pfexec bootadm set-menu timeout=5
edit /rpool/boot/grub/menu.lst and run above to commit changes

NFS aut0 home

/etc/auto_home
* thumper_backup:/export/home/&

enable autfs

change /etc/passwd /home/username

------

http://www.petertribble.co.uk/Solaris/solview.html

vmware nfs share
```
# zfs create -o sharenfs=anon=0 nfspool/nfs1
# zfs set mountpoint=/nfs1 nfspool/nfs1
```

ISCSI
On each storage node, setup iscsitgt and present a lun
```
svcadm enable iscsitgt
zfs create -V 100g testpool/zfsvolume
zfs set shareiscsi=on testpool/zfsvolume
```
On the head node
```
svcadm enable iscsi_initiator
iscsiadm modify initiator-node -A headnode
iscsiadm add discovery-address "ip address of storage node A"
iscsiadm add discovery-address "ip address of storage node B"
iscsiadm add discovery-address "ip address of storage node C"
iscsiadm modify discovery -t enable
devfsadm -c iscsi
```
Run format and see the luns

ISCSI:
```
# svcadm enable iscsitgt
# zfs create -V 100m mypool/myvol
# zfs set shareiscsi=on mypool/myvol
# iscsitadm create target -b /dev/zvol/rdsk/mypool/myvol mytarget
# iscsitadm list target -v

DISABLE:

# iscsitadm delete target -u 0 mypool/myvol
# zfs set shareiscsi=off mypool/myvol
# svcadm disable iscsitgt
```

```
shared key auto login for Sol10 root

ssh-keygen -t dsa

scp id_dsa.pub root@host:/.ssh/authorized_keys2

ssh to host

chmod 600 /.ssh/authorized_keys2
```

http://jeffhigham.blogspot.com/2008/04/cloning-solaris-10-zones-using-zfs.html

Solaris Commands

Interface errors:
netstat -iww

TURN the BEEP oFF!!!!
/usr/openwin/bin/xset b 0

Find all available network interfaces
# prtconf -D | grep network

Change hostname:
```
/etc/hosts
/etc/nodename
/etc/hostname.<network interface
```
m

Createing a RW clone of a snapshot
```
# zfs snapshot zdata/zones@yesterday
# zfs clone zdata/zones@yesterday zdata/zones/today
```
zonecfg -z newzone

zoneadm -z newzone clone currentzone

RBAC:
Tonight, I found out an easy way to allow a non-root user to bind to a privileged port (<1024) on Solaris 10. I've done this before with RBAC (i.e. exec_attr, prof_attr), but knew there was an easy one line command to provide this privilege to a non-root user. Here's the simple command. You must run it as root.
```
  # usermod -K defaultpriv=basic,net_privaddr tomcat
```
Now the user tomcat can run applications that need to bind to privileged ports (i.e. port 80). For those taking notes, this adds a line into the /etc/user_attr file:

```
  tomcat::::type=normal;defaultpriv=basic,net_privaddr
```

Tar to copy

```
tar cf - . | (cd /work/bkup/jane && tar xBf -)
```

Mount ISO:

```
lofiadm -a /path/to/sol10dvd.iso
mount -o ro -F hsfs -o ro /dev/lofi/1 /mnt/cdrom
```

Renaming a zpool

```
zpool export somepool
zpool import somepool newpool
```

Moving a zone

zoneadm -z my-zone move /newpath

```
Find WWN of HBA

HBA info:

fcinfo hba-port
fcinfo hba-port -l
fcinfo remote-port -slp 10000000c9582596
```

You can grow but not shrink a UFS filesystem in Sun Solaris with the command:

# /usr/lib/fs/ufs/mkfs -G -M /current/mount /dev/rdsk/cXtYdZsA newsize

Specifying the current mount point and raw device as well as the new size in 512 byte blocks.

You can do this even when the filesystem is mounted and in use.

To make a copy of your boot disk in Solaris:

# dd if=/dev/rdsk/c0t0d0s2 of=/dev/rdsk/c0t1d0s2 bs=4096

Don't forget to add the bootblock on the mirror disk otherwise you will not be able to boot off of it:

# installboot /usr/platform/`uname -i`/lib/fs/ufs/bootblk /dev/rdsk/c0t1d0s0 **Example**

dd if=/dev/rdsk/c0t0d0s2 of=/dev/rdsk/c0t1d0s2 bs=4096

Here is a one-liner to move data from a partition to another filesystem in Solaris:

# ufsdump 0f - /dev/rdsk/c0t0d0s0 | (cd /restore_dir; ufsrestore xv -)
Example

ufsdump 0f - /dev/rdsk/c0t0d0s0 | (cd /restore_dir; ufsrestore xv -)

To copy a directory to another name more efficiently than using cp:

# pax -rw /oldir /newdir

To get hardware information in Sun Solaris:

```
# prtconf -v
# prtdiag -v
```

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-shell exist. The default password file is /etc/passwd.

```
# pwck [filename]
```

To find the working directory of a process in Sun Solaris:

```
# /usr/proc/bin/pwdx pid
```

To capture TCP/IP (sniff) network information in Solaris:

```
# snoop [ -aPDSvVNC ] [ -d device ] [ -s snaplen ] [ -c maxcount ] [ -i filename ] [ -o filename ] [ -n filename ] [ -t [ r | a | d ] ] [ -p first [ , last ] ] [ -x offset [ , length ] ] [ expression ]
```

Example to capture all packets and display them as they are received:

```
# snoop
```

Example to view only RPC summary lines:

```
# snoop -i rpc.cap -V | grep RPC
```

Example to capture packets with host funky as either the source or destination and display them as they are received:

```
# snoop funky
```

Example to capture packets between funky and pinky and save them to a file:

```
# snoop -o cap funky pinky
```

Then inspect the packets using times (in seconds) relative to the first captured packet:

```
# snoop -i cap -t r | more
```

To look at selected packets in another capture file:

```
# snoop -i pkts -p99,108
```

To look at packet 101 in more detail:

```
# snoop -i pkts -v -p101
```

Example to view just the NFS packets between sunroof and boutique:

```
# snoop -i pkts rpc nfs and sunroof and boutique
```

The test command evaluate a condition and, if its value is true, return a zero exit status; otherwise, return a nonzero exit status. An alternate form of the command uses [ ] rather than the word test.

```
# test -s file
```

or

```
[ -s file ]
```

Example

```
test -s file; [ -s file ]
```

To copy directories or filesystems between servers securely, you can use tar and ssh together.

From the source server, issue the command:

```
# cd /src_dir; tar -cf - dir1 | ssh dest_server (cd /dest_dir; tar -xf -)
```

Description

This script loops until port 22 (ssh) responds on a remote host. Then, it opens an xterm window using ssh.

Very useful when you reboot a server and are waiting for it to come up.

```
#!/usr/bin/perl
#
# Loop until we get a connection to port 22 (ssh) on a remote server
# Then issues an xterm command to the remote host.

# unbuffer output
$| = 1;

use strict;
use IO::Socket;
use Socket;

my $socket;
my $remote_host=$ARGV[0];
my $remote_port = "22";
my @addresses;

# Simple syntax check
if ($remote_host eq "") { print "\nSyntax: $0 hostname\n\n"; exit 1; }

# Simple hostname check (for typos)
@addresses = gethostbyname($remote_host) or die "ERROR: Can't resolve
$remote_host\n";

print "Waiting for connection to $remote_host...";
```

```
while () {
 $socket = IO::Socket::INET->new(PeerAddr => $remote_host,
 PeerPort => $remote_port,
 Proto => "tcp",
 Type => SOCK_STREAM,
 Timeout => 5 );

 if ($socket) {
 print "Connected!\n";
 close($socket);
 system("xterm -geometry 100x24 -sb -sl 5000 -T $remote_host -n
$remote_host -e ssh -X $remote_host &");
 exit 0;
 } else {
 print ".";
 sleep 10;
 }
}
```

Example

wait.pl remote_host

Description

Here is a quick one-liner which performs a search and replace of a regular expression in the same file where it finds the match. It is a combination of the find command and perl:

Example to do a search and replace of first occurance of match in the file site.conf:

```
# find . -name "site.conf" -exec perl -i -p -e "s/csc2.tar.gz/csc3.tar.gz/" {} \;
```

Example to do a search and replace of all matches in the file site.conf:

```
# find . -name "site.conf" -exec perl -i -p -e "s/csc2.tar.gz/csc3.tar.gz/g" {} \;
```

Example to do a search and replace for any files which end with a .pl extension:

```
# find . -name "*.pl" -exec perl -i -p -e "s/old_string/new_string/g" {} \;
```

Example

```
find . -name "*.pl" -exec perl -i -p -e "s/old_string/new_string/g" {} \;
```

Solaris 10 IP NAT using ipfilter

UTSA - mazatlan nge0 (network) - nge1 (internal)

```
map nge0 0/0 -> 0/32 proxy port 21 ftp/tcp
map nge0 192.168.2.0/24 -> 0/32 proxy port ftp ftp/tcp
map nge0 192.168.2.0/24 -> 0/32 portmap tcp/udp auto
map nge0 192.168.2.0/24 -> 0/32
```

Introduction

So, you've got several computers on your home or business network, and you'd like to be able to access the Internet from all of them, probably via a cable (or DSL) modem. Basically you have three options:

1. You connect all your machines and your cable modem to a hub, set them all up as DHCP clients (see this page for how to do this on Solaris), and go for it.
2. You set up one of your machines to do NAT (Network Address Translation), hiding the rest behind a firewall using RFC 1918 compliant addresses on your network.
3. You use one of those Netgear routers, or someting similar (e.g., those from Linksys), as your firewall, and let it perform NAT for you.

The last option is very popular, and is better than nothing, but you can't beat having your own dedicated firewall machine. The first method, as well as being insecure, lacks a certain *je ne sais quoi*, so I'll show you how to set up NAT using Darren Reed's IP Filter. If you want to use the first or last methods, you're on your own!

Hardware

In my experiments, I could only get NAT to work reliably when I had two physical interfaces (i.e., using two virtual interfaces, say hme0 and hme0:1, didn't work). I used hme1 to connect directly to my cable modem, and hme0 as the connection to the rest of my network via a 100 baseT switch. hme1 is under DHCP control per these instructions, and hme0 was set up the conventional way, with the hostname in /etc/hostname.hme0, and the corresponding IP address in /etc/hosts.

Installing IP Filter

By far the best way to get IP Filter is install Solaris 10, which comes with Solaris IP Filter (which is based on IP Filter). For previous versions of Solaris, the best way to get IP Filter is to compile a copy of the latest source code, which can be downloaded from the IP Filter home page. As an alternative, I have a compiled version of the package here. This is IP Filter version 3.3.11, compiled on a Sun SPARCstation 20, running Solaris 2.6. I've also used it on a SPARCstation 2 running Solaris 7, but it is provided here without any support (I currently use the Solaris 10 version of IP Filter on a Sun Netra T1 105). You should probably download a more recent binary from Marauding Pirates.

Configuring IP Filter on Solaris 10

Once you've successfully installed IP Filter, you need to configure it. First of all, you need to make sure that your NAT box will forward IP packets (it's possible this ability was disabled for security reasons). As root, run this command:

```
routeadm
```

If the "Current Configuration" column of the "IPv4 forwarding" row says "disabled", then you must enable it. You do this by running the following command (again, as root):

```
routeadm -u -e ipv4-forwarding
```

The -e ipv4-forwarding option causes IPv4 forwarding to be enabled, and the -u flag causes the change to be applied to the running system (in addition to changing the settings when the system is next rebooted).

When you're happy that IP forwarding is enabled, you need to set up your NAT rules. The file /etc/ipf/ipnat.conf contains the rules you want to use. This is the ipnat.conf file I use, bearing in mind that all of my machines have an IP address in the 192.168.0.1 to 192.168.0.254 range; you should change the addresses between "hme1" and the "->" to suit your needs (note also that I've specified hme1; put the name of your outbound interface here instead):

```
map hme1 192.168.0.0/24 -> 0/32 proxy port ftp ftp/tcp
map hme1 192.168.0.0/24 -> 0/32 portmap tcp/udp auto
map hme1 192.168.0.0/24 -> 0/32
```

The 0/32 stuff is some magic to tell IP Filter to use the address currently assigned to the interface - very useful in DHCP client environments!
The order of the rules is important; don't change them unless you know what you're doing, otherwise things will break! The first rule allows FTP access from all of your hosts. The second maps the source port numbers to a high range (10000 to 40000 by default), and the third rule maps all other TCP traffic.
Once you've set up your NAT rules, you need to enable packet filtering for the interface type you're using. This is done by uncommenting the appropriate line(s) in /etc/ipf/pfil.ap:

```
#le     -1      0       pfil
#qe     -1      0       pfil
hme     -1      0       pfil
```

When you're happy with your configuration, start the IP filter services:

```
svcadm restart network/pfil
svcadm restart ipfilter
```

The interfaces that you enabled packet filtering on by editing /etc/ipf/pfil.ap must be replumbed before you can use them. Here's how to do it, assuming your machine is set up like mine:

```
ifconfig hme1 unplumb
ifconfig hme1 plumb dhcp start
```

Another, perhaps easier, way is to simply reboot your machine. Although it smells like a typical Windoze "admin" kind of way of doing this, it does have the advantage of testing that your modifications will survive a reboot.
Assuming all is well, your firewall should now correctly handle NAT, even after a reboot. Assuming this is the case, enjoy! If this page has been useful to you, please consider buying a copy of my book, Solaris Systems Programming.

Configuring IP Filter for Previous Versions of Solaris

If you're using a version of Solaris prior to Solaris 10, and assuming you have Solaris 10-capable hardware, I don't know why you **wouldn't** use Solaris 10, here is the older version of these instructions. But really, you should upgrade to Solaris 10!
First of all, you need to make sure that your NAT box will forward IP packets (it's possible this ability was disabled for security reasons). As root, run this command:

```
ndd -get /dev/tcp ip_forwarding
```

If the result is "1", you're all set. Zero means that IP forwarding is not enabled. To enable it, delete the file /etc/notrouter, and possibly /etc/defaultrouter too. Create an empty /etc/gateways file, and IP forwarding will be enabled at the next reboot.

One caveat applies, though: if you're using NAT and DHCP on the same server (like I do), IP forwarding will not get enabled. So, I install this script as /etc/init.d/ip_forwarding, with a symbolic link to it from /etc/rc2.d/S69ip_forwarding. With this script in place, IP forwarding will be enabled even if you are using a DHCP client.
When you're happy that IP Filter is running, and IP forwarding is enabled, you need to set up your NAT rules. The file /etc/opt/ipf/ipnat.conf contains the rules you want to use. This is the ipnat.conf file I use, bearing in mind that all of my machines have an IP address in the 192.168.0.1 to 192.168.0.254 range; you should change the addresses between "hme1" and the "->" to suit your needs (note also that I've specified hme1; put the name of your outbound interface here instead):

```
map hme1 192.168.0.0/24 -> 0/32 proxy port ftp ftp/tcp
map hme1 192.168.0.0/24 -> 0/32 portmap tcp/udp auto
map hme1 192.168.0.0/24 -> 0/32
```

The 0/32 stuff is some magic to tell IP Filter to use the address currently assigned to the interface - very useful in DHCP client environments!
The order of the rules is important; don't change them unless you know what you're doing, otherwise things will break! The first rule allows FTP access from all of your hosts. The second maps the source port numbers to a high range (10000 to 40000 by default), and the third rule maps all other TCP traffic.
Use /etc/init.d/ipfboot stop and /etc/init.d/ipfboot start to test your configuration, and when you're happy that all is working well, reboot. This will make sure that everything still works as expected, even after a reboot.
That's about it - enjoy! If this page has been useful to you, please consider buying a copy of my book, Solaris Systems Programming.

Reset MySQL passsword:

1. Log on to your system as either the Unix root user or as the same user that the mysqld server runs as.
2. Locate the .pid file that contains the server's process ID. The exact location and name of this file depend on your distribution, hostname, and configuration. Common locations are /var/lib/mysql/, /var/run/mysqld/, and /usr/local/mysql/data/. Generally, the filename has the extension of .pid and begins with either mysqld or your system's hostname.

   You can stop the MySQL server by sending a normal kill (not kill -9) to the mysqld process, using the pathname of the .pid file in the following command:

   ```
   shell> kill `cat /mysql-data-directory/host_name.pid`
   ```

   Note the use of backticks rather than forward quotes with the cat command; these cause the output of cat to be substituted into the kill command.
3. Create a text file and place the following command within it on a single line:

   ```
   SET PASSWORD FOR 'root'@'localhost' = PASSWORD('MyNewPassword');
   ```

   Save the file with any name. For this example the file will be ~/mysql-init.
4.

Restart the MySQL server with the special --init-file=~/mysql-init option:

```
shell> mysqld_safe --init-file=~/mysql-init &
```

The contents of the init-file are executed at server startup, changing the root password. After the server has started successfully you should delete ~/mysql-init.

5\.

You should be able to connect using the new password.

HBA multipathing:

1. To Determine multipathing on the HBA ports, edit the /kernel/drv/fp.conf file.

2. Perform one of the following:

   * To explicitly enable multipathing on this HBA port, but disable on the rest of the HBA ports, add the following:

     ```
     mpxio-disable="yes";
     name="fp" parent="/pci@6,2000/SUNW,qlc@2" port=0 mpxio-disable="no";
     ```

   * To explicitly disable multipathing on this HBA port, but enable on the rest of the HBA ports, add the following:

     ```
     mpxio-disable="no";
     name="fp" parent="/pci@6,2000/SUNW,qlc@2" port=0 mpxio-disable="yes";
     ```

3. Save and exit the file.

4. Run the stmsboot -u command.

   ```
   # stmsboot -u
   WARNING: This operation will require a reboot.
   Do you want to continue ? [y/n] (default: y) y
   The changes will come into effect after rebooting the system.
   Reboot the system now ? [y/n] (default: y) y
   ```

   During the reboot, /etc/vfstab and the dump configuration will be updated to reflect the device name changes.

5. (Optional) After the reboot, if necessary, configure your applications to use new device names as described in Enabling or Disabling Multipathing.

user directory ownership:

```
for i in user1 user2 user3 user4 user5 ; do /; chown $i:staff $i ; done
```

Zones using zonemgr from http://opensolaris.org/os/project/zonemgr/files/zonemgr-1.8.1.txt :

```
/usr/local/bin/zonemgr -a add -n iron \
-z /zones \
-t w \
-E '' \
-I "129.115.57.86|e1000g1|24|iron" \
-I "10.10.20.86|e1000g0|24|iron_backup" \
-d 129.115.102.150 -D it.utsa.edu \
-C /etc/resolv.conf -C /etc/ssh/sshd_config -C /etc/shadow -C /etc/passwd  \
-C /usr2/home -C /usr2/adm  -C /etc/mail/sendmail.cf -C /etc/group \
-C /etc/ssh/sshd_config.user -C /etc/base.xml -C /usr3/home \
-C /etc/motd -C /etc/issue \
-s "basic|lock" -S ssh \
```

```
/usr/local/bin/zonemgr -a add -n name \
-z /zones \
-E 'passwd hash' \
-I "129.115.x.x|bge1|24|name" \
-I "10.10.x.x|bge0|24|name2" \
-d 129.115.x.x -D it.utsa.edu \
-C /etc/resolv.conf -C /etc/ssh/sshd_config -C /etc/shadow -C /etc/passwd  \
-C /usr2/home -C /usr2/adm  -C /etc/mail/sendmail.cf -C /etc/group \
-C /etc/ssh/sshd_config.user -C /etc/base.xml -C /usr3/home \6
-C /etc/motd -C /etc/issue \
-s "basic|lock" -S ssh \
```

SSH tunnels from RedHat:
http://www.redhatmagazine.com/2007/11/27/advanced-ssh-configuration-and-tunneling-we-dont-need-no-stinking-vpn-software/

.bashrc:
```
export EDITOR=/usr/bin/vi
PS1user="$( test `/usr/ucb/whoami` == root && echo '\[\e[101m\]' )\u\[\e[0m\]"
PS1color='\[\e[1;37;44m\]' # color of working directory
PS1="$PS1user@\h:$PS1color\w\[\e[0m\]$PS1version> "
tty | grep pts > /dev/null && PS1="$PS1\[\e]0;\u@\h\a\]";
export PS1
PATH=/opt/csw/bin:/usr2/home/vhuckaba/bin:/usr/local/bin:/usr/sfw/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/ssl/lib:/usr/local/lib
```

Configure Mysql

Add /opt/coolstack/mysql_32bit/bin /bin to your path, and /opt/coolstack/mysql/man to your manpath.
```
#export PATH=/opt/coolstack/mysql_32bit/bin:$PATH
```

Install the db and mysql user:
```
        # /opt/coolstack/mysql/bin/mysql_install_db
        # groupadd mysql
        # useradd -c "MySQL Server" -g mysql mysql
```

Change ownership

```
# chown -R mysql:mysql /opt/coolstack/mysql_32bit
 # cp /opt/coolstack/mysql_32bit/share/mysql/my-large.cnf /etc/my.cnf
```

Edit my.cnf if necessary. Consider uncomment skip-networking to prevent network access to the database.

Start up the server:

```
$ /opt/coolstack/mysql_32bit/bin/mysqld_safe &
$ ps -ef | grep mysql | grep –v grep  <– Make sure the mysqld process is running
$ /opt/coolstack/mysql_32bit/bin/mysqladmin -u root password
'yourrootpassword'
$ /opt/coolstack/mysql/bin/mysqladmin -u root -h 'yourhostname' password
'yourrootpassword'
```

Snapshot cleanup:

```
#!/usr/bin/bash
# vhuckaba March 2008
PATH=/usr/bin:/usr/sbin
LIST=/tmp/list
echo "Enter the month you want to delete. ie. Jan Feb Mar Apr May Jun Jul
Aug Sep Oct Nov Dec"
read MONTH
zfs list | grep $MONTH > /tmp/list
cat /tmp/list
echo "Are these the snapshots you want to destroy? y/n"
read CHOOSE
if [ "$CHOOSE" = "y" ]
then
for i in $(awk '{ print $1 }' < /tmp/list )
do
echo $i
/usr/sbin/zfs destroy $i
done
fi
```

Solaris 8 zones

1. Find and prepare a sparc box with Solaris 10u4. It is important to have the latest Solaris 10 update. Preparations are usually limited to applying a kernel patch, 127111-01 in my case.

2. Download the Solaris 8 Migration Assitant (current version is 1.0) from this location: Solaris 8 Migration Assistant. The 3 packages in archive are dead easy to install using standard **pkgadd**.

Here are the packages you'll get:

SUNWs8brandr Solaris 8 Migration Assistant: solaris8 brand support (Root)
SUNWs8brandu Solaris 8 Migration Assistant: solaris8 brand support (Usr)
SUNWs8p2v Solaris 8 p2v Tool

3. Make a flar-archive of your Solaris 8 system

Log onto your Solaris 8 box, and run the command. In this and all the following examples, solaris8 is nothing but an arbitrary name I've chosen for my zone. You might as well call it anything you like.

```
bash-3.00# flarcreate -S -n solaris8 solaris8.flar
```

4. Create the basic Solaris 8 zone.

Here's how you do it:

```
bash-3.00# zonecfg -z solaris8
solaris8: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:solaris8-system> create -t SUNWsolaris8
zonecfg:solaris8> set zonepath=/export/solaris8
zonecfg:solaris8> add net
zonecfg:solaris8:net> set address=172.21.7.155/24
zonecfg:solaris8:net> set physical=ce0
zonecfg:solaris8:net> end
zonecfg:solaris8> commit
zonecfg:solaris8> exit
```

Naturally, your IP and network device name will be different. As of now, our zone is fully *configured*, but not yet *installed*.

5. Install the Solaris 8 zone using our flar-archive

```
bash-3.00# zoneadm -z solaris8 install -u -a /export/solaris8.flar
Log File: /var/tmp/solaris8.install.13597.log
Source: /export/solaris8.flar
Installing: This may take several minutes...
Postprocessing: This may take several minutes...
WARNING: zone did not finish booting.

Result: Installation completed successfully.
Log File: /export/solaris8/root/var/log/solaris8.install.13597.log
```

In my case the Solaris 8 zone got stuck on **sys-unconfig**, and so I had to connect to the virtual console of the zone to help it move on:

Here's how you connect to a zone's console:

```
bash-3.00# zlogin -C solaris8
```

That's it! The rest was easy - just a few minutes of configuring the network parameters and DNS/NIS settings. Finally, I was able to ssh into the new zone and run **uname**:

```
solaris8 #uname -a
SunOS solaris8 5.8 Generic_Virtual sun4u sparc SUNW,Sun-Fire-V490
```

```
# prtvtoc /dev/rdsk/c0t1d0s2 | fmthard -s - /dev/rdsk/c0t0d0s2
fmthard: New volume table of contents now in place.

# installboot /usr/platform/sun4u/lib/fs/ufs/bootblk /dev/rdsk/c0t0d0s0
# metadb -f -a /dev/dsk/c0t0d0s5

# metadb -f -a /dev/dsk/c0t0d0s6

# metadb -i
 flags first blk block count
 a u 16 1034 /dev/dsk/c0t0d0s5
 a u 16 1034 /dev/dsk/c0t0d0s6
 a m p luo 16 1034 /dev/dsk/c0t1d0s5
 a p luo 16 1034 /dev/dsk/c0t1d0s6
 o - replica active prior to last mddb configuration change
 u - replica is up to date
 l - locator for this replica was read successfully
 c - replica's location was in /etc/lvm/mddb.cf
 p - replica's location was patched in kernel
 m - replica is master, this is replica selected as input
 W - replica has device write errors
 a - replica is active, commits are occurring to this replica
 M - replica had problem with master blocks
 D - replica had problem with data blocks
 F - replica had format problems
 S - replica is too small to hold current data base
 R - replica had device read errors

# metareplace -e d0 c0t0d0s0
d0: device c0t0d0s0 is enabled

# metareplace -e d1 c0t0d0s1
d1: device c0t0d0s1 is enabled

# metareplace -e d4 c0t0d0s4
d4: device c0t0d0s4 is enabled
```

JUMPSTART:

This JET installation recipe steps through the exact commands needed to install JET onto a Solaris machine. In this example our jumpstart host is a X4200 running Solaris 10 8/07. It also acts as the DNS server for this particular environment.

Most of the JET install is vanilla, you have to do these steps to make it work. The parts that introduce new steps are of course adding the DNS configs and also where we insert machine customization scripts. As well we use a `jumpstart_config.pl` script to do the hard work of tweaking the JET environment. It's written in Perl and hopefully self explanatory. It needs to be run a few times during the whole procedure, once to set up the JET server with all the mods it requires, and from then on it gets run when configuring a host as it edits the hosts templates etc. The hosts in these examples also include a few zones each and I've added a `no_zone_sysidcfg` script which will tweak the zones so that they boot up fully instead of requiring a `zlogin -C <zone>` command to be run so that all the `sysidcfg` questions can be answered. (I was surprised JET didn't do these tweaks).

I've altered two JET scripts slightly. The `jumpstart_config.pl` script does the mods automatically to the `/opt/SUNWjet/Products/zones/postinstall` and the `/opt/SUNWjet/Utils/S99jumpstart` JET scripts. The mods are trivial, adding a hook for the zone `sysidcfg` fixer and putting a final `init 6` on the end of the `/opt/SUNWjet/Utils/S99jumpstart` script so that the machine and zones get rebooted and come up clean.

Installation Recipe

Get the software from the JET website.
Current version of JET (as at 4th December 2007) is version 4.4.
Copy the software to /var/tmp on your jumpstart server.
Install the JET packages
Setup your PATH variable. Add `/opt/SUNWjet/bin` and `/boot/grub/bin` to it.

```
jumphost:/ root# env | grep ^PATH
PATH=/usr/local/bin:/bin:/usr/ccs/bin:/etc:/usr/etc:/usr/bin:/usr/local/
X11/bin
 /usr/bin/X11:/usr/proc/bin:/usr/openwin/bin:/usr/local/sbin:/usr/local/
etc:/opt/SUNWspro/bin
 /usr/sbin:/opt/RICHPse/bin:/usr/ucb:/usr/dt/bin:/usr/platform/sun4u/sbin
 /usr/lib/osa/bin:/opt/SUNWjet/bin:/boot/grub/bin:.
```

Check your umask. It should be `22`.

```
jumphost:/ root# umask
22
```

Install and run the jumpstart configuration script to setup the jumpstart environment.

```
jumphost:/ root# chmod 755 /var/tmp/jumpstart_config.pl
jumphost:/ root# /var/tmp/jumpstart_config.pl
```

Transfer the Solaris media from the DVD into the jumpstart tree.

```
jumphost:/ root# copy_solaris_media -d /data/jumpstart/media/
sol_10_807_x86 -n OS10_X86_807 /cdrom/sol_10_807_x86
Copying Solaris image....
Verifying target directory...
Calculating the required disk space for the Solaris_10 product
Calculating space required for the installation boot image
Copying the CD image to disk...
Copying Install Boot Image hierarchy...
Copying /boot x86 netboot hierarchy...
Install Server setup complete

Added Solaris image OS10_X86_807 at the following location:
 Media: /data/jumpstart/media/sol_10_807_x86

removing directory /data/jumpstart/media/2835

intel-jumphost:/ root# copy_solaris_media -d /data/jumpstart/media/
sol_10_807_sparc -n OS10_SPARC_807 /cdrom/sol_10_807_sparc
sparc-jumphost:/ root# copy_solaris_media -d /data/jumpstart/media/
sol_10_807_sparc -n OS10_SPARC_807 /cdrom/sol_10_807_sparc/s0
Copying Solaris image....
Verifying target directory...
Calculating the required disk space for the Solaris_10 product
Calculating space required for the installation boot image
Copying the CD image to disk...
Copying Install Boot Image hierarchy...
Install Server setup complete

Added Solaris image OS10_SPARC_807 at the following location:
 Media: /data/jumpstart/media/sol_10_807_sparc

removing directory /data/jumpstart/media/7328
```

Verify the media copy

```
jumphost:/ root# list_solaris_locations
Version Location
------- --------
```

```
OS10_X86_807 /data/jumpstart/media/sol_10_807_x86
OS10_SPARC_807 /data/jumpstart/media/sol_10_807_sparc
```

Add software for explo module. Note the `jumpstart_config.pl` script above will have altered the JET settings so that Sun Explorer 5.10 will work.

```
Copy Sun explorer software to /var/tmp/explo
drwxr-xr-x 4 root root 512 Dec 6 18:17 SUNWexplo/
drwxr-xr-x 4 root root 512 Dec 6 18:17 SUNWexplu/

jumphost:/ root# pkginfo -d /var/tmp/explo -l | egrep '(PKGINST|VERSION)'
 PKGINST: SUNWexplo
 VERSION: 5.10,REV=2007.09.20.19.12
 PKGINST: SUNWexplu
 VERSION: 5.10,REV=2007.09.20.19.12

jumphost:/ root# copy_product_media explo 5.10 /var/tmp/explo sparc
jumphost:/ root# copy_product_media explo 5.10 /var/tmp/explo i386
Transferring <SUNWexplo> package instance
Transferring <SUNWexplu> package instance
Packges copied.

jumphost:/ root# list_product_versions
Product Versions
------- --------
explo 5.10
```

Add software for custom module.

```
jumphost:/var/tmp/sol10-x86 root# pkgtrans sudo-1.6.8p9-sol10-intel-local
. SMCsudo
Transferring <SMCsudo> package instance

jumphost:/var/tmp/sol10-x86 root# copy_custom_packages /var/tmp/sol10-x86
i386 SMCsudo
Transferring <SMCsudo> package instance
Packages copied

jumphost:/var/tmp/sol10-sparc root# pkgtrans
sudo-1.6.8p12-sol10-sparc-local . SMCsudo
Transferring <SMCsudo> package instance

jumphost:/var/tmp/sol10-sparc root# copy_custom_packages /var/tmp/
sol10-sparc sparc SMCsudo
Transferring <SMCsudo> package instance
Packages copied
```

Tune the base_config template. Be sure to quote the timezone field as shown.

```
jumphost:/etc/ssh root# vi /opt/SUNWjet/Products/base_config/solaris/
base_config.conf
 base_config_client_allocation="dhcp grub"
 base_config_sysidcfg_root_password="PaSsWoRd"
 base_config_sysidcfg_timezone=\"GMT+10\"
 base_config_sysidcfg_system_locale=en_AU.ISO8859-1
 base_config_sysidcfg_timeserver=192.168.100.14
 base_config_x86_console="ttya"
 base_config_x86_disable_kdmconfig="yes"
 base_config_profile_cluster=SUNWCall
 base_config_profile_swap=4096
 base_config_profile_s4_mtpt="/data"
 base_config_profile_s4_size="free"
 base_config_profile_s5_mtpt="/var"
 base_config_profile_s5_size="40960"
 base_config_profile_del_packages="SUNWimagick SUNWpmowu
SUNWgnome-search-tool"
 base_config_profile_del_clusters="SUNWCpm SUNWCpmx SUNWCdial SUNWCdialx
SUNWCaudd SUNWCstaroffice
 SUNWCpostgr SUNWCpostgr-dev SUNWCpostgr-82 SUNWCpostgr-82-dev SUNWCsppp
SUNWCmoz SUNWCmozdeveloper
 SUNWCbrowser SUNWCbrowserdev SUNWCgna11y SUNWCgna11ydev SUNWCgnapps
SUNWCgndev SUNWCgnex SUNWClx
 SUNWCxscreensaver SUNWCthai SUNWCGtk SUNWCzebra SUNWCevo SUNWCevodev
SUNWCsip SUNWCmco SUNWCsmc SUNWCwbem"
 base_config_notrouter="yes"
 base_config_dns_domain="test.example.com"
 base_config_dns_nameservers="192.168.100.14"
 base_config_dns_searchpath="test.example.com example.com"
 base_config_enable_rootlogin="yes"
 base_config_nfsv4_domain="test.example.com"
```
Tell JET about the subnets it will control
```
/opt/SUNWjet/etc/server_interfaces
192.168.205.0 255.255.255.192 192.168.100.14
192.168.174.64 255.255.255.192 192.168.100.14
192.168.104.0 255.255.255.192 192.168.100.14
```
Define the subnets in the JET `defaultrouters` file
```
/opt/SUNWjet/etc/defaultrouters
192.168.174.64 255.255.255.192 192.168.174.65
192.168.104.0 255.255.255.192 192.168.104.1
192.168.205.0 255.255.255.192 192.168.205.1
```
Setup DHCP subsystem. Will be site specific for your subnets.
```
jumphost:/ root# mkdir /data/dhcp
jumphost:/ root# dhcpconfig -D -r SUNWfiles -p /data/dhcp
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - nsch1mang01.
DHCP server started.

jumphost:/ root# dhcpconfig -N 192.168.174.64 -m 255.255.255.192 -t
192.168.174.65
jumphost:/ root# dhcpconfig -N 192.168.104.0 -m 255.255.255.192 -t
192.168.104.1
jumphost:/ root# dhcpconfig -N 192.168.205.0 -m 255.255.255.192 -t
192.168.205.1
jumphost:/ root# dhtadm -A -m PXEClient:Arch:00000:UNDI:002001 -d
':BootFile="nbp.SUNW.i86pc":BootSrvA=192.168.100.14:'
```
You need to set the DHCP config files to use the floating interface only. The default binding to the physical interfaces will break things. This might already have been done in the host set up recipe.
```
jumphost:/ root# svcadm disable dhcp-server
jumphost:/ root# echo "INTERFACES=nge0:1" >> /etc/inet/dhcpsvc.conf
jumphost:/ root# echo "ICMP_VERIFY=TRUE" >> /etc/inet/dhcpsvc.conf
jumphost:/ root# echo "BOOTP_COMPAT=automatic" >> /etc/inet/dhcpsvc.conf
jumphost:/ root# svcadm enable dhcp-server
```
Setup each host template and client config
hosta01 X4200 hostf01 T2000
When the hosts boot up they will issue a number of SUNWfmd messages as the various machine state changes are extracted from the hardware and logged. These can be ignored as they are historical and will cease once they have been worked through.

http://www.brandonhutchinson.com/Solaris_NIC_speed_and_duplex_settings.html

On the target (server)

1. create zfs 'volume' which will be the backing store (our exported disk)`zfs create -V 50g tank/iscsivol000`
2. create iscsi 'base' directory (config store)`iscsitadm modify admin -d /etc/iscsitgt`
3. create iscsi target`iscsitadm create target -b /dev/zvol/dsk/tank/iscsivol000 target-label`
4. list the targets so far if you want...`iscsitadm list target -v`

On the initiator (client)

1. checkout the client's initiator... `iscsiadm list initiator-node`
   ```
   Initiator node name: iqn.1986-03.com.sun:01:ac7812f012ff.45ed6c53
   ```

Back on the target

1. create an alias for that initiator back on our target (server)... `iscsitadm create initiator -n iqn.1986-03.com.sun:01:ac7812f012ff.45ed6c53 suitable-alias`
2. add an acl to the target... `iscsitadm modify target -l suitable-alias target-label`

On the initiator (client)

1. list discovery modes... `root@host ~ # iscsiadm list discovery`
   ```
   Discovery:
   Static: disabled
   Send Targets: disabled
   iSNS: disabled
   ```
2. sendtargets discovery enable `iscsiadm modify discovery --sendtargets enable`
3. and list... `root@host ~ # iscsiadm list discovery`
   ```
   Discovery:
   Static: disabled
   Send Targets: enable
   iSNS: disabled
   ```
4. add a discovery address (our target server) `scsiadm add discovery-address IP.ADD.RES.SS`
   NOTE: must be IP address
5. show what's exported `# iscsiadm list target`
   ```
   [snip]
   LUN: 0
   Vendor: SUN
   Product: SOLARIS
   OS Device Name: /dev/rdsk/c3t0100001B2BC4078001002A0046AEE439d0s2
   ```
6. enable the device... `devfsadm -Cv -i iscsi`

7. see what's there... `root@host ~ # echo | format`
   ```
   Searching for disks...done
   AVAILABLE DISK SELECTIONS:
   0. c2t0d0
   /pci@7b,0/pci1022,7458@11/pci1000,3060@2/sd@0,0
   1. c2t1d0
   /pci@7b,0/pci1022,7458@11/pci1000,3060@2/sd@1,0
   2. c3t0100001B2BC4078001002A0046AEE439d0
   /scsi_vhci/disk@g0100001a4ba4078000002a0046b6e439
   ```
8. Be surprised, OOO!
9. manage with zfs `root@host ~ # zpool list`
   ```
   NAME SIZE USED AVAIL CAP HEALTH ALTROOT
   tank 62G 17.2G 44.8G 27% ONLINE -
   root@host ~ # zpool create iscsitank
   c3t0100001B2BC4078001002A0046AEE439d0
   root@host ~ # zpool list
   NAME SIZE USED AVAIL CAP HEALTH ALTROOT
   iscsitank 49.8G 53.5K 49.7G 0% ONLINE -
   tank 62G 17.2G 44.8G 27% ONLINE -
   ```
10. do some stuff `root@host ~ # zfs create iscsitank/test`
    ```
    root@host ~ # zfs set mountpoint=none iscsitank
    root@host ~ # zfs set mountpoint=/test iscsitank/test
    root@host ~ # zfs list | grep iscsi
    iscsitank 120K 49.0G 25.5K none
    iscsitank/test 24.5K 49.0G 24.5K /test
    ```

Removing a target

1. On the client, unmount/zpool destroy the drive
2. iscsitadm list target -v
3. iscsitadm delete target -u 0 target-tgt0 where -u <ID> is LUN ID.

# zfs create -o sharenfs=anon=0 nfspool/nfs1
zfs set mountpoint=/nfs1 nfspool/nfs1

How to install Solaris 10 companion CD

I was working on a Solaris 10 x64 update 6 system, and needed a X11vnc application. I know of Sun Freeware site or Blastwave site to grab individual packages, but wanted to try to install the whole Companion CD myself. It involved mounting iso file using lofiadm, which was new to me. Also, use of administration file with pkgadd was new to learn. Here is the procedure.

Download iso image of the Solaris Companion CD from these sites:

1. Solaris freeware from Sun website, or
2. Sun Freeware website, and pick processor/OS on the right hand side of the front page.

Unlike O/S installation CD/DVD, companion CD is a collection of Solaris packages to be installed. Assuming you want to install the packages from the CD on a Solaris machine (Solaris 9, 10, Sparc and x86/x64), login to the system as root.
Unzip the iso file and mount it:

```
# cd <download path>
# unzip sol-10-u6-companion-ga-iso.zip
# lofiadm -a ./sol-10-u6-companion-ga.iso /dev/lofi/1
# mount -F hsfs -o ro /dev/lofi/1 /mnt
```

`/mnt` directory contains the collection of packages under `Solaris_Software_Companion`, and includes `Solaris_i386`, `Solaris_Sparc` and `Source` directories. Under `Solaris_i386` and `Solaris_Sparc` directories, you can find `Packages` directory containing more than 100 packages.

```
# cd /mnt
# ls
Legal_Notice                README
LICENSE_NOTICE              Solaris_Software_Companion
# ls Solaris_Software_Companion
Solaris_i386   Solaris_sparc  Source
```

For installing individual packages, cd into the packages directory and run this:

```
# cd Solaris_Software_Companion/Solaris_i386/Packages
# pkgadd -d `pwd` <package name>
```

For installing all the packages, create an install administration file such as:

```
# cat /var/tmp/admin
mail=
conflict=nocheck
setuid=nocheck
action=nocheck
partial=nocheck
instance=overwrite
idepend=nocheck
rdepend=nocheck
space=check
# cd <Packages directory>
# pkgadd -a /var/tmp/admin -d `pwd`
```

These steps will install packages under `/opt/sfw/bin` directory. More details can be found in the `README` file under the mounted CD file directory. Or user can refer to the Companion CD page of Sun Freeware website.
`lofiadm` man page is here, and **`pkgadd`** man page is here.

The solaris pstack command can be used to view most user process stacks, but I recently encountered a situation where even the pstack -F could not be used to view the process stack.
I issued a zfs recv operation that never returned.

```
# ps -ef | grep zfs
    root  7485  7484   0 01:40:01 ?         0:00 zfs recv -F storage ...
# pstack -F 7485
```

[no output]
pstack could not grab control the process to display the stack.
My next option is to use mdb to view the stack - no quite as convenient as pstack, but more powerful ... here are the steps:

```
# mdb -k
/* get handle to the the zfs process using ::pgrep */
>::pgrep zfs
S    PID   PPID   PGID    SID    UID      FLAGS             ADDR NAME
R   7485   7484   7484   7484      0 0x4a004000 ffffff278437ca88 zfs
/* use the returned ADDR value to get the process threadlist */
>  ffffff278437ca88::threadlist
            ADDR             PROC              LWP CMD/LWPID
ffffff278437ca88 ffffff21da3b68e0                0 /239

/* use the PROC value to view the stack */
> ffffff21da3b68e0::findstack
stack pointer for thread ffffff21da3b68e0: ffffff00f88fc880
[ ffffff00f88fc880 _resume_from_idle+0xf1() ]
  ffffff00f88fc8b0 swtch+0x160()
  ffffff00f88fc960 turnstile_block+0x764()
  ffffff00f88fc9d0 rw_enter_sleep+0x1a3()
  ffffff00f88fca40 dsl_dataset_clone_swap+0x61()
  ffffff00f88fca90 dmu_recv_end+0x57()
  ffffff00f88fcc40 zfs_ioc_recv+0x31e()
  ffffff00f88fccc0 zfsdev_ioctl+0x10b()
  ffffff00f88fcd00 cdev_ioctl+0x45()
  ffffff00f88fcd40 spec_ioctl+0x83()
  ffffff00f88fcdc0 fop_ioctl+0x7b()
  ffffff00f88fcec0 ioctl+0x18e()
  ffffff00f88fcf10 sys_syscall32+0x101()
```

Here's a script to display all stacks for a given process type ...
For example: show all zfs stacks:

```
#!/bin/sh

for p in `pgrep zfs`; do
  echo "-------------------------"
  pargs $p
  echo "0t${p} ::pid2proc|::walk thread|::findstack" | mdb -k
done
```