

```
#!/bin/sh
#
#####
# Copyright (c) 2003 RENSSELAER TECHNOLOGY GROUP, LTD. ALL RIGHTS RESERVED. #
# UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. #
# USE OF A COPYRIGHT NOTICE IS PRECAUTIONARY ONLY AND DOES NOT IMPLY PUBLICATION #
# OR DISCLOSURE. #
# #
# THIS DOCUMENT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF RENSSELAER #
# TECHNOLOGY GROUP, LTD. USE, DISCLOSURE, OR REPRODUCTION IS PROHIBITED WITHOUT #
# THE PRIOR EXPRESS WRITTEN PERMISSION OF RENSSELAER TECHNOLOGY GROUP, LTD. #
# #
# RESTRICTED RIGHTS LEGEND USE, DUPLICATION, OR DISCLOSURE BY THE GOVERNMENT IS #
# SUBJECT TO RESTRICTIONS AS SET FORTH IN SUBPARAGRAPH (C) (1) (ii) OF THE RIGHTS #
# IN TECHNICAL DATA AND COMPUTER SOFTWARE CLAUSE AT DFARS 252.227-7013. #
# RENSSELAER TECHNOLOGY GROUP, LTD., NEW YORK, NEW YORK. #
#####

#####
# By: Milton Vega - Rensselaer Technology Group, LTD. #
# (v1.4 - 06/12/2003) #
# #
# Script Name.....: /opt/local/sbin/OS_diskcp.sh #
#####

#####
# Note: This script contains white space within regular expressions #
# and thus cannot be cut & pasted. #
# #
# The following script copies an entire O/S from/to the user defined #
# disks specified in the variables DISK1 and DISK2. DISK1 & DISK2 #
# are usually specified to be the two disks purposed for the O/S #
# within a particular machine type (e.g. for a 6800 it would be the #
# two disks within the D240 storage unit). #
# #
# For example: #
# DISK1=c0t1d0 and DISK2=c0t0d0 #
# -or- #
# DISK1=c0t0d0 and DISK2=c0t1d0 #
# #
# Because the script DYNAMICALLY determines which disk is ACTIVE and #
# which is STANDBY, both cases above result in exactly the same #
# outcome. #
#####

#####
# The following variables are set in the configuration file which #
# is sourced by this script: #
# #
# ${TESTMODE} #
# ${DISK1} #
# ${DISK2} #
# ${REQUIRE_DRIVE_MCH} #
# #
# The configuration file MUST exist, and each of these variables #
# must be set to non-NULL values. This program will not run #
# otherwise. #
# #
# The configuration file MUST exist in the following location and #
# have the permissions shown: #
# #
# -rw-r--r-- root:root /opt/local/etc/OS_diskcp.conf #
# #
```

```

# -----#
# Sample configuration file contents: #
# -----#
# TESTMODE=YES #
# REQUIRE_DRIVE_MATCH=YES #
# DISK1=c0t0d0 #
# DISK2=c0t1d0 #
# -----#
#####

#####
# Setup the operating environment. #
#####
PATH=/usr/bin:/usr/sbin:/usr/ucb:/etc:/usr/ccs/bin
#
BASENAME=`basename $0`
LOG_FILE=/var/log/"${BASENAME}_log"
#[ x"${TESTMODE}" = x"NO" ] && exec > ${LOG_FILE} 2>&1
exec > ${LOG_FILE} 2>&1
#
FATALERROR=NO
VTOC_TMPFILE="/tmp/destDISK.vtoc.$$"
CONFIG_FILE=/opt/local/etc/OS_diskcp.conf
MNTPT_DIR="/tmp/${MNTPT}.$$"
mkdir -p ${MNTPT_DIR}
BOOTBLK="/usr/platform/`uname -i`/lib/fs/ufs/bootblk"
date '+%nSTART DATE: %m/%d/%y%nSTART TIME: %H:%M:%S%n'
#####

#####
# Make sure that the configuration file exists, is readable, and #
# that it sets the variables needed by this program. #
#####
[ ! -r ${CONFIG_FILE} ] && FATALERROR=YES
[ x"${FATALERROR}" != x"YES" ] && . ${CONFIG_FILE} > /dev/null 2>&1
[ x"${TESTMODE}" = x"" ] && FATALERROR=YES
[ x"${DISK1}" = x"" ] && FATALERROR=YES
[ x"${DISK2}" = x"" ] && FATALERROR=YES
[ x"${REQUIRE_DRIVE_MATCH}" = x"" ] && FATALERROR=YES

if [ x"${FATALERROR}" = x"YES" ]
then
{
echo "====="
echo "FATAL ERROR:"
echo "====="
echo " Either the configuration file '${CONFIG_FILE}'"
echo " does not exist, is not readable, or one of the following"
echo " REQUIRED environment variables is not set within that"
echo " file to an appropriate (non-null) value (example):\n"
echo " TESTMODE=YES|NO "
echo " DISK1=cWtXdY "
echo " DISK2=cWtXdY "
echo " REQUIRE_DRIVE_MATCH=YES|NO "
echo "\n Terminating Abnormally! No changes made to system.\n"
exit 1
}
fi
#####

#####

```

```

# Make sure we can actually access the disk devices specified.      #
# To do this we essentially "PING" the disks using two methods:    #
#                                                                    #
# (1) prtvtoc: to see if we can extract the VTOC table, and        #
# (2) format: to see if we can extract its SCSI inquiry string from #
#             its firmware (via the inquiry sub-command).          #
#                                                                    #
# If either of these fail for either disk, we terminally exit.    #
#####
prtvtoc /dev/rdisk/${DISK1}s2 > /dev/null 2>&1 || FATALERROR=YES
prtvtoc /dev/rdisk/${DISK2}s2 > /dev/null 2>&1 || FATALERROR=YES

> /dev/null 2>&1 format -s -M ${DISK1} << XxX || FATALERROR=YES
inquiry
q
XxX

> /dev/null 2>&1 format -s -M ${DISK2} << XxX || FATALERROR=YES
inquiry
q
XxX

if [ x"${FATALERROR}" = x"YES" ]
then
{
  echo "====="
  echo "FATAL ERROR:"
  echo "====="
  echo "  prtvtoc(1)/format(1M) cannot access either:"
  echo "    /dev/rdisk/${DISK1}s2"
  echo "    /dev/rdisk/${DISK2}s2"
  echo "\n  Terminating Abnormally! No changes made to system.\n"
  exit 1
}
fi
#####

#####
# Get the sizes of the drives, DISK1 & DISK2. Depending on their  #
# sizes relative to one another take action based on the user    #
# specified value of the REQUIRE_DRIVE_MATCH variable.          #
#####
DISK1_SZ=`format ${DISK1} stub << EOF 2> /dev/null | \
sed -ne '/ </s/.*<[a-zA-Z]*\([0-9][0-9]*\).*\/\1/p`
DISK2_SZ=`format ${DISK2} stub << EOF 2> /dev/null | \
sed -ne '/ </s/.*<[a-zA-Z]*\([0-9][0-9]*\).*\/\1/p`

if [ x"${REQUIRE_DRIVE_MATCH}" != x"NO" -a \
x"${DISK1_SZ}" != x"${DISK2_SZ}" ]
then
{
  echo "====="
  echo "FATAL ERROR:"
  echo "====="
  echo "  ${DISK1} and ${DISK2} have different capacities."
  echo "  ${DISK1}=${DISK1_SZ}"
  echo "  ${DISK2}=${DISK2_SZ}"
  echo "\n  Terminating Abnormally! No changes made to system.\n"
  exit 1
}
fi
#####

```

```

#####
# Dynamically determine ACTIVE & STANDBY disks at THIS MOMENT.      #
#####
ACTIVE_DISK=`df -k / | sed -ne '2s/s[0-9][          ][          ]*.*//' -ne '2s/.*/c/c/p'`
STDBY_DISK=${DISK2}
[ x"${ACTIVE_DISK}" = x"${DISK2}" ] && STDBY_DISK=${DISK1}
#####

#####
# Some safety checks...                                             #
#####
# (1) check to see if the ACTIVE_DISK was either DISK1 or DISK2.   #
# We do this because the ACTIVE_DISK might not be of the form     #
# /dev/dsk/c#t#d#s#; it may be of the form /dev/md/dsk/d10 or     #
# /dev/vx/dsk/rootdg/rootvol or something else, in which case we  #
# absolutely want to fatally terminate and not proceed.           #
#                                                                    #
# (2) We also take this opportunity to ensure that the STDBY_DISK is #
# not being used to host any filesystem or swap space.           #
#####
if [ x"${ACTIVE_DISK}" != x"${DISK1}" -a \
    x"${ACTIVE_DISK}" != x"${DISK2}" ]
then
{
    echo "====="
    echo "FATAL ERROR:"
    echo "====="
    echo "    The ACTIVE_DISK is neither ${DISK1} nor ${DISK2}."
    echo "    Perhaps a Logical Volume Manager is being used."
    echo "\n    Terminating Abnormally! No changes made to system.\n"
    exit 1
}
fi

df -kl | grep ${STDBY_DISK} > /dev/null 2>&1 && FATALERROR=YES
swap -l | grep ${STDBY_DISK} > /dev/null 2>&1 && FATALERROR=YES
if [ x"${FATALERROR}" = x"YES" ]
then
{
    echo "====="
    echo "FATAL ERROR:"
    echo "====="
    echo "    The STDBY_DISK is being used to host a File System or SWAP space."
    echo "\n    Terminating Abnormally! No changes made to system.\n"
    exit 1
}
fi

#####
# Build a vtoc table for the STDBY_DISK that is as close as as     #
# possible to ACTIVE_DISK. Since every disk has 512 bytes/sector,  #
# for each partition on the ACTIVE_DISK, we assign exactly the same #
# number of sectors to the corresponding partition of the          #
# STDBY_DISK... !!with two exceptions!!: (1) slice 2 (the partition #
# representing the whole disk), and (2) the highest numbered slice. #
# For slice 2, we take what is specified for the STDBY_DISK directly, #
# since we cannot assume that both ACTIVE and STANDBY disks have   #
# exactly the same total number of sectors. Consequently, the last #

```

```

# partition (the highest numbered one) on the STDBY_DISK will have #
# to be calculated. It (the last partition on the STDBY_DISK) may #
# contain more or less space than on the ACTIVE_DISK depending on #
# its size. #
#####

# =====
# Insert partition 2 from the STDBY_DISK
# verbatim into our VTOC table.
# =====
prvtvoc -h /dev/rdisk/${STDBY_DISK}s2 | \
    sed -ne '/^[
        ][*2[
        ]/p' > ${VTOC_TMPFILE}

# =====
# With the exception of partition number
# 2, insert every partition verbatim from
# ACTIVE_DISK into our VTOC table.
# =====
prvtvoc -h /dev/rdisk/${ACTIVE_DISK}s2 | \
    sed -e '/^[
        ][*2[
        ]/d' >> ${VTOC_TMPFILE}

# =====
# Adjust the last partition in our VTOC
# table with a calculated value.
# =====
LAST_SECTOR=`head -1 ${VTOC_TMPFILE} | awk '{print $6}'`
FIRST_SECTOR=`tail -1 ${VTOC_TMPFILE} | awk '{print $4}'`
SECTOR_COUNT=`expr ${LAST_SECTOR} - ${FIRST_SECTOR} + 1`
#
LAST_ENTRY=`tail -1 ${VTOC_TMPFILE} | awk '{print $1, $2, $3}'`
LAST_ENTRY="${LAST_ENTRY} ${FIRST_SECTOR} ${SECTOR_COUNT} ${LAST_SECTOR}"
#
sed -e "\$s/./${LAST_ENTRY}/" ${VTOC_TMPFILE} > ${VTOC_TMPFILE}.tmp
mv ${VTOC_TMPFILE}.tmp ${VTOC_TMPFILE}
# =====

#####
# Display summary of information that will be acted upon. #
#####
echo ""
echo "#####"
echo "#          SUMMARY OF DYNAMICALLY DETERMINED PARAMETERS          #"
echo "#####"
echo "SOURCE DISK..... ${ACTIVE_DISK}"
echo "DESTINATION DISK.... ${STDBY_DISK}"
echo "FS PARTITION LIST.... \c";
prvtvoc -h /dev/rdisk/${ACTIVE_DISK}s2 | awk '/\// {printf $1","}'

echo "\n\nSOURCE VTOC TABLE (${ACTIVE_DISK}):";          prvtvoc -h /dev/rdisk/${ACTI
VE_DISK}s2
echo "\nDESTINATION VTOC TABLE - BEFORE (${STDBY_DISK}):"; prvtvoc -h /dev/rdisk/${STDB
Y_DISK}s2
echo "\nDESTINATION VTOC TABLE - AFTER  (${STDBY_DISK}):"; sort -n ${VTOC_TMPFILE}
echo "#####"
echo ""
#####

#####
# This is the part of the program where modifications to the system #
# actually occur. As such we test this section with "echo" #
# statements first when installing this program onto any system. #
# This is accomplished by setting the environment variable TESTMODE #
# to anything other than "NO" at the top of this program. #
# These same "echo" statements are useful for logging/debugging when #

```

```

# this script is being tested on a new machine, or actually runs #
# unattended (via cron) and you want to know the status of the O/S #
# copy. #
# #
# This section: #
# - Install the VTOC onto STDBY_DISK. #
# - Generates the STDBY_DISK vfstab file. #
# - newfs(1M)es each relevant partition STDBY_DISK. #
# - Copies the O/S and installs the STDBY_DISK vfstab file. #
# - Installs the bootblock on s0 of STDBY_DISK. #
#####

#####
# Determine whether the program statements that are echo'ed out, #
# should also actually be RUN. This is done by examining #
# the value of the TESTMODE variable (specified by the user at #
# the top of this program) and setting the RUN variable to "true" #
# or "false" accordingly. If TESTMODE is set to "NO" (i.e. RUN=true),#
# ONLY THEN will statements be actually executed. If TESTMODE is #
# set to anything other than "NO", including set to null (i.e. not #
# set at all), then RUN will be set to "false" and statements will #
# only be echo'ed out but not run. This is useful for testing. #
#####
RUN=false; [ x"${TESTMODE}" = x"NO" ] && RUN=true
#####

#####
# Install the VTOC table to the STANDBY DISK. Check that the #
# installation succeeded (i.e. that fmthard succeeded). #
#####
echo "cat ${VTOC_TMPFILE} | fmthard -s - /dev/rdisk/${STDBY_DISK}s2"
${RUN} && cat ${VTOC_TMPFILE} | fmthard -s - /dev/rdisk/${STDBY_DISK}s2
EXIT_STATUS=$?
if [ x"${RUN}" = x"true" -a \
    x"${EXIT_STATUS}" != x"0" ]
then
{
echo "====="
echo "FATAL ERROR:"
echo "====="
echo "      Installation of the VTOC table to (${STDBY_DISK}) failed!"
echo "\n      Terminating Abnormally! No changes made to system.\n"
exit 1
}
fi
#####

echo ""

FS_PARTITION_LIST=`prtvtoc -h /dev/rdisk/${ACTIVE_DISK}s2 | awk '/\/// {print $1}'`
for PARTITION in ${FS_PARTITION_LIST}
do
{
echo "echo y | newfs -m 1 /dev/rdisk/${STDBY_DISK}s${PARTITION}"
${RUN} && echo y | newfs -m 1 /dev/rdisk/${STDBY_DISK}s${PARTITION}

echo "mount -F ufs -o rw /dev/dsk/${STDBY_DISK}s${PARTITION} ${MNTPT_DIR}"
${RUN} && mount -F ufs -o rw /dev/dsk/${STDBY_DISK}s${PARTITION} ${MNTPT_DIR}

echo "ufsdump 0f - /dev/dsk/${ACTIVE_DISK}s${PARTITION} | (cd ${MNTPT_DIR}; ufs
restore rf -)"
${RUN} && ufsdump 0f - /dev/dsk/${ACTIVE_DISK}s${PARTITION} | (cd ${MNTPT_DIR}; ufs

```

```
restore rf -)
```

```
  ${RUN} && [ x"${PARTITION}" = x"0" ] && sed -e "/^[ ]*\dev\/\/s/${ACTIVE_DISK}/${STDBY_DISK}/g" \
    /etc/vfstab > ${MNTPT_DIR}/etc/vfstab
```

```
  echo "rm -f ${MNTPT_DIR}/restoresymtable"
  ${RUN} && rm -f ${MNTPT_DIR}/restoresymtable
```

```
  echo "umount ${MNTPT_DIR}"
  ${RUN} && umount ${MNTPT_DIR}
```

```
  echo ""
```

```
}
done
```

```
  echo "installboot ${BOOTBLK} /dev/rdisk/${STDBY_DISK}s0"
  ${RUN} && installboot ${BOOTBLK} /dev/rdisk/${STDBY_DISK}s0
#####
rm -rf ${MNTPT_DIR} ${VTOC_TMPFILE}
date '+%nFINISH DATE: %m/%d/%y%nFINISH TIME: %H:%M:%S%n'
```