

VeroTek UNIX and OpenBSD Quick Guide

Note: This manual is an on-going project here at VeroTek. It is intended for our internal use, by capturing the common commands and configs that we use, but not often enough to remember. Because of the time span involved, it will have legacy commands from our older flavors of UNIX, so please use with a grain of salt ☺. We have posted it only to help others trying to admin UNIX systems. Please feel free to use it as the start of your own mini-manual.

- Glenn gkennedy@verotek.com

G. Kennedy
9/20/2007
Rev: 4/19/2008

1. Install Notes:

NetraX1 Hardware Platform

SUN Netra X1 500MHz 128M RAM 40GB Disk drive

Pulled cover and installed an IDE CDROM drive

Used Disk3 (sparc) from the OpenBSD CD's

Powered up the unit, at prompt I did a [boot cdrom] and it took off

sd = scsi wd = IDE

Found wd0,

I "z'd" out the old partition table

Created a new table

		Netra	1850R(9G)
a	root	5G	1G
b	swap	1G	1G
c	keeps for full disk		
d	/usr	10G	1G
e	/var	10G	4G
f	/tmp	5G	1G
g	/home	5G	1G

p command prints the table, **q** command saves and exits

Used dc0, the first Ethernet port

Name = ux0001 A.B.C.D/24 DNS 206.13.30.12

I installed the default software sets, minus the games

[date YYYYMMDDHHMM] to set the hardware clock

halt the system, remove the CD and reboot, and its good to go.

2. Shells - are the command processing environment. Similar to command.com in DOS but much more powerful and flexible. They come in different flavors with names like:

Bourne Shell - *sh* - Primary shell developed by Stephen Bourne in the late 1970's. Prompt is \$ **[Note: *sh* is what we use with OpenBSD]**

C-shell - *csh* - Developed by Bill Joy at UC Berkeley. Prompt is %
History mechanism
Job Control
Aliasing

Korn - *ksh* - Developed by David Korn at AT&T in 1983. Extension of the Bourne shell with the features of the C shell.

Restricted shells, - *rsh* - special versions of the shells that establish client environments with limited capability

Using Secure Shell *ssh* to remotely login

If you need to install *ssh*, it is located on j:\data\vendors\openssh. Copy it to a folder and unzip the program and run the installer.

Go to your D: drive (or wherever) and the openssh folder

D:\openssh then down to the \bin folder

D:\openssh\bin ssh A.B.C.20 -l root

Will ssh log you into A.B.C.20 as root, if you don't use the -l root, it will try to log you in as yourself

Note: A file called "known_hosts" is created in a hidden folder on C: called .ssh that has the security hashes for all the devices you've connected to. If the hardware changes, you'll need to open the file in notepad and delete the section for a specific server. Then restart and it will re-establish the hash file.

Shell Scripts - The common UNIX shells support the concept of scripts and script programming. Scripts are text files that contain valid UNIX commands and can be run by the shell in an analogous fashion to DOS batch files. These script environments also support common programming techniques like: if-then-else, do case, variables, etc.

[Note: To run a shell script, you have to pre-pend ./ to the script name. For example, to run the rotate_syslog script, you cd to the /etc directory and:

./rotate_syslog

GUI's

"X Windows" is a typical standard for supporting a UNIX GUI

"Motif" is a typical GUI product from OSF
"Open Look" is a typical product from AT&T

Shell Controls

<Ctrl-S> Stop <Ctrl-C> Break or Interrupt
<Ctrl-Q> Continue <Ctrl-D> End of File (of)

Login

The shell displays

login:

You type your name and [enter]

The shell asks for your password

Password:

You type your password and [enter]

Logout

To logout from the Bourne shell

<Ctrl-D>

To logout from the C shell

logout

Changing Passwords

passwd is used to change a user password. It can also be used by the administrator to clear a user password

Locking the terminal

If there is a reason you need to leave the terminal but can not logout, you can lock the terminal with your password. This will prevent an unauthorized user from accessing the system.

lock will prompt for a password.

Enter your password to unlock the terminal.

System time and calendar

date will display the system date and time to the terminal

cal will display a current calendar to the display

3. Exploring and Communicating

news

Command to display all of the files or all that are new in the */usr/news* directory. Usually included in the user startup environment.

uname -a

Displays information about your system. Helps when you may be logged into several systems.

logname or who am i

Displays the userid or "name" that is logged in.

id

Displays your userid and groupid

tty

Displays the name of your terminal.

who

Displays names of userids currently logged in.

clear

Clears your screen like *cls* in DOS

write userid

Nice, fun way to talk to other users that are logged in. First user issues the *write* command. The terminal of the addressee will beep. They then issue the *write* command and then have dialog. Use *-o-* for over and *-oo-* for over and out. When done, they both <Ctrl-D>.

wall

This is the write to all users command. Available only to a superuser, it allows them to broadcast a message to the entire system.

mesg (-ny)

This will block or enable these messages from other userids.

UNIX mail system

The built-in mail program allows UNIX users to communicate within the local system or around the world. To check your mail, type:

mail

If you have any waiting messages, the program will display their headers. You read them by typing the message number and [enter].

If you want to start mail in order to send a message, type:

mail username

The mail system will ask you for a subject and then put you in insert mode to create your message. When you terminate with a <Ctrl-D>, the message will be sent.

Within the mail program, at the _ prompt, type a ? to get help on the various mail commands. Some of the more common commands are:

r	send a reply to the message back to the sender
f <i>username</i>	forwards the current message to <i>username</i>
h	prints out the list of messages
a	alias <i>group user1 user2 etc.</i>
d	delete the current message
q	quit mail

4. Files

A file to UNIX is any source from which data can be read or any target to which it can write information. This is a much broader concept than we may be used to from either DOS environments or IBM mainframe worlds.

File types

Ordinary - These files store data and are what most of us think of as a file. A typical example is an ASCII text file, but they could be either binary (programs, graphics, etc) or text (ASCII)

Directories - These contain the information used to access other types of files. We use directories to organize our file into groups. Directories can hold ordinary files or other directories.

Special - These files represent physical devices like keyboards, displays, printers or disk drives. When you send I/O to devices in the UNIX world you do it by writing to a file.

File Names

Maximum of 14 characters

They are case sensitive

May use any characters except /

To avoid confusion, suggest you avoid the following

[space] [tab] [backspace] ? @ # \$ ^ & () ` ' [] ; = < >

Inodes

Although we think of directories as containing the files underneath them, in actual fact they contain the pointer to the physical location of these files. UNIX keeps all file information in a special system table. This table has an index node or *inode* entry for every file in the system. The inode has information regarding: file type, size, location, owner, users, times accessed, etc. and number of links.

Directories are actually a series of 16 byte entries that contain 14 characters of a file name and two bytes of an *inode* number. This inode number is used as an index to enter the inode table and get the file information.

The key issue to understand with UNIX is that the inode is the unique aspect, not the file name. You will find later on that files can be given several different names with different attributes which all point to the same inode number or actual location.

When file names are "erased", the inode links are removed until only one remains. When the last link is erased, the file is physically erased.
Note: Unlike DOS, it is actually erased and is not recoverable.

Listing Files

ls command used to list files like *dir* to DOS users. It lists them in alphabetical order.

ls -C equivalent to *dir/w*

ls -F shows functions

ls -a shows all files including hidden (start with a .)

ls -l the long version, gives more information.

- file type and permissions
- number of links or subdirs
- userid
- groupid
- size
- date and time modified
- file name

File type codes

d	directory
-	ordinary
b or c	block or character device

Permissions are three groups of read, write , execute for the owner, group and all others.

Example

Type	Owner	Group	Other
-	rwX	rwX	r--

Creating short files

touch filename

touch is used to directly control the access and modification times of the file. When used with only a file name, it creates an empty file with today's data and time.

cat > filename

This stands for catenate and among many uses, will create a short file by taking text directly from the keyboard. Finish with a <Ctrl-D> to close the file. Analogous to use of *copy con:* in DOS

Displaying file contents*more filename*

will display the file contents to the screen, pausing at each full screen.

File permissions and information

With UNIX, every file has an owner that controls the file. The system files are owned by one of four "system" ids.

root - superuser that owns most system files

bin - owns many binary files

uucp - owns the communications files

lp - owns the printing files

UNIX has three file permissions:

read

write

execute

Note: The superuser has all permissions, and directories must have the execute right for full access.

File modes

Mode	Read	Write	Execute
0			
1			1
2		2	
3		2	1
4	4		
5	4		1
6	4	2	
7	4	2	1

Changing Modes*chmod nnn file*

is the command to change a file mode, where nnn represents the mode for owner, group and others respectively.

File Ownership

Changing the file's owner.

Use the *chown userid file* command. Be careful, once you do, you no longer have rights to change it back.

Changing the group id

Use the *chgrp groupid file* command.

Maintaining files

Wildcards

UNIX supports use of wildcards like:

* ?

Be careful of *, it is more powerful than the DOS version. By itself, it will tag all files except those starting with a period.

rm

This command is used to remove the link between the file name and the inode. If this is the last link, the file is "erased" or "removed".

mv

This command is used to move or rename a file.

copy and cp

These commands are used to copy files. The process creates a duplicate with a new inode number.

ln existingname newname

The link command establishes a subsequent filename and rights pointing to the same inode

find path -name filename -print

Will search all of the specified directories for the filename

5. UNIX Tools

Redirecting inputs and outputs

A program in UNIX expects to get its input from the "standard input device" which is usually the keyboard and send its output to the "standard output device" which is usually the display. UNIX allows you to "redirect" any of these operations with the < for input and the > for output. For example, the `ls` command normally sends a directory listing to the display but you could redirect its output into a file by:

```
ls > filename
```

Concept of pipes

This concept of standard inputs and outputs can be extended to the operation of pipes. In a pipe, the output of one program is "piped" or linked to the input of another program. This can be done at the command line with the | symbol or done by application programs with special calls. One of the basic UNIX philosophy's is for programs to do a single thing well and then "pipe" or link several of them together to perform more complex tasks.

***null* device**

A special device `/dev/null` exists that can take the output from any program and basically make it "disappear", while allowing the programs to function normally. This is often used to make extraneous messages disappear, to avoid confusing end users or to test new programs under development.

The *env* or "environment"

The environment space in a computer is similar to the purpose of a refrigerator door in a household. Everyone knows to stick notes to family members on the door and to look there for any notes to them. In a similar manner, the operating system or programs can place strings of text in the format `name=value` in the environment space. Then other programs can examine this space for these values.

To see what's currently in your environment space, use the *env* command.

```
env
```

Values are commonly placed here at login time when the users `.profile` is read. They can also be added or changed with:

```
env editor=wp
```

Changing shells

Just enter the new name to spawn a sub shell

```
csh
```

Searching with *grep* and *egrep*

"Global regular expression print" used to search a file or files for a text string.

grep [expression] filename

Sorting with *sort*, supports offsets

sort filename

sort +1 filename

Reminder Service

UNIX has a built in reminder service. If you issue the *calendar* command, it will print out any lines of the file named [calendar] in the current directory that contain today or tomorrow's date.

Background Processing

UNIX will run any of your programs in the background and allow you to continue with use of the terminal. To submit a job for background processing, append an **&** to the end of the command line. UNIX will return with a process number.

6. Directories

Directory structure

Try to have each subdirectory hold other subs or files, not both.

Try not to have more than 3 layers.

/ - or root

/bin - binary, executable UNIX programs and script files.

/dev - device files

/etc. - et cetera – All the config files and system admin files

/lib - library

/lost+found

/tmp - cleared every time UNIX restarts.

/var

 /log – for syslog type files

 /cron – for cron files

/usr - user

 /each user

 /bin

 /mail (holds the users mail)

 /spool (printing spool files)

Changing directories

cd

Displaying the current directory (Sorry, DOS users, no prompt)

pwd

Making directories

mkdir

Removing directories

rmdir (they have to be empty and it can not be your current directory)

Finding a file in the structure

find / -name filename -print

7. Text Editing - vi

Starting vi and screen overview

vi filename

Bottom line is command line for ex type commands

The tilde ~ marks unused area

Modes

Insert mode

Several commands drop you into insert mode

Command mode

<esc> takes you back to command mode

If you forget where you are, hit <esc> 'till it beeps.

ex command mode

These are command entered at the 25th line.

Cursor movement

h - left l - right

k - up j - down

Entering commands

ZZ Save and Stop

i insert before cursor

I insert at beginning of current line

a append after cursor

A Append at end of current line.

o open new line below the current.

O open new line above current.

x delete a character

X delete a character to the left of the cursor

dd delete the current line

8. Basic User Administration

Login as *root*

root is called a superuser and has all privileges throughout the UNIX system. When you are logged in as *root*, you will have the special prompt `#`. Many systems restrict the *root* login to the console to support security. Make sure *root* has a secure password.

Users

Added with *adduser*. They are removed with *rmuser* or by editing the */etc/passwd* table. Be sure to have a plan for disposing of the user files and deleting the */usr* directory.

Be sure users understand the importance of passwords.

Groups

Added by editing the */etc/group* file. Create groups that make sense. Users can be in more than one group but only one groupid will be active in their */etc/passwd* entry at one time.

Logfiles

/var/log/messages is a file of error messages received

/etc/ddate is a file of backup dates

Each system may have additional logfiles. Be careful they do not grow out of bounds, particularly accounting logs.

Login Status

The *last* command reads the *wtmp* file and displays a dump of the recent login, logout and reboot activity

9. Basic System Administration

Processes

Processes are the series of programs that are concurrently running in your system. There will be system processes that run constantly like *init* and *swapper* and there will be processes generated as users perform activity at their terminals.

```
ps shows your current processes
ps -a          all
ps -ajkwux    for OpenBSD
```

Killing processes, *kill*

Processes can be stopped or killed by the administrator. This is usually the result of an error and is only necessary to clear a "hung" process. Be very careful when you do this, because if you kill the wrong user process you might lose their data or if you kill a system process the entire system will crash.

```
kill 1234
kill -9 1234 (the "9" represents a signal that kills everything)
kill `cat /var/run/syslog.pid` in OpenBSD for programs [Note be careful of which
` you use]
```

Daemons

This is a term for processes that run in the background, usually at all times. Examples include *lpsched*, *cron*, *syslog*, *swapper*, *pagedaemon*, *update*, etc.

Deferred Processing

UNIX has full capability to handle deferred processing of programs. This can be handled by the *crontab* function, or simple requests can use the *at* function.

```
at 10:30 < runlist
```

where *runlist* is a file that contains the commands that you want executed at the later date and/or time. The *at* command will return a job number for the job and then the *cron* daemon will run the job at the appropriate time. The results of the program will be "mailed" to you.

Additional Monitoring

```
vmstat - Monitors virtual memory
```

```
pstat - A dump of various kernel tables and system information
```

```
uptime - System utilization
```

10. Hardware - Terminals & Disk Drives

UNIX views all I/O devices as special files whose descriptions are contained under the directory */dev*

Terminals

UNIX calls terminals ttys from the old days of teletypes. The device names for these devices will be in the format of */dev/ttyxxx* (usually like */dev/tty1a* or */dev/tty1a*).

Different terminals have very different capabilities with regard to graphics and keyboards. The terminal descriptions are held in a file called */etc/termcap* or *terminfo* on newer systems. UNIX programs like *vi* or *more* and third party application programs go to this database to learn the keystroke and display characteristics of the user's terminal. The programs know which entry to use by using the **TERM=** entry in the environment space as a key for the database.

As soon as a UNIX system is booted, a daemon called *init* is spawned. One of *init*'s tasks is to spawn a process called *getty* (get tty) for every terminal that is enabled. This *getty* process establishes communication with the terminal and runs a program called *login* that allows use of the terminal. *init* knows that a terminal is enabled by examining the */etc/ttys* table.

Typical section of */etc/ttys*

```
0mtty1A
1mtty1a
0mtty1B
1mtty1b
```

If the first character of an entry is a "1" the terminal is enabled, if it's a "0" the terminal is disabled. This table can be edited manually to enable a terminal, but usually the *enable /dev/ttyxxx* command is utilized. To disable a terminal the *disable /dev/ttyxxx* command is used.

As *init* detects an enabled terminal, it uses the second character of the */etc/ttys* entry as a key to enter the */etc/gettydefs* table. From this table, it gets the bit rate and initial communication parameters for the terminal.

The last entry in the */etc/gettydefs* table is the command to be run on the terminal (usually *login*).

New administrators frequently have trouble getting newly wired terminals to work on a UNIX system. The first step in troubleshooting is to determine if the problem is with the terminal and wiring or with something in the UNIX configuration.

A quick way to check a newly wired terminal for operation is to redirect the output of a program directly to the device name. The following command will work even if the terminal is disabled.

```
date > /dev/tty1a
```

So, if this works, you know the terminal and wiring are good.

Hard Disk drives

Hard disk drives are "mounted" in UNIX and form a large file system with a homogenous directory hierarchy. This is a major difference from the way DOS views hard disks as separate drive letters. These hard disks are block oriented, 512 bytes per block.

Never fill the disk !

Use *df* to check space

df -h for "human-readable" format (not on SUN)

df -k to get sizes in kbytes

Use *ulimit* to check the limit on user disk space

system default is 4096 blocks

root can increase, user can decrease

Use *du* for disk usage

du takes a single argument, specifying a pathname for *du* to work; if it is not specified, the current directory is used.

- a display an entry for each file (and not directory) contained in the current directory
- h human format
- H calculate disk usage for link references specified on the command line
- k show sizes as multiples of 1024 bytes, not 512-byte
- L calculate disk usage for link references anywhere
- s report only the sum of the usage in the current directory, not for each file
- x only traverse files and directories on the device on which the pathname argument is specified.

Examples

Sum of directories in Kbytes:

```
# du -sk *
```

disk usage of all subdirectories and files including hidden files within the current directory (sorted by filesize) :

```
# du -sk * | sort -n
```

Floppy Disks

Floppy disks can be treated as archive devices similar to a tape or they can be mounted as part of the file system. They are referred to by their drive number 0 or 1 and their size.

For example:

5 1/4 HD	/dev/fd096ds15
5 1/4 DD	/dev/fd048ds
3 1/2 HD	/dev/fd0135ds18

Formatting - to format a floppy use the format command

```
format /dev/rfd1135ds18
```

Archive Style - If you want to just copy files to a floppy, you would treat it as an archive device and use the tar command to copy files.

```
tar -rv filename
```

To see the files on the floppy use:

```
tar -t
```

File system Style - If you want to use the floppy as part of the active file system, you first create the file system and then mount the floppy, usually at /mnt. It then becomes a continuous part of the overall file system and you can copy or move files as if it were part of the hard disk.

```
mkdev fd to make the file system
```

```
mount /dev/fd1135ds18 /mnt to mount the floppy
```

Note: You must *Umount* the floppy before you remove it from the drive. The easy days of DOS are gone !

Tape Backups

Usually a system will use a third party tape backup system that will come with it's own menu driven software. UNIX versions do have several versions of built in backup software including :

```
dump, restore, cpio, tar
```

11. Hardware - Printers

Printers and Printing

Hardware issues - Printers can be connected to either the parallel or local serial ports of the host machine. The parallel port is preferable if the distance is not a limiting factor. In addition, printers can be connected via modems and remote serial ports.

Queues - Printing in multi-user systems is accomplished by having the users print to a spool file or queue. These are both terms for a system directory or file that receives the print outputs from the users. A print server or scheduling program then prints these files to an actual printer when it is ready.

To create a spooled printer environment use:

mkdev lp

To operate the scheduler that controls print spooling

lpsched is the daemon that controls print spooling

lpshut stops the scheduler

A log is kept in */usr/spool/lp/log*

Individual users or software packages use the *lp* command to print a file

Specific print utilities

lpstat -t

enable name

disable name

cancel printjob

lpmove - moves job to a new queue

Test new printers by *cat filename > /dev/lp* and the file should print directly to the physical printer.

12. System Startup Sequence

POST - Power On Self Test - function of the H/W

Boot - The ROM boot loader reads the initial disk tracks and reads in the kernel of the operating system, and then spawns *init*.

init process

- Opens the `/dev/console`

- Performs a file check *fschk* if necessary

- Reads and executes the `/etc/rc` script or scripts

 - The *rc* stands for run commands and the scripts contain the essential startup housekeeping operations. They mount the file system and spawn the necessary daemons for system operation.

Much of the startup is controlled by the `/etc/rc.conf` file, which has a text entry for each of the major services with = "NO" for most and suggested flags for correct startup.

There is also a `/etc/rc.local` file which is the suggested location for all user defined startup task (i.e. alias type commands or other script files)

One of the last startup steps for *init* is to establish a *getty* process for every terminal that is enabled in `/etc/ttys`

getty is spawned for each enabled terminal in `/etc/ttys`. The second letter of the `ttys` entry is a label that points to an entry in the `/etc/gettydefs` table that lists the speed and details for the terminal. The last entry in `gettydefs` is the program to launch which is usually *login*.

login takes the userid and password and checks them for validity in the `/etc/passwd` file. If they are valid, *login* spawns the shell environment indicated by the `passwd` file

shell is the client command interpreter. As it starts, it reads commands and environment from the `.profile` or `.login` file in each users home directory. It will usually check for the terminal type in `/etc/ttytype`.

When the user logs out, *init* detects the fact and respawns the *getty* & *login* process.

Shutting Down UNIX

It can not be overemphasized how important an orderly shutdown is to UNIX systems. Be sure users have been notified of the impending shutdown. You will usually use the *wall* message command. Then use the *halt* command.

halt

Be sure you wait to power off the machine until you see the "Safe to Power Off" message from UNIX. This might take several minutes in a large system and administrators frequently power off too quickly, thereby damaging the file system.

13. Networking

netstat – a program with a 1,000 uses ☺ to monitor what's happening with IP connectivity

netstat -a shows open ports and existing connections

/etc/services is the file with the list of standard ports and services

fstat lists every open file, pipe or port on a system

Configuring Interfaces

ifconfig is used for most configuration tasks. The challenge is the changes don't stick after the next reboot.

lo0 is the local interface, we have seen dl0 and tl0 used as Ethernet interface names

ifconfig -a will list all the interfaces

ifconfig -A will show the aliases

/etc/netstart is the script file that controls networking config

It gets its information from a series of script files. There is a */etc/hostname.xxx* where xxx is the interface id for every NIC card in the system. For example */etc/hostname.tl0* in the 1850Rs. This file contains the assigned protocol and IP address for that interface, i.e.

```
inet 66.125.20.10 255.255.255.0 NONE
```

[Note: ☺ the interface is tl0 which is a t and an l (as in el) :not a 1 ☺]

/etc/mygate holds the address of the default upstream router

/etc/myname holds the full system name i.e. ws0001.verotek.com

/etc/hosts holds all the static references

Adding multiple IP addresses or aliases

```
ifconfig tl0 alias 66.125.20.11 netmask 255.255.255.255
```

Then add this entry to the */etc/rc.local* file to make it permanent

Routing - is under control of the *route* command

route show will show the current route table

```
route add a.b.c.d -netmask 255.255.255.0 w.x.y.z
```

```
route delete blah blah blah
```

DNS and Name Service

The DNS search path is determined by entries in the */etc/resolv.conf* file

```
lookup file bind
nameserver 206.13.30.12
nameserver 206.13.29.12
```

14. Sys Logging

System logging is handled by a daemon called *syslogd* that is installed by default on the OpenBSD machines. This daemon handles the manipulation and storing of log messages arriving from several internal sources as well as externally arriving files (if configured to do so).

The *syslogd* program is started by the */etc/rc* process, but as with other OpenBSD programs, the configuration should be limited to invoking flags or command line switches from the */etc/rc.conf* file.

In the case of our designated system logger *ux0001*, the */etc/rc.conf* entry has been modified with the addition of the *-u* flag which will instruct the daemon to accept log entries from external machines ... which is what we want in this specific case only. We want the PIX firewall and RT0001 edge router to be able to log to the *ux0001* syslog server.

The syslog process uses **udp port 514** for communication between machines.

The incoming log entries have as their first field a *facility* followed by a *.* and then the *level*. So, incoming alert messages from the user authorization process might look like:

auth.alert

External Machine Examples:

PIX – sends log entries with the Facility Code LOCAL4 which is configurable in the [Configuration] [Properties] [Logging] [Syslog Setup] area.

You must go into */var/log* and *touch* (create) an empty file called *pix* to receive the log entries.

Then you vi the *syslog.conf* file and add an entry like:

Local4.warn (make sure to use tabs in here) */var/log/pix*

And the next time the syslog daemon is restarted you'll be loggin your PIX messages to their own file.

15. FTP

Space holder till we get time to write ☺

15. Packet Filtering – *pf*

The internal packet filter is run via the *pf* command and gets its configuration from the */etc/pf.conf* file

16. Apache Web Server

OpenBSD installs with a current version of Apache Web Server, and has the advantage of a default install that uses *chroot* to run as user “www” with only read privileges.

All of the configuration, log, and content files are in */var/www* structure

/var

/www

/bin

/cgi-bin

/htdocs

/icons

/conf

httpd.conf

 Our system makes use of the “Virtual Host” capability and has 4 different VH at .11\12\13\14

/logs

/vts

/rs

/fm

/bigal

/verotek – site content

/redsails – site content

/fishmore – site content

/bigal – site content

17. Archiving with *tar*

The *tar* (for **t**ape **a**rchive) command is used to compress one or more files into an archive file, and also used to uncompress an archive.

To extract or uncompress an archive, use the following

```
tar -zxvf [somefilename]
  -z for the gz format
  -x for extract
  -v verbose
  -f for the filename
```

18. Cron Scheduling Facility

Unix has the capability to schedule jobs to run using the *cron* daemon. The daemon examines the files in the */var/cron/tabs* directory each minute and runs any tasks so indicated.

[Note: this directory is */var/spool/cron* in SUN systems]

There may be files in there from several users or systems. The principle file is called *root* and contains the cron jobs established by the root user.

Traditional (inherited from Unix) cron format consists of five fields separated by white spaces:

<Minute> <Hour> <Day_of_the_Month> <Month_of_the_Year> <Day_of_the_Week>

The following graph shows what it consists of:

```

* * * * *
| | | | |
| | | | | +-- Year           (range: 1900-3000)
| | | | | +---- Day of the Week (range: 1-7, 1=Monday 0-6 in SUN)
| | | | | +----- Month of the Year (range: 1-12)
| | | | | +----- Day of the Month (range: 1-31)
| | | | | +----- Hour           (range: 0-23)
| | | | | +----- Minute          (range: 0-59)
+-----

```

For example:

```
0 12 * * 1-5 * (0 12 * * Mon-Fri *) At midday on weekdays
```

Here's a typical root crontab file from one of our servers

```

#minute hour mday month wday command
#
# sendmail clientmqueue runner
*/30 * * * * /usr/sbin/sendmail -L sm-msp-queue -Ac -q
# rotate log files every hour, if necessary
0 * * * * /usr/bin/newsyslog
# send log file notifications, if necessary
#1-59 * * * * /usr/bin/newsyslog -m
# do daily/weekly/monthly maintenance
30 1 * * * umask 077; /bin/sh /etc/daily 2>&1 | tee /var/log/daily.out | mail -s "/bin/hostname` daily output" root
30 3 * * 6 umask 077; /bin/sh /etc/weekly 2>&1 | tee /var/log/weekly.out | mail -s "/bin/hostname` weekly output" root
30 5 1 * * umask 077; /bin/sh /etc/monthly 2>&1 | tee /var/log/monthly.out | mail -s "/bin/hostname` monthly output" root
10 0 * * * /etc/restart_snort

```

Crontab on SUN

[Note: I learned this one the hard way ☺ On older UNIX flavors, a simple edit of the crontab file would get you what you wanted. On SUN systems, it appears you must use the crontab command, which will let you edit the file plus “submit” it for operation ... if you don’t, the jobs will never run, and you will pull all your remaining hair out ☺]

Crontab Commands

`crontab -e` Edit your crontab file, or create one if it doesn't already exist.
`crontab -l` Display your crontab file.
`crontab -r` Remove your crontab file.
`crontab -v` Display the last time you edited your crontab file. (This option is only available on a few systems.)

Crontab file

Crontab syntax :-

A crontab file has five fields for specifying day , date and time followed by the command to be run at that interval.

```
* * * * * command to be executed
- - - - -
| | | | |
| | | | +----- day of week (0 - 6) (Sunday=0)
| | | +----- month (1 - 12)
| | +----- day of month (1 - 31)
| +----- hour (0 - 23)
+----- min (0 - 59)
```