Working Directory.") If John Harvard types pwd, he will see the following output:

```
/home01/j/h/jharvard
```

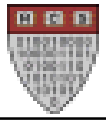
This is the name of John's home directory, and it tells him that his home directory is located in the **h** subdirectory of the **j** subdirectory of the **home01** disk. (There are several different disk partitions in which home directories are stored; it makes no difference which one your own home directory is on. They are organized mostly alphabetically and split into subdirectories by the first two letters of your username in order to allow better system performance.) Note that while DOS and Windows use the backslash (****) as the directory seperator char, and Macintoshes sometimes use a colon (**:**), Unix directory paths are always written using forward slashes (**/**) to seperate the directories.

Your Home Directory's Disk Space Quota

Although fas has a truly tremendous amount of disk space, it is not infinite. As a result, each user has a limited (although large) amount of disk space for their files. This includes your email, web page, and all other files you may put in your home directory. Currently, this disk space quota is 50 MB per user, although this amount may potentially increase in the future. To find out how much of your quota you are currently using, just type **quota** at the prompt.

```
fas% quota
Disk quotas for user jharvard (uid 353):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
        /home01    8529   50000   50000              13    4000    4000
```

The /home01 row gives the quota for and size of John's home directory. The 8529 in the blocks column reports that John has 8529 kilobytes (KB), or roughly 8.5 megabytes (MB), of data in his home directory. These data include any emails in John's email inbox (which resides in the **.inbox** file in his home directory), any files comprising John's home page, and all other files and directories which he may have. The 50000 in the quota and limit columns mean that John's quota is 50,000 KB (or roughly 50 MB), and that he cannot exceed this quota for any reason.



Using ls to List Files in Your Home Directory

To see a listing of the files in your home directory, type **ls**, which stands for "list". You will see an output something like the following:

```
dead.letter mail  News  moral_reasoning_paper.doc
```

These are the files and subdirectories in your home directory. The **mail** subdirectory is where pine stores all of your saved email messages. **News** is a subdirectory containing files relating to news discussion groups, and the other two listings are just files stored in that directory. **dead.letter** is an email message that was canceled before it was sent; pine stores the message in this file in case you later wish to recover the canceled message.

By specifying options to the ls command, you can change the command's output. The **-a** option asks ls to display *all* files, even including files which are normally hidden.

```
fas% ls -a
.                .addressbook.lu  .history         .logout
..               .aliases         .inbox           .pinerc
.addressbook     .cshrc           .login           mail
```

By including the –a option, the items above whose names begin with a period are displayed. These so-called "dotfiles" are normally hidden rather than being displayed. Dotfiles are usually configuration files and setup information—files which you usually do not need to worry about, so they are hidden out of the way. Some typical dotfiles are shown above. These are as follows:

The item called **.** (*i.e.*, a single period) refers to the current directory—*i.e.*, John's home directory. The item called **..** (*i.e.*, two periods) refers to the current directory's parent directory—*i.e.*, the directory in which John's home directory resides. The file called **.addressbook** contains the nicknames that John has defined in Pine for individuals' email addresses. The file called .**aliases** contains shortcuts for Unix commands or programs that either John or system administrators have defined. The file called **.cshrc** contains configuration and startup information for the tcsh shell itself. **.history** contains the listing of the commands that John has recently executed, to allow the scrolling up and down function to work. The **.inbox** file is John's email INBOX, containing all of his incoming messages. The **.login** file contains a list of Unix commands or programs that are executed whenever John logs into fas; these handle setting up his account for use each time. Likewise, .**logout** contains a list of commands that are run when John logs off of fas. The file called **.pinerc** contains Pine's preferences.

The **-l** option requests a long format output, which lists file size, modification date, ownership and other information for files and directories.

```
fas% ls -l
total 567
-rw-------   1 jharvard student      6868 Jul 16 13:36 .dead.letter
drwx------   2 jharvard student      8192 Jul 16 13:36 mail
drw-------   3 jharvard student      8192 Jun 23 16:38 .News
-rw----r--   1 jharvard student    544000 Jul 27 21:40 moral_reasoning_paper.doc
```

The total 567 means that about 567 kilobytes of data exist in this level of John's home directory; *i.e.*, not taking into account the sizes of files and directories nested in subdirectories of his home directory. The first column provides the permission listings for each file and directory in John's home directory, which will be explained below. Note that the initial "d" identifies directories, while regular files have listings that start with a dash. The third column shows the username of the owner of these files—jharvard owns all of the files in his own home directory. The fourth column shows that since John is in the student group on the FAS Unix systems, his files also belong

to that group. The fifth column reports the size in bytes of each item. The sixth column reveals the date and time at which the items in John's home directory were last modified. The last column contains the names of each file or directory.

You can combine the -a and -l options if you wish, in which case ls will show a long format listing of all files, including hidden ones. ls has other options as well, which you can read about in the ls manual page. (Man pages will be discussed shortly.)

Managing Your Files

Unix has many useful commands for managing your files. Some of the most useful are as follows. Note that historically, Unix commands tend to have fairly short names, to minimize the amount of typing needed; many of the most commonly used commands are only two or three letters long.

`cp [file1] [file2]`
Copies [file1] to [file2].

`mv [file1] [file2]`
Moves [file1] to [file2]

`rm [file1]`
Removes (i.e. deletes) [file1].

`mkdir [dir1]`
Creates a subdirectory [dir1] located in the current directory.

`rmdir [dir1]`
Removes the directory [dir1], which must be empty. (i.e. contains no files.)

`cd [dir1]`
Changes the current working directory to be [dir1].

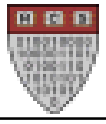
File and Directory Permissions

One of the most important security features of Unix is the ability to control various access permissions to a file. As discussed above, each file is owned by a specific user and group (such as the user "jharvard" and the group "student", in our example above.) For every file you own, you can control nine different security parameters: whether that file can be read, written, or executed by that file's owner (yourself), its group (other students), and everyone else.

Look back at the output of **ls -l** above. The leftmost column in that output reveals the permissions set for each file and directory in John's home directory. Each permission listing consists ten components. As mentioned above, the first component in the permission listing for an item signals whether the item is a directory or a regular file (or one of a number of specialized file types, such as links and named pipes, which we will not discuss here.).

The remaining nine characters in a permission listing should be considered in groups of three: the first triplet refers to the file owner's privileges for that item, the second triplet refers to the group's privileges for that item, and the third triplet refers to everyone else's privileges for that item.

Each triplet consists of three parts: a read permission setting (`r`), a write permission setting (`w`), and an execute permission setting (`x`). If a user has read privileges for a file, that user can view the file; if a user has read privileges for a directory, that user can list the contents of that directory with the ls Unix command or access files and directories within that directory. If a user has write privileges for a file, that user can edit that file; if a user has



write privileges for a directory, that user can delete, move, or rename that directory as well as the files within. If a user has execute privileges for a file, that user can execute that file as a program; if a user has execute privileges for a directory, that user can change his current working directory to that directory with the cd Unix command.

For example, the permission listing for John's .addressbook file (the file containing any nicknames that John has defined in Pine for individuals' email addresses) is `-rw-------`. The first component of this listing signals that .addressbook is a file and not a directory, since it is a hyphen and not a d. The next triplet of components is `rw-`, which means that John has read and write privileges for this file. The next triplet of components is `---`, which means that other members of John's group cannot read, write, or execute this file. The last triplet of components is also `---`, which means that even users outside of John's group (usually called "the world") cannot read, write, or execute this file.

If you would like to change the permissions on a file or directory, you can type

```
chmod [who]±[privilege] [file or directory]
```

where **[who]** indicates the set of users whose permissions you are changing, ± represents the addition (**+**) or subtraction (**-**) of a privilege, and **[privilege]** is one of **r**, **w**, or **x**, representing the privilege to add or subtract for <who>. Valid choices for **[who]** are **u**, **g**, **o**, and **a**, or a combination thereof, where **u** represents the owner of the file or directory, **g** represents all users in the owner's group, **o** represents other users not in the owner's group, and **a** represents *all* users, equivalent to specifying **ugo**.

Hence, if John has created a file "readme.txt" which he would like other users to be able to read, he can allow them to do so by typing

```
fas% chmod a+r readme.txt
```

If he later wishes to only allow others in his group to read the file, but no one else, he could type

```
fas% chmod o-r readme.txt
```

One use for changing permissions is write-protecting a file; if you have a file in your FAS account which you would like to prevent from being changed, you can write protect it by removing all write permissions, i.e. by executing the command

```
fas% chmod a-w importantfile.txt
```

There is an alternate numeric syntax for precisely specifying file permissions, using octal (base 8) numbers. Basically, each permission triplet (user, group, other) is written as a single digit, whose value is the sum of the permissions for that triplet, with read = 4, write = 2, execute = 1. Thus, to grant all permissions to the user, and execute permissions to both group and others, you would invoke something like

```
fas% chmod 711 ~
```

This sets the user permissions of your home directory (~) to be mode 7, which is 4+2+1, so all of rwx are set, while both group and other permissions are set to mode 1, execute only.

For a more in-depth look at permissions, including a discussion of Access Control Lists (ACLs) which allow more precise control of file permissions, see any introductory Unix text such as those mentioned above on page 4.

EDITING FILES IN UNIX

Using Pico

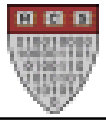
Pico is a Unix program that allows you to create and edit text files. Text editors like Pico resemble typewriters more than they do word processors, as you cannot specify font faces or font sizes for text files. There exist more powerful text editors than pico, such as vim or emacs; however these programs are much more complicated to learn to use. While computer science students will want to learn these more powerful editors, most students will find pico both easiest to use and adequate to their needs.

To use Pico, type **pico [file]** at the fas% prompt, where [file] is the name of an existing file that you want to edit or the name of a file that you want to create. In the screen that appears, you can proceed to edit the file in the normal manner, using your arrow keys to move the cursor and typing the desired text. All of Pico's commands are two-character sequences and are explained at the bottom of the screen, where **^** stands for the Control (Ctrl) key.

To save your text file without quitting Pico, type **Control–O**. When prompted for the file name to write, hit Enter or Return.

To quit Pico, type **Control–X**. If prompted to save the modified buffer, type Y or N accordingly. If then prompted for the file name to write, hit Enter or Return.

(Incidentally, when you compose a message in Pine, the text editor you are using is Pico.)



Basic Pico Editing Commands

The following commands may be used while editing a text file with Pico or while composing, forwarding, or replying to an email with Pine. For additional information on Pico and Pine, visit http://www.washington.edu/pine/ .

On-Screen Help

Control–G	Displays on-screen help

File Management

Control–O	Saves the file (in Pico) or postpone the message (in Pine). (*I.e.*, writes your text out to a file.)
Control–R	Inserts an existing file at the current cursor location. (*I.e.*, reads in an existing file.)
Control–X	Saves the file and exits (in Pico) or sends the message (in Pine).

Moving the Cursor

Control–A	Moves the cursor to the beginning of the current line.
Control–E	Moves the cursor to the end of the current line.
Control–V	Page Down: Moves the cursor forward one screenfull.
Control–Y	Page Up: Moves the cursor backwards one screenfull.

Editing Text

Control–J	Re-justifies the current sentence or paragraph.
Control–L	Re-draws the screen.
Control–K	Cuts the current line
Control–T	Spell check.
Control–U	Pastes the most recently cut line—*i.e.*, uncuts it—or unjustifies a recently justified paragraph or sentence.
Control–W	Searches for text.

Using Aliases

The command shell allows you to define a list of **aliases**, which are shortcuts, abbreviations for longer commands. If you type the same command fairly often (*e.g.*, "pine" or "finger jharvard"), you can create an alias in order to type something shorter to execute that command (*e.g.*, "p" or "fj"). To define an alias, just type

```
alias <shortcut> <long command>
```

at the prompt. This will define <shortcut> to stand for <long command>, but only for your current log-in session; when you log out of fas this alias will be lost. In order to add an alias permanently, you have to modify your startup files.

A Digression: Unix Startup Files

Whenever you log into your Unix account, the shell reads in and processes certain files. These files set up various aspects of your Unix account, ranging from displaying the various messages you see upon login, to setting your prompt to be the string "fas%", to setting environment variables and other configuration items for pine and other programs.

The two most important login files are your **.cshrc** and your **.aliases** files. The **.cshrc** file contains the startup commands for the tcsh shell that reads in and executes all of your commands (**cshrc** stands for "**csh r**esour**c**es"; many configuration files in Unix end in the letters "rc" for this reason.), and the **.aliases** file contains a list of aliases which are loaded by your shell at the end of loading the cshrc. Thus, to permanently add an alias to your account, you have to edit the .aliases file.

To edit your aliases files, type **pico .aliases** at the fas% prompt while in your home directory. A Pico editor window will open up containing all of your current aliases. There are several default aliases which will be there even in new FAS accounts; we recommend leaving these alone, although you can delete them if you wish. Scroll to the bottom of this file and add a new line containing the alias command for the alias you wish to add,

```
alias <shortcut> <long command>
```

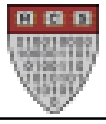
just as above. Then press **Control-X** to exit Pico, saving the file as you do so by typing "Y" when prompted. Because you have added the alias to your .aliases file, the new alias will begin working the next time you log in, and for all times thereafter. .

Some Suggested Aliases:

Here are some suggested aliases you might want to add to your .aliases file.

```
alias pine pine -i
```
This causes pine to start with the Inbox rather than main menu.
```
alias ff find ~ -name '\!*' -print
```
This lets you search for files in your home directory by typing "ff <filename">
```
alias talk ytalk
```
This allows you to always run the superior ytalk command, even if you type talk.
```
alias phe ph 'email=\!*@fas.harvard.edu'
```
This allows you to type phe "email address" to look up someone's directory entry.
```
alias pine 'ps | fgrep pine | fgrep -v grep; [ $? != 0 ] && \pine \!* || %pine'
```
This prevents additional copies of Pine from starting accidentally if you have backgrounded one.

Note that the characters **\!*** represent an argument supplied when invoking the alias; thus for example you would invoke the *phe* alias above as **phe <username>** to look up a person with that username.



COMMUNICATING WITH OTHERS

Using the Pine Email Client

To send and receive email, you can use a program called **Pine** (Program for Internet News and Email). Pine offers a simple menu-driven interface and online help. With Pine, you can send email, receive email, reply to email, forward email to others, save your email into separate folders, print your email, and make an addressbook containing shortcuts to, or nicknames for, others' email addresses. As previously mentioned, Pine uses the text editor Pico to compose, forward, and reply to emails. Thus, similar to pico, all of Pine's commands are one- or two-character sequences. In the program's main menu, all of the commands are one-character sequences; in all other menus, commands are explained at the bottom of the screen.

Perhaps the best feature about pine is that it can be used from any computer, simply by telnetting to fas and running pine there. This is a major advantage for students who are constantly traveling around campus and want to stay in touch with their email. However, many students prefer graphical mail clients such as Eudora or Netscape Mail to Pine's text only interface, and with good reason. The majority of students actually use multiple programs for their email, usually using Eudora when in their own room, and for storing mail for the long term, and using pine when out and about on campus.

In order to better show off Pine and Eudora together, and due to the limitations on the amount of material which can be covered in any one seminar, using the Pine email client is covered in depth in the Using Harvard's Network with your PC and Macintosh seminars, even though it is a Unix program. If you have any questions about Pine, please attend one of those seminars, or else just speak with one of our assistants after this seminar is concluded.

Ph: An Online Phone Directory

Ph is a Unix program that allows you to access Harvard's online directory, which contains contact information for most Harvard faculty, staff, and students. To look up someone's directory information, type **ph <name>** at the fas% prompt, where <name> is the first name, last name, or first and last names of the person for whom you would like directory information. To look up directory information based on someone's email address instead, type **ph email="<email address>"** at the fas% prompt, where <email address> is that particular email address. If the email address ends with harvard.edu, you can replace harvard.edu with an asterisk (*). Were you to type **ph John Harvard** or **ph email="jharvard@fas.*"** at the fas% prompt, the output might resemble the following:

```
----------------------------------------
       name: John Harvard
  telephone: 493-1000
    address: 1 Harvard Yard Mail Center
  residence: University Hall
      email: jharvard@fas.harvard.edu
----------------------------------------
```

Using Finger

Finger is a Unix program that allows you to find out whether or not a specific user is logged into fas. If the user is logged in, finger will give you a variety of information including where they are logged in from and which login host they are using. If the user is not logged into fas, finger will report the when and where they were last logged in. Finger will also tell you whether that user has new mail—this can be a useful feature if you are trying to determine if someone has read a message you have sent them yet.

To finger a user, type **finger <username>** at the fas% prompt. If the corresponding user is logged into fas, you will see output resembling the following:

```
fas% finger jharvard
[fas.harvard.edu]
Login name: jharvard                    In real life: John Harvard
Directory: /home/j/h/jharvard           Shell: /shells/tcsh
Login           Name         Idle TTY   Host              Where
jharvard  John Harvard       0:15 p0  is07  (jharvard.student.harvard.edu)
No unread mail.
No plan.
```

The **In real life** field gives the real name of the user jharvard. The **Idle** field gives the number of minutes since jharvard has last touched his keyboard. The **Host** field states that John is logged into **is07.fas.harvard.edu**, which is one of the FAS Unix systems. Recall that **fas.harvard.edu** is really a cluster of identical machines; although these machines are all identical and interchangeable, they each have a unique name to identify them, of the form **is#**. (is stands for one of Interactive Server, Interchangable Server, or Instructional Server; take your pick!) There are some circumstances in which it is useful to determine which login machine a given user is currently on, and finger provides the easiest way to do that. (To find out the name of the login machine you are currently using, just type **hostname** at the fas% prompt.)

Finally, the **Where** field above explains that John is logged into fas from a computer named jharvard.student.harvard.edu—*i.e.*, John's personal computer.

You can also try to find out whether or not someone is logged into a different Unix system, such as at a different school or university, by typing **finger <email address>** at the fas% prompt. However, not all Unix systems permit such use of Finger, and furthermore not all email addresses are associated with Unix systems.

Customizing Your Own Finger Output

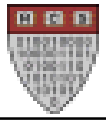
It is possible to customize the output that someone will see when they finger you, by editing what is called a **.plan file**. (The name comes from when the system was originally developed by computer programmers, who used it to inform each other of their current research plans.) Many students like to put messages, quotes, or contact information in their .plan files.

To create your own .plan, simply use the text editor of your choice (usually pico) to create a file called .plan in your home directory. In other words, type **pico .plan** at the fas% prompt. Type your message into the pico edit window; it may be as short or as long as you would like. Save your .plan file and exit.

Now, you must set file permissions on your .plan file to enable other to view it. Remember, by default all your files are private and can be seen only by you. There are two ways to make your .plan file world-readable:

First, you can type the commands:

```
fas% chmod 644 .plan
```



```
fas% chmod 711 ~
```

Alternatively, type **fixfinger** at the fas% prompt. Fixfinger is a custom script that exists only on fas; it simply executes the above chmod commands for you.

You can also create a text file called **.project** for your finger output, which is like a one-line .plan. To create your own .project, type **pico .project** at the fas% prompt, and enter a one-line message. Again, you must then use chmod or fixfinger to set the permissions correctly.

Were John to customize his finger output with a .plan and a .project, it might resemble the following:

```
Login name: jharvard                    In real life: John Harvard
Directory: /home/j/h/jharvard           Shell: /shells/tcsh
Login           Name         Idle TTY   Host              Where
jharvard  John Harvard           p0  is04  (jharvard.student.harvard.edu)
Unread mail since Jan  1 12:00:00 1638.
Project: Surviving freshman week
Plan:
      "I'll wave my hands and move on.  It's a standard procedure."
                             -Professor Joe Johnson
```

Preventing Others from Fingering You

To prevent others from fingering you, type **nofinger** at the fas% prompt and follow the on-screen instructions. From this point onwards, no information about you will be accessible via finger. To re-enable finger, just run nofinger a second time. However, please be aware that other means exist on Unix systems for determining whether or not you are logged in; executing nofinger only removes one of them. If you have any concerns about other people following you excessively or stalking you online, please speak to a User Assistant or contact the FAS Computer Services Security Team at **security@fas**.

Other Ways to Access Public Information

There are other ways to access public login information besides finger. The **who** command lists all users currently logged onto the same login host as you are, and the **rwho** command lists all users currently logged into all FAS Unix machines, including fas, ice, and the ws* workstation. ruptime provides status information on all of the Unix machines, including which of the **is*** boxes are currently configured as **fas** and which as **ice**.

The **w** command is a variation on who; it lists all users currently logged into the same login host as you, and in addition the name of the program they are currently running. This is an important point to remember: *the name of every program you run on fas is visible to other users of the system.*

Finally, if you want to check on the login history of a particular user, the **last <username>** command will list the last several logins for that user. It is a good idea to sometimes run last on yourself to make sure that no one else has stolen your password and logged in as you.

Note: As with any form of publicly available information about individuals, there are proper and improper uses for all of this data. *Don't* abuse these public computer logs; it's one of the fastest ways to get in serious trouble with the administration. Fingering a friend once in a while to see if she's logged in somewhere so you can talk to her is OK; fingering her every five minutes so you can track where she goes all day constitutes online stalking, and is a *very* serious violation of Harvard's network regulations which will very promptly result in disciplinary measures by the Administrative Board. If you have questions about whether a certain action could constitute online stalking or not, or if you feel someone is paying too much attention to you online, do not hesitate to email **security@fas**.

Write: Sending short messages in Unix

Write is a Unix program that allows a user to send a message to another user's screen. Before messaging another user, you must ascertain whether or not that user is logged into fas and, if so, which of the six login hosts the user is on. As explained above, the finger program provides this information. To begin messaging a user, type **write <username>@<Host>** at the fas% prompt, where <username> is the user's username and <Host> is the system reported by finger <username>. Then, begin typing your message, and press Enter or Return after each line. The line of text you have typed will be displayed on the other user's screen, and you can begin typing the next line of your message. You can only type 254 characters in a row before you *must* type Enter or Return, however. To quit Write, press **Control-C**.

Were John Harvard to message a user, the latter's screen would display the following text, followed by John's message:

```
Message from jharvard@is10.fas.harvard.edu on ttyp0 [UNAUTHENTICATED] at 12:00 ...
```

To message someone on a different University's or company's Unix system, type **write <email address>** at the fas% prompt, where <email address> is the email address of that person. Once again, not all Unix systems permit such use of write, and not all email addresses correspond to Unix systems.

Preventing Write Messages

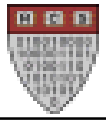
Sometimes, you may wish to disable write messages from being displayed on your screen. This can be useful for example if you are going to be working on an academic assignment and do not wish to be distracted. To prevent anyone from sending you a write message, type

```
fas% mesg n
```

at the prompt. (This stands for MESsaGe No.) "mesg y" will turn back on your ability to receive write messages. It is possible to determine whether another user has mesg turned on or off. When you finger a user, if the user has mesg set to no, then an asterisk (*) will be displayed after their idle time:

```
fas% finger jharvard
[fas.harvard.edu]
Login name: jharvard                    In real life: John Harvard
Directory: /home/j/h/jharvard          Shell: /shells/tcsh
Login           Name         Idle TTY   Host              Where
jharvard  John Harvard       0:05*j7    is06  (ws4.harvard.edu)
```

This means that you cannot send write messages to John until he turns mesg back on.



Using Talk & Ytalk: Chatting with Others in Unix

talk & **ytalk** are Unix programs that allow users to chat with each other in real time. While talking with another user, everything that you type appears on the other user's screen, and vice versa. talk provides a basic version of this functionality, but ytalk is a more sophisticated program and includes support for talk sessions between more than two users. For this and other reasons, ytalk is recommended in place of talk.

Before chatting with another user, you must ascertain whether or not that user is logged into fas and, if so, what host the user is on. As above, the best way to get this information is by fingering that user. To request a chat session with a user, type **ytalk <username>@<Host>** at the fas% prompt, where <username> is the user's username and <Host> is the system reported by finger <username>. Were John Harvard to request a chat session with you in this manner, you would see a message on your screen resembling the following:

```
Message from Talk_Daemon@is05.fas.harvard.edu at 21:06 ...
talk: connection requested by jharvard@is05.fas.harvard.edu.
talk: respond with:  talk jharvard@is05.fas.harvard.edu
```

Simply follow the on-screen instructions to initiate the talk session: as soon as you type **talk jharvard@is05** or **ytalk jharvard@is05** at your fas% prompt, the chat session would begin. Each user will see their screen divided into different sections, one for each user participating in the talk session. Everything typed will show up on the screens of all those participating in the talk session. The benefit of talk and ytalk over write is that write mixes up everything typed by all parties, while talk and ytalk keep different people's messages seperated on the screen, allowing for easier reading.

To request a chat session with someone on a different University's or company's Unix system, type **ytalk <email address>** at the **fas%** prompt, where <email address> is that person's email address. The same caveats apply here as did for finger and write.

To request a chat session with multiple individuals, simply specify multiple <username>@<Host>s, separated by spaces, after ytalk at the fas% prompt. Alternatively, you can use ytalk's Escape menu: while in ytalk, pressing the Escape key will bring up a menu listing various commands. These includeadding another user to the talk session, terminating a user, or logging everything a user types to a file, among others. For more information on these options, see the ytalk manual page.

To quit a chat session or terminate a chat request, type **Control–C**.

mesg n will prevent other users from talk requesting you just as it prevents them from write messaging you.

PRINTING

Public printers exist in dozens of buildings across campus. In addition to the extensive selection of printers in the Science Center, every upperclass House and the freshman dorms Matthews and Hurlbut else each have a computer lab with a printer. Most of these printers are high-speed black and white laser printers, but there are also color and transparency printers and oversized paper printers.

The cost of print jobs to public printers is charged against your FAS account's printing budget. A printing budget with a balance of $0.00 is created automatically with your FAS account, but you must add money to it first in order to be able to print. Harvard doesn't make a profit off of printing; the charges just cover the cost of paper, ink and printer maintenance.

The per-page printing costs depends on the type of printing you are doing. Costs as of September 1999 are as follows.

8.5" x 11" Black and White	**$0.05 per page**
11" x 17" Black and White	**$0.10 per page**
Color Copy Paper	**$0.30 per page**
Color High Brightness Paper	**$0.50 per page**
Color Transparency	**$1.00 per page**

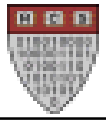
At the end of every print job, an receipt page will be printed listing the username of the person who printed that document and detailing the charges made to that user's print budget. The only exception to this is the color laser printer, sccolor, which does not print receipts due to the higher cost of its paper.

Checking Your FAS Laser Printing Budget

To check how much money you have in your printer budget, type **usage** at the fas% prompt. You will see output resembling the following:

```
Username:  jharvard                      Owner:  John Harvard
Site:      u
Flags:     None
-------------- Laser Budget -----------  --- Laser Usage ---  -- Balance --
   Free      Paid   Overdraft     Total    Used    Available       Due
   0.25     15.00      0.00       15.25    6.15       9.10        0.00
```

The **Paid** field lists the amount of funds that John has contributed to his budget. The **Free** field above lists any free credits that John may have received, either as a reimbursement for the cost of a failed print job, or, rarely, due to enrollment in a computer science class.The **Used** field lists the amount of money that John has spent, in total, on laser printing, and finally the **Available** field lists how much money John currently has available for use. If you overdraft your printer budget by printing more than you can pay for, then the **Balance Due** field will list the amount which you must deposit before you can print again.



Adding Funds to Your FAS Laser Printing Budget

You can deposit funds into you budget in three ways. First, you can insert bills into the cash acceptor in room B-14 of the Science Center, located directly across from the Help Desk. The cash acceptors accept $1, $5, $10, and $20 bills. Secondly, you can make a check for $10 or more out to “Harvard University” and present it to the User Assistants at the Help Desk in room B-14 of the Science Center. Finally, you can type **termbill** at the fas% prompt to charge money against your Harvard College termbill. All additions to your budget take effect within 15 minutes. Please note that there are no refunds for unused balances.

Unix Print Commands

There are a variety of commands you may enter at the fas% prompt to start and control print jobs. In order to print a file, you will need to know the name of the printer you wish to use. All public printers have their names written on labels affixed to the side of the actual printer, so just visit a printer and read its label in order to learn its name. Most commonly, the name of a printer is the name of the building it is located in followed by a number, e.g. “dunster1”. The Science Center printers are all named by room number, e.g. “scb11c” is the printer in room B11C, the Mac classroom. Other commonly used printers include “sccolor” the color laserjet, “scb14_1” in the main terminal room, and barker1 in the Barker Center.

lpr -P [printer_name] [file_name]

This is the basic and most commonly used print command. It sends the file [file_name] to the printer [printer_name].

lpstat -a

This command lists all available printers, in a long and not-terribly-useful format. It’s often easiest to just read the printer’s label to determine its name.

lpq -P [printer_name]

When you send a file to a printer, your print job is placed in a queue (line) with any other jobs that are waiting to be printed. Each job in the queue has a unique ID number, its job number. By running the lpq command, you can see the jobs which are queued for a given printer. This is often useful if you want to see which printer is least busy at a given time.

lprm -P [printer_name] [job_number]

Occaisionally you may print something accidentally, or may print too many copies of something. The lprm command allows you to remove (cancel) a pending print job. You must first run the lpq command to get the job number of your job, and then you can use the job number to cancel the print job. Note that the printer queues and prints jobs very quickly, so it is possible that a job will be printed too fast for you to cancel.

enscript -P [printer_name] -2rG [file_name]

Converts plain text (ASCII) files [filename] to PostScript format and prints them to a laser printer. With the options listed, enscript uses a two–column, landscape format in gaudy mode (page headings, dates, and page numbers are printed in a flashy style).

A Final Note on Printing

Harvard's printing architecture is exceedingly complex. The difficulties of tying together Mac, PC, and Unix printing, and of maintaining several dozen printers spread across campus, means that everything does not always work perfectly smoothly. Sometimes printers will be out of toner or may bleed ink, and very occaisionally printing budgets are charged incorrectly. If you experience any problems with printing, please see a User Assistant, either in your House or Dorm or at the Science Center Help Desk. User Assistants can credit your printer budget for any failed print jobs if necessary.

Conclusion: Where to Go from Here

Man Pages

Most Unix programs come with manual pages (most often referred to as "man pages") that explain their usage. To view man pages, you use the **man** command, followed by the name of the command whose manual you wish to read. Thus

```
fas% man pine
```

will display the manual for pine. Likewise, **fas% man man** will display the manual for the man command itself! Man pages can sometimes seem difficult to understand, because many were originally written with a target audience of computer scientists in mind. However, some man pages, such as that for ytalk, are very clear and easy to read. In any case, reading the manual pages is almost a good idea when you are faced with a problem in Unix.

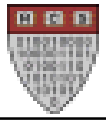
One of the things man pages are most useful for is that they usually list all of the command-line options that may be passed to a program when it is started. For programs such as **ls** or **find** that have many different options, man pages can be an invaluable reference.

Sometimes you don't know the name of a command, but would like to search for a command to do a particular thing. To search through all man pages, use **man -k**. For example, **man -k email** will return a list of various programs related to email, including pine. Likewise, if you know a command name, but not what that command does, you can use the man pages to find out.

Security

We cannot emphasize enough the importance of being conscious of computer security issues. The FAS computer cluster is in general extremely secure, but no system is perfect. Sometimes malicious computer users, or "crackers", break into systems by "sniffing" passwords from the network, in effect eavesdropping on someone else's login, and then use that information to log in to a system themselves. Thus you should never keep truly sensitive information (such as financial records or credit card information) online unless it is protected by encryption. Always use SSH rather than Telnet to log in, if possible, and be sure you choose a hard-to-guess password.

Furthermore, recall that much of what you do on public computer systems is visible to other users. A good analogy is that of a public library: while other people cannot in general see over your shoulder (see what's on your screen), and they can't take things out of your backpack (they can't access your files) unless you let them, they *can* see that you're in the library (logged in) and the title of the book you're reading (the name of the program(s) you're running. If you have any concerns about computer security or about your online visibility, contact **security@fas.harvard.edu**.



To carry the library analogy further, there is an appropriate behavior code for all users of the FAS computer systems, just as there is an appropriate behavior code for patrons of a library. Don't shout (don't send unsolicited bulk email, or "spam"), and don't go around following other people too closely. It's OK to finger a friend once in a while to see if they're logged on so you can talk request them; it's *not* OK to finger someone every five minutes so you can track where they are. All users must read and abide by the FAS Acceptable Use Policy; if you have any questions about it, please don't hesitate to speak with your User Assistant or contact **security@fas**.

Learning More About Unix

For most users, this seminar has covered all the Unix skills you will need; pine, ph, and talk are enough for most student's needs on a daily basis. However, the Unix operating system offers much beyond what we have covered here today, including very powerful programmingenvironments, great reliability and stability, and everything from word processors to games to graphics software used to produce Hollywood's special-effects blockbusters. To learn more about Unix, we highly recommend the extensive series of books by O'Reilly & Associates, which covers most every aspect of Unix in one book or another. In particular, both their *Using Unix* and *Unix in a Nutshell* are highly recommended.

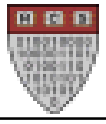
There also exists a version of Unix for PCs and Macintoshes, called Linux, which has gotten great amounts of press lately. If you're interested in learning more about Unix, possibly the best way to do so is to run it on your own computer. However, this step is not for the faint-hearted; Linux is a difficult operating system to install or maintain, and it poses great security risks if not properly maintained. Therefore we do not recommend installing Linux for any but the most serious user who knows what they're getting into.

If you have any other questions about Unix or your FAS Unix Account, please speak with one of our assistants after this seminar, or contact us at **seminar@hcs.harvard.edu**.

The Harvard Computer Society

The **Harvard Computer Society** (HCS) welcomes anyone with an interest in the uses of computing at Harvard and around the world. Our projects include teaching seminars, maintaining and administering several servers, producing various publications both printed and online, developing new software, and anything else at the cutting edge of technology that interests our members.

No computer experience is necessary to participate in the HCS. To get involved or to find out more information, please send e-mail to **info@hcs.harvard.edu**.



SEMINAR EVALUATION FORM

We would appreciate very much your evaluation of this seminar. When you have completed this form, please give it to one of the seminar's staff members or leave it by the door as you leave.

Thank you for your time.

Please write the date and time of this seminar: _____

Please rate this seminar by circling one answer for each category.

Overall experience:	**Poor**	**Good**	**Great**
Instructor:	**Poor**	**Good**	**Great**
Assistants:	**Poor**	**Good**	**Great**
Pace:	**Too slow**	**Just right**	**Too fast**
Level:	**Too basic**	**Just right**	**Too advanced**
Detail:	**Too little**	**Just right**	**Too much**

Do you feel comfortable using your Unix account now? If not, why not?

What can the instructor and assistants improve upon?

What, if anything, should we add to or omit from this seminar?

What additional seminars would you like us to offer? Would you attend?

Do you have any other comments or suggestions?