Dan’s Unix Cheat Sheet

Index

Important Note: I am learning Solaris on a dedicated Intel platform, NOT on a production system. The notes I’m taking are for that system. Just because something is documented here doesn’t mean it is a good idea to use on a production system!

Also note that I am using the Korn (ksh) shell and unless noted assume that’s what is necessary to make the example work. You can get into the Korn shell at any time by typing ksh.

Linux Note: These notes were written primarily for Solaris; however, since I must use RedHat Linux to support certain network test applications, there are note in here that also apply to Linux. In those cases where there is a difference between Linx and Solaris I will try to point out the difference.

bash verses ksh

When I originally wrote much of this document, I was using Sun Unix. The default shell for Sun was sh. I chose to use ksh (Korn Shell) instead and most of these notes were written for ksh. 10+ years later I am far away from Sun and using only linux with bash. For the most part, the syntax for bash is very similar to ksh and most of what I have put in here works fine for linux/bash as it did for Sun/ksh.

I have found a table of differences here, and you will note there aren’t many:

http://www.unix.com/answers-to-frequently-asked-questions/12274-difference-between-ksh-bash-different-shells.html

Getting Help

If you are using Unix regularly, I suggest “Unix in a Nutshell”. This book is a quick summary Solaris Sun’s Unix. There is also “Linux in a Nutshell” for Linux. I also found the “Red Hat Linux 9 Bible” to have a good explanation for configuring various services such as mysql and Apache.

If you need help, ask the Unix group, don’t ask me! I’ll just tell you to talk to them or look at this guide!

The **help command** is

man <command>

Displays the “manual” page for the command in question.

```
man intro
```

will give you some introductory material as well as a list of some commands in alpha order. You can also type

```
apropos “(1)”
```

to get a larger list (note below that this may not work).

apropos <string>

You might be able to search all of the documentation for a string using apropos:

```
qa1#apropos logout
keylogout       keylogout (1)    - delete stored secret key with keyserv
last            last (1)         - display login and logout information about uss
logout          logout (1)       - shell built-in function to exit from a login n
```

This shows you which commands have the work “logout” in them. This depends on if the index was generated (by catman) on the system you are on (if it isn’t you will get an error regarding a missing “windex” file).

whatis <command>

This command will give you a one line summary for a command. Like apropos, this requires the windex file so it may not work for you.

Extremely Useful: Automatic Filename Completion: Some of the Unix files I’ve played with have long names and you have to type the case exact. You only need to type in as much of the filename as is unique.

Solaris?: Type in part of the filename and press the escape key 2 times. It will fill out the rest of the filename.

Linux: ESC-ESC works, but TAB is faster.

If you didn’t type in enough to make the filename unique, press escape then = and you will see a list of files that match what you have typed so far.

Be cool, speak the language

Punctuation Mark	Term Used by the Masses	Unix Enlightened Term
!	Exclamation Point	Bank, Shriek, Ball-bat
"	Quotation Mark	Rabbit Ears
*	Asterisk	Splat, Star
`	Apostrophe	Single Quote, Tick
.	Period	Dot, Point

(1) This command doesn’t exist in Linux
(2) This command doesn’t exist in Solaris

Really Useful Commands	
Unix Command	**Function**
cp	copy
find / -name <fname>	starting in the directory / find the file named <fname>
grep “<string>” *	look in all files (*) for <string>. You can use regular expressions inside of <string>
history	show command history
less	Like more except you can also go forward in the file: f or space: forward b: backward
ls -l	list all files in long format
ls -f	List all files, appending a type indicator
ll	For all systems I use, I have defined ll to be the same as ls ls -l
more <filename>	Output file named <filename>to terminal, stopping at each page. Press “?” at the more prompt for all possible movement commands while in more. less is preferred over more.
mv <source> <destination>	move - there is no rename - you can only move
PRTSCR {the button}	In Linux Mint, this will allow you to copy the entire screen to clipboard or file.
ALT+PRTSCR	In Linux Mint, this will allow you to copy the active window to clipboard or file. If you go to Accessories | Take Screenshot, you can select a region. SHIFT-PRTSCR to do this isn't working for linux mint.

^r	Let's you search for a command to redo and then modify it before execution. ^r then start typing the cmd you wish to execute. You can type some text then repeatedly use ^r to cycle thru all commands that match the string entered.
r `<string>`	do last command starting with <string>. No parameter redoes last command. Using a number will redo the nth command.
rm	delete
^z	“Break” or interrupt the program currently running. Then type “fg” to resume.

Commands Summary	
Unix Command	**Function**
`alias` `ls=”ls -F”`	create “UDC’s”. To make permanent, put in .profile.
`bc`	Calculator. To send commands to it, pipe them like this: echo ‘1 * 2 + 3' | bc
`cancel` `<request id> [<printer>]`	Cancel output sent to a printer

`cat` `<filename>`
 	
`cat` `<filename> | grep xxx`
 	
`zcat` `<filename>`
	Dumps contents of File to terminal.
	
	Often, used with grep to find a particular string in file.
	
	zcat will perform cat function on gzipped files.

`cd` `<dir>`	change directories to <dir> Some special directories: If <dir> completely ommited, same as cd ~ ~ your home directory ~jay jay’s home directory - previous directory - ***very handy for jumping between dirs*** ~+ current dir ($PWD) ~- prev dir
`cp` `<source> <destination>`	copy
`clear`	clear screen (like DOS cls)
`date`	display current date and time
`date` `+%Y-%m-%d`	Display date in a specified format. https://linux.die.net/man/1/date
`date` `-d "<relativeDate>"`	Defaults to *now*. Other possibilities would be *today*, *tomorrow*, *yesterday*, *2 days ago*. https://www.cyberciti.biz/tips/linux-unix-get-yesterdays-tomorrows-date.html
`debsums` `-as <pkgname>`	This is used to verify check sums of all files in a package (e.g., fail2ban). -a will do config files, normally omitted. -s is silent, reporting only bad checksums. `debsums -s` will check all packages reporting only changes.

dd if=/dev/sdax of=myfile.img bs=4M conv=sync,noerr	Copy Partition to a normal file. See also pv command.

dd if=/dev/zero of=bigfile \
  bs=1M count=<n>G \
  status=progress
rm bigfile
	Before zipping the image of a partition, it is a good idea to fill empty space with zeroes. This is done by copying /dev/zero to all empty space.
	
	Use df -h first to get an idea of how much space you will need

df -b	free space -b in bytes (not in linux) -h in Mbytes -k in Kbytes
diff -u oldfile newfile >file.patch	Creates a list of differences between oldfile and newfile. Use patch program to apply changes in file.patch to oldfile to get newfile. See patch.
du -sh <directory>	Report the size of the directory (all files and subdirectories) in human readable form. ncdu is a ‘graphical’ version of du. **Use this instead, if available.**
echo	Like DOS echo
egrep <expression> <files> (1)	Grep using full regular expressions. This version of grep allows you to use the newer grep expressions. In linux ‘egrep’ works, but it is really just calling ‘grep -E’.
env	displays environment variables (like DOS set)
expr <expression>	Calculate an expression. Many operators need to be escaped. Example: echo $(expr 2 * 2)
file xyz	tells you if xyz is a directory or file (and will try to guess the contents)
find / -name <filename>	starting in the directory / find the file named <filename>
find /usr /var -name fname.txt	find in multiple directories

find `/ -size +5000k -printf “%a %s\t%p\n”`	Find all files > 5MB and list them
find `. -atime +7 -exec ls -l {} \;`	Find and list all files older than 7 days
find `.-atime +7 -exec rm {} \;`	Find and delete all files older than 7 days
finger	display logged in users
fuser `-u <file>`	Displays who is accessing <file> `fuser -u /dev/devname` allows you to see who is using a specific device. This is useful when trying to dismount a CD or floppy.
fuser `[-k] 8080/tcp`	Report PID of process that has created port 8080. -k will kill that PID this has not been working reliably for me. Use instead: lsof -i:8080
grep `“<string>” *`	look in all files (*) for <string>. You can use regular expressions inside of <string>
genisoimage `-o myfile.iso /home/danh`	Backs up data into an ISO file which can then be copied to CD using dd if=myfile.iso of=/media/cdrom
grep `-l “<string>” *`	report only the filenames containing <string>.
grep `-v “<string>”`	Find files that DON’T contain “<string>”
gzip `<fname>`	zips file into compressed form. REMOVES the original file. Does only 1 file. Use tar + gzip to do multiple. Also see pigz for multi-threaded zipping
head `[-n <number>] <fname>`	Show the first <number> lines in a file
history	dump command history
host `<host>|<zone>` (2)	DNS lookup
hostname	View or Set (requires root) system’s name
kill `-HUP <pid>`	Send SIGHUP to a process
ldd `<filename>`	Shows all external routines called by the program.

locate <filename> (2)	Returns the directories containing the file. This is much faster than find, but requires updatedb be installed and run to keep the directory database updated.
lp <filename>	Send file to printer
lpstat	Shows output queued to printers
lpstat -v	displays all printers
ln /etc/resolv.conf yippy	creates a link (shortcut) named yippy that points to /etc/resolv.conf.
ls	list current directory -a show hidden files (starting with .) -l long format
ls -d */.	List only the directories in the current directory
ls -d a*	List everything starting with "a" without expanding directories starting with "a"
ls -lt	List everything but sort in date descending order
lsblk	clean output of all block devices
lscpu	List info about CPU
lsusb	List all usb device
mkdir <dir>	make directory named <dir>
mktemp --tmpdir name_XXX	Make a temp file named <name_?> with XXX being replaced with random characters. --tmpdir will use the /tmp dir. My typical usage: tmpf=$(mktemp --tmpdir fname_XXX.tmp)
more <filename>	Output file to terminal, stopping at each page. Press "?" at the more prompt for all possible commands
sudo **mount** -t cifs -o sec=ntlm,user=<uname> //dev/share /mnt/mntpoint	Mount smb: //dev/share at /mnt/mntpoint. Works with RPI.

mount -o big_writes /dev/sda<n> <dir>	This allows for much faster writes, at least to NTFS. If I need to zero out all free space on /dev/sda1, I would use: mount -o big_writes /dev/sda1 /media/ntfs pv /dev/zero >>bigfile.txt
mv <source> <destination>	move - there is no rename - you can only move
od -cx file	Dumps file in hex and char form
passwd	change your password
patch < file.patch	Reads file.patch to change a file from oldfile to newfile. See diff.
pgrep -l <processname>	Lists all processes named <processname>. -l option adds the name of the process. I have had issues with pgrep in scripts and now I tend to use 'pstree | grep | sed' to find exactly what I want.
pigz <filename>	Like gzip except it will use multiple threads to zip the file faster.
pkill <processname>	What pgrep lists, pkill kills. I wouldn't trust this in a script.
postprint	Creates postscript file from ascii file (found in /usr/lib/lp/postscript) Does not exist in linux
dd if=/dev/sdb bs=4M | **pv** | dd of=myfile.img or **pv** myfile.img | dd of=/dev/sdb bs=4M	pv - pipe viewer. Great utility for dd which lets you see the progress of a long dd command To see progress when copying filesets, see rsync --progress below
pwd	displays current directory (Print Working Directory I suppose)
r <string>	do last command starting with <string>. No parameter redoes last command. Using a number will redo the nth command.
reset	Resets TTY settings - fixes MPE emulator.

rename <oldstr> <newstr> <fileset> (2)	Given an old search string, new replacement string, and fileset, this command renames files. For example: rename .htm .html *.htm
rm <filename>	delete (remove). -r recursively go thru every directory -f remove even write protected files -i interactive (prompts)
rm -- <filename>	use to remove a weird filename such as rm -- -a
rmdir <dir>	remove directory named <dir>
rsync -ah --progress <srcDir> <destDir>	Copies files from one location to another and shows progress.
rsync -rv --delete --dry-run sourceDir destDir	Sync destDIR with SourceDir (deletes files in SourceDir). Remove --dry-run to actually execute.
set	Displays all environment variables
shred -u -n 5 <filename>	Shreds a single file by overwriting 5 times, then deleting.
shred -v /dev/sd<x>	Syntax left does a multipass ZERO write. The following will write random: `shred --verbose --random-souce=/dev/urandom -n1 /dev/sd<x>`
cat /dev/zero > zero.file sync **shred** -v -u -z -n 5 zero.file	Shred all free space: create zero.file to contain all free space, then shred it.
sleep <time>	Wait for <time> seconds
smbmount //dev/share /<mount point> -o username=<loginname>[,password=pass]	Mount a Microsoft share. Repl smbmount with mount.cifs in linux mint 17.
stat <fname> (2)	Gives detail info for a file
stty -a	displays current terminal settings
stty erase ^h	Make backspace the backspace key (and not delete)
tail <filename>	Display last page of a file
touch <files>	Sets file’s date/time to now

tput longname	Display the terminal type unix is using for your
tput reset	reset your terminal
uname -v	Display the version of Unix running
uniq	Removes duplicate lines from a file cat <file> | sort | uniq
unset <variablename>	remove variable <variablename>
vi [<filename>]	visual editor. For help on the editor, go here
w	Displays info on current users (showjob)
wall <text>ctrl-d	Send text to all users. Like HP’s WARN. Enter wall on the command line by itself, then the text on the next line(s) finally finishing with ctrl-d.
watch [-n x] 'cmd'	Repeats <cmd> forever, every x secs (default 2).
which grep	Tells you where the grep program is at.
who am i	Displays how you logged in (like showme)
xxx**=**”yyy”; export xxx	Set environment variable <xxx> and export to all shells. You can also use the form: export xxx=”yyy”

(1) This command doesn't exist in Linux
(2) This command doesn't exist in Solaris

apt Package Manager & dpkg	
`apt[-get] install <pkg>`	Install package named <pkg>
`apt remove <pkg>`	Uninstall <pkg>
`apt update`	Download latest package information.
`apt upgrade`	Upgrade ALL packages to the lastest version
`apt autoremove`	Remove all unused packages
`apt check`	Verify there are no broken dependencies
`apt search <string>`	Look for all packages with string anywhere in description. Installed packages denoted with 'i' in column 1.
`apt show <pkg>`	Detailed Info about the package

`dpkg --list`
or
`dpkg --get-selections`
or
`dpkg-query -L <package_name>`
	List all installed packages.

`dpkg -S <filename>`	This will display the package name containing the fully qualified filename.
`dpkg-deb -c <package_name.deb>`	To see the files a .deb file will install

`dpkg --remove <pkgname>`
`dpkg --purge <pkgname>`
	Remove a package or remove package and all config files.

apt-file search <path+fname>	Finds the package containing a specific file (specifying the full path and file name). To use this you may need to install apt-file. Then use apt-file update to update its database.

Redirection	
`> file`	Direct standard output file *file*
`< file`	Take standard input from *file*

`cmd1 | cmd2`	Pipe; take standard output of *cmd1* as standard input to *cmd2*
`>> file`	Direct standard output file *file*; append to *file* if it already exists
`>| file`	Force standard output to *file* even if *noclobber* is set
`<>`	Open *file* for both reading and writing on standard input
`<< label`	Here-document; see below
`<<- label`	Here-document; see below
`2>&1`	Redirect stderr to stdout
`n> file`	Direct output file descriptor *n* to *file*
`n< file`	Set *file* as input file descriptor *n*
`exec >myfile`	exec can be used to change redirection for the current shell. This sends all output for all following commands to myfile.
`exec <myfile`	All input for all following commands comes from myfile
`&0`	stdin
`&1`	stdout
`&2`	stderr

`for x in 1 2 3; do`
`  print %x`
`  done >myfile`
	Redirecting a code block.

`(ls;ls) >myfile`	Another example of redirecting a code block.

`{`
`ls`
`ls`
`} >myfile`
	Yet another example of redirecting a code block

`{`
`  while read rec; do`
`    print ${rec}`
`    done;`
`  } < myfile`
	Even more useful code block. Creates a loop that reads the entire input from the file ‘myfile’.

Here-document: read following lines into program until *label* is reached. << reads each line exactly, <<- will skip leading tabs. For example:

```
cat <<:eod
This is a test.
As is this.
:eod

cat <<-:eod
	This is a test.
	As is this.
:eod
```

If you don’t want parameter substitution to occur in the here-document, single quote the terminator:

```
cat <<‘:eod’
$RANDOM
:eod
```

(1) This command doesn't exist in Linux
(2) This command doesn't exist in Solaris

Job Commands	
For those of us that have worked with operating systems that have real batch jobs "job control" in Unix is a misnomer. It is really process control. There is no such thing as a batch job in Unix.	
`jobs`	Lists all processes you are running
`^z`	Stop the process you are running and return to the shell
`fg [%job#]`	Return *job#* running in foreground
`bg [%job#]`	Start *job#* running in background
`<command> &`	Using a trailing '&' causes the command to start and run in the background.
`nohup <command> &`	This will run a command in the background and then prevent it from being killed when you logout.
`disown -h %job#`	Job will run until termination even if you log off. Turns out disown is not in pdksh (linux) or whatever version of ksh running on solaris. To exit and leave background processes running, type: set -o nohup
`kill %job#`	Kill *job#*
`wait`	Await process completion. If you created a bunch of processes that are running, then use 'wait' to wait for them all to complete before proceeding.

(1) This command doesn't exist in Linux
(2) This command doesn't exist in Solaris

System Management Commands	
Unix Command	**Function**
admintool (1)	System admin w/ graphical interface. You have to be running Xwindows for this to work.
accept `<printer>`	Allows new output to be queued (see reject)
chown `<user> <filename>`	Change the owner of a file
date `mmddhhmmccyy`	Set date. Note the weird format. To set Date time to May 5, 2004 5:30PM: date 0505171302004. Supposedly the year is optional, but if I omitted it, I got 2000.
dispgid (1)	Display all GID's
dispuid (1)	Display all UID's
disable `<printer>`	Disable a printer
dmesg	Display recent logging messages
dstat	Display usage stats. My fav is: `dstat -t -c -d  -n 10`
enable `<printer>`	Enable a printer
fsck `-y`	Cleanup file system and reply Y to all prompts
fsck `-c f -y /dev/sda<n>`	Looks for bad blocks (surface scan) and then marks them unreadable
ftprestart (1)	Restart FTP (see ftpshut)
ftpshut (1)	Stop FTP (see ftprestart)
ftpwho (1)	See who is running FTP
groups `<username>`	See groups to which a user belongs
halt	Stops O/S. Init 0 appears to be favored.
htop	More features than top. Now my preferred. (see also IOTOP)
hwclock (2)	Display the date/time of the hardware clock vs. the 'date' command which shows the date/time of the system clock.

hwclock --hctosys (2)	Set hardware clock to current system clock.
hwclock --systohc (2)	Set system clock to current hardware clock.
iftop	'top' for network connections
init <n>	Restarts system where <n> is: 0 halts the system 1 Single User mode - Linux 2 Single user mode - Solaris 3 Multi user mode - Solaris 3 Don't start Xterminal - Linux 6 Restarts system in the current single/multi user mode
iostat -xc 5 2	Dump extended CPU times once every 5 secs
iostat -xd	Dump extended Disk Stats
iostat -xt (1)	Dump extended TTY stats
iotop	like top, except shows IO used by each process
last	Show user login/logouts
logadm (1)	Log file rotation tool
logins -t (1)	List all defined user names
logger	Logs text to system log
lpadmin	configure the LP printer service
lpc	Line Printer Control. Type ? for help. In /usr/ucb. Not sure this works for us.
lpget list (1)	Display config for all printers
lpmove <reqid> <destination>	Move queued output to a different printer
lpsched (1)	Starts LP print service
lpshut (1)	Stops LP print service
lshw	lists information about ALL hardware installed
lspci	Displays info about PCI devices

lsusb (2)	Displays USB buses and devices
mount	See what volumes are mounted
msg n (1)	Disable write/talk messages to your terminal
nmon	Kind of like top, but allows you to see CPU, disk, memory, network stats. In putty, set remote character set to HP-ROMAN8 to see line draw characters properly.
passwd -sa (1)	Summarizes passwords ps: passworded lk: locked np: no password
pfiles <pid> (1)	Displays open files for a process
pkginfo (1)	Display installed packages
pldd <pid> (1)	Display libraries in use by a process
pmap <pid>	Display memory map for a process.
poweroff	Shuts down system and turns off power (doesn't actually kill the power on my 386 version)
printconf **(2)**	Configure Printers
printmgr (1)	Graphical interface for managing printers (in /usr/sadm/admin/bin)
prstat (1)	See running processes
prtconf (1)	Print HW config (requires root)
prtvtoc /dev/dsk/c0d0s1 (1)	Print partition table for disk (use sysdev to determine disk dev name)
ps [-Al]	Display process info. -A: all processes. -l: long format.
pstack <pid>	Display stack for process

pstree (2) **pstree** <pid> **pstree** -u **pstree** -a	Displays process tree Show process tree starting with processes <pid> Show process tree and users Show process tree and commands
ptime <cmd> (1)	Displays wall/cpu time for execution of a command. See 'time' command for Linux.
ptree [<pid>] (1)	Display process tree
pwck	Validates password file
reboot	Like shutdown -i 6 w/o waits
reject <printer>	Prevents queuing of new output to printer (see accept)
quot -a (1)	Summarize file system ownership
shutdown -g 60 -i 6 "shutting down"	Shuts down system. Gives users 60 seconds to get off. -i is init state: 0: stop os 1: single-user state 2: Multi user state 5: stop so power can be removed 6: reboot into default state Parameters are different for Linux
shred -v /dev/sd<x>	Securely wipe an entire disk
sudo <cmd>	Execute <cmd> as root
sudo su	Login as root
sudo !!	Execute last command entered as root
swap -s (1)	Display swap information
swapon -s (2)	Display Swap information
swapon /dev/sd<x> (2)	Enable swapping into specified partition (typical linux swapping)
swapoff -a (2)	Disable swapping
sync	Writes all buffers to disk

`sys-unconfig`	Remove Network configuration This is a general reconfig utility in Linux
sysdef (1)	Displays device configuration information
su root	Switch to root w/o logging in again
su - root	switch to root and execute root login process
time <cmd> (1)	Displays wall/cpu time for execution of a command. See 'ptime' command for Sun.
top	Simple CPU / performance monitor. See nmon or htop
truss <cmd> (1)	Displays debug info regarding execution of command.
uptime	Shows uptime, user count, and avg CPU load
useradd -s /usr/bin/ksh -c "John Doe" -m -d /export/home/johnd johnd	Create a new user. You must then assign it a password with passwd before it can be used.
usermod [options] johnd	Change user parameters
gpasswd -a user group **usermod** -a -G group user	Adds <user> to <group>. **Use gpasswd!** It is safer.
userdel -r johnd	Removes user. -r removes associated home directory.
vmstat 5	Display virtual memory stats, once every 5 secs.
w	kind of like showjob

Notes
(1) This command doesn't exist in Linux
(2) This command doesn't exist in Solaris

(1) This command doesn't exist in Linux
(2) This command doesn't exist in Solaris

Network commands	
arp -a	Dump arp cache
ifconfig -a	Display network config
ifport eth0 10BaseT (2) **ifport eth0 auto**	Sets the Transceiver type. Only works on some NICs.
iftop	network version of top
in.identd (2)	Provide name of user who's process is running a specified TCP/IP connection
iwconfig (2)	Reports wireless NIC information
lsof | grep TCP | grep <port>	This allows you to determine who is using a particular TCP port (replace TCP with UDP to do UDP). lsof is being installed on the solaris boxes. It is (I believe) available on linux but hasn't been installed.
lsof -p <pid>	Show open files for a specific process
netstat -D (1)	Display DHCP information
netstat -i	Packet counts for each interface
netstat -k <interface> [|grep <countername>] (1)	Dumps all counters for an interface
netstat -r	to see the routing table
netstat -s	Protocol statistics
[watch] **netstat** -tulpn	Shows LISTEN ports with owner process
nmap -sT -O localhost	If nmap is installed and you are root, you can use to see what ports are open on this (or any) host.
nslookup	Examine DNS
ping -s <host> [<packet size>][<count>]	Parameters are not positional, so to specify <count> you must specify <packet size>

route add x.x.x.x/nn y.y.y.y (solaris) route add -net x.x.x.x netmask m.m.m.m gw y.y.y.y (linux)	Add a route x.x.x.x w/ CIDR (you can't use subnet masks) to forward to y.y.y.y
route delete x.x.x.x/nn y.y.y.y (solaris) route del -net x.x.x.x gw y.y.y.y netmask m.m.m.m (solaris)	Delete route
smbclient '\\chq-danh\c' -U chq-danh	Connect to \\chq-danh\c share using user chq-danh and remain in command mode
smbclient '\\chq-danh\c' -U chq-danh -c 'cd bat;dir'	Connect to \\chq-danh\c share using user chq-danh, execute the cd and dir commands, then exit
tcpdump -w <file> -s 0 host <ipaddr>	Dump packets for <ipaddr> to a file. You can then read this file with ethereal. *-s 0* indicates capture full packet length. Requires root. May require fqn: /usr/local/sbin.
tracepath <host>	Like traceroute but faster
traceroute	standard tracert or trace command

Network Utilities

These are Unix based Network utilities. For the most part, they run on Linux and must be located and installed - they are not included as part of the base install for Linux or Solaris.

This section is meant specifically to be used by the networking department.

bing	Bandwidth Ping
lft	Layer Four Traceroute
lsof	List Open Files
netperf	Network Performance Tester
pathchar	Path Characteristics
tcpdump	Packet Capture Program
windump	Windows Version of tcpdump

bing

Bandwidth Ping - given 2 routers it will ping the first, then the second and measure the difference to determine the bandwidth. This runs much faster and produces less traffic than pathchar but is not as comprehensive.

Obtained from rpm.pbone.net. Explanation found at
http://www.freenix.fr/freenix/logiciels/bing.html

There is a man page with this RPM that explains all parameters.

bing [<options>] <firsthop> <secondhop>

Notes:

To determine <firsthop> and <secondhop>, use traceroute.

Always use -S 1000 to force the maximum packet size to 1000. This will give you much more accurate results.

Example: To determine the link speed between rtr-vpn2 and rtr.lhr.ei:

```
bing -S 1000 rtr-vpn2.chq.ei rtr.lhr.ei
BING    rtr-vpn2.chq.ei (10.254.7.27) and rtr.lhr.ei (10.75.1.1)
        44 and 1000 data bytes
15296 bits in -3.738ms: -4092028bps, -0.000244ms per bit
15296 bits in 31.205ms: 490178bps, 0.002040ms per bit
15296 bits in 27.659ms: 553021bps, 0.001808ms per bit
15296 bits in 27.647ms: 553261bps, 0.001807ms per bit
15296 bits in 27.730ms: 551605bps, 0.001813ms per bit
15296 bits in 27.692ms: 552362bps, 0.001810ms per bit
15296 bits in 27.686ms: 552481bps, 0.001810ms per bit
```

```
 15296 bits in 29.063ms: 526305bps, 0.001900ms per bit
 15296 bits in 32.653ms: 468441bps, 0.002135ms per bit
 15296 bits in 32.618ms: 468944bps, 0.002132ms per bit
 15296 bits in 32.607ms: 469102bps, 0.002132ms per bit
 15296 bits in 30.610ms: 499706bps, 0.002001ms per bit
 15296 bits in 27.957ms: 547126bps, 0.001828ms per bit
 15296 bits in 26.728ms: 572284bps, 0.001747ms per bit
 15296 bits in 25.057ms: 610448bps, 0.001638ms per bit
```

after the display slows down or stops, press control-C to get the results

```
--- rtr-vpn2.chq.ei statistics ---
bytes   out    in   dup  loss   rtt (ms): min       avg       max
   44    60    60          0%           1.935     2.754     8.438
 1000    60    60          0%           5.754     6.405    10.544

--- rtr.lhr.ei statistics ---
bytes   out    in   dup  loss   rtt (ms): min       avg       max
   44    60    59          1%         151.610   175.405   349.769
 1000    59    58          1%         180.486   199.691   330.719

--- estimated link characteristics ---
estimated throughput 610448bps
minimum delay per packet 148.522ms (90665 bits)

average statistics (experimental) :
packet loss: small 1%, big 1%, total 1%
average throughput 741301bps
average delay per packet 171.497ms (104690 bits)
weighted average throughput 728842bps
```

Disregard the ***experimental*** section. Even the man page says it is way out of whack.

lft

Layer Four Traceroute - This utility does a traceroute using SYN/FIN packets instead of the normal UDP packets. This allows it to run much faster and sometimes trace devices that the normal traceroute won't see.

For more information and the source, see http://www.mainnerve.com/lft

Usually you'll just use it without options:

```
/home/danh>lft 10.75.1.1

Tracing ______________.

TTL  LFT trace to rtr.lhr.ei (10.75.1.1):80/tcp
 1   rtr.chq.ei (10.1.1.1) 2.4ms
 2   rtr-vpn2.chq.ei (10.254.7.27) 3.8ms
 3   [target] rtr.lhr.ei (10.75.1.1):80 169.1ms
```

The most probable option is -E which tries several different ways to determine each hop:

```
/home/danh>lft -E 10.75.1.1

Tracing _________?????_____?????___?__????____?________________________.

TTL  LFT trace to rtr.lhr.ei (10.75.1.1):80/tcp
 1   rtr.chq.ei (10.1.1.1) 2.1/1.1ms
**   [firewall] the next gateway may statefully inspect packets
 2   rtr-vpn2.chq.ei (10.254.7.27) 2.9/5.2ms
 3   [target] rtr.lhr.ei (10.75.1.1):80 158.1/157.0/*/*/*ms
```

lsof

List Open Files - This utility will show you what files a user or process has open or who has a specific file open. There are many options for this program. Only the most useful is covered here.

This comes installed on Linux Redhat 9. To obtain for other systems:

```
ftp://vic.cc.purdue.edu/pub/tools/unix/lsof
```

To list the files a specific user has open:

```
lsof -u root
```

To List the file open for a specific process:

```
lsof -p 7763
```

To find who has a specific file open:

```
lsof /usr/sbin/myfile
```

netperf

Network Performance Analyzer - this program requires a server (netserve) running on the far end and netperf running on the near end. This allows it to send TCP packets to the server and measure performance much more precisely than ICMP based tools.

Obtained from www.netperf.org which is NOT accessible from our network as it is a 192.6 address (I had to use dial-up). I have a linux RPM in \\chq-danh\downloads\redhat9.

I have not been able to get the linux version from www.netperf.org to work; however, the RPM version runs fine. The solaris and windows versions at www.netperf.org run fine.

```
Usage: netperf [global options] -- [test options]
```

```
Global options:
    -a send,recv      Set the local send,recv buffer alignment
    -A send,recv      Set the remote send,recv buffer alignment
    -c [cpu_rate]     Report local CPU usage
    -C [cpu_rate]     Report remote CPU usage
    -d                Increase debugging output
    -f G|M|K|g|m|k    Set the output units
    -F fill_file      Pre-fill buffers with data from fill_file
    -h                Display this text
    -H name|ip        Specify the target machine
    -i max,min        Specify the max and min number of iterations (15,1)
    -I lvl[,intvl]    Specify confidence level (95 or 99) (99)
                      and confidence interval in percentage (10)
    -l testlen        Specify test duration (>0 secs) (<0 bytes|trans)
    -o send,recv      Set the local send,recv buffer offsets
    -O send,recv      Set the remote send,recv buffer offset
    -n numcpu         Set the number of processors for CPU util
    -p port           Specify netserver port number
    -P 0|1            Don't/Do display test headers
    -t testname       Specify test to perform
    -v verbosity      Specify the verbosity level
    -W send,recv      Set the number of send,recv buffers
```

Notes:

On the system to be tested, you must first install and run netserver which will simply run in the background and respond to netperf.

I find running the win32 version of netperf at a cmd line much easier than running the linux based one. You have an option of running it on every platform.

With no paramaters, this does a short test to local test which doesn't do much. For this to be truly useful, you minimally need:

```
netperf -H <hostname> -l 60 -f k
```

This will do a 60 second test to <hostname> and tell you the speed in kbits/sec (def is MBits which is hard to read). 60 seconds isn't very long. A serious test needs to be at least 5 minutes.

There are several different tests you can perform: TCP_STREAM (default), TCP_RR (transaction rate), UDP_STREAM, and UDP_RR (transaction rate for UDP)

TCP_STREAM is probably the most common. It will tell you the bandwidth available on the line:

```
[c:\]netperf -H 10.1.11.3 -l 60 -f k
TCP STREAM TEST to 10.1.11.3
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^3bits/sec

 87380   8192   8192    60.00    8314.33
```

Here you can see the bandwidth is 8.3M.

TCP_RR will tell you the number of “transactions / seconds”. That is, netperf sends a one byte packet and waits for a one byte response, then sends another:

```
[c:\]netperf -H 10.1.11.3 -l 60 -f k -t TCP_RR
TCP REQUEST/RESPONSE TEST to 10.1.11.3
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

8192   8192   1        1       60.00    1500.98
16384  87380
```

An unloaded 10BaseT network can support about 1500 transactions / sec.

A UDP_STREAM test is also available:

```
[c:\]netperf -H 10.1.11.3 -l 60 -f k -t UDP_STREAM
UDP UNIDIRECTIONAL SEND TEST to 10.1.11.3
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^3bits/sec

  8192    8192   61.00        8256      0    8870.06
 65535           61.00        8253           8866.84
```

This shows the transmit rate and the receive rate. The manual indicates that if they are very different, use the receive rate as the true rate (wnd rate).

There is also a UDP transaction test:

```
[c:\]netperf -H 10.1.11.3 -l 60 -f k -t UDP_RR
UDP REQUEST/RESPONSE TEST to 10.1.11.3
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

8192   8192   1        1       60.00    1468.18
65535  65535
```

Interestingly, the transaction rate for UDP on this system was a little slower (I would not have expected that).

There is a connect/request/response test which mimics a normal HTTP request (you have to pay for the connection setup overhead on each request):

```
[c:\]netperf -H 10.1.11.3 -l 60 -f k -t TCP_CRR
TCP Connect/Request/Response TEST to 10.1.11.3
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

8192   8192   1        1       60.00     287.97
16384  87380
```

pathchar

Path Characteristics - This program, given an IP address will try to determine the latency and bandwidth of each hop. It is intensive on the network and quite slow but does seem to work.

Obtained from ee.lbl.gov. Usage info found at
http://www.caida.org/tools/utilities/others/pathchar/pathcharnotes.html

Syntax is: (this program provides no help whatsoever)

```
pathchar [<options>] <host>

-f <n>            Start on hop # <n>. Using -f 2 will skip the path from
                  the test PC to the first router.
-m <n>      Maximum packet size
-M <n>      Minimum packet size
-q <n>            #queries, default 32
-Q <n>      bytes,
            if (-), packet size increment per query, defaults to 92.
            if (+), number of sizes, defaults to 32
-n          No DNS name resolution
-v          verbose
-w <n>      Timeout value in seconds?
```

Notes:

Use -m 1500, it takes forever for it to figure out the MTU.

Also, you are normally going to care about the BW from your PC to the first (or probably even 2nd router, so use -f 2 or -f 3)

This program generates a lot of traffic and can take quite a while to run.

Example:

```
/root>./pathchar -m 1500 10.75.1.1
pathchar to 10.75.1.1 (10.75.1.1)
 can't find path mtu - using 1500 bytes.
 doing 32 probes at each of 45 sizes (64 to 1500 by 32)
 0 danh-linux-ws.chq.ei (10.1.11.3)
```

```
        |   7.7 Mb/s,   605 us (2.76 ms)
        1 rtr.chq.ei (10.1.1.1)
        |   8.3 Mb/s,   154 us (4.51 ms)
        2 rtr-vpn-lhr.chq.ei (10.254.7.26)
        |    449 Kb/s,   74.3 ms (180 ms)
        3 rtr-lhr-chq-t1.lhr.ei (192.10.75.75)
       3 hops, rtt 150 ms (180 ms), bottleneck  449 Kb/s, pipe 10094 bytes
```

This gives you the available bandwidth between hops and the propgation delay (latency).

tcpdump

tcpdump is the primary network protocol analyzer for unix (see windump in this section for a windows version of tcpdump). It output is character-mode based so it is now usually used just to collect data. Once you have collected a trace then you can review it using ethereal.

tcpdump will be (should be) installed on every one of our systems although you probably will need root (or sudo) to access it. If you don't have the ability you'll get the completely meaningless (to me) message:

```
$ tcpdump
tcpdump: no suitable device found
```

The parameters are:

```
Usage: tcpdump [-adeflnNOpqRStuvxX] [ -c count ] [ -C file_size ]
               [ -F file ] [ -i interface ] [ -r file ] [ -s snaplen ]
               [ -T type ] [ -w file ] [ -E algo:secret ] [ expression ]
```

and a man page is available on the main unix systems (not the nss boxes).

Notes:

Since you will primarily (I would think) want to use this just to capture data to then process with ethereal you will always want to write to a file using -w

```
tcpdump -w filename
```

Generally you are not going to want to capture everything, but filter out everything except what you really want to see otherwise you will have to wade through a lot of data. The rest of these notes will simply go over some of the ways to filter data.

To play back a file captured with -w

Once you've captured a data file you can play it back and filter it by using the -r option

```
tcpdump -w myfile           (records the file)
tcpdump -r myfile           (plays the file to your monitor)
```

and you can use filters to reduce the output as it comes to your terminal.

To stop capturing after 100 packets

```
tcpdump -c 100
```

All traffic to/from a host

```
tcpdump host 205.153.63.30
```

All traffic to/from a NIC

```
tcpdump ether host 0:10:5a:e3:37:0c
```

Traffic to a specific IP destination

```
tcpdump dst 205.153.63.30
```

Traffic to/from a specific port

```
tcpdump port 53
```

Traffic to/from a specific UDP port

```
tcpdump udp port 53
```

windump

windump - Windows version of tcpdump. See tcpdump in this section regarding usage.

Obtained from windump.polito.it. Also requires winpcap (available from the same location) to run.

eiprofile

eiprofile is the login script file that is standard for all users on netboxu and for the netmgr user on all unix systems. eiprofile attempts to provide a standard environment across all unix platforms and gets reasonably close.

eiprofile consists of a collection of environment variables and commands available on all systems.

Prompt

eiprofile only works in the korn shell (ksh). All netboxu users and netmgr on all systems are setup to use this shell by default.

When eiprofile is invoked correctly, it is fairly obvious based on the standard prompt it produces:

```
netboxu.chq: danh$
```

You are presented with the domain name of the box, the current directory, and the user/priv prompt as shown above.

Command Line Editing and Redo Environment

eiprofile provides a consistent command line editing and redo environment. Command line editing is describe here and the redo command is found in Really Useful Commands.

Command History

eiprofile implements command history by making it unique within each shell. Further, the command history is removed at the end of each shell session and is in a secured file so others may not examine it.

Environment Variables

eiprofile provides the following variables. To follow the backwards unix standard, a boolean variable uses 0 for true and 1 for false.

EIBATCHMODE: If set, script is being run with ‘eibatchmode on’. This isn't something you would normally need to reference.

EIBRCD: This is the branch code (such as ord) of the box. This setting is 100% accurate for network owned boxes. For others, it depends on if the siteinit UDC or resolv.conf file are correctly configured. There is a pretty good chance this will be accurate for most machines.

EIBRNO: This is the branch number (such as 4) of the box. This setting is 100% accurate for network owned boxes. For others, it depends on if the siteinit UDC file is present and configured correctly. There is probably only a 50% chance of this being accurate on a non-networking owned box.

EIDEBUG: If true, any use of command debugprint will execute.

EIHOSTNAME: This is the hostname only (etms part of etms.ord.ei). This is 100% accurate on all network owned boxes and very accurate on other boxes. Only some test machines seem to be wrong.

EIINTERACTIVE: True if the shell is being run interactively (e.g. a user at a terminal).

EINETOWNED: If set, this machine is 'owned' by the networking department. This would apply to nss boxes as well as netboxu.

EIOS: The operating system which can be one of 'linux', 'solaris', 'unknown'.

EIROOT: 0 if the user has root capability; else 1.

Commands

debugprint <text>

If EIDEBUG is true, then <text> is displayed.

dir <ls options>

dir is the same as using ls -Fl

eibatchmode on [emailaddress]
eibatchmode off
eibatchmode quiet

eibatchmode is used to produce a psuedo batchjob environment like we had on the HP3000. A 'batch job' on the 3000 had every command listed as it was executed and if an error occurred, the batch flushed rather than assume all is well and keep blindly going on. A unix script, on the other hand, won't show you what is being executed and if an error occurs it goes on its merry way.

Using the 'on' option causes all commands to be listed as they are executed and if any command generates an error code (non-zero result code) then the script is aborted immediately with an error:

```
netboxu.chq: danh$ eibatchmode on
netboxu.chq: danh$ rm myfile
rm myfile
```

```
rm: cannot lstat `myfile': No such file or directory
Command terminated in an error state.
Remainder of job flushed.
```

You may optionally include an email address after on such as:

```
eibatchmode on daniel.hallock@expeditors.com
```

This performs the same as eibatchmode on, except if an error occurs (and only if an error occurs), the output of the script is mailed to the email address.

The side-effect of using an email address is all of the output of the command must be redirected into a file until the command has completed executing. If an error occurs the contents of the file are mailed and in either case the output of the file is then displayed on the terminal. *That means that during the execution of the script, nothing is displayed.* It is only displayed at the end of execution.

‘eibatchmode quiet’ is used if you want the script to abort if an error occurs, but you don’t care about seeing the commands listed as they execute.

eicontinue <commands>

If eibatchmode is on, then any command that produces an error will cause the entire script to abort. By prefacing the command with eicontinue, the commands that cause errors will not flush the script.

For example, before creating a new file, you may want to delete any old version of the file; however, if the file doesn’t exist rm would generate an error and flush the script, so use eicontinue:

```
eibatchmode on
eicontinue rm myfile.txt
```

Note: you can’t use redirection with eicontinue such as:

```
eicontinue ls >myfile
```

When you use eicontinue, {?} is not available. Instead use ${EICC}:

```
eicontinue grep xxx fset*
if (( ${EICC} != 0 )); then
  whatever
  fi
```

grep, honestly, is a constant problem for me when using eibatchmode on. I have found the best way so far to deal with grep is in a subshell by using ():

```
eibatchmode on          #causes script to abort on any error
(                       #start subshell
 grep xxx fset*         #do grep which may return an error
 if (( ${?} ...         #handle the error
)                       #exit subshell
```

eiint2commas <number>

Given a number, this command formats it with commas so 1234 becomes 1,234.

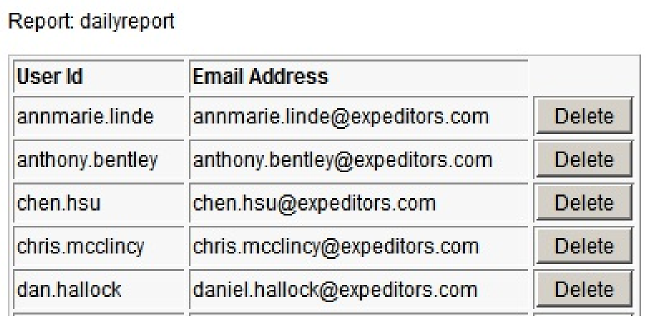
eilapsedtime

This command displays the current value of the timer (in seconds) since eistarttimer was called.

einotify <reportid> <mail options>

einotify is the EI version of the 'mail' command. The difference is that instead of specifying destination email addresses, you specify a report id. The email is then sent to everyone in that report id.

For example, the report id 'dailyreport' has been set up in the notifications database using the Notification DB Reports Table Maintenance web program (found on the internal networking home page) and various users are assigned to this report id:



Report: dailyreport

User Id	Email Address	
annmarie.linde	annmarie.linde@expeditors.com	Delete
anthony.bentley	anthony.bentley@expeditors.com	Delete
chen.hsu	chen.hsu@expeditors.com	Delete
chris.mcclincy	chris.mcclincy@expeditors.com	Delete
dan.hallock	daniel.hallock@expeditors.com	Delete

When the dailyreport script is run, instead of having to specify individually all of these users, you simply use einotify with the report id:

```
einotify dailyreport -s 'Daily Report' </tmp/filename
```

einotify will take all of the options of 'mail' except you don't list user names.

As a side note, the real benefit to using einotify is that end-users can directly update the list of email addresses for a report id instead of the programmer having to modify the script.

einotifylist <report id>

This command is used by einotify (see that command) but may be used by others when trying to use <report id> for other purposes. This is most useful for utilities such as mailquery which mails the results of a query to a list of users. Instead of listing email addresses, a report id can be used:

```
mailquery -d netboxu.chq.ei,readonly,,netflow \
      -q “select * from constants.branches” \
      -s ‘email subject’ \
      -f ‘netflow@expeditors.net’ \
      -c ‘<column list>’ \
      -t $(einotifylist criticalnotes | sed -e's/[ ][ ]*\(.\)/,\1/g')
```

note that $(einotifylist criticalnotes) returns:

```
john.moore@expeditors.com john.janssen@expeditors.com
hussain.mohammed@expeditors.com
```

sed is used to insert commas:

```
john.moore@expeditors.com,john.janssen@expeditors.com,hussain.mohammed@e
xpeditors.com
```

eirdns <ip addr>

This returns the dns name for an ip address:

```
netboxu.chq: danh$ eirdns 10.1.1.1
ntp.chq.ei
```

It can also resolve dns addresses into IP addresses if you fully qualify the name and terminate it with a period:

```
netboxu.chq: danh$ eirdns rtr.chq.ei.
10.1.1.1
```

It is most handy when used with other commands:

```
print $(eirdns 10.1.1.1)
```

This is not a fast command - it must execute an SQL query to function. You probably don’t want to use it in a heavy manner.

eiscp <password> <scp parameters>

This is a front-end to the eiscp command to allow it to be used in a script. The scp command will prompt separately for a password which makes it unsuitable when using it inside of a script.

This form of the command allows you to embed the password in the command line so the script doesn't stop to ask.

eistarttimer

This command starts an internal timer. See eilapsedtime. Call this routine again to reset the timer.

eisu

This is the same as 'su root -c ksh eisu'

help

Displays a quick list of all eiprofile commands.

listd <wildcarded filename>

Displays list of files with just last modified and accessed dates:

```
netboxu.chq: danh$ listd

------------------------File------------------------ ---Modify--- ---Access---
blah.sed*                                            Jan 13 16:14 Jan 13 16:18
booboo*                                              Nov 18 13:59 Nov 18 14:04
datex/                                               Sep 15 10:49 Jan 26 04:03
datex.tar                                            Sep 15 09:50 Sep 15 09:52
dead.letter*                                         Aug 24  2004 Jun 21  2005
dhcp/                                                Jan 13 10:11 Jan 26 04:03
dhcp-old.MYD                                         Jan 12 11:06 Jan 13 10:16
dhcp-old.MYI                                         Jan 12 11:06 Jan 13 10:12
dhcp-old.frm                                         Jan 12 11:06 Jan 13 10:12
```

listf <wildcarded filename>

Lists files with just the size in bytes.

```
netboxu.chq: danh$ listf

--------------------------File--------------------------- -----Bytes-----
blah.sed*                                                             308
booboo*                                                               218
datex/                                                              4,096
datex.tar                                                          51,200
dead.letter*                                                        8,572
dhcp/                                                               4,096
dhcp-old.MYD                                                    4,189,420
dhcp-old.MYI                                                    3,471,360
dhcp-old.frm                                                        8,684
```

listg <wildcarded filename>

Lists just the group to which each file belongs

```
netboxu.chq: danh$ listg

-group- --file--
root    blah.sed
users   booboo
www     datex/
users   datex.tar
users   dead.letter
users   dhcp/
mysql   dhcp-old.MYD
mysql   dhcp-old.MYI
mysql   dhcp-old.frm
```

listo <wildcarded filename>

Lists just the owner of each file

```
netboxu.chq: danh$ listo

-owner- --file--
danh    blah.sed
danh    booboo
billh   datex/
danh    datex.tar
danh    dead.letter
danh    dhcp/
mysql   dhcp-old.MYD
mysql   dhcp-old.MYI
mysql   dhcp-old.frm
```

listuser <username>

List all users on the system (by examining the passwd file). If <username> is present, then lists just that username.

```
netboxu.chq: danh$ listuser

--------user-------- --------desc--------
adm                  adm
amanda               Amanda user
apache               Apache
```

trace <traceroute options>

trace is the same as 'traceroute'.

Command Line Editing	
This assumes you are running ksh shell and editing is set correctly in your profile.	
Common commands	
`^u`	erase entire line
`backspace`	Move left one character, erasing it
`left&right arrow`	Move cursor without erasing
`^d`	erase character to the right of cursor
`^a`	go to beginning of command line
`^e`	go to end of command line
`up&down arrows`	Move thru command history
Not so Common Commands	
`^i (tab)`	command completion
`^k`	delete (kill) to end of line
`^r`	search history file

Command Line Editing in bash	
This assumes you are running bash shell.	
Common commands	

`left&right arrow`
`^b & ^f`
	Move cursor without erasing

`alt-b`	Move cursor back one word
`alt-f`	Move cursor forward one word
`^a`	go to beginning of command line
`^e`	go to end of command line
`^u`	erase entire line
`backspace` `^h`	Move left one character, erasing it

`delete` `^d`	erase character to the right of cursor
`^k`	erase everything from cursor to end
`^_`	Undo last line edit
`up&down arrows`	Move thru command history
`^i (tab)`	command completion (if nothing happens, tab twice to get a list).
`^k`	delete (kill) to end of line
`^r`	search history file. ● Press control-r to invoke. ● Start to type cmd to find. ● As you type, qualified cmd is displayed. ● ^r again finds next cmd matching string. ● Enter executes cmd. ● arrows, ^a, ^e lets you edit cmd.
`history`	see all commands typed
`history -c`	clear history
`history | grep ll`	list just commands with ll
`!<n>`	execute history line numbered <n>
`!-2`	execute command 2 back from end
`!!`	execute last command (like !-1)
`sudo !!`	redo last command, doing sudo first
`!!<string>`	redo last command starting with <string>
`!!?<string>?`	redo last command containing <string>
`!!:s/<s1>/<s2//`	redo last command, changing string <s1> to <s2>

Command Line Editing in Other Shells

Command line editing is very easy in Korn and evidently ugly in the others. If you have to use another shell for whatever reason, it can be done to some extent.

fc -l	This will dump the command history (like history command in ksh).
fc -r	This will take the last command and put it into your text editor so you can edit it. When you exit the text editor, it will then execute the command you saved in the text editor.
fc <n>	Redo command numbered <n>

Command Line Editing Revisited (Using vi Style Editing)

Since writing my initial text, I've learned how to invoke and use vi style editing at the command line. I won't say this is any easier than the emacs form listed above, but it is consistent with vi: if you are using vi to edit text, may as well use it to edit the command line.

To invoke vi command line editing

To turn on vi editing, type:

```
      set -o vi
```

If this fails, you cannot use command line editing in the lowly default solaris shell sh (I don't know if this is still being force on everyone or not). Use Korn or Bash shells by typing ksh or bash, then use set -o vi.

To test that this works, type a command such a "test" and press ESCAPE. If the cursor backs up under the last "t" in test, then vi editing is working.

vi Command Line Editing Commands	
Movement	

`space`
`right arrow`
`(linux)`
	Move 1 character right

`backspace`
`left arrow`
`(linux)`
	Move 1 character left

`w`	Right 1 word
`b`	Left 1 word
`e`	end of current word
`0`	beginning of line
`$`	end of line
`f<x>`	move to next <x> character on line
`;`	repeat last f command
`F<x>`	move to prev <x> character on line
Adding Text	
`i`	insert before cursor

`a`	append after cursor
`I`	insert at beginning of line
`A`	append at end of line
`r`	replace 1 character
`R`	replace until Escape
Deleting Text	
`x`	delete character on cursor
`X`	delete character left of cursor
`D`	delete to end of line
`dd`	delete entire line
`u`	undo last change
`U`	undo all changes
Moving thru History File	

`-`
`up arrow (linux)`
	go back a line in history file

`+`
`down arrow (linux)`
	go forward a line in history file

`G`	first line in history
`/<text>`	search back thru history for <text>
`?<text>`	search forward thru history for <text>
Filename Completion	
`\`	complete filename.
`*`	Complete with all filenames
`=`	list files. In linux you can type this after partially entering a command name and all possible commands are listed.
`<n>/`	After using ‘=’ to list all files, use <n>/ to select the file you want

FTP	
`get <fname> | put <fname>`	To get/put a single file
`mget <fileset> | mput <fileset>`	To get/put files using wildcards
`prompt`	To prevent being asked before every transfer
`bin`	put FTP in binary mode! You should always do this as it produces a faster transfer.

Environment Variables	
env to display	setenv to set
`EDITOR`	Name of your preferred editor
`HOME`	absolute path of your home directory
`LOGIN`	your login name
`MANPATH`	MAN pages search path
`PATH`	directory search path
`PRINTER`	name of your preferred printer
`PS1`	Shell prompt \a beep \d date \h unqualified host name \H qualified host name \s shell name \t time (12h) \T time (24h) \@ (am/pm) \u username \$ priv level ($ or #)
`PWD`	Absolute path name of your current directory (set by CD)
`SHELL`	Absolute path of your shell
`TERM`	name of your terminal type

Samba Notes

Configuring Samba

Additional setup information for Samba and SWAT can be found in my RedHat Installation notes,
`http://www.chq.ei/is/NetStats/NetTasks/Admin/Misc/Red%20Hat%209%20Installation.pdf`

SWAT

I use SWAT to configure Samba. It has been setup on netboxu and my own test Linux server.

To access SWAT, use your web browser on port 901 such as:

```
http://netboxu.chq.ei:901
```

Configuring SWAT

On an initial installation, SWAT will not work from an IP address other than 127.0.0.0 - in other words you cannot use SWAT from any system except the system with SWAT installed on it.

In /etc/services, make sure the following line is present:

```
swat  901/tcp
```

Go into the directory /etc/xinet.d and there should be a file called swat already there:

There should be a line:

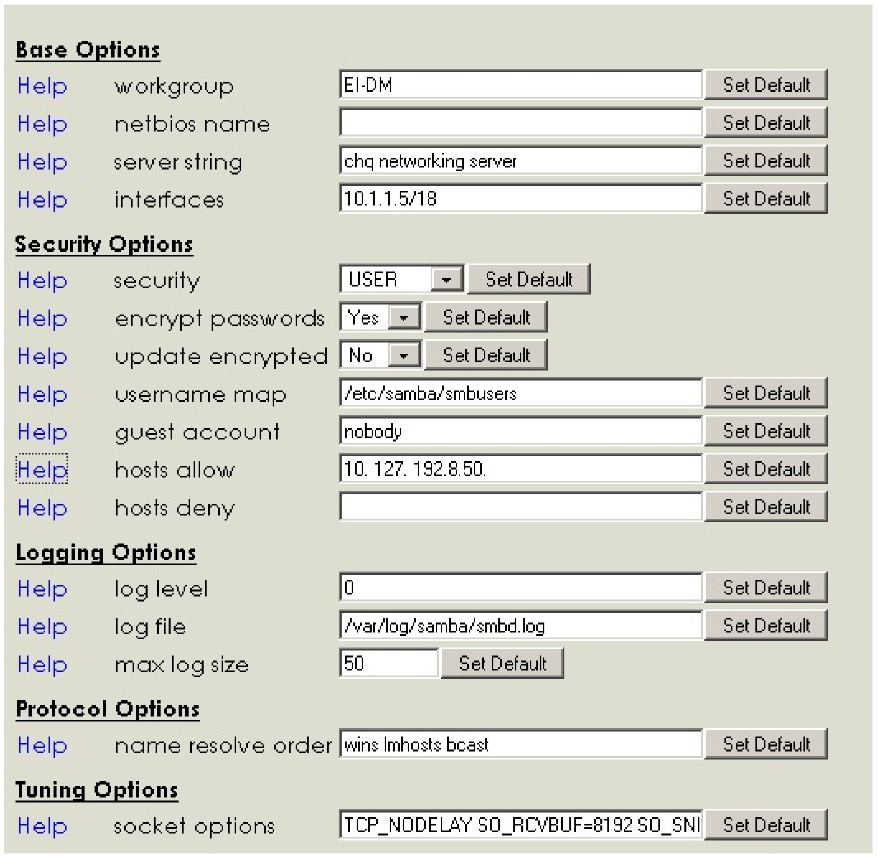
```
only_from        = 127.0.0.1
```

Comment this line by putting a # in column 1. Reboot the system (or locate and kill the xinetd process).

Configuration Examples

These are the values being used in netboxu:

Global Variables



Base Options

workgroup	EI-DM
netbios name	
server string	chq networking server
interfaces	10.1.1.5/18

Security Options

security	USER
encrypt passwords	Yes
update encrypted	No
username map	/etc/samba/smbusers
guest account	nobody
hosts allow	10. 127. 192.8.50.
hosts deny	

Logging Options

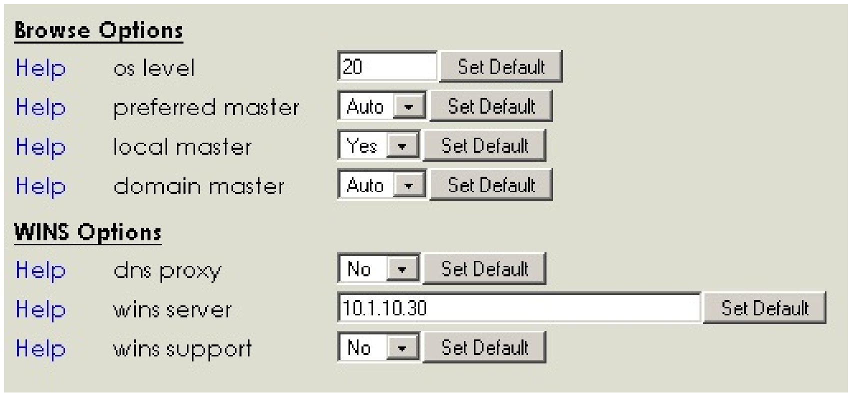
log level	0
log file	/var/log/samba/smbd.log
max log size	50

Protocol Options

name resolve order	wins lmhosts bcast

Tuning Options

socket options	TCP_NODELAY SO_RCVBUF=8192 SO_SNI



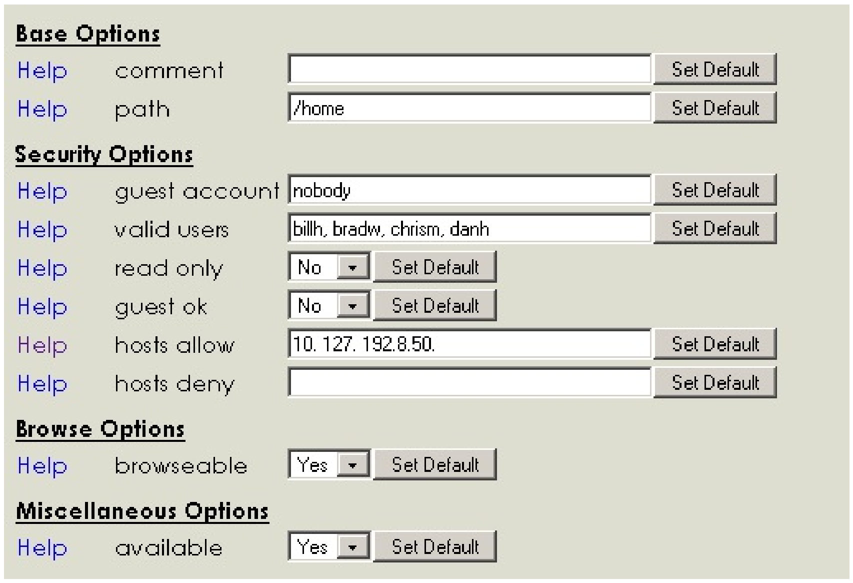
Browse Options

os level	20
preferred master	Auto
local master	Yes
domain master	Auto

WINS Options

dns proxy	No
wins server	10.1.10.30
wins support	No

Here is an example of a share:


Base Options

Help	comment		Set Default
Help	path	/home	Set Default

Security Options

Help	guest account	nobody	Set Default
Help	valid users	billh, bradw, chrism, danh	Set Default
Help	read only	No	Set Default
Help	guest ok	No	Set Default
Help	hosts allow	10. 127. 192.8.50.	Set Default
Help	hosts deny		Set Default

Browse Options

Help	browseable	Yes	Set Default

Miscellaneous Options

Help	available	Yes	Set Default

Configuring a User

The process of mapping a windows users to a linux user is rather confusing. Essentially you must tell Samba that if you logged on as windows user "X", that must map to linux user "Y". This mapping of windows names to linux names is found in the file /etc/samba/smbusers.

To add a new mapping, edit the file and add a line. For example, my linux user is danh and my domain user is chq-danh, so the mapping line is as follows:

danh = chq-danh

Once the user is setup, you must tell Samba your Windows password using smbpassword:

To add:

```
smbpasswd -a danh
```

To change:

```
smbpasswd danh
```

Using Samba

Windows Client to Unix Samba Server

Accessing a Samba Server from a Windows client is just like accessing a Windows server.

To access the directory /ei/netflow on the server netboxu (netboxu being the netbios name), simply access \\netboxu\ei\netflow.

Unix Client to Windows Server using smbclient

smbclient is a quick way to connect to a windows server to transfer files. This is probably the preferred way as using smbmount is probably considered a bit more permanent (in fact by default won’t work for a normal user).

Assume you want to move the file \\chq-fs\sys\temp\test.txt to your Unix system into the directory /home/danh.

First, cd to the destination directory (or use lcd once in smbclient):

```
cd ~danh
```

Now connect to the file server using smbclient:

```
smbclient //chq-fs/sys -U chq-danh
```

smbclient is very much like ftp. Use cd go to the correct directory:

```
cd \temp
```

then get the file

```
get test.txt
```

Unix Client to Windows Server using smbmount

To allow a Unix system access to a Windows server, use smbmount.

To access chq-fs/sys, first you must have an empty directory to attach it to:

```
mkdir ./testmnt
```

Now do the mount:

```
smbmount //chq-fs/sys ./testmnt -o username=chq-danh
```

If you get a message indicating smbmount is not a known command, do it this way:

```
mount -t cifs //chq-fs/sys ./testmnt -o username=chq-danh
```

or

```
mount.cifs //chq-fs/sys ./testmnt -o username=chq-danh
```

For Raspberry Pi (RPI) I am finding, I need to do it like this:

```
mount -t cifs -o sec=ntlm,user=danh //dev/share /mnt/mntpoint
```

Note: So far I've not figured the trick to doing this w/o being root. In reading the Samba Server Admin Book, it appears that in order to allow normal users the ability to use smbmount, and smbumount, you must chmod the executables:

```
chmod a+s /usr/bin/smbmnt /usr/bin/smbumount
```

(Indeed, I've tested that and it does work for me).

When you are done, unmount:

```
smbumount ./testmnt
```

Printing a File from a Unix System to a Windows Printer

Get your output into a flat file such as:

```
ls >/tmp/output.txt
```

Connect to the Windows printer (lj_PS in this case):

```
smbclient //chq-danh-xp/lj_PS -U chq-danh
```

Turn on CR/LF translation:

```
translate
```

and finally print the file:

```
print /tmp/output.txt
```

Notes for General Unix

To see what shell you are using:

```
echo $SHELL
```

Shell filename wildcard matches:

?	Any single character
*	Any group of zero or more characters
[ab]	Either a or b
[a-z]	Any character between a and z, inclusive

Some standard extensions

.a	Archive file (library)
.c	C program source file
.f	FORTRAN program source file
.F	FORTRAN program source file to preprocess
.gz	gzipped file
.h	C program header file
.html	HTML file for World Wide Web servers
.o	Object file (compiled and assembled code)

.s	Assembly language code
.z	Packed file
.Z	Compressed file
.1 to .8	Online manual source file
.txt	ASCII text file
.tar	tar archive (19.5)
.shar	Shell archive (19.2)
.sh	Bourne shell script (1.5)
.csh	C shell script (47.2)
.mm	Text file containing troff's mm macros (43.14)
.ms	Text file containing troff's ms macros (43.14)
.ps	PostScript source file

Set the prompt to include full path:

```
PS1='$PWD\$'
```

Append “/” to directories in ls command

```
alias ls=”ls -F”
```

To dump all accounts with root access:

```
grep “:0:” /etc/passwd
```

To prevent overwriting files with redirection (>)

```
set noclobber
```

ls >myfile	fails if myfile exists
ls >| myfile	will overwrite myfile

cd ...

In dos you can type cd ... to go back 2 directories or cd .... to go back 3. Can’t do that in unix. Instead cd ../..

Maintaining current directory

If you want to do something in another directory but don’t want to cd back to the directory you are in, put the command in ():

```
cd
(cd /etc; ls)
pwd
```

The final pwd will print your home directory, not /etc.

tar

To create a “tarball”:

```
tar cf <filename.tar> <dir>
gzip <filename.tar>
```

To extract:

```
gunzip <filename.gz>
tar xf <filename.tar>
```

To list contents of a tarball:

```
tar tf <filename.tar>
```

To make Gnome’s terminal respond correctly to arrow keys

Select Edit | Current Profile from the Terminal menu
Select Title and Commands Tab
Enable “Run command as login shell”

Use \ to escape

`echo 1 > 2`	Send “1" to the file 2
`echo 1 \> 2`	Send “1 > 2" to terminal

Use \ for line continuation

```
echo This is a really \
  long line
```

To create unique history files for every session (not just every user):

In .profile add:

```
HISTSIZE=1024
HISTFILE=~/.hist$$
```

Using pscp

This program allows you to transmit/receive files between a unix host and your pc. To get the parameters, just type

```
pscp
```

to transfer c:/xyz.txt to my unix box, I type:

```
pscp c:/xyz.txt danh@danh-solaris:/export/home/danh/
```

Getting Rid of find errors

When doing a “find /” command, you will see a lot of errors regarding directories you don’t have access to. It is hard to see where the file is between all of the errors. Get rid of these by terminating the command with 2>/dev/null:

```
find / -name “*test*” 2>/dev/null
```

To enter text directly into a file

```
cat > filename
```

/etc/motd

Contains the message of the day (displayed at login). To display the file, /etc/profile must be modified to include:

```
cat /etc/motd
```

Shell Expressions

Here are some notes on how to use expressions (and the IF statement) in ksh/bash:

test expression will set $?

```
test 1 -eq 2
if $? ...
```

[ ] is the same (note that [ ] must have spaces around them)

```
[ 1 -eq 2 ]
if $? ...
```

if you use [ ] then things with special meaning have to be escaped:

```
[ \( 1 -eq 2 \) ]
```

so you can use *[[ ]]* instead:

```
[[ (1 -eq 2) ]]
```

this can be joined into the if statement (I guess endif would be a bit too much to have to type, so they decided on the ever so mnemonic fi to terminate an if block):

```
if [[ (1 -eq 2) ]]
then
echo whatever
fi
```

the then can be put on the same line (almost like a real programming language):

```
if [[ (1 -eq 2) ]]; then
```

```
echo whatever
fi
```

! is used for negation. Note that ! must have spaces around it too:

```
if [[ ! (1 -eq 2) ]]; then
echo whatever
fi
```

Notes for Solaris

Fixing the line edit problem at the :: prompt

When you first login, the MPE emulator screws up the terminal settings so you cannot use backspace or even control-U.

When you first get the “::” prompt, type “FIXBS” and that should fix it.

If FIXBS doesn’t exist, then type “reset” and that too will fix it although it may have undesirable side effects.

To print a large banner:

```
banner <text>
```

To create windex file (so apropos/whatis will work)

```
su
catman
```

wait quite a while

Removable Media

Device names

/floppy/floppy0
/cdrom/cdrom0

To mount a floppy:

Insert into drive
```
volcheck -v
```

To eject a floppy:

eject floppy

To format a floppy (this is not a DOS format, see below):

```
rmformat -F quick -U /dev/rdiskette0
```

To write a unix file system to the floppy:

```
/usr/sbin/newfs -v /vol/dev/aliases/floppy0
```
eject
```
volcheck -v
```
Now /floppy/floppy0 should be accessible.

To format a floppy w/ FAT file system:

Insert floppy
```
volcheck -v
fdformat -d
```
eject floppy
```
volcheck -v
```
floppy will be mounted as /floppy/floppy0

Locating CDRW drives:

```
cdrw -l
```

Erasing a CDRW disk:

```
cdrw -b all
```

Writing the directory /export/home/danh to CD (note: I was unable to test this):

```
mkisofs -r /export/home/danh | cdrw -i
```

To mount a CD:

Just insert it

To eject a CD:

```
eject cdrom
```

Packages

Packages are applications setup to be installed easily with the package manager.

Packages can be managed with admintool and smc graphically (in Gnome, type admintool & or smc & at a cmd line).

To list installed packages:

```
pkginfo | more
pkginfo | grep “<pkgname>”
```

To get detailed info about an install:

```
pkginfo -l <pkgname>
```

To install a package

```
pkgadd -d <pkgname>
```

To verify installation was goo:

```
pkgchk -v <pkgname>
```

To remove a package

```
pkgrm <pkgname>
```

Patches:

To list all patches:

```
patchadd -p
```

To list a specific patch

```
patchadd -p | grep 12345
```

To install patches

get the latest patches by going to www.sun.com.

If you need to be in single user state (pretty good chance for a big patch, type “init 1" to stop the system and go to single user state.

(my experience with single use mode is you have no echo and must use ^j to end a line instead of enter).

```
patchadd <path>
```
where <path> points to the directory of the patch.

To Remove a patch

You cannot remove a patch if it was obsoleted by another patch. You also cannot remove it if you specified patchadd -d (this doesn’t save backup info).

`patchrm <patchid>` such as patchrm 113998-05

System Log Files

These are located in /var/adm:

```
more /var/adm/messsages
```

or just

```
dmesg
```

ftpconfig

To enable anonymous FTP:

```
ftpconfig <directory path>
```

to remove:

```
ftpconfig -d <directory path>
```

snoop

requires super user

to monitor traffic between <src> and <dst> Ip addresses

```
snoop <src> <dst>
```

to see whole packets

```
snoop -v <src> <dst>
```

to monitor a specific port

```
snoop tcp port 23 <src> <dst>
```

To restart networking services

```
ps -e | grep “inetd”
kill <pid>
```

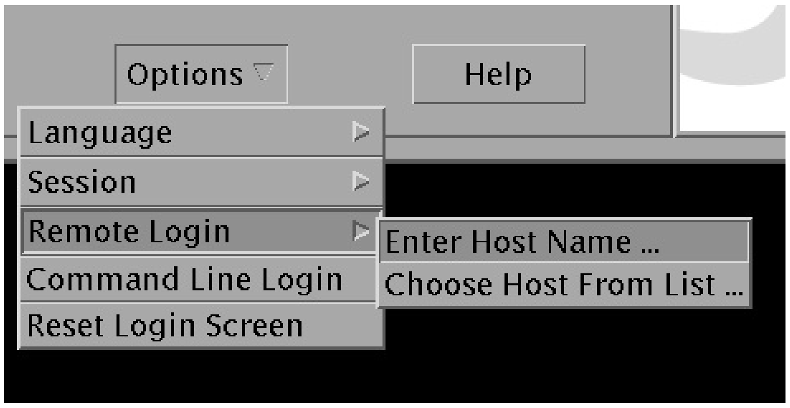
To force system to look for new devices on next reboot:

```
touch /reconfigure
init 0
```
(preps system to be turned off to allow new devices)

Xterminal Access

If you are running Solaris on your desktop (such as my Vmware 386pc solaris install), you can login to another solaris system, as long as it is in the same building (performance and router filters stop it elsewhere).

To do this, at the login screen, select enter hostname before logging in as follows:



Now enter a user/pw for the remote system.

You can also install XWin32 and access a Unix system via Gnome or CDE from the Windows desktop.

Formatting Hard Drives

In VMWare, you can add an additional disk by using the new hardware wizard.

When adding a new disk, remember:

```
touch /reconfigure
init 0
```

To see a list of disks, just use format:

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0d0 <DEFAULT cyl 16641 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@0,0
       1. c0d1 <DEFAULT cyl 2078 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@1,0
Specify disk (enter its number):
```

To see if a disk is formatted, enter it's number. If unformatted you will get a message:

```
AVAILABLE DISK SELECTIONS:
       0. c0d0 <DEFAULT cyl 16641 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@0,0
       1. c0d1 <DEFAULT cyl 2078 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@1,0
Specify disk (enter its number): 0
selecting c0d0
Controller working list found
[disk formatted, defect list found]
```

Formatting a disk

Note: you cannot format a disk created in vmware - it is already formatted (I suspect this is a low layer format that won't work on any IDE drive).

```
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
0. c0t1d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0
1. c0t3d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0
Specify disk (enter its number):1
Selecting c0t3d0
[disk formatted]
format> format
Ready to format. Formatting cannot be interrupted
and takes 23 minutes (estimated). Continue? yes
Beginning format. The current time is Thu Dec 6 09:54:40 2001
Formatting ...
done
Verifying media...
pass 0 - pattern = 0xc6dec6de
2035/12/18
pass 1 - pattern = 0x6db6db6d
2035/12/18
```

```
Total of 0 defective blocks repaired.
format>
```

Creating Solaris partition using format’s fdisk command. You do this on an Inter machine, not a Sparc machine:

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0d0 <DEFAULT cyl 16641 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@0,0
       1. c0d1 <drive type unknown>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@1,0
Specify disk (enter its number): 1

AVAILABLE DRIVE TYPES:
        0. DEFAULT
        1. other
Specify disk type (enter its number): 0
selecting c0d1
No current partition list
No defect list found
[disk formatted, no defect list found]
format> fdisk
No fdisk table exists. The default partition for the disk is:

  a 100% "SOLARIS System" partition

Type "y" to accept the default partition, otherwise type "n" to edit the
partition table.
n
             Total disk size is 2080 cylinders
             Cylinder size is 1008 (512 byte) blocks

                                               Cylinders
      Partition   Status    Type          Start   End   Length    %
      =========   ======    ============  =====   ===   ======   ===

WARNING: no partitions are defined!

SELECT ONE OF THE FOLLOWING:

   1. Create a partition
   2. Specify the active partition
   3. Delete a partition
   4. Exit (update disk configuration and exit)
   5. Cancel (exit without updating disk configuration)
Enter Selection: 1
Select the partition type to create:
   1=SOLARIS   2=UNIX        3=PCIXOS     4=Other
   5=DOS12     6=DOS16       7=DOSEXT     8=DOSBIG
   9=DOS16LBA  A=x86 Boot    B=Diagnostic C=FAT32
   D=FAT32LBA  E=DOSEXTLBA   0=Exit? 1
Specify the percentage of disk to use for this partition
(or type "c" to specify the size in cylinders). 50
Should this become the active partition? If yes, it will be activated
each time the computer is reset or turned on.
Please type "y" or "n". n
```

```
               Total disk size is 2080 cylinders
               Cylinder size is 1008 (512 byte) blocks

                                                  Cylinders
        Partition   Status    Type          Start   End   Length    %
        =========   ======    ============  =====   ===   ======   ===
            1                 Solaris           1  1040    1040     50
```

To save the partition table, you must exit correctly:

```
SELECT ONE OF THE FOLLOWING:

   1. Create a partition
   2. Specify the active partition
   3. Delete a partition
   4. Exit (update disk configuration and exit)
   5. Cancel (exit without updating disk configuration)
Enter Selection: 4
```

Creating the Disk Slices (partitions) and labeling the Disk

This part is rather confusing. Even though you created partitions with fdisk, you evidently need to assign those to slices which is also called partitions. If you are building a single partition disk, this isn't too big of a deal, it all goes into a single slice/partition which appears to always be slice 2 (unless you override that).

Assuming the above partition was built with fdisk, do the following:

```
format> partition


PARTITION MENU:
        0      - change `0' partition
        1      - change `1' partition
        2      - change `2' partition
        3      - change `3' partition
        4      - change `4' partition
        5      - change `5' partition
        6      - change `6' partition
        7      - change `7' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        !<cmd> - execute <cmd>, then return
        quit
partition> modify
Select partitioning base:
        0. Current partition table (my table)
        1. All Free Hog
Choose base (enter number) [0]? 1

Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0               0         (0/0/0)         0
  1       swap    wu       0               0         (0/0/0)         0
  2     backup    wu       0 - 1037      510.89MB    (1038/0/0) 1046304
  3 unassigned    wm       0               0         (0/0/0)          0
  4 unassigned    wm       0               0         (0/0/0)          0
  5 unassigned    wm       0               0         (0/0/0)          0
  6        usr    wm       0               0         (0/0/0)          0
  7 unassigned    wm       0               0         (0/0/0)          0
  8       boot    wu       0 -    0        0.49MB    (1/0/0)       1008
  9 alternates    wm       1 -    2        0.98MB    (2/0/0)       2016

Do you wish to continue creating a new partition
table based on above table[yes]? **yes**
Free Hog partition[6]?
Enter size of partition '0' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '5' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '7' [0b, 0c, 0.00mb, 0.00gb]:

Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0               0         (0/0/0)          0
  1       swap    wu       0               0         (0/0/0)          0
  2     backup    wu       0 - 1037      510.89MB    (1038/0/0) 1046304
  3 unassigned    wm       0               0         (0/0/0)          0
  4 unassigned    wm       0               0         (0/0/0)          0
  5 unassigned    wm       0               0         (0/0/0)          0
  6        usr    wm       3 - 1037      509.41MB    (1035/0/0) 1043280
  7 unassigned    wm       0               0         (0/0/0)          0
  8       boot    wu       0 -    0        0.49MB    (1/0/0)       1008
  9 alternates    wm       1 -    2        0.98MB    (2/0/0)       2016

Okay to make this the current partition table[yes]?
Enter table name (remember quotes): **"Dan Partition Table"**

Ready to label disk, continue? **y**
```

Now exit the partition section and verify the disk (look for errors right after the verify command - they tend to scroll off):

```
partition> quit


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        fdisk      - run the fdisk program
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        save       - save new disk/partition definitions
        volname    - set 8-character volume name
        !<cmd>     - execute <cmd>, then return
        quit
```

```
format> verify

Primary label contents:

Volume name = <        >
ascii name  = <DEFAULT cyl 1038 alt 2 hd 16 sec 63>
pcyl        = 1040
ncyl        = 1038
acyl        =    2
bcyl        =    0
nhead       =   16
nsect       =   63
Part      Tag    Flag     Cylinders        Size            Blocks
  0 unassigned    wm       0                0         (0/0/0)          0
  1 unassigned    wm       0                0         (0/0/0)          0
  2     backup    wu       0 - 1037      510.89MB    (1038/0/0) 1046304
  3 unassigned    wm       0                0         (0/0/0)          0
  4 unassigned    wm       0                0         (0/0/0)          0
  5 unassigned    wm       0                0         (0/0/0)          0
  6 unassigned    wm       3 - 1037      509.41MB    (1035/0/0) 1043280
  7 unassigned    wm       0                0         (0/0/0)          0
  8       boot    wu       0 -    0        0.49MB    (1/0/0)       1008
  9 alternates    wm       1 -    2        0.98MB    (2/0/0)       2016

format>
```

Verify the partition information with the prtvtoc command:

```
# prtvtoc /dev/dsk/c0d1s2
* /dev/dsk/c0d1s2 partition map
*
* Dimensions:
*     512 bytes/sector
*      63 sectors/track
*      16 tracks/cylinder
*    1008 sectors/cylinder
*    1040 cylinders
*    1038 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       2      5    01          0   1046304   1046303
       6      0    00       3024   1043280   1046303
       8      1    01          0      1008      1007
       9      9    00       1008      2016      3023
```

Installing the file system

```
# newfs /dev/rdsk/c0d1s2
newfs: construct a new file system /dev/rdsk/c0d1s2: (y/n)? y
/dev/rdsk/c0d1s2:      1046304 sectors in 1038 cylinders of 16 tracks,
63 sects
        510.9MB in 65 cyl groups (16 c/g, 7.88MB/g, 3776 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 16224, 32416, 48608, 64800, 80992, 97184, 113376, 129568, 145760,
 887520, 903712, 919904, 936096, 952288, 968480, 984672, 1000864,
1017056,
 1032224,
```

Mounting the new Disk

In the above example, slice 2 contains the user data (slice 6 contains the unused space). This disk is channel 1, disk 1, slice 2 or c0d1s2.

Now assume we want to mount c0d1s2 to /dansvol

Edit /etc/vfstab to include the appropriate entry (use export/home as an example). Make sure you get everything correct as a mistake will blow out the operating system and you will have to repair it in single-user mode.

```
cp /etc/vfstab /etc/vfstab.org
```
{Just in case}

and add the line (note I’m enabling UFS logging here):

```
/dev/dsk/c0d1s2 /dev/rdsk/c0d1s2 /dansvol  ufs  1 yes logging
```

create the /dansvol directory:

```
mkdir /dansvol
```

Mount:

```
mountall
```

verify it is mounted

```
# mount
```

to unmount:

```
umount /dansvol
```

Using an FAT32 Volume

Although the manuals allude to being able to fdisk, mkfs, and mount a FAT32 file system, I have not been able to get it to work. The solaris fdisk program doesn’t seem to put a drive letter on the partition, but you must have a drive letter for mkfs and mount to work.

To get around this,

```
touch /reconfigure
init 0
```
{tell system a config change is occurring}
{halt the operating system}

boot the system using an MSDOS diskette. Use the MSDOS fdisk and format commands to create and format the partition.

Once the partition is created, start solaris and mount the partition in solaris using:

```
mkdir /dos
mount -F pcfs /dev/dsk/c0d1p0:c /dos
```

`mkdir /dos`	{or something appropriate}
`mount -F pcfs /dev/dsk/c0d1p0:c /dos`	{where p0 is partition 0 and :c is the drive letter}

Ethereal

Found on the companion CD. Once installed, run from /opt/sfw/bin.

Companion CD MAN pages

The companion software is installed into /opt/swf. To see the MAN pages, you need to add the following to your MANPATH environment variable:

```
:/opt/swf/man
```

as in:

```
:/usr/man/:/usr/local/man:/usr/man/:/usr/local/man:/opt/sfw/man
```

tcpdump

tcpdump is on companion software CD. Once installed, requires super-user and is installed in /opt/swf/sbin.

top

top in on companion software CD. Once installed, resides in /opt/swf/bin.

To allow telnet sessions to use root:

```
vi /etc/default/login
```

comment out line CONSOLE=/dev/console

and while you are at it you may want to set SLEEPTIME to 0.

To permanently set showmode for vi

create the file .exrc in your home directory with *set showmode* in it.

Configuring a Network Printer

The following configures a printer with the local name m02n5 and description M02N5 to send data to the printer queue m02n5 on chq-fs:

```
lpadmin -p m02n5 -D “M02N5" -I any -o nobanner -s chq-fs!m02n5
```

and this causes that printer to become the default:

```
lpadmin -d m02n5
```

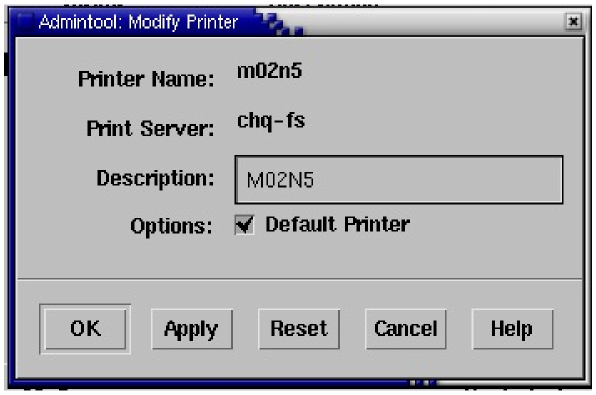
once configured, verify:

```
/>lpstat m02n5

                              Windows 2000 LPD Server
                                    Printer \\10.1.10.30\m02n5

Owner         Status           Jobname           Job-Id    Size
Pages
```

and verify by printing:

```
/>lp /etc/profile
request id is m02n5-1 (1 file)
```

This can also be setup in admintool as follows


Admintool: Modify Printer

Printer Name: m02n5

Print Server: chq-fs

Description: M02N5

Options: [✓ Default Printer]

To print to a non-default printer:

```
/>lp -d m02n4 /etc/profile
request id is m02n4-2 (1 file)
```

Deleting a Network Printer

```
lpadmin -x m02n5
```

To disable Desktop Manager at System Startup

```
/usr/dt/bin/dtconfig -d
```

to reenable it

```
/usr/dt/bin/dtconfig -e
```

To Start the Desktop Manager manually

```
/usr/dt/bin/dtlogin -daemon; exit
```

Dealing with Memory Questions

If you think there are memory issues:

```
prtconf
```

To see the total physical memory

```
sar -k 1 1
```

to se the amount of kernel memory. Add together sml_mem, lg_mem, and ovsz_alloc to determine total memory usage.

```
vmstat 3
```

This will execute the vmstat command every 3 seconds. Disregard the first iteration. The real free memory figure is on the 2nd line. The output is in KB.

See Solaris Internals, chapter 7 for more information.

Notes for Red Hat Linux

To create whatis file (so apropos/whatis will work)

```
su
/usr/sbin/makewhatis
```

wait quite a while

If you have a text-based login prompt and you want to use the GUI

login and then type “startx”

To restart network

```
service network restart
```

To add terminal to the panel

Right click on Panel (towards right where it is empty) then:
add to panel | launcher from menu | system tools | Terminal

Accessing Windows systems

In Nautilus, type “smb:” to see workgroups.

To configure KDE

Click on red hat | control center

To config Graphical Login Screen

Run gdmsetup as root user

Allows you to alter various things about login screen including xdmcp (remote login)

To add your own commands or scripts

put files in */home/<user>/bin*. This directory is automatically searched.

RPM’s

```
rpm -qa
```
To list all installed rpm’s

```
rpm -qi <rpm name>
```
To display info about rpm

```
rpm -ql <rpm name>
```
To display all files in an (installed) RPM

```
rpm -qf <fully qualified filename>
```
Tells what rpm the file belongs to

```
rpm -q –whatrequires <package>
```
Tells which packages depend on <package>

```
rpm -e <package>
```
Removes package

```
rpm –rebuilddb -v
```
Rebuild the rpm database (sadly I had to use this after my first attempt to remove an RPM.

My favorite location for RPMs has been rpm.pbone.net.

Screensaver

You don’t want to run a screen saver on vmware - it will burn up a lot of cpu cycles. To disable, click redhat | preferences|screensaver

gftp <ftp server>

Graphical FTP (kind of like CuteFTP)

System Logs Utility

System Tools | System logs

To mount cdrom

```
mount /mnt/cdrom
```

In Fedora Core 3, this isn’t working. Instead, you have to know the actual hard drive descriptor and mount that to a directory. On my current install, the descriptor is hda. To find this, you can type:

```
dmesg | grep “^hd.:”
```

Then create a directory such as:

```
mkdir /cdrom
```

and then mount:

```
mount /dev/hda /cdrom
```

To mount floppy

```
mount /mnt/floppy
```

To unmount cdrom

```
umount /mnt/cdrom
```

or

```
umount /cdrom
```

To unmount floppy

```
umount /mnt/floppy
```

User Config utility

System Settings | Users and Groups

To stop Xterminal:

```
Init 3
```

Single User mode

`Init 1` (init 2 in solaris)

To determine run level:

```
runlevel
```

To add a printer

```
printconf-gui
```

or

`printconf` (runs from a telnet console)

FTP’ing to vsFTPd fails

If you are running the vsFTPd daemon on your server you will not be able to login to any user that uses ksh as the shell. Use anonymous.

Installing NTP

(from http://www.siliconvalleyccie.com/linux-hn/ntp.htm)

Make sure NTP is installed:

```
/root>rpm -qa | grep ntp
chkfontpath-1.9.7-1
ntp-4.1.2-0.rc1.2
```

Edit the NTP configuration file:

```
vi /etc/ntp.conf
```

and add:

```
restrict 10.1.1.1 mask 255.255.255.255 nomodify notrap noquery
server 10.1.1.1
restrict 127.0.0.1
```

```
            #restrict default ignore
```

save config file and type:

```
            /etc>date
            Wed Jul  7 07:57:21 PDT 2004
            /etc>ntpdate 10.1.1.1
             7 Jul 09:12:41 ntpdate[25093]: step time server 10.1.1.1 offset
4511.868602 sec
            /etc>ntpdate 10.1.1.1
             7 Jul 09:12:44 ntpdate[25111]: adjust time server 10.1.1.1 offset
0.015902 sec
            /etc>date
            Wed Jul  7 09:12:50 PDT 2004
```

and to force NTP to start at boot, type:

```
            chkconfig --level 35 ntpd on
```

To start, stop NTP:

```
            /etc/init.d/ntpd start
            /etc/init.d/ntpd stop
            /etc/init.d/ntpd restart
```

Updating O/S with up2date

This is the normal manner used to keep the O/S and the various RPMs up to date.

1. First, you must go to the redhat website and create a logon id.
2. Next run up2date –register to tell it your login/password
3. Use up2date –dry-run to see what it will update
4. Go to single user mode
5. Use up2date -u to update the operating system

up2date parameters	
`-show-channels`	list subscribed channels
`-showall`	list all available files
`-i`	installs a package
`–dry-run`	shows what would happen w/o actually doing it.
`-u`	Updates the system.

Lost root Password

(from Getting Started Guide)

If you loose root’s password, you reset it by telling the boot loader to boot linux in single user mode and then setting the password to something new

- Reboot the system
- When grub comes up, type “e” to enter edit mode. Select the kernel. Type “e” again to edit it. At the end of the line, add “single”. Now type “b” to boot.
- The system boots up in single user mode.
- Use passwd root to fix the password
- Reboot
- edit grub again and remove “single”. (Actually this seems to have been done automatically).

Copying files to a CD in Linux

You will need the programs mkisofs and cdrecord. Locate and install these RPM’s if you are missing the programs. Mkisofs creates a cd image of the files you want to copy and then cdrecord records the image to CD.

Note: I had to install this from the Redhat install CD. The official version of cdrecord (now called cdrtools) would fail on the cdrecord -scanbus command.

1. Check cdrecord version.

```
cdrecord --version
Cdrecord-Clone 2.01-dvd (i686-pc-linux-gnu) Copyright (C) 1995-2004 JC6rg Schilg
```

2. Check mkisofs version

```
mkisofs --version
mkisofs 2.01 (i686-pc-linux-gnu)
```

3. To locate the IDE/ATAPI device name for the CD Rom drive, type:

```
dmesg | grep '^hd.:'
hda: SONY CD-RW CRX300E, ATAPI CD/DVD-ROM drive
hda: ATAPI 48X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
```

In this case, my CDROM is on hda.

4. To create an ISO file:

```
mkisofs  -iso-level 3 -J -r -V disklabel -o OutputFileName file[ file...]
```

For example:

```
mkisofs -o /tmp/test.iso  -iso-level 3 -J -r -V label /root
INFO:   UTF-8 character encoding detected by locale settings.
```

```
        Assuming UTF-8 encoded filenames on source filesystem,
        use -input-charset to override.
Unknown file type (unallocated) /root/.. - ignoring and continuing.
  7.81% done, estimate finish Mon Jan 24 20:36:34 2005
 15.59% done, estimate finish Mon Jan 24 20:36:34 2005
 23.37% done, estimate finish Mon Jan 24 20:36:38 2005
 31.14% done, estimate finish Mon Jan 24 20:36:37 2005
 38.95% done, estimate finish Mon Jan 24 20:36:39 2005
 46.72% done, estimate finish Mon Jan 24 20:36:38 2005
 54.50% done, estimate finish Mon Jan 24 20:36:39 2005
 62.30% done, estimate finish Mon Jan 24 20:36:40 2005
 70.08% done, estimate finish Mon Jan 24 20:36:39 2005
 77.85% done, estimate finish Mon Jan 24 20:36:41 2005
 85.65% done, estimate finish Mon Jan 24 20:36:41 2005
 93.44% done, estimate finish Mon Jan 24 20:36:40 2005
Total translation table size: 0
Total rockridge attributes bytes: 79011
Total directory bytes: 215288
Path table size(bytes): 906
Max brk space used a5000
64222 extents written (125 MB)
```

5. Mount the new volume to test it:

```
mkdir /MountPoint
mount -t iso9660 -o ro,loop=/dev/loop0 ISOFile /MountPoint
```

such as

```
#mkdir /test_iso
#mount -t iso9660 -o ro,loop=/dev/loop0 /tmp/test.iso /test_iso
#cd /test_iso
# dir
total 68M
-r--r--r--   1 root root 1.2K Jan  4 20:36 anaconda-ks.cfg
-r-xr-xr-x   1 root root 1.2M Jan  7 13:14 ast-ksh*
-r--r--r--   1 root root 548K Jan  5 17:57 cdrecord-2.01.1-5.i386.rpm
dr-xr-xr-x   3 root root 2.0K Jan  9 18:07 gpc/
-r--r--r--   1 root root 7.9M Jan  9 18:06 gpc-20041218-with-gcc.i686-pc-linux-z
-r--r--r--   1 root root  51K Jan  4 20:36 install.log
```

```
umount /MountPoint
```

6. Determine the (psuedo) SCSI port the CD drive is on:

```
cdrecord -scanbus
scsidev: 'ATA'
devname: 'ATA'
scsibus: -2 target: -2 lun: -2
Linux sg driver version: 3.5.27
Using libscg version 'schily-0.8'.
cdrecord: Warning: using inofficial libscg transport code version (schily - Red.
scsibus0:
        0,0,0     0) 'SONY    ' 'CD-RW  CRX300E  ' 'KYS2' Removable CD-ROM
        0,1,0     1) *
        0,2,0     2) *
        0,3,0     3) *
        0,4,0     4) *
```

```
          0,5,0     5) *
          0,6,0     6) *
          0,7,0     7) *
```

7. Write the file to CD:

```
cdrecord -v dev=SCSIAddr file
```

such as

```
cdrecord -v dev=ATAPI:0,0,0 /tmp/test.iso
```

Other options:

speed=*multiplier*
By default, it will burn as fast as possible. Use speed= to slow it down if you are having problems.
-eject
At end of writing, eject drive.
-dummy
Don't actually turn the laser on.

Sample Session:

```
cdrecord: No write mode specified.
cdrecord: Asuming -tao mode.
cdrecord: Future versions of cdrecord may have different drive dependent defaul.
cdrecord: Continuing in 5 seconds...
Cdrecord-Clone 2.01-dvd (i686-pc-linux-gnu) Copyright (C) 1995-2004 JC6rg Schilg
Note: This version is an unofficial (modified) version with DVD support
Note: and therefore may have bugs that are not present in the original.
Note: Please send bug reports or support requests to http://bugzilla.redhat.coma
Note: The author of cdrecord should not be bothered with problems in this versi.
TOC Type: 1 = CD-ROM
scsidev: 'ATAPI:0,0,0'
devname: 'ATAPI'
scsibus: 0 target: 0 lun: 0
Use of ATA is preferred over ATAPI.
Warning: Using ATA Packet interface.
Warning: The related Linux kernel interface code seems to be unmaintained.
Warning: There is absolutely NO DMA, operations thus are slow.
Using libscg version 'schily-0.8'.
SCSI buffer size: 64512
atapi: 1
Device type    : Removable CD-ROM
Version        : 0
Response Format: 2
Capabilities   :
Vendor_info    : 'SONY    '
Identifikation : 'CD-RW  CRX300E  '
Revision       : 'KYS2'
Device seems to be: Generic mmc2 DVD-ROM.
Current: 0x0009
Profile: 0x0010
Profile: 0x000A
```

```
Profile: 0x0009 (current)
Profile: 0x0008
Profile: 0x0002
Using generic SCSI-3/mmc   CD-R/CD-RW driver (mmc_cdr).
Driver flags   : MMC-3 SWABAUDIO BURNFREE FORCESPEED
Supported modes: TAO PACKET SAO SAO/R96P SAO/R96R RAW/R16 RAW/R96P RAW/R96R
Drive buf size : 1422080 = 1388 KB
FIFO size      : 4194304 = 4096 KB
Track 01: data   125 MB
Total size:      144 MB (14:16.32) = 64224 sectors
Lout start:      144 MB (14:18/24) = 64224 sectors
Current Secsize: 2048
ATIP info from disk:
  Indicated writing power: 6
  Is not unrestricted
  Is not erasable
  Disk sub type: Medium Type C, low Beta category (C-) (6)
  ATIP start of lead in:  -11231 (97:32/19)
  ATIP start of lead out: 359847 (79:59/72)
Disk type:    Short strategy type (Phthalocyanine or similar)
Manuf. index: 27
Manufacturer: Prodisc Technology Inc.
Blocks total: 359847 Blocks current: 359847 Blocks remaining: 295623
Forcespeed is OFF.
Speed set to 1411 KB/s
Starting to write CD/DVD at speed   8.0 in real TAO mode for single session.
Last chance to quit, starting real write    0 seconds. Operation starts.
Waiting for reader process to fill input buffer ... input buffer ready.
trackno=0
BURN-Free is ON.
Turning BURN-Free off
Performing OPC...
Starting new track at sector: 0
Track 01:  125 of  125 MB written (fifo 100%) [buf  99%]   8.0x.
Track 01: Total bytes read/written: 131526656/131526656 (64222 sectors).
Writing  time:  112.729s
Average write speed   7.9x.
Min drive buffer fill was 99%
Fixating...
Fixating time:   30.940s
cdrecord: fifo had 2072 puts and 2072 gets.
cdrecord: fifo was 0 times empty and 1996 times full, min fill was 93%.
```

8. To create a new multi-session CDROM

```
cdrecord -v -multi dev=SCSIAddr file
```

such as:

```
cdrecord -v -multi dev=ATAPI:0,0,0 /tmp/test.iso
```

9. To add new files to a multi-session CDROM:

During creation of the ISO file, you must extract the starting/ending track #'s of the CD (the CD must be inserted for this to work):

```
mkisofs  -iso-level 3 -J -r -V disklabel -o OutputFileName \
-C $(cdrecord dev=SCSIAddr -msinfo) -M SCSIAddr file[ file...]
```

such as

```
mkisofs -o /tmp/test.iso  -iso-level 3 -J -r -V label \
-C $(cdrecord dev=ATAPI:0,0,0 -msinfo) -M ATAPI:0,0,0 /etc
```

and then write:

```
cdrecord -v -multi dev=SCSIAddr file
```

such as:

```
cdrecord -v -multi dev=ATAPI:0,0,0 /tmp/test.iso
```

Garbage Characters in man or pstree

I’ve had a problem for several years where text in man is screwed up, for example ‘don’t’ shows up as ‘donbt’ and pstree also display’s ‘b’ instead of ‘+’ and ‘|’.

This is caused by LANG being set to ‘`en_US.UTF-8`' (unicode). By clearing this (enter the command ‘LANG=’), the problem is fixed.

eiprofile has been modified to set the LANG to nothing.

vi - the least possible to get by	
Vi command	**Function**
`i`	insert text before cursor. ESC ends insertion.
`a`	append text after cursor. ESC end appending.
`x`	Delete character at cursor
`dd`	delete current line
`:x`	Save changes and exit.
`/text`	Search file for “text”
`:q!`	Quit without saving changes

vi Boot Camp

To start vi editor, type

vi <filename>

If system crashes while you are in vi, type “vi -r” when you log in again to pickup the changes you had made.

Here are some additional vi and vim Manuals (vim is the *improved* vi available on Linux. It is not on the Sun systems at this time.)

vi Quick Reference Guide
vi Tutorial
vim Manual

COLORS

If you install the full version of vim on linux, syntax highlighting is turned on and there are a zillion colors. I really don't like this, myself. To get rid of it:

- edit ~/.vimrc (this file may well not exist)
- add `syntax off`
- Save, exit, test.

Vi command	Function
Text Commands	

`A`	insert text at end of line
`a`	insert text after cursor
`dd`	delete current line
`D`	delete rest of line
`I`	Insert text at beginning of line
`J`	Join next line with current
`p`	Inserted last deleted text after curser
`P`	Inserted last deleted text before cursor
`R`	begin overwriting
`r`	replace just character the cursor is on
`i`	insert text before cursor
`u`	Undo last change
`U`	Restore current line
`yy`	Copy current line (use p or P to paste)
`escape`	Exit text input mode back to command mode
`ctrl-L`	Redraw screen
Movement	
`h j k l`	left down up right
`gg or :0`	1st line of file (:100 would be 100th line)
`G`	end of file
`w`	forward a word
`b`	backward a word
`space`	right
`0`	beginning of line
`$`	end of line
`return`	next line

`H`	first line of screen
`L`	last line of screen
`ctrl-F`	forward 1 screen
`ctrl-B`	backward 1 screen
Searching	
`/xxx`	search forward for “xxx”. Drop “xxx” to repeat last /
`?xxx`	search backward for “xxx”. Drop “xxx” to repeat last ?
`:nnn`	go to line # nnn
`m<a-z>`	Mark location and name it a,b,c, etc.
`:%s/<x>/<y>/g`	Search for <x> and replace it with <y> everywhere
`‘<a-z>`	Return to marked location named a,b,c, etc.
Saving and Exiting	
`:x`	Save and Quit
`:w [<file>]`	Write text to <file>. If no <file>, simply saves current changes.
`:w!`	write even if file is readonly
`:n`	edit next file (assumes you open more than one)
`:q`	exit
`:q!`	exit w/o saving
Manipulating Blocks of text	
`yy then p`	*yy* copies the line you are on. Move cursor to new location and type *p* to paste.
`y<n> then p`	*y5* copies the current line + the next 5 lines (6 lines total) and *p* pastes them
`dd then p`	*dd* deletes the current line you are on. *p* then pastes them.
`d<n> then p`	*d1* deletes the current line + 1 next line and *p* pastes them.
`V`	In vim (not vi) use *V* to turn on visual line mode. You can then mark whole lines. Once marked, use *y* or *d* to copy/delete them and p to paste them.

`v`	In vim, use *v* to turn on visual character mode to mark characters then *y* or *d* to copy/delete and *p* to paste.
`ctrl-v`	As above except this works in block mode (selects a rectangle of text).
Miscellaneous	
`:set showmode`	Tells you which mode you are in
`:sh`	Enter Shell mode
`:!<cmd>`	Do the command
`!!<cmd>`	Do the command <cmd> and put the output into the editor
Special commands while in Insert Mode (vim only)	
`Arrow keys`	These will move you anywhere in the text
`ctrl-o`	Escapes insert mode and allows a normal command and returns to insert mode.
`ctrl-u`	Undo all changes to the current line
`ctrl-x then ctrl-e or ctrl-y`	Enters scroll mode. You can then repeatedly use ctrl-e & ctrl-y to move up and down.
`Del key`	The Del key doesn’t work for me, but I can make ^d work like it by using this: :set t_kD=^v^d {actually type the control characters}

Unix Performance Notes

These notes are taken from “System Performance Tuning”.

SAR

SAR can provide long term statistics.

Use /usr/lib/sa/sa1 and sa2 in crontab to collect stats (pg 21).

use SAR -A to dump all stats.

Examining a processes’ Priority in Linux

Use ps -el to see priority.

Use renice to change the nice portion of the priority. Only root can increase nice.

Examining a processes’ Priority in Sun

use ps -efcL so see priority for lightweight threads

To see Interrupt (IRQ) assignments

cat /proc/interrupts

Load Average

uptime shows the average load for the last 1, 5, and 15 minutes

vmstat columns

Use vmstat <interval time>. Ignore the first interval - it’s always bogus.

```
  procs                      memory      swap          io     system      cpu
 r  b  w   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id
 1  0  0 1147444 137524  72228 728948    0    0     1  3262  255   465 49 15 36
```

r: run queue length
b: runable but blocked
w: runable but swapped

swpd: the amount of virtual memory used (kB).
free: the amount of idle memory (kB).
buff: the amount of memory used as buffers (kB).

si: Amount of memory swapped in from disk (kB/s).
so: Amount of memory swapped to disk (kB/s).
bi: Blocks sent to a block device (blocks/s).
bo: Blocks received from a block device (blocks/s).

in: The number of interrupts per second, including the clock.
cs: The number of context switches per second.

us: % of CPU time system in usermode
sy: % of CPU time sytstem in system (kernel) mode
id: % of CPU time system is idle and/or blocked on io.

mpstat - multiprocessor stats

use mpstat -P ALL to see stats based on each cpu:

```
netboxu.chq: danh# mpstat -P ALL 10
Linux 2.4.20-8smp (netboxu)     03/22/06

10:21:41     CPU   %user   %nice %system   %idle    intr/s
10:21:51     all   10.10    0.00    1.35   88.55    232.60
10:21:51       0    8.00    0.00    1.20   90.80    232.60
10:21:51       1   12.20    0.00    1.50   86.30      0.00
```

Examining swap space size

On Sun:
sar -r <interval>:

```
$ sar -r 10

SunOS nss-dev 5.8 Generic_117350-33 sun4u    03/22/06

10:27:22 freemem freeswap
10:27:32   53869  1101354
```

on linux:

```
cat /proc/meminfo | head -3 | grep -vi mem
        total:    used:    free:  shared: buffers:  cached:
Swap: 3980992512 1175191552 2805800960
```

To see elapsed and cpu time for a process

Proceed the command with time such as:

```
netboxu.chq: ei# time whoami
root
    0.03s real     0.00s user     0.01s system
```

when the command terminates, the wall time, cpu time spent in user mode, and cpu time spent in system mode will be reported.

Expect Notes

Expect is a scripting language primarily meant to drive interactive programs that would otherwise be hard to script. My primary interest in Expect is to use it to drive Telnet essentially allowing scripts to be written to run on a unix system to telnet into another system much like a reflections script would be written to run on a PC to telnet into another system.

Creating A Script File

First, you must create a file that will invoke Expect correctly. To do this, on the first line of your new script file, type:

```
#!/usr/bin/expect
```

Save the file and alter it to allow execution:

```
chmod 755 <file>
```

Set a default timeout

In reflections, we would specify a timeout in the wait command (such as wait 0:0:30 for 'xxx'). In expect the timeout is set with a separate command:

```
set timeout 60
```

Tell Expect what program to run

Expect can run anything besides telnet, but since that is our focus that is what we will do. You simply use the 'spawn' command and then specify what you would normally type at the unix command prompt:

```
spawn telnet netboxu.chq.ei
```

Using Expect for a very simple telnet session

If you will be responding to just a few prompts that are all unique you can get by with a single expect command:

```
expect {
      "login:"        {send "netmgr\r";  exp_continue}
      "assword:"      {send "password\r"; interact}
      timeout         {puts "timeout";  exit}
      }
```

This simple command waits for 'login:' and 'password:' (the P is omitted since it isn't always upper or lowercase). When 'login' is found, 'netmgr' is transmitted ('\r' means do a carriage return at the end). The command exp_continue means to redo that expect command again.

When ‘password’ is found, the password is transmitted. ‘interact’ means to stop looking for input and let the user ‘interact’ with the program.

If a timeout occurs, an error message is displayed and the program exits.

Handling many different prompts

Usually when writing a reflections script, we transmit something, wait for something, transmit something, wait for something, etc until we complete the script.

Here is a sample script for getting connected to a router. It must wait for the ‘login’ prompt, then the password prompt, then a general prompt, and finally another password prompt. The simple expect used above won’t work because ‘password’ is requested twice and the password will be different the second time.

We’ll call this script ‘openrtr’ and the first (and only) parameter is the ip/dns of the router such as ‘openrtr rtr.chq.ei’.

```
#!/usr/bin/expect
set timeout 60

spawn telnet $argv

expect "assword:"
send "pass1\r"
expect ">"
send "enable\r"
expect "assword:"
send "pass2\r"
interact
```

This script starts telnet (passing the parameter to telnet that you passed to the script). It then waits for the password prompt, send the password, waits for the ‘>’ command prompt, sends the enable command, waits for the enable password, sends that password, and then lets the user start using telnet.

Expect Patterns

If you want to match a whole line, you probably want to include \n:

```
expect “this is a test\n”
```

You can anchor begin/end line as in a regexp:

```
expect “^test”
expect “test$”
```

You can use ‘?’ and ‘*’ as you would for glob patterns (e.g. filename matching as is done in the ls command). The manual warns that using ‘*’ at the end of a pattern doesn’t tend to do as you may want.

```
      expect “*hi”
      expect “?assword”
      expect “\[pP]assword”
      expect “\[a-f0-9]”
```

To use regexp with expect, use the ‘-re’ flag:

```
      expect -re “a*”
```

Expect IF statement

The underlying language for expect is Tcl. So if you need to know how the control structures in Expect work, you will actually study Tcl.

The IF statement is in the form

```
      if {expression} {
       <statements>
       }
      else {
       <statements>
       }
```

My primary use for ‘if’ in reflection scripts was to look for a timeout when waiting for a string - if I didn’t get what I was expecting then there is an error such as:

```
      transmit ‘^m’
      wait 0:0:10 for ‘#’
      if not found
        stop
        end
```

In Expect you would actually do this in the expect command:

```
      send “\r”
      expect {
            timeout           {puts “timeout”; exit}
            “#”
            }
```

Here is a simple IF statement:

```
      if {$count < 0) {
            set total 1
            }
```

FOR Loop

This is a real FOR statement unlike the poor excuse for FOR in the shells albeit in yet another format:

```
      for {<init>} {<logical>} {<update> {
       <statements>
       }
```

Here is an example of doing a FOR k:=1 to 10:

```
      for {set k 1 } {k>10} {incr k} {
            <statements>
            }
```

WHILE Loop

```
      while {<logical>} {
       <statements>
       }
```

Detecting the command prompt using EIPROFILE

The HP3000 always transmitted a DC1 any time it was waiting for input so we could easily write scripts to detect when the HP was waiting for input. When using eiprofile, the command prompt includes the DC1 (XOn) control character (aka control-Q). This allows you to detect exactly when it is awaiting a prompt at the shell (unfortunately this can’t be extended to work everywhere like it could on the HP).

So if you are using eiprofile and the prompt is working correctly, you can simply wait for control-Q:

```
      expect {
            timeout        {puts "timeout"; exit}
            "\021"
            }
```

Pausing The Script

Use sleep to cause a pause. This can be handy when working with modems where we’ve seen some problems sending a response to them too quickly:

```
      send “AT\m”
      sleep .5
```

Debugging

Expect will display information as it runs a script if you alter line one of the script to include -d as in this example:

```
      #!/usr/bin/expect -d
```

This produces a lot of output, but it will show the data stream coming from the process and how expect is matching that data stream.

Sample Script

This script connects to a router and does a ‘show version’:

```
#!/usr/bin/expect
set timeout 5

# This script is called in the format:
#   openrtr <rtr> <loginpw> <enablepw1> <enablepw2>

#Invoke telnet on the specified router.
# To access the first element of the argv array you must use lindex:

spawn telnet [lindex $argv 0]

#Wait for login pw prompt and send login pw
expect {
   timeout     {puts "timeout waiting for initial password prompt";
exit}
    -re "\[pP]assword:"
   }
send [lindex $argv 1]
send "\r"

#Wait for std user prompt and then send 'enable'
expect {
   timeout     {puts "timeout waiting for initial cmd prompt"; exit}
   -re "^.*>$"
   }
send "enable\r"

#Wait for enable pw prompt and send enable pw
expect {
   timeout     {puts "timeout waiting for 2nd password prompt"; exit}
   -re "\[pP]assword:"
   }
send [lindex $argv 2]
send "\r"

#If the normal prompt is returned, set pw to false. If a pw prompt
#is received, set pw to true.
expect {
   timeout     {puts "timeout waiting for 1st # prompt"; exit}
   -re "\[pP]assword:" {set pw true}
   -re "^.*#$" {set pw false}
   }

#If we received a 2nd pw prompt, try the 2nd enable password
if {$pw == "true"} {
   send [lindex $argv 3]
   send "\r"
   expect {
      timeout  {puts "Neither enable pw matched"; exit}
      -re "^.*#$"
      }
   }

send "term len 0\r"
expect {
   timeout     {puts "timeout waiting for # prompt"; exit}
   -re "^.*#$" {set pw false}
   }

send "show clock\r"
expect {
   timeout     {puts "timeout waiting for # prompt"; exit}
   -re "^.*#$" {set pw false}
   }

send "show version\r"
expect {
   timeout     {puts "timeout waiting for # prompt"; exit}
   -re "^.*#$" {set pw false}
   }

send "exit\r"
sleep 1
exit
```

ksh Script Writing Tips

There are some tips to help write scripts in ksh. Note that ksh and bash are extremely close so these should work in either.

I’m going to assume you already know how to write scripts. This isn’t a tutorial on how to write scripts but rather just various notes to help you transition to writing scripts in ksh.

I’ve found many short script writing pages on the net, but the best by far (at 600+ pages) is *Advanced Bash-Scripting Guide*. Since ksh is a superset of bash, this manual is very useful.

Contents

Script Header

At the top of each of my scripts, you will find:

```
#!/usr/bin/ksh
. /etc/eiprofile
eibatchmode quiet
```

You need to tell the script which shell is being used. Since everything I do is with ksh, the first line invokes ksh.

The second line forces eiprofile to be executed. Although this is done for the login shell automatically, it isn’t done when a subshell is executed and I almost always want access to things that eiprofile contains.

Finally, I use eibatchmode to cause the script to abort when an unexpected error occurs.

Constants

numbers

123	base 10
0123	base 8
0x123	base 16
4#123	base 4

strings

‘abc’
“abc”

Using “ ” allows variables and \ (like \t) to be expanded so
print “$RANDOM”
results in a random number being printed whereas
print ‘$RANDOM’
results in printing the word $RANDOM

You should always use ‘ ’ unless you know you need “ ”.

Variables

A variable is represented by ${varname} or $varname. ${varname} will work anywhere whereas $varname won’t such as:

```
print abc${myvar}xyz
print abc$myvarxyz
```

To assign a variable a value use '='. There CANNOT be a space on either side of the equal sign:

```
myvar=123
```

Special Variables

$?	exit status
$$	process id
$!	pid of last job run in background
$_	last command entered

Expressions

Numerical expressions are evaluated using (( )) such as; however, you must use $(()) for assignment (got to love Unix):

```
a=$(( 1 + 1 ))
```

Note that the spaces around (( )) are not needed (e.g. you can use $((1+1))); however, to be consistent with [[ ]] I always use them (it isn't option when using [[ ]]).

Inside of the (( )), you do not need to delimit variables. So

```
a=$(( a + 1 ))
```

is the same as

```
a=(( ${a} + 1 ))
```

Inside of a condition test, you don't use the $:

```
if (( a > 1 )); then
```

Operators

The following operators work inside of (( )):

+ - * /	
^	exponentiation
%	modulus
<<	bit shift left
>>	bit shift right
&	bit AND
|	bit OR
~	bit negate
!	bit NOT
^	bit XOR

&&	logical and
||	logical or

String Manipulation

String manipulation inside of ksh is not great. Bash is much better. In fact, had I not already started using ksh for so much I would probably have moved to bash just because it does strings much better.

`${varname:-default}`	if not set, use <default>
`${varname:=default}`	if not set, set varname to default and return default.
`${#string}`	length of string
`expr “${<str>}” : ‘regexp’`	finds location of regexp in str.
`expr index “${<str1>}” ‘<str2>’`	index of str2 in str1
`expr substr “${str}” <position> <len>`	return a substring

Commands

Normally a single command is placed on each line such as

```
dir
ps
```

You can string commands together using ; but this makes little sense in a script:

```
dir;ps
```

You can cause commands to be executed in a new subshell using ( ):

```
(dir;ps)
```

You can create a code block using {}:

```
{
dir
ps
}
```

&& executes a list of commands. As soon as one returns an error, the rest are skipped:

```
dir && ps && who
```

If ps returns an error, who is never executed.

|| execuites a list of commands. As soon as one does NOT return an error, the rest are skipped:

```
dir || ps || who
```

If dir works, ps and who will never be executed.

Assigning command output to a variable

Quite often you will find that you want to assign the output of a program to a variable. This is done quite simply using $():

```
myvar=$(ps)
```

You can string together commands. I commonly want to know how many lines are in a file which is done with:

```
myvar=$(cat file | wc -l |bc)
```

and that can easily be placed in a condition:

```
if (( $(ps | grep ‘ksh’ | wc -l | bc) > 0 )); then
```

if command

The if command performs a logical test and branches based on the outcome of the test.

The forms for the if command are:

```
if <test>; then
     <cmd>
     fi

if <test>; then
     <cmd>
else
     <cmd>
fi

If <test>; then
     <cmd>
elif <test>; then
     <cmd>
else
     <cmd>
fi
```

Numeric tests

If you are testing numbers, do so inside of (( )):

```
if (( a > 1 )); then
```

If you are testing strings, do so inside of [[ ]]:

```
if [[ a == ‘abc’ ]]; then
```

Normal logical tests:

==	equal
<	less than
<=	less then or equal
>	greater than
>=	greater than or equal
!=	not equal
-z	string is null
-n	string is not null

File tests

-e	file exists
-f	file is regular (not dir or device)
-s	file is not zero size
-d	file is dir
-r	file has read permission
-w	file has write permission
-x	file has execute permission
-N	file modified since last read
f1 -nt f2	file f1 is newer than f2
f1 -ot fw	file f1 is older than f2

true is a program that returns a 0 exit code.
false is a program that returns a non-zero exit code.

for example, to test if file abc.txt exists:

```
if [[ -e abc.txt ]]; then
```

to reverse the test (to see if the file doesn’t exist):

```
if [[ ! -e abc.txt]]; then
```

for command

The for command is the basic looping construct. Unfortunately it is nothing like the for command in any real language. It essentially executes a block of code for every item in a list:

```
for arg in [list]; do
```

```
          statement
          done
```

To process a list of files:

```
for filename in *.c; do
```

To process each line in a file:

```
for filename in $(cat filename); do
```

This is the closest thing to a normal for statement:

```
for x in 1 2 3 4 5 6; do
```

while command

As long as a condition is true, loop thru the statements:

```
while <condition>; do
      statement
      done
```

until command

Nearly like the while command. The while command tests the condition first. The until command tests the condition AFTER executing the statements. Unfortunately, this is completely unclear from the syntax:

```
until <condition>; do
      statement
      done
```

case command

You couldn’t possibly make an uglier command than this one:

```
case ${var} in
      value      )
      statement
      ;;
      value )
      statement
      ;;
      * )
      statement
```

```
          ;;
          esac
```

such as

```
     case ${name} in
          ‘hallock’ ) print ‘dude’;;
          ‘smith’   ) print ‘wow’;;
          * ) print ‘yikes’;;
          esac
```

break command

stop the loop

continue command

next iteration of loop

print command

More than likely you are going to be reporting things in your scripts. Displaying constants and variables is done with print (or echo).

```
     print ‘hello ‘${name}
     print “hello ${name}”
```

The normal print command prints a CR/LF at the end of each line. If you don’t want this, use the -n option:

```
     print -n ‘Enter your name:’
```

printf command

If you need fancier formatting, look at the printf command. This works nearly like the printf function in C.

read command

To get data from the user (or perhaps a file) into your script, you use the read command:

```
     read <varname>
```

So to get input from a user:

```
     print -n ‘What is your name: ‘
     read name
```

```
print 'Hi' ${name} '!'
```

There are a lot of options for read, use the man page.

Using a Code Block and read to process a file

If you want to create a loop to read from a file, here is a simple way

```
{                        #Start Code block
  while read rec; do
    print ${rec}
    done
  } < myinputfile        #redirect file for read to process
```

Functions

Happily, you can create functions in ksh although parameter passing is weak.

Definition

To create a function:

```
function funcname {
      commands
      }
```

Parameters

You cannot declare anything about the parameters passed. They are simply passed positionally and you refer to them as ${1} for the first parameter thru ${n} for the nth parameter. Note, you can use $1 .. $9. To go beyond $9 REQUIRES the ${} format.

So this function adds 2 numbers:

```
function add {
      print $(( ${1} + ${2} ))
      }
```

Some special parameters:

$#	number of parms
$*	All parms as a single word
$@	All parms, each quoted individually
$-	flags passed to script

Return Value

Functions can use the exit code to return information. The problem with this is any exit code besides 0 is an error and if you are trapping errors (as eibatchmode does), the exit code is going to get you into trouble. Also, you can only return 0-255.

If you want to do this use exit such as

```
function add {
     exit $(( ${1} + ${2} ))
     }
```

Another way to do this is to use a global variable; however, this is considered poor programming practice:

```
function add {
     addresult=$(( ${1} + ${2} ))
     }
```

About the best way to do this is to return the value in stdout (using print) and then use $() to assign the output:

```
function add {
     print $(( ${1} + ${2} ))
     }

result=$(add 1 2)
```

This works OK but don't forget that you can't be printing other things to stdout. If you want to print some kind of status, print it to stderr:

```
function add {
     print ${1} + ${2} >&2
     print $(( ${1} + ${2} ))
     }
```

Local Variables

You can create local variable in a function using local:

```
function add {
     local result
     result=$(( ${1} + ${2} ))
     print ${result}
     }
```

Parallel subprocesses

Occassionally I need to do something that is not cpu intensive (usually it is network bound) and so it makes sense to execute many iterations of a command at the same time.

A simple way to do this is just create some subprocesses using ():

```
(commandlist)&
(commandlist)&
```

Typically I'm reading IP addresses out of a file and then doing something to each IP address and I can do them all at once. I do this with a while loop:

```
for x in $(cat filename); do
     somecommand ${x} &
     done
```

This can fire off too many processes so usually I have some kind of test to see how many jobs are running and I limit it to 10-20:

```
for x in $(cat filename); do
     while (( $(jobs | wc -l | bc) > 10; do
          sleep 10
          done
     somecommand ${x}
     done
```

Using a Pipe File

When you need to do something where you can't simply use the pipe command ('|'), you can create a pipe file.

For example, I have a situation where I want to run mysql, create a lock, and then just let it sit until I finish doing another task.

To do this, first I create a pipe file:

```
mknod testpipe p
```

Now I start mysql in the background so that it reads from this pipe:

```
mysql -u root -pxxxxx netflow <testpipe &
```

This next part is optional and is used to force the pipe to stay open. At this point, mysql is reading an empty pipe. If you send output to this pipe and close it, mysql will exit because an EOF is put into the pipe. If you want the pipe to stay open across multiple commands, you must have something else open the pipe for output and keep it open:

```
sleep 60m >testpipe &
```

Now I just send commands to the pipe:

```
print ‘show tables;’ >testpipe
print ‘show processlist;’ >testpipe
```

mySQL will stay up reading commands until the 60m timer on sleep expires or you send it an exit command:

```
print ‘exit’ >testpipe
```

Note that the sleep command will still be there and the pipe still exists until you exit.

To get around this, you must keep the PID of sleep and then kill it. When you create the sleep process, save the PID like this:

```
sleep 60m >testpipe &
sleepPid=${!}
```

Now when you are ready to close and delete the pipe (after sending an exit command to the process reading from the pipe):

```
kill ${sleepPid}
rm testpipe
```

Using MySQL

mySQL isn't a Unix-only thing. In fact it really has nothing to do with Unix. However, netflow and its database are implemented on a Linux host, so I will throw some notes for MySQL in here.

The best place for exact mySQL syntax is the mysql.com website. There is a link to the manual in our internal home page.

Typically mysql is run from the linux/unix server; however, there is a mysql client for dos and various 3rd party clients for Windows. My favorite is an old version of mysql-front because of the way it exports data.

Contents

Running mySQL

The general form for running mySQL is:

```
mysql -u <username>
```

For netflow, the user for reading data is ‘readonly’. If you need to modify data, then you must use the root user.

Other useful switches:

-p	Supply the password on the command line
-h	specify a host if you are trying to access mySQL on a different system.
-e	specify a command to run, then exit
-B	run in batch mode (no formatting of output)
-N	don’t write column names in the output (useful in batch mode)
-H	output in HTML mode

Getting Information about Databases and Tables

To retrieve information from mySQL, you will need to know the names of databases, tables, and fields within tables.

To get a list of all the databases which you can access, type

```
show databases;
```

To start using a database, type:

```
use <databaseName>;
```

Now to see the tables in that database, type:

```
show tables;
```

Now that you know the tables, you will probably want to know what fields are in each table. There are two ways to do this. If the table you are interested in is called netflows, then the easy way is just to type:

```
select * from netflows limit 1;
```

This will display a single record from the table so you can see all of the fields. If the table is really big, the following formats each field on a separate line:

```
select * from netflows limit 1 \G;
```

If you want to get detailed information about the fields in the table, use this command instead:

```
show create table netflows \G;
```

Using the Select Command

The select command is used to retrieve records from a table (or multiple tables and databases if necessary).

The general form of select is:

select <fieldlist> | *
from <table>
[where <qualifier>]
[order by <fieldlist>]
[limit <n>]

The *fieldlist* is a list of what will be shown in the report. For example

```
select primaryIP, iosversion
```

will show you those two fields in the report whereas

```
select *
```

will show you all fields and you can use that to see what all of the fields are and then rewrite the query to select only the fields you care about.

from <*table*> tells the system which table to get the data from.

where <*qualifier*> limits the data that will be reported. This is some kind of conditional statement listing one or more fields. More than likely there are 2 that you will use the most:

```
<field>="value"
```

such as primaryIP = "10.1.1.1"

or

```
<field> like "value%"
```

which means the field starts with 'value' such as primaryIp like "192.1.60.%"

You can use AND OR and NOT in mysql like you can in most languages:

where (field1 = ‘a’ or field1 = ‘b’) and field2 = ‘1'

Order by <fieldlist> tells how to sort the data before reporting it. If you don't use Order By, then no telling what the order will be - probably random. In most cases you will use a single field or perhaps a field list:

```
order by PrimaryIp
order by model,iosversion
```

To reverse sort use 'desc' such as

```
order by model desc, iosversion
```

When trying to sort by an IP address, you don't really want to sort by the alpha format of an IP address. It is best to convert it to a 32bit integer before sorting which is the purpose of inet_aton:

```
order by inet_aton(primaryip)
```

limit is used to limit the # of records returned. This is VERY handy when you are first setting up a query and you don't know how many records will be returned. If you say 'limit 1' then only 1 record is returned. When writing a new query I usually am using 'limit 100' to prevent a mistake from taking forever to complete.

IPTABLES

Chains:
    INPUT: packets to router
    FORWARD: packets going THRU router (**this is probably what you want**)

To list everything

```
iptables -L
```

To list just FORWARD

```
iptables -L FORWARD
```

To list verbose AND get line numbers:

```
iptables -L FORWARD -v --line-numbers
```

To delete rule #3 from the FORWARD chain:

```
iptables -D FORWARD 3
```

To append a rule to the end of the FORWARD chain:

```
iptables -A FORWARD -i eth1 --protocol tcp --sport 443 -j DROP
```

To insert a rule at the beginning of the FORWARD chain:

```
iptables -I FORWARD -i eth1 --protocol tcp --sport 443 -j DROP
```

dump / restore

For the moment, this just covers how I'm using dump / restore to create a full system image of my local linux systems.

full dump

For most systems, I simply do a full dump once a month using /danz/stdbackup. This script executes dump with:

```
dump -0 -h0 -f /backup/dumpfile.dmp /etc /home ... #list of directories to backup
```

-0 indicates this is a full backup.

-h0 indicates the level at which the nodump flag (assigned to files/directories with lsattr) is ignored. -h0 indicates never to ignore it.

-f indicates the output file, /backup/dumpfile.dmp in this example.

For bigger systems that need a partial backup, my backup command is more like:

```
dump -h0 -0uj -f /backup/dmp/full-backup.dmp /
```

Here, the backup is much like above except

-u indicates the /etc/dumpdates file will be updated, necessary when doing partials.

-j indicates to compress the output data

Note that in this example, I only specify backup up "/" - the entire system.

partial dump

On large systems, the full dump occurs once a month. Partials are then done daily or weekly. The partial backup is executed with:

```
dump -$(date +%d) -uj -f ${fname}.dmp /
```

-$(date +%d) - This evaluates to the current day such as -8 or -21. For this partial backup, I maintain backups of only the files that have changed on each day thus using -<day> as the level of the backup.

-u indicates the /etc/dumpdates file will be updated.

-j indicates to compress the output data

-f output file name. In this case I want the dump to be placed in a file called partial-backup-nn.dump where nn is the day of the month. This is computed with:

fname=/backup/dmp/partial-backup-$(date +%d)

Displaying list of backups

You can see the list of backups as follows:

```
cat /var/lib/dumpdates
/dev/sda1 0 Thu Aug  1 02:30:02 2019 -0700
/dev/sda1 1 Thu Aug  1 01:30:01 2019 -0700
/dev/sda1 12 Mon Aug 12 01:30:01 2019 -0700
/dev/sda1 13 Tue Aug 13 01:30:01 2019 -0700
/dev/sda1 14 Wed Aug 14 01:30:01 2019 -0700
/dev/sda1 15 Thu Aug 15 01:30:01 2019 -0700
...
```

Creating a list of files backed up

In my backup scripts, every backup is followed by running restore against the .dmp file to create a list of files backed up:

```
restore -tf full-backup.dmp > full-backup.lst
```

Restoring Individual Files

Use restore in interactive mode. We will run it INSIDE the /backup directory so restored files are restored there. Then we can copy/move restored files into position.

restore -if full-backup.dmp

To verify you have the correct tape, use what:

```
restore > what
Dump   date: Thu Aug  1 02:00:02 2019
Dumped from: the epoch
Level 0 dump of / (dir etc) on nsdude:/dev/root
Label: none
```

use ls & cd to get to the proper location:

```
restore > ls
.:
danz/ etc/  home/ root/

restore > cd danz
restore > ls
./danz:
danzprofile       fossil-arm        sensorMonitor     teensy_loader_cli
```

```
danzreboot        fossil-x86        sjlogmon          tping
distall           gsmGpsUdpDemo     sjlogmonSed       tw
dog               mntbigdude        smartTest.sh      webreport
f                 monitorWH1080     stdbackup         xscp
fossil            popfind           t
```

I want to restore the file stdbackup, so I add it:

```
restore > add stdbackup
```

Once all files to be restored are added, extract them:

```
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
restore: <name unknown>: ftruncate: Invalid argument
set owner/mode for '.'? [yn] n
restore > quit
```

and the file is found in /backup/danz:

```
/backup:ls ./danz/*
./danz/stdbackup*
```

VirtualBox Management

I use Sun's (now Oracle) VirtualBox for all of my virtualization needs. Here are some notes.

New VMs

When creating a new VM, use vbox's own VDI files for the drive volumes. Since these are the native files, it's utilities will work directly on them, rather than having to do a time consuming conversion from VDMK to VDI, do the operation, then convert back to VDMK.

Compacting a VDI/VDMK File

This is a summary from
https://vladimir-ivanov.net/how-to-compact-virtualboxs-vdmk-file-size/

- Inside the running VM, fill free space with zeros

```
dd if=/dev/zero of=bigfile  bs=1M count=<n>G  status=progress
sync
rm bigfile
```

- Shut down the VM.

- Using vboxmanage, compress the VDI:

```
VBoxManage modifyhd <filename>.vdi --compact
```

Setting Up Arduino Dev System

I have always used Windows for software development, but really everything I do in the development realm can be done in Linux now so I converted my dev system to linux when I upgraded hardware.

Of course, there are still some problems trying to get everything I use to run properly under Linux. This section looks at the problems encountered and how they were resolved.

Code::Blocks

I use c::b to write PC code (extremely rarely) but mostly to write MCU code - arduino or Teensy. I use c::b to edit code ONLY. I run the Arduino IDE to do the actual compilation.

I install code blocks using the package manager.

Installed, c::b doesn't know about INO files. I didn't keep my notes on how to fix this but I know that I modified the Arduino Wizard. I used https://github.com/provideyourown/CodeBlocks-Arduino to get c::b to the point of creating a new INO file. I did have to tweak /usr/share/codeblocks/templates/wizard/arduino/wizard.script. The best thing to do is get that directory of my dev system for any future system.

I copied all settings from my old c::b install on windows to the new install in linux. That fixes many things, but not all.

ctl-tab works different on linux and is basically useless. Use ctl-f6 or ctl-, instead.

Indentation was not working properly (as I like it). Make sure the settings | editor | indent options are set properly (I used old config as template).

I want **ctl-Y to delete the current line** (as it does on all my other editors). To do this you need to install the keyboard shortcuts plug in (install the plugins package for c::b). Once installed, you can go into settings | editors | keyboard shortcuts. Follow the tree edit | special commands | line | delete and add ctl-Y.

The biggest problem I have had moving from Windows c::b to Linux c::b is **code completion**. Basically, I cannot get it to recognize all of the arduino externals by using compiler search directories as it is supposed to. Instead, I've had to implement a kind of ugly solution.

My tweaked wizard.script has been modified to deal with the missing externals. This script will add any files found in /usr/share/codeblocks/templates/wizard/arduino/files/. I removed all of the *default* files and instead of trying to guess at a ton of files to copy in, I created symbolic links to various directories containing all the files that I need.

Here is the content's of that directory:

```
labdudex/file:ls -l
total 4.0K
lrwxrwxrwx 1 bigdan bigdan 72 Jul 28 21:15 arduinocores ->
/home/bigdan/.arduino15/packages/arduino/hardware/avr/1.8.6/cores/arduino//
lrwxrwxrwx 1 bigdan bigdan 31 Jul 28 21:13 arduinolib -> /home/bigdan/.arduino15/libraries/
lrwxrwxrwx 1 bigdan bigdan 19 Jul 28 21:21 locallibs -> /home/bigdan/Arduino//
lrwxrwxrwx 1 bigdan bigdan 58 Jul 28 21:16 teensy ->
/home/bigdan/.arduino15/packages/teensy/hardware/avr/1.59.0//
```

Essentially, I link

/home/bigdan/.arduino15/packages/arduino/hardware/avr/1.8.6/cores/arduino/
/home/bigdan/.arduino15/libraries
/home/bigdan/.arduino15/packages/teensy/hardware/avr/1.59.0/

into the files directory.

In working with the c::b arduino problem, I was constantly needing to open files in the hidden directory .arduino15, except open file dialog box wouldn't show it. To **display hidden files in the open file box**, type ctl-H.

Rather than try to get RWIN running under Linux, I am using putty to connect to MCU's via a serial port (ttyUSB0 or ttyACM0). For the most part this is working fine, except once I get putty running, I can't figure out how to get into settings to tweak them. To **open the settings window in running putty**, do a ctl-right-click with mouse in the window.