Introduction to Unix & SDF

Gene Michael Stover

created Saturday, 2003 December 6
updated Wednesday, 2012-12-19T08:34:26-0800

Contents

Chapter 1

To Do

1. Consider decomissioning this document. Has someone else done a more up-to-date one? Look on SDF. If so, link to it at the start of this.
2. Add another link for Vi: `http://www.onlineitdegree.net/vi-editor/`.[1]
3. How to make a web site on SDF. Probably mention the `mkhomepg` (or whatever it's called) command & give pointers to other articles about HTML & CGI.
4. Mention Tweak membership on SDF.
5. Mention some games.
6. Better discussion of membership levels on SDF. This is a toughie to do because this is mostly an introduction to unix, with only some SDF-specific goodies, not a full SDF policy document. Maybe it would be suitable to mention that any unix system might allow only certain users to run certain programs, & I could give brief details of SDF's membership levels. This was requested or suggested by at least two people on *bboard* / GENERAL, so it's not a frivolous request. Hmmm...
7. Discuss Zmodem (*rz*, *sz*).
8. Put it in gopher space.

[1] Thanks, Melissa!

Chapter 2

Introduction

This is for people who are new to unix, especially people who are new to unix & new to http://sdf.lonestar.org/[1] (SDF), a public access unix system.

I sometimes hang out in the chat rooms on SDF, & almost every day someone enters who has recently created an account & asks how to get started using unix & the features of SDF. It's an awkward situation because there is a lot to know to get started with unix; it would be difficult to explain it all in a chat room. What's more, most of the regulars in those chat rooms have seen & answered the question enough times that they understandably do not look forward to doing so again.

So I'm answering that question here, once. If all goes well, when someone enters a chat room & asks how to start learning about unix, someone can point them to this article.

2.1 What to read first

If you already know what unix is, or if you don't but you also don't enjoy reading about petty controversies about a definition, skip the "What is Unix" chapter & start reading Chapter 4 "Some Basic Commands".

[1] Super Dimension Fortress

Chapter 3

What is Unix?

What is unix? That's a loaded question.

3.1 Short Answer

The short answer is "Unix is an operating system". So is Gnu/Linux, which is often just called Linux. So are HP/UX, AIX, NeXT STEP (now called Apple Macintosh OS 10), & Microthought Winders (often called many uncomplimentary things).

If this short answer is enough for you, congratulations on knowing the value of keeping things simple. Now skip ahead to Chapter 4 "Some Basic Commands".

3.2 Long Answer

The long answer is "It depends on who you ask".

3.2.1 My Opinion

In my opinion, unix is a description of the function of an operating system. I even say that unix is an Application Programmer's Interface (API). Any operating system which implements the unix API is a unix; it's an implementation of unix. Gnu/Linux is a unix.[1] Gnu/Linux is often called Linux, but strictly speaking, Linux is a unix kernel, but not a whole unix operating system. There are many other implementations of unix. I can name original BSD, OpenBSD, FreeBSD, NetBSD, HP/UX, AIX, Minix, Apple Macintosh OS 10, NeXT STEP, QNX, & Lynx. There are probably others.

Now for some legal bullshit. "UNIX" is a trademark of http://www.opengroup.org/[2]. From what I can infer from their web site about their opinions of what unix

[1] Gnu stands for Gnu's Not Unix, but I say it is.
[2] The Open Group

is, they would agree with me that it's a description of the function of a family of operating systems, but they would also add "that we have certified to be UNIX". So legally, it's not a UNIX unless The Open Group certifies it as a UNIX. So a lot of those operating systems I listed as unices are not UNIXes. It's a thoroughly sad case of legalities getting in the way of simplicity & sanity.

Anyway, I say that if an operating system behaves like unix, then it's a unix, though not necessarily a UNIX$^{(tm)}$.

3.2.2 Another Opinion

Another opinion, which I don't share, is that Unix is a particular product. HP/UX is another. AIX is another. Gnu/Linux is another. You get the idea.

Fine, except that there is no product simply called "Unix" for sale today. You can't even point at a product that is the direct descendant of the original Unix that Ritchie & Thompson wrote at AT&T. Their Unix was *the* definitive Unix, the original, the one named Unix. You can't buy it today, methinks. And many of those that you can obtain today shared code with the original Unix. BSD is Unix as modified by students at the University of California in Berkeley. That's where sockets were invented. BSD branched into many implementations, several of which are alive & well today. Theoretically, each of those implementations has at least some of the original Unix code in them. Yet somehow BSD isn't a unix? Get real. I'm sure there are similar examples of cross-pollination between the other unix operating systems available today.

Another problem with the claim that BSD, Gnu/Linux, & the others are not unix is that it takes little effort to write a program that runs on all of them. There are enough differences that a non-trivial program requires a little care to be portable, but it's not like portability concerns affect the architecture of your program. All in all, it is easy to port code between HP/UX, AIX, Gnu/Linux, & the BSDs. This isn't an accident. Those products are implementations of a single idea of what an operating system should be. That idea needs a name so that when you ask me what kind of operating system I program for a living, I don't have to reply with "I'm an HP/UX, Gnu/Linux, AIX, FreeBSD, NetBSD, OpenBSD, QNX, & others programmer". I say the name for that concept is "unix". If not, then it's POSIX, but POSIX is also a trademarked term with a certification schedule from The Open Group (together with IEEE).

So somewhat out of contempt for our legal system's petty concerns for terminology, I say it's all "unix". There you have it, in more detail & controversy than anyone with a sense of perspective would ever want.

Chapter 4

Some Basic Commands

4.1 The Command Line Shell

When you login to a unix system, you'll be on a command line. It'll have a prompt like "`$`" or "`bash-2.04$ `" or maybe "`%`". There are other variations.

The command line is how you communicate with the *shell*. The shell is a program that reads commands from a command line & runs other programs. It does some other things, but they are mostly details. The main point of a shell is to run programs.

The traditional, original shell on unix is called the Bourne shell. If you're wondering why it's called the Bourne shell, ask yourself who might have written it.

The Bourne shell program is named `/bin/sh`, but that program on most modern unices is actually a pointer to the Korn shell or the Born Again shell. Those two shells are more modern than the Bourne shell, but they are backward compatible with it.

There are other shells, including C shell (`/bin/csh`) & the restricted shell. You can change your shell on many unices with the `chsh` command. Since a shell is just a program, no magic behind it, you can use damned near any program you want as your shell. You could use Perl as your shell, I've used clisp as a shell (just for an experiment), or you could write your own shell.

The important thing about a shell is that most of the things you type to it are interpreted as requests to run a program. If you type "`ls`", you're asking your shell to run a program called *ls*. If you type "`bogie -k lap -f lap woo woo -d hickie`", you're asking it to run a program called *bogie* with all that other crap as *command line arguments*.

Enough about shells for now.

4.2 Listing Files

On unix, the *ls* program gives you lists of files.

If you just type "`ls`", you'll get a list of the files in the current directory. Like this:

```
bash-2.04$ ls
#unx.tex#  RCS unx.aux  unx.bbl  unx.blg  unx.log  unx.tex  unx.tex~  unx.toc
bash-2.04$
```

Notice that *ls*, when used in this plain way, doesn't show you files sizes or permissions or anything like that. Unix is traditionally tery terse. Programs show you only the information you requested, plus error messages. You don't have progress messages like "2003-12-06T16:16 beginning to read files" and "2003-12-06T16:17 reading first file" and other nonsense cluttering your screen. The programs just print what's important.

If you want *ls* to print the details about the files, give it the "`-l`" option. That's a minus sign (`-`) followed by a letter *el* (`l`). Like this:

```
bash-2.04$ ls -l
total 52
-rw-r-----    1 gene     gene         9181 Dec  6 16:18 #unx.tex#
drwxrwx---    2 gene     gene         4096 Dec  6 14:03 RCS
-rw-rw----    1 gene     gene         1390 Dec  6 14:47 unx.aux
-rw-rw----    1 gene     gene           49 Dec  6 14:47 unx.bbl
-rw-rw----    1 gene     gene          878 Dec  6 14:47 unx.blg
-rw-rw----    1 gene     gene         4201 Dec  6 14:47 unx.log
-rw-r-----    1 gene     gene         6387 Dec  6 14:47 unx.tex
-rw-rw----    1 gene     gene         1212 Dec  6 13:49 unx.tex~
-rw-rw----    1 gene     gene          658 Dec  6 14:47 unx.toc
bash-2.04$
```

The first column, which has all those minus signs, *r*s, & *w*s, shows the http://www.dartmouth.edu/ rc/help/faq/permissions.html[1]. The next column is the number of links to the file.[2] The next column is the user who owns the file. The next is the group of the file. Then there's the number of bytes in the file. Then the date the file was modified. Then the file's name.

Here's an exercise: Type "`echo *`" on the command line, like this:

```
bash-2.04$ echo *
#unx.tex# RCS unx.aux unx.bbl unx.blg unx.log unx.tex unx.tex~ unx.toc
bash-2.04$
```

Now go figure out why or how *echo* differs from *ls*.

[1] file permissions

[2] The link count is intimately related to the structure of the unix file system. There's no need to worry about it for now.

4.3 Moving Around the File System

All unix systems have a file system whose root directory is `/`. Your home directory is somewhere inside a directory in the file system. Maybe your home directory is `/home/you` or `/usr/hm0/you` or whatever. The details depend on the unix. To learn your home directory, type "`echo $HOME`", like this:

```
bash-2.04$ echo $HOME
/home/gene
bash-2.04$
```

Unix has a notion of a *current working directory*. To see what your current working directory is, type "`pwd`", like this:

```
bash-2.04$ pwd
/home/gene/library/website/docsrc/unx
bash-2.04$
```

To change your current working directory, use the *cd* command. Give it the name of the new directory you want. If you want to move to the current working directory's parent, use "`..`". Here are some examples:

```
bash-2.04$ pwd
/home/gene/library/website/docsrc/unx
bash-2.04$ cd ..  # move to parent
bash-2.04$ pwd
/home/gene/library/website/docsrc
bash-2.04$ cd .. # move to parent again
bash-2.04$ pwd
/home/gene/library/website
bash-2.04$ cd share # move into the "share" directory
bash-2.04$ pwd
/home/gene/library/website/share
bash-2.04$ cd /tmp  # move to this absolute directory
bash-2.04$ pwd
/tmp
bash-2.04$
```

4.4 The VI Cursor Commands

vi (Section [**?**]) is an editor, but its cursor-movement commands are often used by other programs. Figure 4.1 shows *vi*'s cursor-movement commands.

The *vi* commands might not make much sense at first, but notice that you can use all of them with the fingers from the right hand. They may not be the most mnemonic commands, but they are very functional.

Some programs besides *vi* which use the *vi* cursor-movement commands include:

command	action
h	move cursor left
j	move cursor down
k	move cursor up
l	move cursor right

Figure 4.1: The cursor-movement commands from *vi*

- the mail reader called `mutt`4.5.1, but just the *j* & *k* keys in its menus
- `/usr/games/worm`, an old game & an excellent way to learn the *vi* commands
- VIPER, a *vi*-emulator for `emacs`4.9

4.5 E-mail

Most unix systems give you multiple options for how to read your e-mail.

If you want to read your e-mail with a text-only program you run from the command line (which is how I read e-mail), some common programs are *mutt* and *pine*. Just type the name of either of those programs. If you see a list of e-mail, it means the program is available, & you are running it. If you see an error message that says "command not found", it means your shell couldn't find it, maybe because it's not available on your unix.

You can figure out the basics of *mutt* or *pine* just by using them. People become almost religiously devoted to their favorite e-mail programs, so you can learn more detailed or subtle ways of using a mail program by asking someone who prefers that program. People are usually happy to tell you more than you want to know about how to use their favorite e-mail program. So pick one of them, figure out the basics on your own, & then go find a long-time user of that program & strike up a conversation about it with him.

I've never used *pine*, so I can't tell you how. I sometimes use *mutt*, so here is some super-quick info on using *mutt*.

4.5.1 mutt

The official documentation about *mutt* is at http://www.mutt.org/[3] .

When you run *mutt*, you'll see a list of e-mail messages. Each line on your screen will show a subject, a date, & who sent the message.

One of the messages will be selected. To view the selected message, type the Return, Enter, or Space-bar key (I can't remember which). You'll see the body of the message.

When you are viewing the body of a message, type the Space bar to page down. When you come to the end of the message, type the Space bar to view

[3] `http://www.mutt.org/`

the next unread message. If there are no more unread messages, you'll see the list of messages again.

When you are viewing the body of a message, you can type "`q`" to quit that view & return to the list of messages.

In the list of messages, the "`j`" key moves the cursor down & selects that message. The "`k`" key moves the cursor up & selects that message. The "`q`" key exits *mutt*.

4.5.2 Graphical E-mail Readers

If you want to read your e-mail with a graphical e-mail reader, maybe the one in your web browser, you'll need a more elaborate setup. You'll need to instruct your mail client to use your unix account for sending & receiving messages. It sends messages via the Simple Mail Transport Protocol (SMTP). It downloads messages that you receive via the Post Office Protocol version 3 (POP3). Of course, the unix system where you have your account must be configured to allow these operations. You'll need to refer to the documentation for that computer system to do all this, but it can be done if the unix system allows it.

4.5.3 My Own E-mail

For the record, I usually use the *rmail* mode in Gnu Emacs for reading & sending e-mail.

4.6 Viewing Files

If you have a file you want to read, use the *less* program to read it. For example, if the file is called `README` & is in your current directory, type "`less README`". If that gets you a "command not found" error, try "`more README`".

You'll see the first part of the file on your screen. (If it's a really short file, you'll see all of it on your screen.) To see the next screen-full, type the Space bar. To see the previous screen-full, type the "`b`" key. To quit viewing the file & return to the command line, type the "`q`" key.

4.7 Surf the Web

A good web browser to use on a text-only command line is *lynx*. You can give it an URL to read on the command line, like this: "`lynx http://lisp-p.org/`" or "`lynx http://google.com/`". Use the Space bar to page down, the "`b`" key to page up, the arrow keys to select hyperlinks, & the Return or Enter key to follow a hyperlink. Use the "`q`" key to quit.

Unless the unix system where you have an account is very trusting, you probably cannot run a graphical web browser on it. Technically it can be done using the graphical system called X (which is common on unices), setting permissions

with the *xhost* program, & setting your DISPLAY environment variable, but it usually is not allowed on a public access unix.

4.8 Upload & Download

4.8.1 FTP

To transfer files between your home computer & a remote unix, use the program called *ftp*. Your home computer almost certainly has a copy, even if you run Microthought Winders at home.

To run *ftp*, first go to a command line. Then type "`ftp ftp.freeshell.org`" to connect to the FTP server at SDF.

The *ftp* client has a command line interface. Here are some of the commands it understands:

cd *directory* Change directory on the remote FTP server.

dir Print a lsit of the files in the current directory on the remote FTP server.

ascii Tell FTP to assume that files you transfer are text files.

binary Tell FTP to assume that files you trnasfer are binary files.

hash Tell FTP to print an octothorpe character (#) for each block it sends or receives. It helps you see the progress of the file transfer.

put *filename* Upload a file.

get *filename* Download a file.

In general, use the *cd* command to enter the correct directory on the remote FTP server. Then set the correct mode (*ascii* or *binary*). Then use *get* to download a file or *put* to upload one.

The *ftp* client program supports many more commands. Use "`man ftp`" for more information about FTP.

In case you are interested: FTP refers to the File Transfer Protocol; it was one of the three requirements that created the Arpa Net which became the Internet. *Ftp* is also the name of the File Transfer Program which implements FTP.

4.8.2 scp

scp is the Secure Shell Copy Program. It's name is also a reference to *rcp*, the Remote Copy Program.

Use *scp* like you would use *cp*, but it can copy from one computer to another.

For example, if I want to copy `myfile.cpio.bz2` on my computer into the `tmp` directory in my account on SDF, I would run this: "`scp myfile.cpio.bz2`

`gms@freeshell.org:tmp/`". The *scp* program will probably prompt me for my password, then it will copy the file.

Run "`man scp`" to get more information about using *scp*.

In my experience, *scp* takes longer than *ftp* to transfer files, & it gives up easily. If either computer or the network is slow, *scp* gives up quickly. So I prefer *ftp*.

Ed Chang has told me that there is an implementation of *scp* for Microthought Winders called WinSCP, at http://winscp.sourceforge.net/eng/about.php[4] . I haven't tried it myself, but it sounds like it might be pretty good.

4.8.3 rcp

You can't use *rcp* on SDF, so this section applies when copying files between other unix systems.

An old but pretty cool program for copying files between two unices is *rcp*, the Remote Copy Program.

For *rcp* to work, you must first configure the `.rhosts` files in your accounts on the unix systems. Then you can copy files from one system to another with the *rcp* program.

For example, if I want to copy `myfile.cpio.bz2` on the local computer into the `tmp` directory in my account on a computer called `overthere`, I would run this: "`rcp myfile.cpio.bz2 overthere:tmp/`". The *rcp* program will copy the file; it won't ask me for my password.

4.8.4 When FTP Doesn't Work

If you don't have FTP on your home computer, you can transfer files via e-mail.

First, make sure you have an e-mail account that allows you to send files as attachments. You might use the e-mail account that your ISP supplied with your Internet connection, or you might create a Web e-mail account on Yahoo!, Hotmail, or any of the many other systems that provide free Web e-mail accounts.

To transfer a file from your home computer to SDF, you create an e-mail message on your e-mail account that is *not* on SDF. Attach the file; with a Web e-mail account, it'll allow you to upload the file into the e-ail message. Then send the message. Give it a minute to arrive, then login to your account on SDF. Receive the message with an e-mail program which knows how to deal with attached files. Mutt & pine both know about attachments. Use that e-mail program to save the attachment to a file.

To transfer a file from SDF to your home computer through e-mail, login to SDF & send the file to your non-SDF account as an attachment to an e-mail message. Then receive the message on your non-SDF e-mail account & save the file to your home computer's file system.

[4] `http://winscp.sourceforge.net/eng/about.php`

4.9 Editing Files

There are two great religions in the modern world: *emacs* and *vi*.

The two most popular file editors on unix are *emacs* and *vi*.

Emacs is more properly called Gnu Emacs because emacs is a way of programming file editors, & Gnu Emacs is just one editor written in that way. I believe Gnu Emacs was the first big production product of Gnu. Richard Stallman wrote it himself. It is a Lisp with screen-handling functions built-in. It's a great big powerful editor, an operating environment, & I love it. To try emacs, type "`emacs`", wait for the screen to update, then type the Escape key, the `x` key, then "`info`", then press Return. Carefully read & follow the instructions you see on the screen. It's kind of difficult to learn emacs from the start, but once you do, it's great.

The other popular editor on unix is *vi*. I don't use vi much. Two web sites that discuss it are http://www.emerson.emory.edu/services/editors/vi/vi.html[5] and the http://www.thomer.com/vi/vi.html[6].

[5] Mastering the VI editor
[6] Vi Lovers Home Page

Chapter 5

Learning More

5.1 The Man

Most of a unix system is documented in the online manual, called the *man*. If you want to know about a program, you look it up in the *man*. For example, if you want to learn more about the *ls* program, type "`man ls`".

The man also has a simple search capability. The "`-k`" command line option tells the man to give you a list of entries that mention a particular keyword, which you also supply on the command line. If you want to get a list of the manual entries that menion networking, you would type "`man -k network`".

The man is divided into sections. If you don't specify a section when you ask the man to lookup documentation for a program, the man takes a guess about what section you mean, & it usually assumes you mean the most general, high-level section available (section 1). You can specify a section as the second argument to *man*, before the keyword. For example, if I want to read about the `select` function in section 2 (the kernel API), I would type "`man 2 select`".

Here are some things I recommend reading in the man to get started with unix. I give you the exact command to type, then an explanation of why I recommend that entry in the man.

man man This tells you how to use the online manual.

man 1 intro This tells you about the general section of the online manual. You should also try "`man 2 intro`" & all the other intros for sections 3 through 9.

man mutt Tells you how to use *mutt*, the e-mail reader.

man pine Tells you how to use *pine*, the e-mail reader.

man sh Tells you how to use & program the shell.

man 4 tcp Tells you how to program TCP sockets. I suggest it here as an example of the breadth of information that's available in the online manual.

5.1.1 The man is for real

I tell newbies about the man all the time, but they usually don't try it.

I don't know why they don't believe me. I started using unix in college in 1985. In a class, the professor gave everyone an account & told us to login after class & type "`man man`". He didn't tell us anything else, not even what "man man" would do. By exploring the manual on my own time for a couple of evenings, plus a couple of conversations with the unix gurus that worked in the lab, I was pretty good with unix. It was an excellent way to start using & programming unix.

Maybe the newbies do believe me, but they're lazy.

5.2 Gnu info

Besides the unix *man*, Gnu/Linux computers usually have some information in a hypertext system call *info*.

To use *info* from the command line, just type "`info`". If you are a Gnu Emacs user, you can run emacs & type Meta-x `info`, & you'll be presented with the same data in the same format that *info* would.

Info is an interactive hypertext system. It predates the Web by at least five years, methinks. Read & follow the instructions on the screen & you'll be alright.

Info systems are not always maintained as well as the man, but most are pretty good, & some have a lot of excellent information. It is often well-written, too. It's an enjoyable read, & since you can page through the entire system one screen-full at a time by typing the Space bar, it's convenient to read.

Though it's not for beginning users, I'll mention here that one of my favorite sections in *info* is the documentation for Autoconf.

5.3 Books

I can't recommend any books on using unix. I learned it so long ago that I don't remember what books I used, or even if I used any books. Here are some interesting books about unix, though not exactly about *using* unix.

I like Mike Gancarz's book, *The Unix Philosophy*, a lot. It talks about the reasons behind writing applications the unix way: with small programs piped together, & with flat, plain text data files. I have heard that Eric Raymond's book, *The Art of UNIX Programming*, is similar & also good, but I haven't read it myself.[1]

Even though we're talking about using unix, I think an understanding of how operating systems work is useful to know, & Andrew S. Tanenbaum's *Modern Operating Systems* is an excellent book. It includes a good section about unix. He points out that the defining characteristic about early unix was its 40 system

[1] I infer from the book's title that Eric Raymond would agree with me that "unix" is a functional description of a class of operating systems, not a particular implementation.

calls. We've added more system calls, but the unix kernel still has a fairly small number of elegant, well-chosen system calls. That's an important part of what makes unix what it is.

5.4 Folklore

One of the cool things about unix is that it has played an intimate part in the folklore of computing since the late 1970s. There are all sorts of interesting tales & personalities to explain or help you remember why something was done a particular way. The history is not hidden inside a corporation. It's out in the open, & lots of people know it.

That's also one of the bad things about unix because it means you need to talk to people to learn what you need to learn next. Hopefully people will write downt he folklore & also link to it so that new unix users & programmers can find it.

You can learn the folklore by talking (or typing) to an experienced unix user or programmer. A good place to read about it online is http://catb.org/ esr/jargon/html/[2].

[2]The Jargon File

Chapter 6

Super Dimension Fortress

Here is some quick reference information about using http://sdf.lonestar.org/[1] (SDF). I'm not a member of SDF's staff, so the information here is not definitive. If you find a discrepancy between what I say & how things actually work on SDF, it's an error in what I've said, not in how SDF behaves.

6.1 Getting Information on SDF

Use the *faq* or *help* commands to get information about SDF. These are not standard unix programs. They are specific to SDF.

The *help* command mostly gives you lists of other commands. If you type "`help`", you'll see a menu like this:

```
SDF Help System - v8, 1993 - sdf!smj

 [a] Directory and File Utilities
 [b] Email Information
 [c] USENET Information
 [d] User and Process Information
 [e] Tutorials and Very Useful Commands
 [f] ARPA Services (internetworking)
 [g] Homepage, VHOST and MetaARPA Utilities

 [q] Quit SDF Help System

Your Choice?
For more help, type 'faq'
For a list of basic UNIX commands, type 'unix'
to remove your account, type 'delme'
```

[1] Super Dimension Fortress

Most of the letters in the menu will give you lists of commands. After you quit the *help* program with "`q`", you could run the commands or read more about them in the online manual I discussed earlier. Some of the programs are specific to SDF & are not discussed in the man. SDF's excellent *com* program is an example of a program that is not discussed in the man.

To me, the *faq* program is more useful. In it, you can find short explanations of unix file permissions, uploading & downloading, SDF's policies, & the history of SDF.

To enter the faq, type "`faq`" on the command line, & the *faq* program gives you a list of sections. This is faq's main menu. You can exit by typing "`q`" here. If you type "`g`", it will ask you for the name of a section. You enter a section name & press return, & you'll see the section's menu.

In a section's menu, you have a numbered list of questions or topics. If you type "`g`", & press return, you'll see main menu again. If you type "`t`", you'll be prompted for the number of a question or topic. Type that & press return. You should type the number exactly as it appears in the menu, including leading zeros.

Some things you can learn from the *faq* include:

- SDF's policies (& prices, methinks)
- brief history of unix

6.2 COM Mode

SDF has its own chat system, called `com`. It's like a stripped-down IRC, but much cooler than IRC because people actually talk, & you don't get unwanted pop-up download files & you don't have tons & tons of bots waiting to download "warez". Anyway, `com` is like IRC except that `com` is cool.

To use com, type "`com`" on the command line. You'll be plopped into the lobby, which is the default room.

Com uses *single-key commands* mostly. If you are new to `com`, you might want to read COM Command Discussion COM Command Discussion (Section [**?**]) in this same article. If you are having trouble exiting from the `com` program, then you definitely need to read COM Command Discussion COM Command Discussion (Section [**?**]).

Table 6.1 shows the most frequently needed `com` commands.

Also, from COM Mode, you can compete with others in games of Nethack, Netris, & some others. If you can master the "h" command in COM, it will show you the commands for playing these games.

6.2.1 COM Commands Discussion

The `com` commands are *hot*. By that, I mean that you type a command character, & `com` immediately takes action. You must be careful to type only the command character you intend. If you type other characters along with the

key	meaning
h	Get a list of all the commands
?	Get a list of all the commands
space bar	Enter text that you "say" into the room
return	Enter text that you "say" into the room
g	goto another room
q	Quit COM Mode

Table 6.1: The most frequently needed commands for `com`

character you intend, one of those other characters might be interpreted as a command, & the character you intend will be interpreted as an argument to that command. It might sound like I'm being pedantic, but people who are new to `com` often have this problem.

Table 6.2 shows the proper sequence of steps for entering a command in `com`. It assumes you are already running `com` & that you are not currently processing any `com` commands. You are just reading the text in the room.

Step number 4 in Table 6.2 is critically important & deserves further discussion. In step number 4, you touch (or "type") the command key. You do not touch (or type) any other keys on your keyboard or on any other keyboard. You don't need to "enter" the command by typing the Return or Enter key on your keyboard; `com` will acknowledge your command key immediately.[2]

A common error for new users is to type the Return (or Enter) key after they type the command key. `Com` executes the command key fine, then it notices the Return key. Return is a command, so `com` executes it. The Return command gets you a prompt so that you may enter text that you "say" for other people to "hear" in the room. Because of the Return key, `com` is not interpreting the new user's keystrokes as commands.

Then the new user tries to type another command, such as the "q" (quit) command, but `com` interprets it as text to say into the room. The new user is under the impression that he needs to end commands with Return, so he does that, which causes `com` to print "q" into the room for everyone to see. The new user is confused about why his command doesn't execute, so he types "h" & Return again. The "h" is interpreted as a command & gets him a menu, but the Return is also interpreted as a command & puts him into enter-a-line-of-text mode. He sees the "q" command in the menu, types it & Return, & the cycle repeats.

All of this discussion may seem silly to most people, but a lot of new users have amazingly huge amounts of trouble when they learn to enter a command into `com`.[3]

[2] "Immediately" is a relative term. Lag might prevent `com` from reacting instantly, but it will react without you pressing any other keys.

[3] Some people keep track of the longest time a new user has required to exit from `com`. I have heard that the record is over 23 minutes.

1. Put both hands in your lap so that neither hand is touching the keyboard or any of the keys on it.
2. Raise one hand, but don't touch the keyboard with it yet.
3. On your raised hand, extend the index finger. Curl the other fingers & the thumb into your palm.
4. Use the extended index finger of your raised hand to touch the command key on your keyboard.
   (a) Touch & release the command key. In other words, "type" the key.
   (b) Do not touch the Space Bar before or after touching the command key.
   (c) Do not touch the Return key before or after touching the command key.
   (d) Do not touch any other keys before or after touching the command key.
   (e) Do not touch any keys at all with any of the other fingers on your raised hand.
   (f) Do not touch any keys at all with the thumb of your raised hand.
   (g) Your other hand remains in your lap. Do not use it or the fingers on it to touch the keyboard or any of the keys on the keyboard.
5. Without touching any keys, return the raised hand to your lap.
6. Relax the fingers & the thumb on the hand that was raised & which is now in your lap.
7. Read the text on the screen to see what happened.
8. If you see the result of your command, you did it correctly.
9. If you see something else, you did it incorrectly. In particular, if you see "`[`*`your-handle`*`]` *`the-command`*", where *your-handle* is your login name & where *the-command* is the command key you intended, you did it incorrectly.

Table 6.2: The proper & detailed sequence of events for executing a command in `com`

6.2.2 COM Session Example

Let's walk a new user named *newbie* through his first session in `com`.

1. Newbie starts on the unix command line.
2. Newbie enters COM Mode by typing "`com`" & pressing the Return key.
3. Newbie might have to wait a few seconds, but she eventually sees `com` start. (I think new users see a menu of the `com` commands & have to answer Yes that they have read & understood the menu.) After `com` starts, Newbie will be in the *lobby* & will see a list of the people in that room. (The lobby is the default room & is where most of the chatting happens.)
4. Newbie wants to say hello to everyone, so she types the Space Bar & waits until she sees a "`[newbie]`" prompt. Then Newbie types "Hi all. I'm new." & then types the Return key.
5. Everyone in the room will see "`[newbie] Hi all. I'm new.`".
6. *Newbie is doing well so far, but now let's have her make a mistake.*
7. Newbie wants to leave `com`, but she can't remember the command, so she types "h" to get a menu & pressed Return. *This was a mistake. Newbie should have typed "h" alone & not pressed Return.*
8. `Com` *sees Newbie's "h" command & prints the help menu. That's good, but* `com` *also sees Newbies Return & assumes Newbie wants to "say" a line of text into the room. So* `com` *prints a "*`[newbie]`*" prompt & waits for Newbie to type some text, but Newbie is busy reading the help menu & doesn't notice the prompt.*
9. Newbie sees the "q" (quit) command on the help menu & types "q" & then Return. *This is a double mistake.* `Com` *thinks Newbie is typing a line of text to say into the room, so her "q" is not interpreted as a command. Also, Newbie is still under the incorrect belief that she must end* `com` *commands with a Return.*
10. Because of Newbie's mistake in the previous step, `com` prints "`[newbie] q`" into the room.
11. Newbie is confused that her "q" command didn't work, so she types "h" & Return. This takes her back to step 7.

    She'll repeat this who knows how many times. Other people in the lobby will notice her problem & try to help, but Newbie has to go through the error many times before she stops typing Return after her commands.

    The problem is compounded by `com`'s modality & Newbie's failure to realize that `com` is modal. Sometimes, `com` is waiting for Newbie to type a command. Some commands are single key. (The "q" & "h" commands

are examples.) Other commands wait for Newbie to type something, & during that time, `com` does not interpret her keystrokes as commands. (The Space Bar and Return commands, both of which are for saying text into the room, are examples.)

The other people in the room see what Newbie is going through, but there isn't much they can do because they can't see Newbie's screen & help her figure out whether `com` is waiting for her to type a command or waiting for her to type text. What's more, whenever they say something like "Type *h* by itself", Newbie types an "h" & then a Return. So Newbie keeps going around & around.

12. Eventually, Newbie figures it out while `com` is waiting for her to type a command. At this time, Newbie types "q" (& does not type any other characters at all).

13. Newbie might have to wait a few seconds, but she eventually sees `com` print an "Unlinking tty" message, & then Newbie is dropped back into her unix command line.

6.2.3 What to Do if You Are Caught in the Newbie Loop

The problem is that `com` is modal & new users sometimes fail to realize that.

What does it mean for a program to be **modal**? It means that sometimes, `com` is waiting for you to type a command, & sometimes it is waiting for you to type arguments to a command. Some commands are single key. (The "q" & "h" commands are examples.) Other commands wait for you to type something, & during that time, `com` does not interpret her keystrokes as commands. (The Space Bar and Return commands, both of which are for saying text into the room, are examples.)

If you find yourself in the `Com` Newbie loop, here's how to get out of it. Follow the exact steps from Table 6.3. Type only what it says to type. Do not type anything else. Do not type Return unless a step in the table says to type Return.

Here's why the steps in Table 6.3 work:

1. If `com` is waiting for you to type a command, then you could type the command anyway, but you type Space, which gets you a prompt (`com` isn't interpreting commands). Then you type Return, which ends that prompt. So now `com` is waiting for a command again. So the Space Bar & Return sequence was unnecessary, but it did not harm. (People in the room won't even see it because if you try to "say" an empty line, `com` doesn't print anything.)

2. If `com` has prompted you & is not interpreting your commands, the Space Bar won't do anything useful, but the Return key will end that prompt. If you had mistakenly typed Return earlier & `com` was waiting for you to say something into the room, the Space Bar will do nothing, but the Return

1. Type the Space Bar.
2. Type the Return (or Enter) key.
3. Type "q".
4. Wait for `com` to print "Unlinking tty".
5. You will be at the unix command line prompt. (Test this by typing "pwd" & then Return.)
6. You may now re-enter `com` or go do something else.

Table 6.3: The steps to get out of the `Com` Newbie loop. Do exactly what the steps say. Do only what the steps say. Do not type anything else.

will end that mode, & `com` will now wait for you to type a command.. (If you had typed things before the Space Bar, you will "say" them into the room.)

6.2.4 COM History

I have heard that early versions of COM Mode were fairly simple shell scripts that ran *tail* on a common file. Excellently unixly elegant. COM Mode is in about version 6 now, & it works across a bunch of computers networked at SDF, so I don't know if it's still just some shell scripts.

6.3 Bulletin Board

If you type "`bboard`", you'll enter the bulletin board. Its commands resemble those of the *faq*:

g Prompts you for a section name, which you type. Then takes you to that section.

t From a section's menu, prompts you for a message number (which you must enter exactly, including leading zeros). Then it prints the message & all replies to your screen.

q Quits the bulletin board.

Chapter 7

Programming

So you want to learn to program? Good. Programming is fun, creative, & fascinating. (It used to be a well-paying career, too. I guess it still is, if you are in India or China.)

7.1 Language

There are billions & billions of programming languages. Well, not that many, but there's a damned lot.

You'll probably want to learn one of the most popular ones, which is too bad because most of the currently popular languages aren't very good. They are complex versions of the exact same features that have been available in programming languages since the 1960s. That's right: Even after forty years of hyped, new languages, we've effectively come full-circle to Algol 1960.

But I'm a bitter & opinionated old programmer, so ignore what I just wrote. Here's some actually useful information:

You can use just about any language you want on unix. At the moment, it might be difficult to find a fully functioning C# compiler for unix, but one is coming, & there is no technical reason that C# won't work on unix. Many other programming languages are available for unix. Heck, many were developed *on* unix. So you can pick pretty much whichever you want.

No matter what you pick, I highly recommend that you also do some plain, vanilla C. That's because the programming language of unix is C. When you need to write some programs to experiment with some feature of unix to learn the details of how it works before you use it in a larger program you are writing, the best language for writing those experimental programs is C because it hides the least from you. It's not that C is better, & I'm not saying it's better for writing applications. I mean that because unix's API is for C, C is the best language for discovering how unix's system calls work. C doesn't hide anything from you.

There are two books you need to do serious C programming.

The first is the original & the classic *The C Programming Language*, by Kernighan & Ritchie ([**?**]). Yeah, it's old & small, but since when is information bad just because it's old? And since when would you rather take days to read a humongous tome that contained no more information than in a small book like K&R?

The second book you need for C programming is *The Standard C Library*, by P. J. Plauger ([**?**]).

7.2 Languages on SDF

Here are some languages that are available on SDF. You may have to be validated or even ARPA to use some of them, but I don't know for sure.

language	**pathname**
Bourne (or other)	/bin/sh
C	/usr/bin/gcc
C++	/usr/bin/g++
Open Lisp (not Common)	/sys/pkg/uxlisp/uxlisp
Perl	/usr/pkg/bin/perl
Python	/usr/pkg/bin/python

To learn a language, I recommend studying at least one book, not just online sources.

7.3 Theory

Obtain, read, & understand a book about data structures. Learn it & learn to love it. Data structures separates the scripters from the programmers.

Chapter 8

Hacking

So you want to hack? To break into computers? Take them over? Get root access? Crash them or use their CPU cycles to do work for you?

Aim high, but I have some advice. Those activities aren't http://catb.org/ esr/jargon/html/H/hack.html[1] . Those activities are part of http://catb.org/ esr/jargon/html/C/cracking.html[2] . For more information about the meaning of *hack*, you might read two other chapters from The Jargon File:

- http://catb.org/ esr/jargon/html/meaning-of-hack.html[3] and
- http://catb.org/ esr/jargon/html/crackers.html[4] .

You might find some people in COM Mode who would like to talk about these things, but most of them are really tired of kiddies coming into the lobby & asking "Do any of you know how to hack?" So you'll need to be more subtle. It's like real life; you don't walk into a room, interrupt the current conversation, & loudly ask if anyone will sell you nose candy. It's just rude.

Information about security holes is pretty easy to find without help. Just use Google to search for things like "computer security virus hole exploit". What you need is the knowledge to make use of that information. You need to learn straightforward programming techniques, especially networking[5], before you can make use of all that security information that is readily available. So instead of asking people to teach you to crack, it might be more worthwhile & fun to talk programming with people.

While we're on the topic, I guess I can throw out a bone. Take a look at "CIFS: Common Insecurities Fail Scrutiny" ([**?**]).

Also, be aware that most of the protocols in use on the Internet are defined in the Request For Comment (RFC) documents. One database of RFCs is

[1] `hacking`
[2] `cracking`
[3] `The Meaning of 'Hack'`
[4] `Crackers, Phreaks, & Lamers`
[5] Lookup *socket*, *connect*, *listen*, & *accept* in the man.

http://www.rfc-editor.org/[6]. Another such database is http://www.cis.ohio-state.edu/cs/Services/rfc/index.html[7].

Why do you care about the RFCs? Because to find a hole in some protocol, you need to know the actual protocol, not just heresay that people slip to you in a chat room. To know the protocol, you need to refer to its definitive source. For most protocols on the Internet, that's the RFCs.

For example, let's say you wanted to research holes in Internet multicast. You could go to either of those RFC databases & search for "multicast". You conclude that any of these RFCs were worth your time

id	title
RFC1112	Host extensions ofr IP multicasti
RFC3513	Internet Protocol Version 6 (IPv6
RFC3261	SIP: Session Initiation Protocol

That's just an example. By the way, if none of those title make you at least a little curious to know what's in those documents, then you might not be cut-out to be a computer hacker or a cracker.

[6]The RFC Editor
[7]RFCs at Ohio State University

Chapter 9

Meta

This document is available online in several formats:

- HTML is at `http://lisp-p.org/unx/`.
- PostScript is `http://lisp-p.org/unx/unx.ps`.
- DVI is `http://lisp-p.org/unx/unx.dvi`.

There are no plans to make it available in Pointless Document Format (PDF).

Bibliography