

## NAME

**copyroot** – Make an alternative boot disk.

## SYNOPSIS

**copyroot** help=y

**copyroot**

## DESCRIPTION

Those who manage Sun/Solaris system know the importance of the boot disk. To protect the boot disk, administrators have opted placing the OS on online mirrored disks in addition to a regular backup.

The great advantage of online mirror is hardware failure protection; When one of the mirrored disks fails, the system will still be up and running.

However, if there is a file error, file system corruption, or a human error, both of the mirrored disks are at fault. It is slow and difficult to restore from backup tape – especially when the backup/restore software is unavailable because of an error.

In this article, I introduce a simple alternative approach; Copyroot, listing 2, makes Solaris alternative bootable backup boot disk.

Compared with online mirror, copyroot creates a delayed, “offline” mirror. Changes on the primary disk do not immediately propagate to the backup boot disk, providing a chance to correct the error, or fall back to the last known good OS. Compared to an offline tape backup, an alternative backup boot disk is a online backup, and ready to boot.

Backup boot disk does not prevent the server from going down, when the primary boot disk crashes. However, simply type a boot command to boot from the backup disk, or it may be possible to automatically boot when “boot-device” is set to multiple boot disks.

Solaris allows defining a device alias using “nvalias”, e.g.:

```
nvalias bkupboot /pci@1f,4000/scsi@3/disk@8,0
```

The above alias allows this boot command:

```
boot bkupboot
```

which boots the system from the backup boot disk. The boot-device can be defined to have multiple boot disks with, e.g.:

```
ok> setenv boot-device primboot bkupboot
```

Whenever a server needs to be shutdown for maintenance tasks, I always perform the ritual of “boot bkupboot, shutdown, boot” to verify the backup boot disk.

Also, you do not need to choose from using online mirrors, and backup boot disk if you can afford a third disk, you can have both.

Copyroot was written in the Solaris 2.3 days, and works all the way up to Solaris 9. I recently updated it to support a configuration where the primary boot disk is mirrored using Solaris Volume Manager, which is part of Solaris 9.

I did not test copyroot for creating a boot disk under Veritas Volume Manager, but it should work. However, one of the Sun BluePrints paper [1] “strongly discourages” Veritas Volume Manager on boot disk in favor of Solaris Volume Manager, citing that “VERITAS volumes do not, by default, correspond to partitions”, and mentions “extra complications”.

Now, let’s review how copyroot works, following the program flow, and emphasizing the major points.

## Setting Up Working Environment

The shell used is ksh93d, which is part of Solaris distribution at least since Solaris 2.6. During the recent update, some new features of ksh93 comes in handy.

Following shell definition, I always define PATH, which solves the “can run on command line, but not on cron” problem. “builtin -d unname” removes unname as a ksh93d built-in command, forcing unname to be

/bin/uname. The builtin “uname -i” which previous delivered unexpected results, is fixed in the recent ksh93 shell.

Lines 15–24 are generic; It sets up the OPT (“OPT Power Tools”) directory structure, one of my earlier designs:

- All jobs reside in a common directory (\$my\_dir).
- The configuration files, or associated programs, if any, are in the same directory as jobs, and can be derived from the job names (\$my\_name). For example, the configuration corresponding to \$my\_name would be \$my\_name.cf.
- The log directory \$my\_logdir for each job is \$my\_dir/log/\$my\_name.
- Log file (\$my\_logfile) has the time\_stamp appended, and the alphabetic order (ls listing) corresponds to the time order;
- Redirect all the standard error and output to the log file, which catches all the errors – especially unexpected errors.
- Log files are cleaned by the program itself. The retention period is defined in the configuration file, or set to the default value.

### Sourcing the Configuration File

Copyroot needs some information to start, which is obtained by sourcing sources the configuration file, copyroot.cf.

This is a configuration file sample:

```
machine_name="ziqui"           # for verification purpose
p_boot=$(                     # primary boot disk partitions
    print /dev/dsk/c0t0d0s0
    print /dev/dsk/c0t0d0s1
    print /dev/dsk/c0t0d0s3
    print /dev/dsk/c0t0d0s4
    print /dev/dsk/c0t0d0s6
)
b_boot=$(                     # backup boot disk partitions
    print /dev/dsk/c0t8d0s0
    print /dev/dsk/c0t8d0s1
    print /dev/dsk/c0t8d0s3
    print /dev/dsk/c0t8d0s4
    print /dev/dsk/c0t8d0s6
)
backup_dir="/apps2"          # optional directory to hold boot disk backup
```

The variables defined in the configuration file are:

Define the machine\_name. The purpose is to compare the definition with output from “uname -n”, making sure copyroot runs on the correct machine.

Define the device names for the primary boot disk, and define the backup boot disk. copyroot assumes that the first item in the list is the device name for the root file system, which every Solaris system must have. The second item is the dedicated swap partition; If you do not have one, or if it is not on the boot disk, use “-” as place holder. The primary device names and backup device names should correspond one-to-one.

Reverse copy is done simply by swapping the definition of p\_boot and b\_boot; This is needed when the primary boot disk crashes and is replaced.

“backup\_dir” is optional directory to hold boot disk backup, which I clarify later.

When the boot disk is under the Solaris Volume Manager, the definition of p\_boot looks like this:

```
p_boot=$(
    print /dev/md/dsk/d0
    print /dev/md/dsk/d1
    print /dev/md/dsk/d3
    print /dev/md/dsk/d4
    print /dev/md/dsk/d6
)
```

### Checking the Run Machine

Ensure the script executes only on the box where you want it to run. To disable the program, simply define `machine_name` as something that does not match “`uname -n`”.

This might be an option, for example, when the OS on the primary boot disk is upgraded, and you want the backup boot disk to still have the old OS for an extended period of time, lines 37–40.

This erroring out also reminds you to change the `machine_name` back to “`uname -n`”, after you feel the upgrade is ok.

### Checking the Mounted File Systems

In order to perform duplication using `ufsdump` and `ufsrestore`, mount the backup disk slices under `/copyroot`. Initially, nothing should be mounted under `/copyroot`. If there is, perhaps something else is mounted. Also, the backup disk slices should not be mounted at all. If these conditions happen, something is wrong and `copyroot` should not continue, lines 42–46.

### Performing the Dry Run

Lines 48–67 performs the dry run. `Copyroot` cycles through each slice except the swap slice, running `ufsdump` on each slice, and outputting to `/dev/null`, or a backup file under `$backup_dir` if it is defined in the configuration file.

`copyroot` uses `ufsdump` to check the primary boot disk. If the primary disk has a corruption, it most likely shows up in a dry run. I then terminate the program, preventing the corruption to propagate to the backup disk. This is not a perceived caution, but a lesson learned the hard way.

Since I do not know what error messages `ufsdump` may produce, I parse the messages, line by line, checking for known, expected output. Anything other than expected output is considered an error, and the program terminates immediately. This is more reliable than checking the exit code.

Saving the `ufsdump` file, and using it later on for restore not only avoids the need to read the boot disk twice, but also is safer since the restore uses the verified backup, instead of `ufsdump` output on the fly.

### Partitioning Backup Disk

A prerequisite of using `copyroot` is partitioning the backup disk, which needs to be partitioned only initially, and in the rare case when the primary disk is repartitioned. If the backup disk is of the same size as the primary, you can, and should partition the backup disk as the primary. The syntax is shown in a comment, line 69, or you can do it manually.

`Copyroot`, which uses `ufsdump` and `ufsrestore`, does not require the backup disk be the same size or have the same partitions as the primary. It only requires the backup partitions be large enough to hold the same amount of data as the primary.

### Performing the Actual Run

Lines 71–106 creates the backup disk.

For each slice of the backup disk except the swap partition, execute `newfs` to make a new UFS file system, and check it with `fsck`, lines 77–79. The `fsck` may not be necessary for newly created file systems, but won't hurt.

Next, mount the slice and verify the mount directly, lines 81–82. I `ufsrestore` every slice, skipping the swap partition, from the primary to the backup, using `ufsdump` output on the fly or the saved `ufsdump` file, lines 84–88.

The root file system needs special attention for two reasons:

First, the `vfstab` file contains the boot disk device name for boot strapping purposes. The file on the backup boot should contain the device name of the backup disk instead of the primary. Code lines 91–97 perform the changes.

If the primary boot disk is mirrored using Solaris Volume Manager, the `/etc/system` file contains a line added by “metaroot”:

```
rootdev: /pseudo/md@0:0,0,blk
```

which defines the root device to be the mirrored volume. This line needs to be commented out in the backup `/etc/system`, lines 99–101, when the alternative boot disk is not mirrored. On the other hand, when going from regular boot disk to the mirrored alternative boot disk, this line needs to be uncommented out, lines 101a–101c.

Each slice is synchronized from the primary, `/copyroot` is unmount’ed, and the partition is fsck’ed again – just to be sure.

On a Sun E250 with 2x300MHz CPU, 896 MB memory, all SEAGATE ST318405LC disks, running Solaris 9 full installation, `copyroot` requires about 12 minutes to backup the primary disk to the `$backup_dir` on the third disk, and restore to the backup boot disk. Although relatively short, the synchronization process is inefficient and vulnerable. Considering most of the data are static, I’m seeking a better method.

### Install the Boot Block

Last, but certainly not least, install the boot block, otherwise it won’t boot, line 108. (For those familiar with Linux, this is analogous to `lilo`, or `grub`).

In conclusion, `copyroot` creates a backup boot disk using a simple, standard method, which provides protection for the primary disk data. In case of hardware failure, `copyroot` reduces down time by booting up from the backup boot disk. It can be used with Volume Manager mirrored boot disk to provide both online availability, and data protection.

The article describing this program is published at

Wang, Michael. “Copyroot.” Backup Scripts from UnixReview.com, 3rd Edition. Ed. Ed Schaefer. Sys Admin 12.12 (December 2003): 28–33.

Future version of `copyroot`, if any, will be found at

```
http://www.unixlabplus.com/unix-prog/copyroot.
```

### SEE ALSO

- Erik Vanden Meersch and Kristien Hens, Configuring Boot Disks With Solaris Volume Manager Software, <http://www.sun.com/solutions/blueprints/0402/solstice.pdf>, October 2002.
- Sun Microsystems, Solaris Volume Manager Administration Guide, <http://docs.sun.com/db/doc/806-6111>, 2002.
- John S. Howard and David Deeths, Boot Disk Management: A Guide for the Solaris Operating Environment, Prentice Hall PTR, First Edition, December 27, 2001.

### AUTHOR AND ACKNOWLEDGMENT

```
Michael Wang <F<xw73@columbia.edu>>.
circa 1996.
1999-11, minor modification.
2003-06, major modification for publication.
```

This script was based on an idea by co-worker Dave Singer circa 1996. Both of us worked for Lucent, then AT&T Bell Labs, in Whippany, NJ. Dave originally used the `dd` command but I rewrote it using `ufsdump` and `ufsrestore`, and updated for Solaris Volume Manager.

### Program List

```

001 #!/usr/dt/bin/dtksh
002 #
003 # Listing 2: copyroot
004 # Author: Michael Wang
005 # This script creates a backup boot disk from primary.
006 #
007 # This script was based on an idea by co-worker Dave Singer circa 1996.
008 # Both of us worked for Lucent, then AT&T Bell Labs, in Whippany, NJ.
009 # Dave originally used the dd command but I rewrote it using ufsdump and
010 # ufsrestore, and updated for Solaris Volume Manager.
011
012 PATH=/usr/xpg4/bin:/usr/bin:/usr/sbin
013 builtin -d uname
014
015 my_name=${0##*/}
016 my_dir=${0%/*}
017 my_logdir=$my_dir/log/$my_name
018 [[ -d $my_logdir ]] || mkdir -p $my_logdir
019 my_logfile=$my_logdir/$my_name.$(date +%m-%d_%H:%M)
020
021 exec 1>$my_logfile 2>&1
022
023 date
024 my_conf=$my_dir/$my_name.cf
025
026 [[ -r $my_conf ]] || {
027     print "The configuration file ($my_conf) does not exists."
028     exit 10
029 }
030
031 . $my_conf
032 p_boot=$(print $p_boot)
033 b_boot=$(print $b_boot)
034 : ${log_keep:=30}
035 find $my_logdir -type f -mtime +$log_keep -print -exec rm {} \;
036
037 [[ "$(uname -n)" == "$machine_name" ]] || {
038     print "This script should be run on $machine_name only."
039     exit 20
040 }
041
042 pipe="|"
043 mount | egrep "^/copyroot | on (${b_boot// /$pipe}) " && {
044     print "/copyroot or backup disk mounted already."
045     exit 30
046 }
047
048 set -A pa -- $p_boot
049 for (( slice = 0; slice <= ${#pa[@]}-1; slice++ )); do
050     (( slice == 1 )) && continue
051     [[ -n $backup_dir ]] && dump_file=$backup_dir/$slice.dump ||
052         dump_file="/dev/null"
053     ufsdump 0f $dump_file ${pa[$slice]} 2>&1 < /dev/null | while read j; do
054         print -r -- "$j" | egrep "\

```

```

055 DUMP: Writing [0-9.]+ Kilobyte records|\
056 DUMP: Date of (this|last) level 0 dump:|\
057 DUMP: Dumping /dev/. * \(. * \) to|\
058 DUMP: (Mapping|Dumping) \ (Pass . * \) \ [(regular files|directories) \] |\
059 DUMP: Estimated [0-9.]+ blocks \ ([0-9.]+(MB|KB) \) |\
060 DUMP: [0-9.]+ blocks \ ([0-9.]+(MB|KB) \) on . * at|\
061 DUMP: [0-9.]+% done, finished in|\
062 DUMP: DUMP IS DONE" && continue
063     print -r -- "$j"
064     print "Primary root filesystem has problems."
065     exit 40
066 done
067 done
068
069 # prtvtoc /dev/rdisk/c0t0d0s0 | fmthard -s - /dev/rdisk/c0t9d0s0
070
071 set -A ba -- $b_boot
072 [[ -d /copyroot ]] || mkdir /copyroot
073 tab=$(print "\t")
074 cd /
075
076 for (( slice = 0; slice <= ${#ba[@]}-1; slice++ )); do
077     (( slice == 1 )) && continue
078     print y | newfs ${ba[$slice]}
079     fsck -y ${ba[$slice]}
080
081     mount ${ba[$slice]} /copyroot
082     df -Pk /copyroot | grep "^${ba[$slice]} " || exit 60
083
084     if [[ $dump_file == "/dev/null" ]]; then
085         ufsdump 0f - ${pa[$slice]} | ( cd /copyroot && ufsrestore rf - )
086     else
087         ( cd /copyroot && ufsrestore rf $backup_dir/$slice.dump )
088     fi
089
090     (( slice == 0 )) && {
091         for (( s=0; s <= ${#pa[@]}-1; s++ )); do
092             pd=${pa[$s]} bd=${ba[$s]} pr=${pd/dsk/rdisk} br=${bd/dsk/rdisk}
093             sed "s:$pd\([ $tab \):$bd\1;"
094             s:$pr\([ $tab \):$br\1;"
095             " /copyroot/etc/vfstab > /copyroot/etc/vfstab.tmp
096             cp /copyroot/etc/vfstab.tmp /copyroot/etc/vfstab
097         done
098
099         grep "^rootdev:" /etc/system && {
100             sed "/^rootdev:/s:^:* :/" /etc/system > /copyroot/etc/system
101         }
101a     [[ ${ba[0]} == /dev/md/dsk/* ]] && grep "^* rootdev:" /etc/system && {
101b         sed "/^* rootdev:/s:* :/" /etc/system > /copyroot/etc/system
101c     }
102 }
103
104 umount /copyroot
105 fsck -y ${ba[$slice]}

```

```
106 done
107
108 installboot /usr/platform/$(uname -i)/lib/fs/ufs/bootblk ${ba[0]}/dsk/rdisk}
109
110 rmdir /copyroot
111 date
112 print "copyroot finished on $(uname -n) successfully."
113 exit 0
```