fileserver-backup

- Tägliches inkrementelles Backup der Daten von manjaro.domain.inside nach backup.domain.srv
- Läuft täglich um 22:10 Uhr

```
#!/bin/bash

# ##############################################################################################
#
# Skript: do_fileserver_backup
#
# Zweck: Automatisiertes backup der Daten vom Fileserver
#
# Version: 1.0
#
# Benutzt den Logging Code von Michael Wayne Goodman
# http://www.goodmami.org/2011/07/simple-logging-in-bash-scripts/
#
# Benutzt den Spinner Code von William Pursell
# http://stackoverflow.com/questions/12498304/using-bash-to-display-a-progress-working-indicator
#
# Dieses Script benoetigt das Paket "msmtp"
#
# ##############################################################################################



# ###############
#
# Initialisierung
#
# ###############

# /etc/profile sourcen
source /etc/profile;

# Systemsprache explizit setzen um zu erzwingen, dass die Ausgabe der Befehle auf Englisch erfolgt.
export LANG="en_US.UTF-8";



# ###################################
#
# Konfiguration / Variablen / Strings
#
# ###################################

# Konfiguration des Skripts
FQDNFILESERVER="manjaro.domain.inside";                    # FQDN des Fileservers
FQDNBACKUPSERVER="backup.domain.srv";                      # FQDN des Backupservers
USER="root";                                               # SSH-user
BACKUPCONFIG="/etc/storebackup.d/backup-everything.conf";  # Konfigfile für das Backup
MOUNTPOINT="/mnt/fileserver";                              # Mountpunkt fuer das Filesystem der
virtuellen Maschine per SSHfs

# Strings
SUBJECTHOSTNAME="backup.domain.srv";                       # Hostname für das Subject der Mail
SUBJECTSTATEBAD="- [ERROR] - ";                            # Subject-Teilstring im Fall von Fehlern
SUBJECTSTATEGOOD="- [SUCCESS] - ";                         # Subject-Teilstring fuer erfolgreiche
Backups
ERRORSTATUSSTRING="";                                      # initialisierung der Variable
SPIN="-\\|/";                                              # Zeichen für den Spinner

# Mail Konfiguration
MSMTPCONFIG="/root/scripts/backup/msmtprc";                # Pfad zur Konfiguration von msmtp
SENDMAILIFGOOD="yes";                                      # Soll eine Mail gesendet werden wenn das
Backup erfolgreich war? [yes | no]
```

```
SENDMAILIFBAD="yes";                                    # Soll eine Mail gesendet werden wenn das
BAckup fehlgeschlagen ist? [yes | no]
MAILRECEIVER="email@gmail.com,email@domain.de";         # Empfänger der Email
MAILSENDER="email@gmail.com";                           # Absenderadresse der Email
MSMTPDEBUG="no";                                        # Setzt das debug-flag fuer msmtp

# Logging
LOGFILE="/root/scripts/backup/backup.log";              # Vollstaendiger Pfad zum Logfile
LOGVERBOSITY="2";                                       # Standard-Loglevel [0 - 4]
SILENT_LVL="0";                                         # Definiert Silent (Nicht editieren)
ERR_LVL="1";                                            # Definiert Loglevel von Error (Nicht
editieren)
WRN_LVL="2";                                            # Definiert Loglevel von Warn (Nicht
editieren)
INF_LVL="3";                                            # Definiert Loglevel von Info (Nicht
editieren)
DBG_LVL="4";                                            # Definiert Loglevel von Debug (Nicht
editieren)




# ##########
#
# Funktionen
#
# ##########

# Ein formatiertes Datum bauen
function f_makeadate ()
{
        local FORMATTEDDATE=$(date +"%Y-%m-%d %H:%M:%S")
        echo $FORMATTEDDATE
}

# Mailversand
function f_sendanemail ()
{
        if [ "$MSMTPDEBUG" != "no" ]
        then
                echo -e  $3 | msmtp --debug -C $1 $2;
        else
                echo -e  $3 | msmtp -C $1 $2;
        fi
}

# Logging
function f_notify () { f_log $SILENT_LVL "NOTE: $1"; }      # Always prints
function f_error () { f_log $ERR_LVL "ERROR: $1"; }
function f_warn () { f_log $WRN_LVL "WARNING: $1"; }
function f_inf () { f_log $INF_LVL "INFO: $1"; }            # "info" is already a command
function f_debug () { f_log $DBG_LVL "DEBUG: $1"; }
function f_log ()
{
        LOGDATE=$(f_makeadate)
        if [ $LOGVERBOSITY -ge $1 ]
        then
                # Ins Logfile schreiben
                echo -e "${LOGDATE} - $2" >> $LOGFILE;

                #Den Error-String erweitern
                if [ $1 = $ERR_LVL ]
                then
                        ERRORSTATUSSTRING+="  $2\n";
                fi
        fi

        # Die Meldung an STDOUT ausgeben
        echo "$2";
```

```
}



# Man werfe eine Nachricht
BACKUPSTARTDATE=$(f_makeadate);
f_notify "Backup-Skript gestartet ${BACKUPSTARTDATE}";



# ####################################
#
# Initiale Tests und Konfiguration
#
# ####################################

f_inf "Initiale Tests gestartet.";

# Teste SSH-Verbindung zum Fileserver
ssh -l $USER $FQDNFILESERVER 'uname -a | grep -q Linux' && f_inf "Die SSH-Verbindung zu $FQDNFILESERVER
konnte aufgebaut werden." || f_error "Mit der SSH-Verbindung zu $FQDNFILESERVER stimmt etwas nicht.";


# Check ob der Mountpoint für das Backup verwendet wird
mountpoint -q $MOUNTPOINT;
if [ $? -eq 0 ]
then
        f_error "An ${MOUNTPOINT} ist ein Filesystem gemountet.";
else
        f_inf "An ${MOUNTPOINT} ist kein Filesystem gemountet.";

        # Einhaengen des Quell-Filesystems vom Fileserver
        mount $MOUNTPOINT && f_inf "Das Quell-Filesystem konnte an ${MOUNTPOINT} gemountet werden." ||
f_error "Das Quell-Filesystem konnte nicht an ${MOUNTPOINT} gemountet werden.";
fi

f_inf "Initiale Tests und Konfiguration beendet.";



# ########################
#
# Durchfuehrung des Backups
#
# ########################

# Backup starten sofern bisher kein Fehler auftrat
if [ "$ERRORSTATUSSTRING" = "" ]
then
        f_inf "Backup-Routine gestartet.";

        # Backup fahren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                f_inf "Sicherung der Daten per storeBackup."
                # Der primäre Prozess wird in den Hintergrund geschickt, damit man einen Spinner
anzeigen kann während das Script auf der Shell läuft um eine optische Rückmeldung zu geben, dass das
Script noch laeuft.
                storeBackup -f ${BACKUPCONFIG} && f_inf "Das Backup der Daten von ${FQDNFILESERVER}
wurde auf ${FQDNBACKUPSERVER} geschrieben." || f_error "Das Backup der Daten von  ${FQDNFILESERVER}
wurde nicht auf ${FQDNBACKUPSERVER} geschrieben." &
                PID=$!
                i=0
                while kill -0 $PID 2>/dev/null;
                do
                        i=$(( (i+1) %4 ));
                        printf "\r${SPIN:$i:1}";
                        sleep .1;
                done
        fi
```

```
        # Das per sshfs eingehaengte Filesystem aushaengen.
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                umount ${MOUNTPOINT}  && f_inf "Das per sshfs auf ${FQDNBACKUPSERVER} an ${MOUNTPOINT}
eingehaengte Filesystem wurde erfolgreich ausgehaengt." || f_error "Das per sshfs auf
${BACKUPSERVERFQDN} an ${MOUNTPOINT} eingehaengte Filesystem konnte nicht erfolgreich ausgehaengt
werden.";
        fi


        f_inf "Backup-Routine beendet.";
fi

# Man werfe eine Nachricht
if [ "$ERRORSTATUSSTRING" = "" ]
then
        f_notify "Das Backup war erfolgreich.";
else
        f_error "Das Backup war nicht erfolgreich.";
fi

BACKUPENDDATE=$(f_makeadate)
f_notify "Backup-Skript beendet ${BACKUPENDDATE}"
f_notify "================================================================================================"


# ########################
#
# Status per Mail versenden
#
# ########################

# Sollen wir eine Mail senden?
if [ "$SENDMAILIFGOOD" == "yes" ] || [ "$SENDMAILIFBAD" == "yes" ]
then
        f_notify "Mailen des Status des Backups vom ${BACKUPSTARTDATE} gestartet.";
        f_inf "Konfiguration:";
        f_inf " * Mail bei Erfolg: $SENDMAILIFGOOD";
        f_inf " * Mail bei Fehler: $SENDMAILIFBAD";

        # Ermitteln des Datums
        SENDDATE=$(date -R);
        SUBJECTDATE=$(f_makeadate);

        # Bauen des Subjects abhaengig vom  Fehlerstatus
        MAILSUBJECT="[${SUBJECTHOSTNAME}]: ${SUBJECTDATE} ";
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                MAILSUBJECT+="${SUBJECTSTATEGOOD}";
        else
                MAILSUBJECT+="${SUBJECTSTATEBAD}";
        fi
        MAILSUBJECT+="Backup";

        # Bauen des E-Mail-Headers
        MAILHEADER+="To: ${MAILRECEIVER}\n";
        MAILHEADER+="From: ${MAILSENDER}\n";
        MAILHEADER+="Date: ${SENDDATE}\n";
        MAILHEADER+="Subject: ${MAILSUBJECT}\n";

        # Bauen des Mail-Bodies abhaengig vom Fehlerstatus
        MAILBODY+="\n";
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                MAILBODY+="Das Backup der Daten von ${FQDNFILESERVER} am ${SUBJECTDATE} auf
${FQDNBACKUPSERVER} war erfolgreich.\n";
        else
                MAILBODY+="Das Backup der Daten von ${FQDNFILESERVER} am ${SUBJECTDATE} auf
${FQDNBACKUPSERVER} schlug fehl.\n";
                MAILBODY+="\n";
```

```
                MAILBODY+="Folgende Fehler sind aufgetreten:\n";
MAILBODY+="=============================================================================================\n"
;
                MAILBODY+="${ERRORSTATUSSTRING}";
MAILBODY+="=============================================================================================\n"
;
        fi

        # Mailheader und Mailbody zusammenfuehren
        EMAILTEXT=${MAILHEADER}${MAILBODY};

        # Senden der E-Mail

        # TODO - Was passiert wenn die email nicht gesendet werden kann?

        f_sendanemail "$MSMTPCONFIG" "$MAILRECEIVER" "$EMAILTEXT" && f_notify "Die Email wurde
gesendet." || f_error "Die Mail konnte nicht gesendet werden.";

        f_notify "Mailen des Status des Backups vom ${BACKUPSTARTDATE} beendet.";
        f_notify
"=============================================================================";

fi
```

```
# Example for a user configuration file

# Set default values for all following accounts.
defaults
tls on
# tls_trust_file /etc/ssl/certs/ca-certificates.crt
tls_certcheck off
logfile /root/scripts/backup/msmtp.log

# gmail account
account gmail
tls on
auth on
host smtp.gmail.com
port 587
user email@gmail.com
from email@gmail.com
password ************************

# Set a default account
account default : gmail
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#

# 01 21 9,19,29 * *               root    /usr/local/bin/backup10.sh
# 01 21 5,10,15,20,25,30 * *      root    /usr/local/bin/backup5.sh
# 01 22 * * *                     root    /usr/local/bin/backup.sh
```

```
#00 6    * * *   root    storeBackup -f /etc/storebackup.d/backup-everything.conf
10 22    * * *   root    /root/scripts/backup/do_fileserver_backup
```

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    nodev,noexec,nosuid 0       0
# / was on /dev/md2 during installation
UUID=6ddfd527-89af-4b2f-97cb-945118df5cbf /               ext4    errors=remount-ro 0       1
# /boot was on /dev/md0 during installation
UUID=d3b23960-5565-4882-b9cc-92ba2ac94836 /boot           ext4    defaults        0       2
# swap was on /dev/md1 during installation
UUID=23a43ec9-2ff1-4438-a7cf-a15c7b99bc37 none            swap    sw              0       0

root@manjaro.domain.inside:/data /mnt/fileserver fuse.sshfs user=root,rw,_netdev,noauto  0 0
```

```
# configuration file for storeBackup.pl
# Generated by storeBackup.pl, 3.2, build 361

# You can set a value specified with '-cf_key' (eg. logFiles) and
# continue at the next lines which have to begin with a white space:
# logFiles = /var/log/messages  /var/log/cups/access_log
#      /var/log/cups/error_log
# One ore more white spaces are interpreted as separators.
# You can use single quotes or double quotes to group strings
# together, eg. if you have a filename with a blank in its name:
# logFiles = '/var/log/my strage log'
# will result in one filename, not in three.
# If an option should have *no value*, write:
# logFiles =
# If you want the default value, comment it:
logFile = /var/log/storeBackup/backup-everything.log
# You can also use environment variables, like $XXX or ${XXX} like in
# a shell. Single quotes will mask environment variables, while double
# quotes will not.
# You can mask $, {, }, ", ' with a backslash (\), eg. \$
# Lines beginning with a '#' or ';' are ignored (use this for comments)
#
# You can overwrite settings in the command line. You can remove
# the setting also in the command by using the --unset feature, eg.:
# '--unset doNotDelete' or '--unset --doNotDelete'


# source directory (*** must be specified ***)
sourceDir=/mnt/fileserver

# top level directory of all linked backups (*** must be specified ***)
# storeBackup must know for consistency checking where all your backups
# are. This is done to make sure that your backups are consistent if you
# used --lateLinks.
backupDir=/data/backup

# series directory, default is 'default'
# relative path from backupDir
series=everything

# ------------------------------------------------------------------------
# you do not need specify the options below to get a running configuration
# (but they give you more features and more control)
#
```

```
# directory for temporary file, default is /tmp
;tmpDir=

# List of other backup directories to consider for
# hard linking. Relative path from backupDir!
# Format (examples):
# backupSeries/2002.08.29_08.25.28 -> consider this backup
# or
# 0:backupSeries    -> last (youngest) backup in <backupDir>/backupSeries
# 1:backupSeries    -> first before last backup in <backupDir>/backupSeries
# n:backupSeries    -> n'th before last backup in <backupDir>/backupSeries
# 3-5:backupSeries  -> 3rd, 4th and 5th in <backupDir>/backupSeries
# all:backupSeries  -> all in <backupDir>/backupSeries
# This option is useful, if you want to explicitly hard link
# to backup series from different backups. You can specify eg. with
# 0:myBackup to the last backup of series 'myBackup'. If you specify
# backup series with otherBackupSeries, then only these backups will be
# used for hard linking.
# Default value is to link to the last backup of all series stored in
# 'backupDir'.
;otherBackupSeries=

# lock file, if exist, new instances will finish if
# an old is allready running, default is /tmp/storeBackup.lock
lockFile=/tmp/storeBackup-backup-everything.lock

# remove the lock files before deleting old backups
# default ('no') is to delete the lock file after deleting
# possible values are 'yes' and 'no'
;unlockBeforeDel=

# continue if one or more of the exceptional directories
# do not exist (no is stopping processing)
# default is 'no', can be 'yes' or 'no'
;contExceptDirsErr=

# Directories to exclude from the backup (relative path inside of the backup).
# You can use shell type wildcards.
# These directories have to be separated by space or newline.
;exceptDirs=

# Directories to include in the backup (relative path inside of the backup).
# You can use shell type wildcards.
# These directories have to be separated by space or newline.
;includeDirs=

# rule for excluding files / only for experienced administrators
# !!! see README file 'including / excluding files and directories'
# EXAMPLE:
# searchRule = ( '$size > &::SIZE("3M")' and '$uid eq "hjc"' ) or
#    ( '$mtime > &::DATE("3d4h")' and not '$file =~ m#/tmp/#' )'
;exceptRule=

# For explanations, see 'exceptRule'.
;includeRule=

# write a file name .storeBackup.notSaved.bz2 with the
# names of all skipped files, default is 'no', can be 'yes' or 'no'
;writeExcludeLog=

# do not save the specified types of files, allowed: Sbcfpl
# S - file is a socket
# b - file is a block special file
# c - file is a character special file
# f - file is a plain file
# p - file is a named pipe
# l - file is a symbolic link
# Spbc can only be backed up if GNU copy is available.
;exceptTypes=
```

```
# Activate this option if your system's cp is a full-featured GNU
# version. In this case you will be able to also backup several
# special file types like sockets.
# Possible values are 'yes' and 'no'. Default is 'no'
cpIsGnu=yes

# make a hard link to existing, identical symlinks in old backups
# use this, if your operating system supports this (linux does)
# Possible values are 'yes' and 'no'. Default is 'no'
;linkSymlinks=

# exec job before starting the backup, checks lockFile (-L) before
# starting (e.g. can be used for rsync) stops execution if job returns
# exit status != 0
;precommand=

# exec job after finishing the backup, but before erasing of old
# backups reports if job returns exit status != 0
;postcommand=

# follow symbolic links like directories up to depth 0 -> do not
# follow links
followLinks=1

# use this only if you write your backup over a high latency line
# like a vpn over the internet
# storebackup will use more parallelization at the cost of more
# cpu power
# possible values are 'yes' and 'no'; default is 'no'
;highLatency=

# If this option is disabled, then the files in the backup will not
# neccessarily have the same permissions and owner as the originals.
# This speeds up backups on network drives a lot. Correct permissions
# are restored by storeBackupRecover.pl no matter what this option is
# set to. Default is 'no'
;ignorePerms=

# suppress (unwanted) warnings in the log files;
# to suppress warnings, the following keys can be used:
#   excDir (suppresses the warning that excluded directories
#          do not exist)
#   fileChange (suppresses the warning that a file has changed during
#               the backup)
#   crSeries (suppresses the warning that storeBackup had to create the
#             'default' series)
#   hashCollision (suppresses the warning if a possible
#                  hash collision is detected)
#  This option can be repeated multiple times on the command line.
#  Example usage in conf file:
#  suppressWarning = excDir fileChange crSeries hashCollision
#  By default no warnings are suppressed.
;suppressWarning=

# do *not* write hard links to existing files in the backup
# during the backup (yes|no)
# you have to call the program storeBackupUpdateBackup.pl
# later on your server if you set this flag to 'yes'
# default = no: do not write hard links
;lateLinks=

# only in combination with --lateLinks
# compression from files >= size will be done later,
# the file is (temporarily) copied into the backup
# default = no: no late compression
;lateCompress=

# Files with specified suffix for which storeBackup will make an md5 check
# on blocks of that file. Executed after --checkBlocksRule(n)
;checkBlocksSuffix=
```

```
# Only check files specified in --checkBlocksSuffix if there
# file size is at least this value, default is 100M
;checkBlocksMinSize=

# Block size for files specified with --checkBlocksSuffix
# default is 1M (1 megabyte)
;checkBlocksBS=

# if set, the blocks generated due to checkBlocksSuffix are compressed
# Possible values are 'yes' and 'no'. Default is 'no'
;checkBlocksCompr=

# length of queue to store files before block checking,
# default = 1000
;queueBlock=

# Files for which storeBackup will make an md5 check depending
# on blocks of that file.
# The rules are checked from rule 1 to rule 5. The first match is used
# !!! see README file 'including / excluding files and directories'
# EXAMPLE:
# searchRule = ( '$size > &::SIZE("3M")' and '$uid eq "hjc"' ) or
#    ( '$mtime > &::DATE("3d4h")' and not '$file =~ m#/tmp/#' )'
;checkBlocksRule0=

# Block size for option checkBlocksRule
# default is 1M (1 megabyte)
;checkBlocksBS0=

# if set to 'yes', blocks generated due to this rule will be compressed
# possible values: 'yes' or 'no', default is 'no'
;checkBlocksCompr0=

# Filter for reading the file to treat as a blocked file
# eg.   gzip -d   if the file is compressed. Default is no read filter.
;checkBlocksRead0=

# Read files specified here in parallel to "normal" ones.
# This only makes sense if they are on a different disk.
# Default value is 'no'
;checkBlocksParallel0=

;checkBlocksRule1=
;checkBlocksBS1=
;checkBlocksCompr1=
;checkBlocksRead1=
;checkBlocksParallel1=

;checkBlocksRule2=
;checkBlocksBS2=
;checkBlocksCompr2=
;checkBlocksRead2=
;checkBlocksParallel2=

;checkBlocksRule3=
;checkBlocksBS3=
;checkBlocksCompr3=
;checkBlocksRead3=
;checkBlocksParallel3=

;checkBlocksRule4=
;checkBlocksBS4=
;checkBlocksCompr4=
;checkBlocksRead4=
;checkBlocksParallel4=

#  List of Devices for md5 ckeck depending on blocks of these
#  Devices
;checkDevices0=
```

```
# Directory where to store the backups of the devices
;checkDevicesDir0=

# Block size of option checkDevices0
# default is 1M (1 megabyte)
;checkDevicesBS0=

# if set, the blocks generated due to checkDevices0 are compressed
;checkDevicesCompr0=

# Read devices specified here in parallel to "normal" ones.
# This only makes sense if they are on a different disk.
# Default value is 'no'
;checkDevicesParallel0=

;checkDevices1=
;checkDevicesDir1=
;checkDevicesBS1=
;checkDevicesCompr1=
;checkDevicesParallel1=

;checkDevices2=
;checkDevicesDir2=
;checkDevicesBS2=
;checkDevicesCompr2=
;checkDevicesParallel2=

;checkDevices3=
;checkDevicesDir3=
;checkDevicesBS3=
;checkDevicesCompr3=
;checkDevicesParallel3=

;checkDevices4=
;checkDevicesDir4=
;checkDevicesBS4=
;checkDevicesCompr4=
;checkDevicesParallel4=

# write temporary dbm files in --tmpdir
# use this if you have not enough RAM, default is no
;saveRAM=

# compress command (with options), default is <bzip2>
;compress=

# uncompress command (with options), default is <bzip2 -d>
;uncompress=

# postfix to add after compression, default is <.bz2>
;postfix=

# maximal number of parallel compress operations,
# default = choosen automatically
;noCompress=

# length of queue to store files before compression,
# default = 1000
;queueCompress=

# maximal number of parallel copy operations,
# default = 1
;noCopy=

# length of queue to store files before copying,
# default = 1000
;queueCopy=

# write statistics about used space in log file
# default is 'no'
```

```
;withUserGroupStat=

# write statistics about used space in name file
#                   will be overridden each time
# if no file name is given, nothing will be written
# format is:
# identifier uid userName value
# identifier gid groupName value
;userGroupStatFile=

# do not compress files with the following
# suffix (uppercase included):
# (if you set this to '.*', no files will be compressed)
#  Default is \.zip \.bz2 \.gz \.tgz \.jpg \.gif \.tiff \.tif \.mpeg \.mpg \.mp3 \.ogg \.gpg \.png
exceptSuffix=\.zip \.bz2 \.gz \.tgz \.jpg \.gif \.tiff \.tif \.mpeg \.mpg \.mp3 \.ogg \.gpg \.png \.avi
\.divx \.flv \.mp4 \.mkv \.vidx

# like --exceptSuffix, but do not replace defaults, add
;addExceptSuffix=

# Files smaller than this size will never be compressed but always
# copied. Default is 1024
;minCompressSize=

# alternative to exceptSuffix and minCompressSize:
# definition of a rule which files will be compressed
# If this rule is set, exceptSuffix, addExceptSuffix
# and minCompressSize are ignored.
# Default rule generated from the options above is:
# comprRule = '$size > 1024' and not
#   '$file =~ /.zipZ|.bz2Z|.gzZ|.tgzZ|.jpgZ|.gifZ|.tiffZ|.tifZ|.mpegZ|.mpgZ|.mp3Z|.oggZ|.gpgZ|.pngZ/i'
;comprRule=

# default is 'no', if you do not want to compress, say 'yes'
;doNotCompressMD5File=

# permissions of .md5checkSumFile, default is 0600
;chmodMD5File=

# verbose messages, about exceptRule and includeRule
# and added files. default is 'no'
;verbose=

# generate debug messages, levels are 0 (none, default),
# 1 (some), 2 (many) messages
;debug=

# reset access time in the source directory - but this will
# change ctime (time of last modification of file status
# information
# default is 'no', if you want this, say 'yes'
;resetAtime=

# do not delete any old backup (e.g. specified via --keepAll or
# --keepWeekday) but print a message. This is for testing.
# Values are 'yes' and 'no'. Default is 'no' which means to delete.
;doNotDelete=

# delete old backups which where not finished
# this will not happen if doNotDelete is set
# Values are 'yes' and 'no'. Default is 'no' which means to delete.
;deleteNotFinishedDirs=

# keep backups which are not older than the specified amount
# of time. This is like a default value for all days in
# --keepWeekday. Begins deleting at the end of the script
# the time range has to be specified in format 'dhms', e.g.
# 10d4h means 10 days and 4 hours
# default = 30d;
# An archive flag is not possible with this parameter (see below).
```

```
keepAll=35d

# keep backups for the specified days for the specified
# amount of time. Overwrites the default values choosen in
# --keepAll. 'Mon,Wed:40d Sat:60d10m' means:
# keep backups from Mon and Wed 40days + 5mins
# keep backups from Sat 60days + 10mins
# keep backups from the rest of the days like spcified in
# --keepAll (default 30d)
# you can also set the 'archive flag'.
# 'Mon,Wed:a40d5m Sat:60d10m' means:
# keep backups from Mon and Wed 40days + 5mins + 'archive'
# keep backups from Sat 60days + 10mins
# keep backups from the rest of the days like specified in
# --keepAll (default 30d)
# If you also use the 'archive flag' it means to not
# delete the affected directories via --keepMaxNumber:
# a10d4h means 10 days and 4 hours and 'archive flag'
;keepWeekday=

# do not delete the first backup of a year
# format is timePeriod with possible 'archive flag'
;keepFirstOfYear=

# do not delete the last backup of a year
# format is timePeriod with possible 'archive flag'
;keepLastOfYear=

# do not delete the first backup of a month
# format is timePeriod with possible 'archive flag'
;keepFirstOfMonth=35d

# do not delete the last backup of a month
# format is timePeriod with possible 'archive flag'
;keepLastOfMonth=

# default: 'Sun'. This value is used for calculating
# --keepFirstOfWeek and --keepLastOfWeek
firstDayOfWeek=Mon

# do not delete the first backup of a week
# format is timePeriod with possible 'archive flag'
;keepFirstOfWeek=8d

# do not delete the last backup of a week
# format is timePeriod with possible 'archive flag'
;keepLastOfWeek=

# keep multiple backups of one day up to timePeriod
# format is timePeriod, 'archive flag' is not possible
# default is 7d
;keepDuplicate=

# Keep that miminum of backups. Multiple backups of one
# day are counted as one backup. Default is 10.
;keepMinNumber=10

# Try to keep only that maximum of backups. If you have more
# backups, the following sequence of deleting will happen:
# - delete all duplicates of a day, beginning with the old
#   once, except the oldest of every day
# - if this is not enough, delete the rest of the backups
#   beginning with the oldest, but *never* a backup with
#   the 'archive flag' or the last backup
;keepMaxNumber=35

# Alternative deletion scheme. If you use this option, all
# other keep options are ignored. Preserves backups depending
# on their *relative* age. Example:
#
```

```
#   keepRelative = 1d 7d 61d 92d
#
# will (try to) ensure that there is always
#
# - One backup between 1 day and 7 days old
# - One backup between 5 days and 2 months old
# - One backup between ~2 months and ~3 months old
#
# If there is no backup for a specified timespan (e.g. because the
# last backup was done more than 2 weeks ago) the next older backup
# will be used for this timespan.
;keepRelative =

# print progress report after each 'number' files
# Default is 0, which means no reports.
;progressReport=

# print depth of actual readed directory during backup
# default is 'no', values are 'yes' and 'no'
;printDepth=

# ignore read errors in source directory; not readable
# directories does not cause storeBackup.pl to stop processing
# Values are 'yes' and 'no'. Default is 'no' which means not
# to ignore them
;ignoreReadError=

# after a successful backup, set a symbolic link to
# that backup and delete existing older links with the
# same name
;linkToRecent=

# name of the log file (default is STDOUT)
;logFile=

# if you specify a log file with --logFile you can
# additionally print the output to STDOUT with this flag
# Values are 'yes' and 'no'. Default is 'no'.
;plusLogStdout=

# output in logfile with time: 'yes' or 'no'
# default = no
;suppressTime=

# maximal length of log file, default = 1e6
;maxFilelen=

# number of old log files, default = 5
;noOfOldFiles=

# save log files with date and time instead of deleting the
# old (with [-noOldFiles]): 'yes' or 'no', default = 'no'
;saveLogs=

# compress saved log files (e.g. with 'gzip -9')
# default is 'bzip2'
;compressWith=

# write log file (also) in the backup directory:
# 'yes' or 'no', default is 'no'
# Be aware that this log does not contain all error
# messages of the one specified with --logFile!
# Some errors are possible before the backup
# directory is created.
;logInBackupDir=

# compress the log file in the backup directory:
# 'yes' or 'no', default is 'no'
;compressLogInBackupDir=
```

```
# filename to use for writing the above log file,
# default is '.storeBackup.log'
;logInBackupDirFileName=
```