

Umsetzung der beim CTServer verwendeten Lösung "Backup mit Snapshots¹" auf dem Eisfair-1

Der Clou an diesem Backup ist neben dem sparsamen Umgang mit dem Festplattenplatz (dank Hardlinks), daß jeder Benutzer über eine einfache Freigabe an die Backup-Daten kommt, aber nur an all die Daten, auf die er normaler Weise Zugriff hat. Die Backup-Daten in der Freigabe sind zudem schreibgeschützt und somit manipulationssicher. Die Backup-Partition ist nur für die kurze Zeit des Backups beschreibbar.

Neben **rsync** (kann über die Paketverwaltung gefunden und herunter geladen werden – eine Konfiguration ist nicht erforderlich) sind der **Samba-** und **NFS-Server** erforderlich.

Als erstes wird das Verzeichnis festgelegt, daß die Backup-Daten aufnimmt und auf spezielle Weise freigegeben wird. Ich habe dafür eine eigene, interne Festplatte vorgesehen. Auf dieser Festplatte befindet sich eine komplette Basis-Installation eines Eisfair-1-Server mit der 4. Partition als Datenpartition. D.h. ich könnte von dieser Festplatte notfalls auch booten, aber es ist natürlich nicht erforderlich, es so einzurichten.

Daher sieht meine **/etc/fstab** also so aus:

```
/etc/fstab 755/755 100%
/dev/sda3 / ext3 defaults,errors=remount-ro 0 1
/dev/sda1 /boot ext3 defaults,errors=remount-ro 0 1
/dev/sda2 none swap sw 0 0
/dev/sda4 /data ext3 defaults 0 2
proc /proc proc defaults 0 0
/dev/fd0 /media/floppy auto defaults,user,noauto 0 0
/dev/cdrom /media/cdrom iso9660 defaults,ro,user,noauto 0 0
devpts /dev/pts devpts defaults 0 0
usbfs /proc/bus/usb usbfs defaults 0 0
/sys /sys sysfs defaults 0 0
#UUID=e40f8e78-81be-4c11-87d9-268d7a90bec3 /mountpoint ext3 defaults, 0 0
/dev/sdb4 /mountpoint ext4 defaults 0 0
192.168.170.200:/mountpoint/data /backup nfs ro,user,noauto 0 0
```

So oder besser per blkid die Datenpartition mounten

Kein automatischen Mouten, weil der Startvorgang des Server dadurch stark verzögert wird. Deshalb ein Aufruf in /etc/init.d/local

Ich habe auf meinem Server also im Rootverzeichnis einen neuen Ordner **/mountpoint** mit einem Ordner **data** angelegt. Hier hänge ich meine Backup-Datenpartition ein. Dies kann per **/dev/sdxy** oder **blkid** umgesetzt werden. Zu der letzten Zeile mit der IP-Adresse komme ich gleich.

1 http://www.ctserver.org/magic_viewtopic.php?f=2&t=884&hilit=backups

Nachdem das Paket **NFS-Server** über die EISfair-Paketverwaltung herunter geladen und installiert wurde, muß es analog zum folgendem Screenshot konfiguriert (mit **F4** alle Optionen einblenden) werden. Die **IP-Adresse** ist die des **EISfair-1-Servers**.

```

/etc/config.d/nfsserver
-----
general settings
-----

START_NFSSERVER                = yes
NFSSERVER_SUPPORT_NFS3        = yes
NFSSERVER_SUPPORT_NFS4        = yes

NFSSERVER_SHARE_N              = 1
NFSSERVER_SHARE_1_ACTIVE      = yes
NFSSERVER_SHARE_1_PATH        = /mountpoint/data
NFSSERVER_SHARE_1_NAME        =

NFSSERVER_SHARE_1_HOSTS_N     = 1
NFSSERVER_SHARE_1_HOSTS_1    = 192.168.170.200
HOST                            =
NFSSERVER_SHARE_1_HOSTS_1    = no
RW                              =
NFSSERVER_SHARE_1_HOSTS_1    =
ANONUSER                        =
NFSSERVER_SHARE_1_HOSTS_1    =
ANONGROUP                       =
NFSSERVER_SHARE_1_HOSTS_1    = no
ROOT_SQUASH                     =
NFSSERVER_SHARE_1_HOSTS_1    = no
ALL_SQUASH                      =
NFSSERVER_SHARE_1_HOSTS_1    = secure
OPTION                          =

```

Für den einfachen Zugriff der Benutzer auf die Backup-Daten ist noch ein weiterer Ordner erforderlich. Ich habe dazu im Rootverzeichnis den Ordner **/backup** angelegt. Dieser Ordner wird per **Samba**, wie im nächsten Screenshot der Samba-Konfiguration zu sehen sehen ist, **freigegeben**.

```

/etc/config.d/samba
-----
SAMBA_SHARE_5_ACTIVE          = yes
SAMBA_SHARE_5_NAME            = backup
SAMBA_SHARE_5_COMMENT         = backup on %h
SAMBA_SHARE_5_RW              = no
SAMBA_SHARE_5_BROWSE          = yes
SAMBA_SHARE_5_PATH            = /backup
SAMBA_SHARE_5_USER            = +users
SAMBA_SHARE_5_PUBLIC          = no
SAMBA_SHARE_5_READ_LIST       =
SAMBA_SHARE_5_WRITE_LIST      =
SAMBA_SHARE_5_FORCE_CMODE     = 0600
SAMBA_SHARE_5_FORCE_DIRMODE   = 0700
SAMBA_SHARE_5_FORCE_USER      =
SAMBA_SHARE_5_FORCE_GROUP     =

```

Diesen Ordner vorher erstellen

Jetzt komme ich auf die letzte Zeile in der `/etc/fstab` zurück. In den per Samba freigegebenen Ordner `/backup` wird der Inhalt der Backup-Partition eingehängt (die IP-Adresse ist die des Eisfair-1-Servers). Ich habe herausgefunden, daß die Dauer des Bootvorganges drastisch reduziert wird, wenn man den Eintrag mit der Option `noauto` versieht und statt dessen einen entsprechenden `mount`-Befehl in der `etc/init.d/local` einfügt.

```
/etc/init.d/local 683/703 97
#!/bin/sh
#-----
# /etc/init.d/local - rc script for gerneral purpose
#
# Creation:      19.07.2003  fm
# Last Update:  20.07.2003  fm
#
# Copyright (c) 2003 Frank Meyer <frank@eisfair.org>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#-----

case $1
in
  start)
    # /opt/avg/avgadmsrv/bin/avgadmsrv &
    #mount /dev/sdb4
    mount /backup
    ;;

```

Zum Mouneten der NFS-Freigabe

Wenn einmal der Bedarf bestehen sollte, die Backup-Festplatte zu „umounten“, hier die von mir ermittelte Vorgehensweise (alle Befehle erfolgen an der Konsole).

`umount /backup` = NFS-Freigabe umounten

`/etc/init.d/nfsserver stop` = NFS-Server stoppen, weil /mountpoint sonst nicht umountbar

`umount /mountpoint` = umount von /mountpoint

`/etc/init.d/nfsserver start` = NFS-Server wieder starten

Nun komme ich zum eigentlichen Backup-Script, das den Namen **snapshot** trägt. Dieses habe ich unter `/usr/bin` abgelegt und `chmod` ausführbar gemacht.

```
/usr/bin/snapshot [----] 3 L:[ 28+41 69/289] *(1720/6501b) 10 0x00A
#####
# Start User Config Stuff
# This is the local root of the backup directories. (where to backup to)
backuproot=/mountpoint
# directory, e.g. /backup/server-data.
snapshot="data"
# You need to set the newdayhour to 00-23 depending on what hour you run
# the script in cron. Ich starte das script z.B. um 2, 18 und 23 Uhr, also
# setze ich newdayhour auf 02
# wenn chour == 02 (current hour) 02 dann Kopie/Link anlegen
newdayhour=13
# number of snapshots to save
savehours=1
savedays=7
saveweeks=4
savemonths=12
saveyears=10
# list directories to backup
allsrcdir="
/home/
/etc/
/var/log/
/var/spool/
/data/www/
/data/VMs/
"
# list directories to skip during in backup
# or leave blank
# EXCLUDES=""
EXCLUDES="--filter=. /tmp/excludes-`${snapshot}`"
```

Diesen Ordner anlegen, falls er nicht existiert

Dieser Ordner muß angelegt werden bzw. unter /mountpoint existieren

Ein cronjob mit dieser Uhrzeit muß angelegt werden

Das Script selber ist gut dokumentiert. Besonders wichtig ist allerdings der Eintrag **newdayhour**. Da ich nur 1x täglich eine Sicherung um 13UHR mache, steht hier also `newdayhour=13`. Es muß zwingend ein Cronjob zum Aufrufen des Snapshot-Backupscripts mit genau dieser Uhrzeit erstellt werden.

```
/etc/config.d/cron
-----
Cron Jobs
-----
START_CRON                = yes
CRON_N                    = 3
CRON_1_NAME               =
CRON_1_ACTIVE             = yes
CRON_1_TIMES              = 0 0 * * *
CRON_1_COMMAND            = /etc/cron.daily/logrotate

CRON_2_NAME               = Plattenplatz prüfen
CRON_2_ACTIVE             = yes
CRON_2_TIMES              = 0 */4 * * *
CRON_2_COMMAND            = /usr/bin/test-plattenplatz

CRON_3_NAME               = Rotations-Sicherung
CRON_3_ACTIVE             = yes
CRON_3_TIMES              = 0 13 * * *
CRON_3_COMMAND            = /usr/bin/snapshot > /dev/null
```

Wer die fcron-Meldung nach dem Laufen des Scripts nicht unterdrücken möchte, läßt das > /dev/null einfach weg. Ich habe dem Original-snapshot-Script noch eine zusätzlich, letzte Zeile mit dem Inhalt **/usr/bin/snapshot-report** hinzugefügt, damit ich eine Benachrichtigung über das Backup per Email zugestellt bekomme. Dazu habe ich in /usr/bin eine ausführbare Datei **snapshot-report** mit folgendem Inhalt erstellt.

Inhalt von /usr/bin/snapshot-report

```
#!/bin/sh
#
#
( echo "To: root"
  echo "Subject: Ordner /backup/hourly auf Backup-Server"
  echo "Größe des Ordners"
  du -sh /backup/hourly
  echo "-----"
  echo "Ende des letzten Durchlaufs der Rotationssicherung"
  ls -la /backup/lastrun.time
  echo "-----"
) | /usr/lib/sendmail root
```

Meine Version von **snapshot**:

```
#!/bin/sh
# mirror
# make a snapshot from directories
# an create hourly/daily/monthly/yearly archieves
# -----
# neobiker (2006)
# originally based on mirror Version 3.11 By Stu Sheldon stu@actusa.net
# ...
# ...
# The directories are named 'snapshot-<year>-<month>-<day>-<hour>'
#
# Each time snapshot runs, it date stamps a file in the <snapshot> directory called
# 'lastrun.time'. This file's date when listing it using ls -laF shows the last
# date and time snapshot ran on that host.
#
# The last thing you need to do is add snapshot in your crontab with the proper
# times and switches.
#
# If you are going for hourly sync's, add the following to your crontab:
# 0 * * * * /usr/local/sbin/snapshot
#
# Every four hours would be:
# 0 0,4,8,12,16,20 * * * /usr/local/sbin/snapshot
#
# you get the idea
# ...

#####
# Start User Config Stuff

# This is the local root of the backup directories. (where to backup to)
backuproot=/mountpoint

# directory, e.g. /backup/server-data
snapshot="data"

# You need to set the newdayhour to 00-23 depending on what hour you run
# the script in cron. Ich starte das script z.B. um 2, 18 und 23 Uhr, also
# setze ich newdayhour auf 02
# wenn chour == 02 (current hour) 02 dann Kopie/Link anlegen
newdayhour=13
```

```
# number of snapshots to save
savehours=1
savedays=7
saveweeks=4
savemonths=12
saveyears=10
```

```
# list directories to backup
allsrcdir=""
/public/
/home/
/etc/
/netlogon/
/root/
/var/www/htdocs/webdav/
"
```

```
# list directories to skip during in backup
# or leave blank
# EXCLUDES=""
```

```
EXCLUDES="--filter=. /tmp/excludes-`${snapshot}`"
```

```
cat > /tmp/excludes-`${snapshot}` << EOF
+ *
EOF
```

```
# Mount and Dismount commands for all reasons are in the following functions
# you can also mount windows-shares via smbclient e.g.
# here: mount for writing during backup, and mount readonly afterwards
```

```
mounting ()
{
    precom=0
    mount -o remount,rw `${backuproot}` || mountfail
}
```

```
umounting ()
{
    postcom=0
    mount -o remount,ro `${backuproot}` || umountfail
}
```

```

# Typical Unix saving localhost
rsync="rsync -a -R -q --delete"

# Typical Unix using ssh and public keys
# rsync="rsync -aR -q --numeric-ids -e ssh --delete"
# This has settings for windows and expects to use a mounted share.
# rsync="rsync -a -R -q --delete --modify-window=10"

# End User Config Stuff
#####
##
# Start Static Code

cmdline=$1
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin

cmonth=`date +%m`
dowweek=`date +%W`
cdays=`date +%d`
chour=`date +%H`

yearidir=${backuproot}/${snapshot}/yearly
monthdir=${backuproot}/${snapshot}/monthly
weekdir=${backuproot}/${snapshot}/weekly
daydir=${backuproot}/${snapshot}/daily
hourdir=${backuproot}/${snapshot}/hourly

snapshotdir=`date +'snapshot-%Y-%m-%d-%H'`

lockfile=${backuproot}/${snapshot}/syncing-now

lsync=`/bin/ls -1r ${hourdir} | head -1`
lday=`date +'snapshot-%Y-%m-%d-%H' -d '1 day ago'`
lweek=`date +'snapshot-%Y-%m-%d-%H' -d '1 week ago'`

makedirs ()
{
    [ -d ${yeardir} ] || mkdir -p ${yeardir}
    [ -d ${monthdir} ] || mkdir -p ${monthdir}
    [ -d ${weekdir} ] || mkdir -p ${weekdir}
    [ -d ${daydir} ] || mkdir -p ${daydir}
    [ -d ${hourdir} ] || mkdir -p ${hourdir}
}

```



```

dosync ()
{
  [ -d ${hourdir}/${snapshotdir} ] || mkdir -p ${hourdir}/${snapshotdir}
  [ -n "${lsync}" ] && rsync="${rsync}" --link-dest=${hourdir}/${lsync}"

  for srcdir in ${allsrcdir}; do
    ${rsync} "${EXCLUDES}" ${srcdir} ${hourdir}/${snapshotdir}
  done
}

```

```

doyearly ()
{
  [ ${saveyears} -eq 0 ] && return
  if [ -d ${monthdir}/${snapshotdir} ]; then
    cp -alF ${monthdir}/${snapshotdir} ${yeardir}
  else
    echo >&2 "I can't find snapshot ${monthdir}/${snapshotdir}..."
  fi
}

```

```

domonthly ()
{
  [ ${savemonths} -eq 0 ] && return
  if [ -d ${hourdir}/${snapshotdir} ]; then
    mv ${hourdir}/${snapshotdir} ${monthdir}
    ln -s ${monthdir}/${snapshotdir} ${hourdir}/${snapshotdir}
  else
    echo >&2 "I can't find snapshot ${hourdir}/${snapshotdir}..."
  fi
}

```

```

doweekly ()
{
  [ ${saveweeks} -eq 0 ] && return
  if [ -d ${daydir}/${lweek} ]; then
    mv ${daydir}/${lweek} ${weekdir}
  else
    echo >&2 "I can't find snapshot ${daydir}/${lweek}..."
  fi
}

```

```

dodaily ()
{
  [ ${savedays} -eq 0 ] && return

```

```

if [ -d ${hourdir}/${lday} ]; then
  mv ${hourdir}/${lday} ${daydir}
else
  echo >&2 "I can't find snapshot ${hourdir}/${lday}..."
fi
}

docleanup ()
{
  [ ${savehours} -lt 1 ] && savehours=1
  count=0
  for i in `bin/lis -r ${hourdir}`; do
    let count=count+1
    if [ ${count} -gt ${savehours} ]; then
      rm -Rf ${hourdir}/${i}
    fi
  done

  [ ${saveweeks} -gt 0 -a ${savedays} -lt 7 ] && savedays=7
  count=0
  for i in `bin/lis -r ${daydir}`; do
    let count=count+1
    if [ ${count} -gt ${savedays} ]; then
      rm -Rf ${daydir}/${i}
    fi
  done

  count=0
  for i in `bin/lis -r ${weekdir}`; do
    let count=count+1
    if [ ${count} -gt ${saveweeks} ]; then
      rm -Rf ${weekdir}/${i}
    fi
  done

  count=0
  for i in `bin/lis -r ${monthdir}`; do
    let count=count+1
    if [ ${count} -gt ${savemonths} ]; then
      rm -Rf ${monthdir}/${i}
    fi
  done

  count=0

```

```

for i in `bin/lis -r ${yeardir}`; do
    let count=count+1
    if [ ${count} -gt ${saveyears} ]; then
        rm -Rf ${yeardir}/${i}
    fi
done
}

```

```

inuse ()
{
    echo >&2 "I am already syncing snapshot ${snapshot}"
    exit 1
}

```

```

mountfail ()
{
    echo >&2 "I can't mount filesystem ${backuproot}"
    exit 1
}

```

```

umountfail ()
{
    echo >&2 "I can't unmount filesystem ${backuproot}"
    exit 1
}

```

```

####
#### Programm starts here ... #####
####

```

```

[ -f ${lockfile} ] && inuse
mounting
mkdirs
touch ${lockfile}

```

```

case ${cmdline} in
-y)
    doyearly
    docleanup
;;
-m)
    domonthly
    docleanup
;;

```

```
*)
dosync
if [ ${newdayhour} -eq ${chour} ]; then
  dodaily
  if [ ${dowweek} -eq 0 ]; then
    doweekly
  fi
  if [ ${cday} -eq 1 ]; then
    domonthly
    if [ ${cmonth} -eq 1 ]; then
      doyearly
    fi
  fi
fi
docleanup
;;
esac

touch ${backuproot}/${snapshot}/lastrun.time
rm -f ${lockfile}
rm -f /tmp/excludes-${snapshot}

umounting
echo "Backup for ${snapshot} is complete..."

/usr/bin/snapshot-report
```

Original von **snapshot**

```
#!/bin/sh
# mirror
# make a snapshot from directories
# an create hourly/daily/monthly/yearly archives
# -----
# neobiker (2006)
# originally based on mirror Version 3.11 By Stu Sheldon stu@actusa.net
# ...
# ...
# The directories are named 'snapshot-<year>-<month>-<day>-<hour>'
#
# Each time snapshot runs, it date stamps a file in the <snapshot> directory called
# 'lastrun.time'. This file's date when listing it using ls -laF shows the last
# date and time snapshot ran on that host.
#
# The last thing you need to do is add snapshot in your crontab with the proper
# times and switches.
#
# If you are going for hourly sync's, add the following to your crontab:
# 0 * * * * /usr/local/sbin/snapshot
#
# Every four hours would be:
# 0 0,4,8,12,16,20 * * * /usr/local/sbin/snapshot
#
# you get the idea
# ...

#####

# Start User Config Stuff
```

```
# This is the local root of the backup directories. (where to backup to)
backuproot=/backup

# directory, e.g. /backup/server-data
snapshot="server-data"

# You need to set the newdayhour to 00-23 depending on what hour you run
# the script in cron. Ich starte das script z.B. um 2, 18 und 23 Uhr, also
# setze ich newdayhour auf 02
# wenn chour == 02 (current hour) 02 dann Kopie/Link anlegen
newdayhour=00

# number of snapshots to save
savehours=4
savedays=7
saveweeks=4
savemonths=12
saveyears=10

# list directories to backup
allsrcdir="
/home/
/srv/home/
/srv/daten/
/srv/daten/neobiker/
/var/lib/cyrus/
/var/ftp/
/var/log/
/var/spool/
/var/www/
"
```

```
# list directories to skip during in backup
# or leave blank
# EXCLUDES=""

EXCLUDES="--filter=. /tmp/excludes-`${snapshot}`"
```

```
cat > /tmp/excludes-`${snapshot}` << EOF
- /backup/
- /snapshot/
- /srv/daten/
- /var/spool/squid/
+ *
EOF
```

```
# Mount and Dismount commands for all reasons are in the following functions
# you can also mount windows-shares via smbclient e.g.
# here: mount for writing during backup, and mount readonly afterwards
```

```
mounting ()
{
    precom=0
    mount -o remount,rw `${backuproot}` || mountfail
}
```

```
umounting ()
{
    postcom=0
    mount -o remount,ro `${backuproot}` || umountfail
}
```

```
# Typical Unix saving localhost
```

```
rsync="rsync -a -R -q --delete"
```

```
# Typical Unix using ssh and public keys
```

```
# rsync="rsync -aR -q --numeric-ids -e ssh --delete"
```

```
# This has settings for windows and expects to use a mounted share.
```

```
# rsync="rsync -a -R -q --delete --modify-window=10"
```

```
# End User Config Stuff
```

```
#####
```

```
##
```

```
# Start Static Code
```

```
cmdline=$1
```

```
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin
```

```
cmonth=`date +%m`
```

```
dowweek=`date +%w`
```

```
cdays=`date +%d`
```

```
chour=`date +%H`
```

```
yeardir=${backuproot}/${snapshot}/yearly
```

```
monthdir=${backuproot}/${snapshot}/monthly
```

```
weekdir=${backuproot}/${snapshot}/weekly
```

```
daydir=${backuproot}/${snapshot}/daily
```

```
hourdir=${backuproot}/${snapshot}/hourly
```

```
snapshotdir=`date +'snapshot-%Y-%m-%d-%H`
```

```
lockfile=${backuproot}/${snapshot}/syncing-now
```



```
lsync=`/bin/ls -1r ${hourdir} | head -1`  
lday=`date +'snapshot-%Y-%m-%d-%H' -d '1 day ago`  
lweek=`date +'snapshot-%Y-%m-%d-%H' -d '1 week ago`
```

```
makedirs ()
```

```
{  
  [ -d ${yeardir} ] || mkdir -p ${yeardir}  
  [ -d ${monthdir} ] || mkdir -p ${monthdir}  
  [ -d ${weekdir} ] || mkdir -p ${weekdir}  
  [ -d ${daydir} ] || mkdir -p ${daydir}  
  [ -d ${hourdir} ] || mkdir -p ${hourdir}  
}
```

```
dosync ()
```

```
{  
  [ -d ${hourdir}/${snapshotdir} ] || mkdir -p ${hourdir}/${snapshotdir}  
  [ -n "${lsync}" ] && rsync="${rsync} --link-dest=${hourdir}/${lsync}"  
  
  for srcdir in ${allsrcdir}; do  
    ${rsync} "${EXCLUDES}" ${srcdir} ${hourdir}/${snapshotdir}  
  done  
}
```

```
doyearly ()
```

```
{  
  [ ${saveyears} -eq 0 ] && return  
  if [ -d ${monthdir}/${snapshotdir} ]; then  
    cp -alf ${monthdir}/${snapshotdir} ${yeardir}  
  else  
    echo >&2 "I can't find snapshot ${monthdir}/${snapshotdir}..."
```

```

    fi
}

domonthly ()
{
    [ ${savemonths} -eq 0 ] && return
    if [ -d ${hourdir}/${snapshotdir} ]; then
        mv ${hourdir}/${snapshotdir} ${monthdir}
        ln -s ${monthdir}/${snapshotdir} ${hourdir}/${snapshotdir}
    else
        echo >&2 "I can't find snapshot ${hourdir}/${snapshotdir}..."
    fi
}

```

```

doweekly ()
{
    [ ${saveweeks} -eq 0 ] && return
    if [ -d ${daydir}/${lweek} ]; then
        mv ${daydir}/${lweek} ${weekdir}
    else
        echo >&2 "I can't find snapshot ${daydir}/${lweek}..."
    fi
}

```

```

dodaily ()
{
    [ ${savedays} -eq 0 ] && return
    if [ -d ${hourdir}/${lday} ]; then
        mv ${hourdir}/${lday} ${daydir}
    else
        echo >&2 "I can't find snapshot ${hourdir}/${lday}..."
    fi
}

```

```

    fi
}

docleanup ()
{
    [ ${savehours} -lt 1 ] && savehours=1
    count=0
    for i in `bin/lis -r ${hourdir}`; do
        let count=count+1
        if [ ${count} -gt ${savehours} ]; then
            rm -Rf ${hourdir}/${i}
        fi
    done

    [ ${saveweeks} -gt 0 -a ${savedays} -lt 7 ] && savedays=7
    count=0
    for i in `bin/lis -r ${daydir}`; do
        let count=count+1
        if [ ${count} -gt ${savedays} ]; then
            rm -Rf ${daydir}/${i}
        fi
    done

    count=0
    for i in `bin/lis -r ${weekdir}`; do
        let count=count+1
        if [ ${count} -gt ${saveweeks} ]; then
            rm -Rf ${weekdir}/${i}
        fi
    done
}

```

```
count=0
for i in `bin/lis -r ${monthdir}`; do
    let count=count+1
    if [ ${count} -gt ${savemonths} ]; then
        rm -Rf ${monthdir}/${i}
    fi
done
```

```
count=0
for i in `bin/lis -r ${yeardir}`; do
    let count=count+1
    if [ ${count} -gt ${saveyears} ]; then
        rm -Rf ${yeardir}/${i}
    fi
done
```

```
}
```

```
inuse ()
```

```
{
    echo >&2 "I am already syncing snapshot ${snapshot}"
    exit 1
}
```

```
mountfail ()
```

```
{
    echo >&2 "I can't mount filesystem ${backuproot}"
    exit 1
}
```

```
umountfail ()
```

```
{
```

```
echo >&2 "I can't unmount filesystem ${backuproot}"
exit 1
}
```

```
####
```

```
#### Programm starts here ... #####
```

```
####
```

```
[ -f ${lockfile} ] && inuse
```

```
mounting
```

```
makedirs
```

```
touch ${lockfile}
```

```
case ${cmdline} in
```

```
-y)
```

```
doyearly
```

```
docleanup
```

```
::
```

```
-m)
```

```
domonthly
```

```
docleanup
```

```
::
```

```
*)
```

```
dosync
```

```
if [ ${newdayhour} -eq ${chour} ]; then
```

```
do daily
```

```
if [ ${dowweek} -eq 0 ]; then
```

```
do weekly
```

```
fi
```

```
if [ ${cday} -eq 1 ]; then
```

```
do monthly
```

```
if [ ${cmonth} -eq 1 ]; then
    doyearly
fi
fi
fi
docleanup
;;
esac

touch ${backuproot}/${snapshot}/lastrun.time
rm -f ${lockfile}
rm -f /tmp/excludes-${snapshot}

umounting
echo "Backup for ${snapshot} is complete..."
```