By Stef Bon
Published: 2007-08-27 14:18

An attempt to complete automatic discovery and mounting of SMB (Windows and Samba) networksharesIndex

Introduction

For some time I've been looking for a good way to let my computer discover the Windows network and mount shares on demand. After trying the kioslaves in the desktop of my choice (KDE) and the FUSE programs Fusesmb and SmbNetFs, I was not really satisfied. There had to be an way to let my computer discover all the workgroups/domains, the hosts and the shares, and mount a share on a hosts on demand.

The following issues are necessary imho:

- automatic discovery of the SMB network (domains, hosts and shares)
- building of a representation which makes sense to the users logged in in their homedirectory
- automatic mounting of shares on demand
- automatic unmounting of shares after some time unused
- use of a kernel fs like SMBFS or CIFS

- independant of a Desktop Manager

The FUSE program Fusesmb comes very near to what I'm looking for, but lacks the extended capabilities CIFS has, like the right presentation of the permissions, support of inotify, symlink and ATTR/ACL. It took me some time to find out why the FUSE program lacks these features. FuseSMB is based upon the smbclient library smbclient.so, which is part of the Samba package, and has nothiong to do with smbfs or cifs.

A few weeks ago I ran into two howto's describing on different ways how SMB shares can be mounted using autofs:

Accessing windows or samba shares using autofs

a guide to setup autofs for SMB shares, using existing files;

Integrating Your Linux Host into a Windows Environment

a guide to setup autofs to access shares in an alternative way.

Especially the latest gave me the push in the right direction.## Summary

This howto basically describes two techniques combined. Those two are:**automatic mounting of SMB shares with autofs using symlinks**

Some weeks ago I read a **hint at Cool Solutions (a Novell site)**hint which describes how with a triplet of auto files (there the author calls them auto.master, an auto.mydomain and an auto.mydomain.sub) very easily shares on a SMB host can be mounted:

The auto.master file in /etc:

```
/mydomain /etc/auto.mydomain
```

The auto.mydomain file:

```
* -fstype=autofs,-Dhost=& file=:/etc/auto.mydomain.sub
```

The auto.mydomain.sub file:

```
* ${host}:/&
*    -fstype=smbfs,workgroup=mydomain,uid=myuser,credentials=/home/myuser/.smb/mydomain ://${host}/&
```

Now when I create a symlink into the directory /mydomain:

```
install /home/mysuser/network/mydomain/hostA/

ln --symbolic /mydomain/hostA/firstshare /home/myuser/network/mydomain/hostA/firstshare
```

Now when I list the contents of share (and thus following the symlink) autofs will mount the share:

```
ls -l /home/myuser/network/mydomain/hostA/firstshare
```

will show the contents of the mounted share.

The format of the three auto files to configure autofs and the ability to use symlinks am I using here.

More information about how I've done this you'll find on the following pages:

- **Page 5** Start the automount program for a user and service
- **Page 6** Stop all the automount programs for a user.
- **Page 7** Run scripts with KDM**the automatic discovery of SMB hosts and building a cache**

Here I'm using the utility nbtscan. It finds all the SMB hosts in the network and all the services they are offering. Here this information is used to build a "networkcache". Basically it is nothing more than a directory tree of ipnumbers, containing subdirectories related to the services found. More info about that you'll find in chapter/page ... This networkcache is very important in this construction. It provides network information for several scripts/utilities here.

- Building a "global service tree". More information in **page 2**.

- Building a representation of the Windows Network in the homedirectory of connected users. See **page 3**.
- Provide information to the (auto)mounting process to prevent unnecessary mounts/actions

Here this networkcache is build with the only provider of networkinformation is the scanning with nbtscan. Other sources of informationcould very be used as well. Think of Avahi and OpenSLP. More information on **page 9**.
Futher, there are and allways have been programs which need information about the network (services available), and using their own technique to find this information and to keep it in a cache. I've the idea to maintain this information (cache) central, and let every program and user which needs it make use of it.

Description

To discover all the SMB hosts in the network I use the tool nbtscan . It's an utility, like nmblookup, which can get information about SMB hosts in the network. The mode I'm interested in is the the lookup of all the SMB hosts and their services in a networkrange.

A command like:

```
nbtscan -v -q -s : 192.168.0.0/24
```

gives information like:

```
192.168.0.1:ROUTER          :00U
192.168.0.1:ROUTER          :03U
192.168.0.1:ROUTER          :20U
192.168.0.1:ROUTER          :00U
192.168.0.1:ROUTER          :03U
192.168.0.1:ROUTER          :20U
192.168.0.1:__MSBROWSE__:01G
192.168.0.1:CWWERKGROEP         :1dU
192.168.0.1:CWWERKGROEP         :1bU
192.168.0.1:CWWERKGROEP         :1eG
192.168.0.1:CWWERKGROEP         :00G
192.168.0.1:CWWERKGROEP         :1dU
```

```
192.168.0.1:CWWERKGROEP         :1bU
192.168.0.1:CWWERKGROEP         :1eG
192.168.0.1:CWWERKGROEP         :00G
192.168.0.1:MAC:00-00-00-00-00-00
192.168.0.2:LFS20060812         :00U
192.168.0.2:LFS20060812         :03U
192.168.0.2:LFS20060812         :20U
192.168.0.2:LFS20060812         :00U
192.168.0.2:LFS20060812         :03U
192.168.0.2:LFS20060812         :20U
192.168.0.2:__MSBROWSE__:01G
192.168.0.2:BONONLINE         :1dU
192.168.0.2:BONONLINE         :1bU
192.168.0.2:BONONLINE         :1cG
192.168.0.2:BONONLINE         :1eG
192.168.0.2:BONONLINE         :00G
192.168.0.2:BONONLINE         :1dU
192.168.0.2:BONONLINE         :1bU
192.168.0.2:BONONLINE         :1cG
192.168.0.2:BONONLINE         :1eG
192.168.0.2:BONONLINE         :00G
192.168.0.2:MAC:00-00-00-00-00-00
```

Now this information is very usefull to determine the SMB hosts and the workgroups/domains they are part of. First of allit's easy to see (and to determine) that there are only two hosts, ROUTER with ipaddress 192.168.0.1 and LFS20060812 with ipaddress 192.168.0.2. The names ROUTER and LFS20060812 are netbiosnames. To determine the netbiosname given an ipaddress, I've used the OOU record.

To determine the workgroup the OOG is important. Note that the command nbtscan does not create this records, it reports only.

So this gives the following information:

Two hosts with smb services:

192.168.0.1, netbiosname ROUTER and part of workgroup/domain CWWERKGROEP
192.168.0.2, netbiosname LFS20060812 and part of workgroup/domain BONONLINE

Script

I've written a script for this:

```
#!/bin/bash
#
# scan the network for smb hosts
#
#
if [ -z "$TMPDIR" ]; then
    if [ ! -d /tmp ]; then

 exit

    else

 TMPDIR=/tmp/networkcache

    fi

fi
if [ ! -d $TMPDIR ]; then
    install --directory $TMPDIR

fi
if [ -z "$NBTSCAN_COMMAND" ]; then
    # try to find the nbtscan command

    NBTSCAN_COMMAND=$(which nbtscan 2>>/dev/null)

    if [ -z "$NBTSCAN_COMMAND" ]; then
```

```
 do_log "The program nbtscan not found. This program is necessary for the scanning"
 do_log "of the network for smb hosts. Cannot continue."
 exit

    fi

fi
if [ ! -x "$NBTSCAN_COMMAND" ]; then
    do_log "The command $NBTSCAN_COMMAND is not executable. This program is necessary for the scanning"
    do_log "of the network for smb hosts. Cannot continue."
    exit

fi
if [ -z "$BASE_NETWORK_CACHE_DIR" ]; then
    if [ ! -f /etc/networkcache/networkcache.conf ]; then
 do_log "The file /etc/networkcache/networkcache.conf is not found."
 exit

    else
 source /etc/networkcache/networkcache.conf

 if [ -z "$BASE_NETWORK_CACHE_DIR" ]; then

     do_log "The variable BASE_NETWORK_CACHE_DIR is not set."
     exit

 fi

    fi

fi

if [ ! -d $BASE_NETWORK_CACHE_DIR/by-ip/ipv4 ]; then
```

```
 install --mode=755 --directory $BASE_NETWORK_CACHE_DIR/by-ip/ipv4

fi
if [ ! -d $BASE_NETWORK_CACHE_DIR/by-service/smb ]; then

 install --mode=755 --directory $BASE_NETWORK_CACHE_DIR/by-service/smb
fi
#
# determine the networks connected to this machine
#
ip route show scope link | awk '{ print $1 }' > $TMPDIR/localhost.iprange

for iprange in $(cat $TMPDIR/localhost.iprange ); do
    network=$( echo $iprange | cut -d "/" -f 1)

    if [ -f $TMPDIR/nbtscan.raw.$network.output ]; then

 mv --force $TMPDIR/nbtscan.raw.$network.output $TMPDIR/nbtscan.raw.$network.output.old

    else

 # create an empty old browselist

 touch $TMPDIR/nbtscan.raw.$network.output.old

    fi

    $NBTSCAN_COMMAND -v -q -s : $iprange > $TMPDIR/nbtscan.raw.$network.output

    if [ -f $TMPDIR/nbtscan.ipv4.$network.list ]; then

 mv -f $TMPDIR/nbtscan.ipv4.$network.list $TMPDIR/nbtscan.ipv4.$network.list.old
```

```
  else

touch $TMPDIR/nbtscan.ipv4.$network.list.old

  fi
  cat $TMPDIR/nbtscan.raw.$network.output | cut -d ":" -f 1 | sort --unique > $TMPDIR/nbtscan.ipv4.$network.list

  for ipv4_host in $(cat $TMPDIR/nbtscan.ipv4.$network.list); do

if [ -n "$ipv4_host" ]; then

   #
   # get the netbiosname of host: look for record with the 00U value
   #
   # transform everything to *case
   #
   # get the 00U record

      netbiosname_host=$(cat $TMPDIR/nbtscan.raw.$network.output | grep --max-count 1 "^$ipv4_host:.*:00U" | tr --complement --delete "[:graph:]" | cut --delimiter ":" --fields 2 | tr "[:lower:]" "[:upper:]")

   if [ -n "$netbiosname_host" ]; then


 #
 # get the workgroupname of host: look for record with the 00G value
 #
    # transform everything to *case
 #
 # get the 00G record

 workgroup_host=$(cat $TMPDIR/nbtscan.raw.$network.output | grep --max-count 1 "^$ipv4_host:.*:00G" | tr --complement --delete "[:graph:]" | cut --delimiter ":" --fields 2 | tr "[:lower:]" "[:upper:]")
```



```
 if [ ! -d $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host ]; then

    install --mode=755 --directory $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host

 fi

 if [ ! -d $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb ]; then

    install --mode=755 --d $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb
 fi
 touch $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb/config

 echo "workgroup=$workgroup_host" > $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb/config
 echo "netbiosname=$netbiosname_host" >> $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb/config
 echo "ipnumber=$ipv4_host" >> $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/$ipv4_host/smb/config

   fi
fi
  done
  for fipv4_host in $BASE_NETWORK_CACHE_DIR/by-ip/ipv4/*; do
if [ -d $fipv4_host/smb ]; then
   ipv4_host=$(basename $fipv4_host)
   if [ -z "$(cat $TMPDIR/nbtscan.ipv4.$network.list | grep -w $ipv4_host)" ]; then
 # this ipnumber is not found anymore in the smbbrowse list
 #
 # remove the service smb from this host
 rm --recursive --force  $fipv4_host/smb
   fi
fi
  done

done
```

Notes

First it determines all the networks this hosts is connected to. This is necessary, because nbtscan cannot scan more than one network a time. Further, for all the hosts found, it will check the OOU and the OOG records only. With this information it builds a directory tree:

```
BASE_NETWORK_CACHE_DIR/by-ip/ipv4/
                                192.168.0.1/smb
                                              config
                                192.168.0.2/smb
                                              config
```

The script does add an ipnumber/smb directory when a new host is found, and delete only the smb directory when a host is in the directory, but not detected (anymore) by the nbtscan command.

The config file contains important information like the workgroup, the netbiosname and the ipnumber. The lastest is redundant, but I can not avoid this.

Note that this script uses a configurationfile(here /etc/networkcache.conf), which is used to set the directory where this networkinformation is kept (BASE_NETWORK_CACHE_DIR).

Not all SMB hosts are detected

I've got a LinkPro Printserver which is not detected. For now I do not have a sollution for that found. I do not know any other tool which detects all SMB hosts on a subnet.

Organisation of the various scripts

As already said above there are scripts which provide system-wide information, and script meant for the usersessions. Futher this construction is build with the service SMB: make information about the SMB network available and make mounting of the shares on demand for users possible. But this can also work with other services like FTP. In building this I've tried to concentrate on the organisation of the various scripts and configuration files to make it easy to understand and to extend.***Organisation of files***

Name of script: ***scan_network_with_nbtscan.sh***

Description: scan the SMB network for hosts and the services they are offering

Directory: ***/etc/networkcache/service/smb***

How called: link ***/etc/networkcache/run.d/system/smb-10.sh --> ../../service/smb/scan_network_with_nbtscan.sh***

Runs when a script - which is on his turn run periodically by fcron - does run all the script found in the /etc/networkcache/run.d/system.

Type: system related

Description

I assume the networkcache is build and available. It has the following format:

```
BASE_NETWORK_CACHE_DIR/by-ip/ipv4/
                                  192.168.0.1/smb
                                                config
                                  192.168.0.2/smb
                                                config
```

This information is only not very usable for the user. They know it in the following form:

```
BASE_NETWORK_CACHE_DIR/by-service/smb/
                                  CWWERKGROEP
                                             ROUTER
                                  BONONLINE
                                           LFS20060812
```

This "service" tree represents the data, stored in the "ip" tree a different way, and does not add any information or change it. The netbiosnames (ROUTER and LFS20060812) are symlinks to the directories 192.168.0.1/smb and 192.168.0.2/smb.

The following script keeps the smb tree synchronisched with the ip tree:***Script***


```
#!/bin/bash
#
# copy the ip tree to a smb service tree
#
#

if [ -z "$TMPDIR" ]; then
    if [ ! -d /tmp ]; then

 exit

    else

 TMPDIR=/tmp/networkcache

    fi

fi
if [ ! -d $TMPDIR ]; then
    install --directory $TMPDIR

fi
if [ -z "$BASE_NETWORKCACHE_DIR" ]; then
    if [ ! -f /etc/networkcache.conf ]; then
 do_log "The file /etc/networkcache.conf is not found."
 exit

    else
 source /etc/networkcache.conf

 if [ -z "$BASE_NETWORKCACHE_DIR" ]; then

     do_log "The variable BASE_NETWORK_CACHE_DIR is not set."
```

```
    exit

 fi

   fi

fi

if [ ! -d $BASE_NETWORKCACHE_DIR/by-service/smb ]; then

   install --mode=755 --directory $BASE_NETWORKCACHE_DIR/by-service/smb
fi



# check all hosts
cd $BASE_NETWORKCACHE_DIR

# all ipv4 hosts with a smb service are checked
# this tree is based on the ipnumber v4
#
# also the tree based on the service smb is build
# next is to check this tree
find by-ip/ipv4 -maxdepth 2 -type d -name smb | (
   while read fsmb; do

 if [ -f $fsmb/config ]; then

    source $fsmb/config

    if [ -n "$workgroup" ]; then
  if [ ! -d $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup ]; then
     install --mode=755 --directory $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup
  fi
```

```
  if [ -h $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup/$netbiosname ]; then
     rm --force $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup/$netbiosname

  fi

  ln --symbolic --force ../../../$fsmb $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup/$netbiosname
    fi

 fi

   done
)
#
# first thing is to delete the netbiosnames which are not
# in the ip tree anymore
cd $BASE_NETWORKCACHE_DIR/by-service/smb
for fworkgroup_host in $BASE_NETWORKCACHE_DIR/by-service/smb/*; do
   workgroup_host=$(basename $fworkgroup_host)
   for fnetbiosname_host in $fworkgroup_host/*; do
 netbiosname_host=$(basename $fnetbiosname_host)
 if [ -h $fnetbiosname_host ]; then
    if [ ! -d $fnetbiosname_host ]; then

  # symbolic link exist: target does not exist

     rm $fnetbiosname_host
    else

  if [ ! $(cat $fnetbiosname_host/config | grep workgroup | sed "s@^workgroup.*=@@") = $workgroup_host ]; then

     # host does not belong to this workgroup
```

```
    rm $fnetbiosname_host

 fi

   fi

else

   # something wrong : not a link
   rm $fnetbiosname_host
fi
  done


  if [ $(ls -A $fworkgroup_host | wc --words) -eq 0 ]; then

# no host anymore in this workgroup

 rmdir  $fworkgroup_host

  fi

done
```

Organisation of files

Name of script: ***create_global_smb_tree.sh***

Description: build from the information in the ipbased tree/cache a smb based tree/cache

Directory: ***/etc/networkcache/service/smb***

How called: link ***/etc/networkcache/run.d/system/smb-20.sh --> ../../service/smb/create_global_smb_tree.sh***

Runs when a script - which is on his turn run periodically by fcron - does run all the scripts found in the /etc/networkcache/run.d/system.

Type: system related

Description

The next thing is to find all shares available on a SMB host. This has to be done on a per user basis. On the server is configured which share a user has access to. So in fact you cannot speak of "the shares available". Better is: the shares a user has access to.

Determing the shares available on a host for user sbon is simple with the utility ***smbclient***:

```
smbclient -g -L LFS20060812 -A /home/sbon/.autofssession/mount.smb.cred 2>>/dev/null
```

```
Disk|ftp|Sources Linux
Disk|bononline|HTML files
Disk|public|Public share
IPC|IPC$|IPC Service (Linux Samba 3.0.25a server)
Disk|sbon|Networkfolder of IPC_
Server|LFS20060812|Linux Samba 3.0.25a server
Workgroup|BONONLINE|LFS20060812
Workgroup|CWWERKGROEP|ROUTER
```

Without credentials it's possible to configure ***smbclient*** to act as guest:

```
smbclient -g -L LFS20060812 -N 2>>/dev/null
```

```
Anonymous login successful
Disk|ftp|Sources Linux
Disk|bononline|HTML files
Disk|public|Public share
IPC|IPC$|IPC Service (Linux Samba 3.0.25a server)
```

```
Anonymous login successful
Server|LFS20060812|Linux Samba 3.0.25a server
Workgroup|BONONLINE|LFS20060812
Workgroup|CWWERKGROEP|ROUTER
```

By taking only the lines starting with Disk, and selecting the second field, (with the | as seperator) gives the shares available to user sbon:

```
smbclient -g -L LFS20060812 -A /home/sbon/.autofssession/mount.smb.cred 2>>/dev/null | grep "^Disk" | cut -d "|" -f 2
```

```
ftp
bononline
public
sbon
```

This information is necessary when building a "Windows network" tree in the homedirectory of the user sbon, when he is logged in. This is what the first part of following script does. First, it stores the available shares for this user in the file ***shares.list*** in a subdirectory in the SMB service tree:

```
BASE_NETWORKCACHE_DIR/by-service/smb/
                                  BONONLINE
                                     LFS20060812
                                        shares
                                           sbon
                                              shares.list
                                  CWWERKGROEP
                                     ROUTER
                                        shares
                                           sbon
                                              shares.list
```

The shares available to another user (mbon for example) are stored in another subdirectory ***shares/mbon/***.

Now all the domains/workgroups, servers and shares for a user are discovered, the next thing that can be done is the building of a representation of the

"Windows network neighbourhood" in the homedirectory of the user sbon. This may look like:

```
/home/sbon/Global Network/Windows Network/
                                        BONONLINE
                                           LFS20060812
                                              bononline
                                              ftp
                                              public
                                              sbon
                                        CWWERKGROEP
                                           ROUTER
                                              ftp
                                              public
                                              sbon
```

The following script does that:

```
#!/bin/bash
ICON_WORKGROUP="smb4k"
ICON_NETWORK="network_local"
ICON_SERVER="server"
userid=$1
if [ -z "$TMPDIR" ]; then
   if [ ! -d /tmp ]; then

 exit

   else

 TMPDIR=/tmp/networkcache

   fi
```

```
fi
if [ ! -d $TMPDIR ]; then
   install --directory $TMPDIR

fi
if [ ! -f /etc/networkcache.conf ]; then
   do_log "The file /etc/networkcache.conf is not found."
   exit

else
   source /etc/networkcache.conf

   if [ -z "$BASE_NETWORKCACHE_DIR" ]; then

 do_log "The variable BASE_NETWORKCACHE_DIR is not set."
 exit

   fi

fi
if [ ! -f /etc/autofs.usersession.conf ]; then
   do_log "Configuration file autofs.usersession.conf not found."
   exit

else
   source /etc/autofs.usersession.conf

   if [ -z "$BASE_AFSUS_CONFIG_DIR" ]; then

 do_log "The variable BASE_AFSUS_CONFIG_DIR is not set."
 exit

   elif [ -z "$BASE_AFSUS_FILES_DIR" ]; then
```



```
 do_log "The variable BASE_AFSUS_FILES_DIR is not set."
 exit

   elif [ ! -f $BASE_AFSUS_CONFIG_DIR/service/smb/usersession.conf ]; then

 do_log "The configurationfile usersession.conf not found."
 exit

   fi

   source $BASE_AFSUS_CONFIG_DIR/service/smb/usersession.conf

   GLOBAL_NETWORK_NAME=${GLOBAL_NETWORK_NAME:-"Global Network"}

   SMB_NETWORK_NAME=${SMB_NETWORK_NAME:-"Windows Network"}

fi
if [ -n "$userid" ]; then
   useridlist="$userid"


else
   useridlist=""

   for fuserid in $BASE_AFSUS_FILES_DIR/*; do

 if [ -f $fuserid/smb/auto.master ]; then

    userid=$(basename $fuserid)
    if [ -z "$useridlist" ]; then
  useridlist="$userid"
    else
```

```
 useridlist="$useridlist $userid"

    fi
fi

  done

fi
for userid in $useridlist; do
  userproperties=$(getent passwd | grep --max-count 1 -E "^$userid:")

  if [ -n "$userproperties" ]; then

homedir=$(echo $userproperties | cut -d ":" -f 6)
gidnr=$(echo $userproperties | cut -d ":" -f 4)
uidnr=$(echo $userproperties | cut -d ":" -f 3)

  else
do_log "Something strange: userproperties for user $userid not found."

continue

  fi
  # the network is scanned and the all the hosts and workgroups are noted
  #
  # next is to find the shares per netbioshost
  # on a per user basis
  # important to note is that all relevant names are names used in the
  # smb service (netbiosnames) and not hostnames
  for fworkgroup in $BASE_NETWORKCACHE_DIR/by-service/smb/*; do
workgroup=$(basename $fworkgroup)
for fnetbiosname in $fworkgroup/*; do
   netbiosname=$(basename $fnetbiosname)
```



```
  if [ -f $fnetbiosname/config ]; then

ipnumber=$(cat $fnetbiosname/config | grep "^ip.*=" | sed "s@^ip.*=@@" )

if [ ! -d $fnetbiosname/shares/$userid ]; then

   install --mode=700 --owner=$uidnr --group=$gidnr --directory $fnetbiosname/shares/$userid
fi

# find all the shares available for this user
#
# check for credentials
#

if [ -f $homedir/.autofssession/mount.smb.cred ]; then

   smbclient -g -L $netbiosname -A $homedir/.autofssession/mount.smb.cred 2>>/dev/null | grep "^Disk" | cut -d "|" -f 2 > $fnetbiosname/shares/$userid/shares.list

else

   smbclient -g -N -L $netbiosname 2>>/dev/null | grep "^Disk" | cut -d "|" -f 2 > $fnetbiosname/shares/$userid/shares.list

fi

# create the directories representing the shares

   for share in $(cat $fnetbiosname/shares/$userid/shares.list | tr "[:upper:]" "[:lower:]" ); do

   if [ -n "$share" ]; then

 if [ ! -d "$fnetbiosname/shares/$userid/$share" ]; then
    install --mode=0755 --directory "$fnetbiosname/shares/$userid/$share"
```



```
 fi

   fi

done

for fshare in "$fnetbiosname/shares/$userid/*"; do

   if [ -d "$fshare" ]; then

 share=$(basename "$fshare")

 if [ -z "$(cat $fnetbiosname/shares/$userid/shares.list | grep --word-regexp $share)" ]; then

    # the share does not exist any more

    rm --force --recursive $fshare

 fi

   fi

done
  fi

  # create the networkfolders in the homedirectory of this user
  # note that the quotes are necessary

  if [ ! -d "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME" ]; then

install --mode=755 --owner=$uidnr --group=$gidnr --directory "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME"

  fi
```



```
  cd "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME"

  if [ ! -d $workgroup ]; then

install --mode=755 --owner=$uidnr --group=$gidnr --directory $workgroup
  fi
  if [ ! -f $workgroup/.directory -a -n "$ICON_WORKGROUP" ]; then

touch $workgroup/.directory
chown $uidnr:$gidnr $workgroup/.directory
echo "[Desktop Entry]" > $workgroup/.directory
echo "Icon=$ICON_WORKGROUP" >> $workgroup/.directory
  fi

  if [ ! -d $workgroup/$netbiosname ]; then
install --mode=755 --owner=$uidnr --group=$gidnr --directory $workgroup/$netbiosname
  fi

  if [ ! -f $workgroup/$netbiosname/.directory -a -n "$ICON_SERVER" ]; then

touch $workgroup/$netbiosname/.directory
chown $uidnr:$gidnr $workgroup/$netbiosname/.directory

echo "[Desktop Entry]" > $workgroup/$netbiosname/.directory
echo "Icon=$ICON_SERVER" >> $workgroup/$netbiosname/.directory
  fi

  # check the base autofs dir exist
  # if not it is of no use to let the user shares point to this directory
  # it's just a check, this directory should exist
  if [ -d $BASE_AFSUS_FILES_DIR/$userid ]; then
```

```
 # create symbolic links to the autofs part

 if [ -d $fnetbiosname/shares/$userid -a $(ls -A $fnetbiosname/shares/$userid/ 2>>/dev/null | wc --words) -gt 0 ]; then
    for fshare in $fnetbiosname/shares/$userid/*; do

  if [ -d $fshare ]; then
     share=$(basename $fshare)

     # check the link does exist AND points to an existing share
     if [ ! -h "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME/$workgroup/$netbiosname/$share" ]; then
   #
   # this is what it is all about!!!
   # create a link to the autofs part
   # when the user enters this directory autofs will handle the rest....

   ln --symbolic --force $BASE_AFSUS_MOUNT_DIR/user/$userid/smb/$netbiosname/$share "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME/$workgroup/$netbiosname/$share"

     fi

  fi

    done
 fi

   fi
done

  done

  pwd_keep="$PWD"

  cd "$homedir/$GLOBAL_NETWORK_NAME/$SMB_NETWORK_NAME"
  for fworkgroup in ./*; do
```



```
workgroup=$(basename "$fworkgroup")
if [ -d $fworkgroup ]; then
   for fnetbiosname in $fworkgroup/*; do
 if [ -d "$fnetbiosname" ]; then
    netbiosname=$(basename $fnetbiosname)
    for fshare in $fnetbiosname/*; do
     share=$(basename $fshare)
     if [ ! -d $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup/$netbiosname/shares/$userid/$share ]; then

      if [ -z "$(mount -t cifs | grep --word-regexp $BASE_AFSUS_MOUNT_DIR/user/$userid/smb/$netbiosname/$share)" ]; then
   find $fshare -xdev -delete
      fi
     fi
    done
    if [ $(find "$fnetbiosname" -maxdepth 1 -type l | wc --lines) -eq 0 ]; then
  if [ ! -d $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup/$netbiosname ]; then
    if [ -z "$(mount -t cifs | grep --word-regexp $BASE_AFSUS_MOUNT_DIR/user/$userid/smb/$netbiosname)" ]; then
```



```
  find $fnetbiosname -xdev -delete
      fi
 fi
    fi
fi
  done
  if [ $(find "$fworkgroup" -maxdepth 1 -mindepth 1 -type d | wc --lines) -eq 0 ]; then
if [ ! -d $BASE_NETWORKCACHE_DIR/by-service/smb/$workgroup ]; then
    find $fworkgroup -xdev -delete
fi
   fi
fi
  done
  cd "$pwd_keep"

done
```

Organisation of files

Name of script: ***create_user_smb_tree.sh***

Description: create a tree in the homedirectory of the user which represents the SMB network to make browsing possible. The shares are symlinks to the automount part to allow mounting of the shares on demand

Directory: ***/etc/networkcache/service/smb***

How called: link ***/etc/networkcache/run.d/session/smb-10.sh --> /etc/networkcache/service/smb/create_user_smb_tree.sh***

First called by ***start_smb_usersession.sh*** which runs every scripts it will find in the ***/etc/networkcache/run.d/session*** directory.

Secondly called by the script - run periodically by fcron - which runs every script it will find in the ***/etc/networkcache/run.d/session*** and the ***/etc/networkcache/run.d/system*** directories.

Type: session related

Description

As I've already described in the introduction, it is very easy/possible to mount SMB shares using a three files, auto.master, auto.hosts and auto.share.

So first the three auto. files are necessary. In my case, being user sbon, the following directories are used:

```
base directory where the three auto. files for this user and this service will go: /var/run/autofs/user/sbon/smb
base directory where the automount program will do the actual mount: /mnt/autofs/user/sbon/smb
base directory where the networkinformation for the SMB service is kept: /var/lib/network/cache/by-service/smb
```

The auto. files (in /var/run/autofs/user/sbon/smb) looks like:

```
/mnt/autofs/user/sbon/smb/    /var/run/autofs/user/sbon/smb/auto.hosts
```

The auto.hosts looks like:

```
#!/bin/bash
host=$(echo "$1" | sed 's@^/*@@' )
pwd_keep=$PWD
cd /var/lib/network/cache/by-service/smb
hostsfound=$(find . -maxdepth 2 -mindepth 2 -name $host)
for fhost in ${hostsfound}; do
  if [ -d $fhost ]; then
    cd $fhost
    break
  fi
done
if [ -d shares ]; then
  echo "-fstype=autofs,-Dhost=${host} file:/var/run/autofs/user/sbon/smb/auto.share"
fi
cd $pwd_keep
```

An the auto.share file:

```
* ${host}:/&
*    -fstype=cifs,credentials=/home/sbon/.autofssession/mount.smb.cred ://${host}/&
```

The auto.hosts does a simple lookup (in the smbcache) the host does exist and has shares. In the construction here this is not really necessary, the user does not access this directory directly, only through symlinks pointing to existing hosts and shares.

Futher the auto.master is not really necessary. This file is not needed by the automount program, but only by the (init)script which starts the automount program for the various mountpoints.

Now the next thing is to determine and to create the directory for this user and service. The directory

/mnt/autofs/user/sbon/smb

seems to be a good choice.

The next thing is to start the automount program:

```
automount --pid-file /var/run/autofs.mnt-autofs-user-sbon-smb.pid --timeout 300 \
   /mnt/autofs/user/sbon/smb program \
   /var/run/autofs/user/sbon/smb/auto.hosts
```

This starts the automount program which will mount SMB shares on Samba or Windows hosts, using the CIFS filessytem. Now after creatinga SMB representation in the homedirectory of user sbon, where the shares are symlinks pointing to virtual/autofs mountpoints, usersbon can browse the SMB network and access shares whenever he wants. ***Generalising: use of templates***

As you can see, the auto.* files do depend on the service (SMB), the filesystem(CIFS) and the user(sbon). If for example another user (mbon) do want to make use of this construction, the auto.* files will look different. This also counts if in stead of CIFS SMBFS is used. Futher, I've tried to make this construction usable when extendeing it with another service, like FTP, Novell Netware or SSH.

To still make this construction user and service independent, I'v chosen the use of templates. All data which do depend on the user or the configuration are substituted in the files the moment the session starts. The template for the auto.master file (auto.master.tmpl) looks for example like:

```
%BASE_AFSUS_MOUNT_DIR%/user/%USER%/smb %BASE_AFSUS_FILES_DIR%/%USER%/smb/auto.hosts
```

And the auto.hosts.tmpl:

```
#!/bin/bash
host=$(echo "$1" | sed 's@^/*@@' )
pwd_keep=$PWD
```

```
cd %BASE_NETWORKCACHE_DIR%/by-service/smb
hostsfound=$(find . -maxdepth 2 -mindepth 2 -name $host)
for fhost in ${hostsfound}; do
  if [ -d $fhost ]; then
    cd $fhost
    break
  fi
done
if [ -d shares ]; then
  echo "-fstype=autofs,-Dhost=${host} file:%BASE_AFSUS_FILES_DIR%/%USER%/smb/auto.share"
fi
cd $pwd_keep
```

To create the right auto.share file, I've used two auto.share.tmpl. One to use if credentials are available, one to do without:

```
* ${host}:/&
*    -fstype=cifs,credentials=%HOMEDIR%/.autofssession/mount.smb.cred ://${host}/&
```

and

```
* ${host}:/&
*    -fstype=cifs,guest ://${host}/&
```

The templates are stored in the directory /usr/share/autofs/usersession/filesystem/cifs/.***Script***

The following script does all this:
- read the settings from various configurationfiles
- create the directory where autofs will mount the shares for this user if it does not exist
- check the automount process is already running

- copy the right templates to the configurationdirectory for this user and service
- substitute the right values in the templates
- build a SMB tree in the homedirectory
- start the automount program

```
#!/bin/bash
. /etc/session.d/scripts/misc/loganddebug.functions
copy_and_substitute_autofs_settings()
{
local ltmplfile=$1
local lautofile=$2
if [ ! -f $BASE_AFSUS_TMPL_DIR/$FILESYSTEM/$ltmplfile ]; then
    do_log "Template $ltmplfile not found."

    exit

else
    cp --force $BASE_AFSUS_TMPL_DIR/$FILESYSTEM/$ltmplfile $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile
    sed -i -e "s@%USER%@$userid@g" $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile
    sed -i -e "s@%BASE_AFSUS_MOUNT_DIR%@$BASE_AFSUS_MOUNT_DIR@g" $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile
    sed -i -e "s@%HOMEDIR%@$homedir@g" $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile
    sed -i -e "s@%BASE_NETWORKCACHE_DIR%@$BASE_NETWORKCACHE_DIR@g" $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile
    sed -i -e "s@%BASE_AFSUS_FILES_DIR%@$BASE_AFSUS_FILES_DIR@g" $BASE_AFSUS_FILES_DIR/$userid/smb/$lautofile

fi
}
start_usersession_automount()
{
local ldir="$1"
local ltype=""
local lmap="$2"
local lpidfile="$3"
if [ ! -d "$ldir" ]; then
```



```
    do_log " Creating mountpoint $ldir..."
    install --mode=0755 --directory "$ldir"
fi
if [ ! -e "$lmap" ]; then

    do_log "Map file $lmap does not exist!"
    return

elif [ -x "$lmap" ]; then
    ltype="program"

else
    ltype="file"
fi
do_log "Starting automount program on $ldir."
$AUTOMOUNT_PROGRAM --pid-file $lpidfile --timeout 300 $ldir $ltype $lmap
}
userid=$1
service=$2
logpriority=$3
userproperties=$(getent passwd | grep -m 1 -E "^$userid:")
homedir=$(echo $userproperties | cut -d ":" -f 6)
gidnr=$(echo $userproperties | cut -d ":" -f 4)
uidnr=$(echo $userproperties | cut -d ":" -f 3)
if [ -z "$userproperties" ]; then
    # something wrong : the basic properties for this user are not found!
    exit 0
fi
if [ -z "$logpriority" ]; then
    if [ -f $homedir/.xlogpriority ]; then

 logpriority=$(cat $homedir/.xlogpriority)
```

```
    else
        logpriority="local3.info"

    fi

fi
if [ -z "$TMPDIR" ]; then
    if [ ! -d /tmp ]; then

 exit

    else

 TMPDIR=/tmp

    fi

fi
if [ -f /etc/sysconfig/autofs.conf ]; then
    source  /etc/sysconfig/autofs.conf

fi
if [ -n "$automount" ]; then
    AUTOMOUNT_PROGRAM=$(which $automount 2>>/dev/null)

else
    AUTOMOUNT_PROGRAM=$(which automount 2>>/dev/null)

fi
if [ -z "$AUTOMOUNT_PROGRAM" ]; then
    do_log "Automount program not found."
    exit
```

```
fi
PIDROOT_NAME=${pidroot:-"autofs"}
PIDROOT_DIR=${piddir:-"/var/run"}

if [ ! -f /etc/networkcache.conf ]; then
    do_log "The configuration file networkcache.conf not found."
    exit

else
    source /etc/networkcache.conf

    # check some variables

    if [ -z "$BASE_NETWORKCACHE_DIR" ]; then

 do_log "The variable BASE_NETWORKCACHE_DIR is not set."
 exit

    elif [ ! -d "$BASE_NETWORKCACHE_DIR" ]; then

 install --mode=0755 --directory $BASE_NETWORKCACHE_DIR

    fi

fi
if [ ! -f /etc/autofs.usersession.conf ]; then
    do_log "The configuration file autofs.usersession.conf not found."
    exit

else
    source /etc/autofs.usersession.conf

    # check some variables
```

```
  if [ -z "$BASE_AFSUS_MOUNT_DIR" ]; then

do_log "The variable BASE_AFSUS_MOUNT_DIR is not set."
exit

  elif [ ! -d $BASE_AFSUS_MOUNT_DIR ]; then

install --mode=0755 --directory $BASE_AFSUS_MOUNT_DIR

  fi
  if [ -z "$BASE_AFSUS_CONFIG_DIR" ]; then

do_log "The variable BASE_AFSUS_CONFIG_DIR is not set."
exit

  elif [ ! -f $BASE_AFSUS_CONFIG_DIR/service/smb/usersession.conf ]; then

do_log "The file usersession for this service not found."
exit

  else

source $BASE_AFSUS_CONFIG_DIR/service/smb/usersession.conf

if [ -z "$FILESYSTEM" ]; then

    do_log "Filesystem for this service not set."
    exit

elif [ -z '$(cat /proc/filesystems | grep "nodev.*$FILESYSTEM\$")' ]; then
    do_log "Filesystem $FILESYSTEM not supported."
    exit
```

```
 fi

   fi

   if [ -z "$BASE_AFSUS_FILES_DIR" ]; then

 do_log "The variable BASE_AFSUS_FILES_DIR is not set."
 exit

   fi

   if [ -z "$BASE_AFSUS_TMPL_DIR" ]; then

 do_log "The variable BASE_AFSUS_TMPL_DIR is not set."
 exit

   elif [ ! -d $BASE_AFSUS_TMPL_DIR ]; then

 do_log "The directory $BASE_AFSUS_TMPL_DIR does not exist and it should."
 exit

   fi
fi
# create a folder where the autofs.master etcetera go
if [ ! -d $BASE_AFSUS_FILES_DIR/$userid/smb ]; then

   install --mode=0755 --directory $BASE_AFSUS_FILES_DIR/$userid/smb
fi
# create teh folder where autofs will do the mounts for this user and this service (smb)
if [ ! -d $BASE_AFSUS_MOUNT_DIR/user/$userid/smb ]; then

   install --mode=0755 --directory $BASE_AFSUS_MOUNT_DIR/user/$userid/smb
```



```
fi
PID_FILE=$(echo $BASE_AFSUS_MOUNT_DIR/user/$userid/smb | sed -e "y/\//-/" | sed -e "s/^-//")
PID_FILE=${PIDROOT_DIR}/${PIDROOT_NAME}.${PID_FILE}.pid
# check the autofs for this user and service is not already running
if [ -f $PID_FILE ]; then
    # rely completely on the (non) existence of the pidfile the service is already running or not
    do_log "Already running: pidfile $PID_FILE does exist."
    exit

fi
# copy the right template to the config directory
copy_and_substitute_autofs_settings auto.master.tmpl auto.master
copy_and_substitute_autofs_settings auto.hosts.tmpl auto.hosts
if [ -f $homedir/.autofssession/mount.smb.cred ]; then
    copy_and_substitute_autofs_settings auto.share.cred.tmpl auto.share

else

    copy_and_substitute_autofs_settings auto.share.guest.tmpl auto.share

fi
if [ -d $BASE_NETWORKCACHE_CONF_DIR/run.d/session ]; then
    # this runs every smb script found in the session directory
    if [ $(ls -A $BASE_NETWORKCACHE_CONF_DIR/run.d/session/smb-*.sh 2>>/dev/null | wc --words) -gt 0 ]; then
 for script in $(ls $BASE_NETWORKCACHE_CONF_DIR/run.d/session/smb-*.sh); do

     if [ -x $script ]; then
  eval $script $userid &

     fi

 done
```

```
    fi

fi
start_usersession_automount $BASE_AFSUS_MOUNT_DIR/user/$userid/smb $BASE_AFSUS_FILES_DIR/$userid/smb/auto.hosts $PID_FILE
```

Note that this scripts has to started when a usersession begins. More info about how to do this in the next chapter.

Organisation of files

Name of script: ***start_smb_usersession.sh***

Description: start automount process to enable mounting of smb shares on demand for a user

Directory: ***/etc/session.d/scripts/start***

How called: link ***/etc/session.d/kdm/startup/20start_smb_usersession.sh --> /etc/session.d/scripts/start_smb_usersession.sh***

Only run when a user logs in with KDM.

Type: session related

Name of templates: auto.master.tmpl, auto.hosts.tmpl and auto.share.cred.tmpl and auto.share.guest.tmpl

Description: generalised templates to use with the cifs filesystem

Directory: ***/usr/share/autofs/usersession/filesystem/cifs***

Description

When a user terminates his/hers session, the automount program which is started should also stop. Here is discussed the autmount program for the SMB

service, so here only this program should stop. But it appears to be very simple to terminate any automount program started for the user.

This is done by looking at the pidfiles. Apart from they are containing the pid of the corresponding automount program, their name is derived from the path where the automount does mount the filesystems. In our case this is:

```
ls /var/run/autofs.*
```

/var/run/autofs.mnt-autofs-user-sbon-smb.pid

Looking at this file, it's very easy to extract the path:

```
ls /var/run/autofs.* | cut --delimiter "." --fields 2 | sed 's@-@\/@g'
```

mnt/autofs/user/sbon/smb

Note the lack of the starting slash.

Now when a session terminates, before stopping the automount program, first all the filesystems attached here by this automount program should be unmounted:

```
mount -t cifs | grep /mnt/autofs/user/sbon/smb/
```

//LFS20060812/bononline on /mnt/autofs/user/sbon/smb/LFS20060812/bononline type cifs (rw,mand)

So at this moment there is one share mounted. It's usefull to look at all the mounts here:

```
mount | grep /mnt/autofs/user/sbon/smb
```

```
automount(pid7217) on /mnt/autofs/user/sbon/smb type autofs (rw,fd=4,pgrp=7217,minproto=2,maxproto=4)
automount(pid10279) on /mnt/autofs/user/sbon/smb/LFS20060812 type autofs (rw,fd=4,pgrp=7217,minproto=2,maxproto=4)
//LFS20060812/bononline on /mnt/autofs/user/sbon/smb/LFS20060812/bononline type cifs (rw,mand)
```

The first automount is the "base" automount program for this user (sbon) and service (smb). The second (started by the auto.hosts file) creates the actual mountpoints (shares) on the virtualhosts (submounts) (as far as I can understand).

Script

The following script does that:
- determine all the base automount processes for this user, not only smb;
- umount any existing mounts done by the automount processes;
- terminate the base automount processes

```
#!/bin/bash
. /etc/session.d/scripts/misc/loganddebug.functions
umount_shares()
{
local lbasemountpoint=$1
local lfilesystem=$2
lmountpoints=$(mount -t $lfilesystem | grep $lbasemountpoint | awk '{ print $3 }')
if [ -n "$lmountpoints" ]; then
    for lmountpoint in $lmountpoints; do
 if [ -n "$lmountpoint" -a -n "$(mount -t $lfilesystem | grep --word-regexp $lmountpoint)" ]; then

     do_log "Umounting $lmountpoint."
     umount $lmountpoint
 fi
   done
fi
}
userid=$1
service=$2
logpriority=$3
userproperties=$(getent passwd | grep -m 1 -E "^$userid:")
```

```
homedir=$(do_log $userproperties | cut -d ":" -f 6)
gidnr=$(echo $userproperties | cut -d ":" -f 4)
uidnr=$(echo $userproperties | cut -d ":" -f 3)
if [ -z "$userproperties" ]; then
    # something wrong : the basic properties for this user are not found!
    exit 0
fi
if [ -z "$TMPDIR" ]; then
    if [ ! -d /tmp ]; then

 exit

    else

 TMPDIR=/tmp

    fi

fi
if [ -z "$logpriority" ]; then
    if [ -f $homedir/.xlogpriority ]; then

 logpriority=$(cat $homedir/.xlogpriority)

    else
        logpriority="local3.info"

    fi

fi
if [ ! -f /etc/autofs.usersession.conf ]; then
    do_log "The configuration file autofs.usersession.conf not found."
    exit
```

```
else
   source /etc/autofs.usersession.conf

   # check some variables

   if [ -z "$BASE_AFSUS_CONFIG_DIR" ]; then

 do_log "The variable BASE_AFSUS_CONFIG_DIR is not set."
 exit

   fi

   if [ -z "$BASE_AFSUS_MOUNT_DIR" ]; then

 do_log "The variable BASE_AFSUS_MOUNT_DIR is not set."
 exit

   fi
fi
if [ -f /etc/sysconfig/autofs.conf ]; then
   source  /etc/sysconfig/autofs.conf

fi
PIDROOT_NAME=${pidroot:-"autofs"}
PIDROOT_DIR=${piddir:-"/var/run"}
if [ $(ls -A ${PIDROOT_DIR}/${PIDROOT_NAME}.*-user-$userid-*.pid 2>>/dev/null | wc --words) -gt 0 ]; then
   for pidfile in ${PIDROOT_DIR}/${PIDROOT_NAME}.*-user-$userid-*.pid; do
 pidnr=$(cat $pidfile)

 autofs_mountpoint=$(basename $pidfile | cut --delimiter "." --fields 2 | sed -e "y/-/\//")

 autofs_mountpoint="/$autofs_mountpoint"
```



```
 if [ -d "$autofs_mountpoint" ]; then

    service=$(basename "$autofs_mountpoint" )

    if [ -f $BASE_AFSUS_CONFIG_DIR/service/$service/usersession.conf ]; then

 source $BASE_AFSUS_CONFIG_DIR/service/$service/usersession.conf

 if [ -n "$FILESYSTEM" ]; then

     umount_shares "$autofs_mountpoint/" $FILESYSTEM

 fi

    fi

 fi

#       COLUMNS=1024 ps ax | grep "[0-9]:[0-9][0-9] $automount " |
#       (
#           while read pid everything_else
#           do
#               boot_mesg "Stopping automount with pid $pid..."
#               kill $pid
#               evaluate_retval
#           done

 if [ -n "$pidnr" ]; then

    do_log "Stopping automount on $autofs_mountpoint."

    kill $pidnr
```

```
else
    rm $pidfile
fi
unset FILESYSTEM
  done
fi
```

Organisation of files

Name of script: ***stop_usersessions.sh***

Description: stops all automount processes started for this and umounts all related mounts

Directory: ***/etc/session.d/scripts/stop***

How called: link ***/etc/session.d/kdm/reset/10stop_usersessions.sh --> /etc/session.d/scripts/stop_usersessions.sh***

Run when a user stops a session (by logging out) which is started by KDM.

Type: session related

This part maybe hard to understand. I'm trying to explain the use of a few configurationfiles to let you get an overview how this construction works. I hope everything is clear. If not, do not hesitate to post a message.

Configuration of the networkcache

The settings for the networkcache I've put in a configurationfile ***/etc/networkcache***. In my case it looks like:

```
cat /etc/networkcache.conf
```

```
#
# BASE directory where the networkinformation is kept
#
BASE_NETWORKCACHE_DIR=/var/lib/network/cache
#
# directory where the configuration goes
#
BASE_NETWORKCACHE_CONF_DIR=/etc/networkcache
```

There are only two variables in it:

- BASE_NETWORKCACHE_CONF_DIR
- description: directory for configuration of networkcache
- default: /etc/networkcache

- BASE_NETWORKCACHE_DIR
- description: directory where the networkcache is
- default: /var/lib/network/cache

In my case this looks like:

```
/etc/networkcache/
                  create_network_cache.sh
                  networkcache.conf -> ../networkcache.conf
                  run.d
               session
                  smb-10.sh -> ../../service/smb/create_user_smb_tree.sh
                       system
```

```
                    smb-10.sh -> ../../service/smb/scan_network_with_nbtscan.sh
      smb-20.sh -> ../../service/smb/create_global_smb_tree.sh
service
   smb
  create_global_smb_tree.sh
  create_user_smb_tree.sh
  scan_network_with_nbtscan.sh
```

Explanation:

Subdirectory:

- BASE_NETWORKCACHE_CONF_DIR/service/smb
- description: scripts to discover the smb services and to setup the global and user trees

Subdirectory:

- BASE_NETWORKCACHE_CONF_DIR/run.d/system and BASE_NETWORKCACHE_CONF_DIR/run.d/session
- description: symbolic links to scripts (which are system or session related) in the BASE_NETWORKCACHE_CONF_DIR/service/smb/ directory. I've chosen for this construction to make it possible to run the scripts in a particular order and to make a difference between the system and session related scripts.

The mainscript - which is run periodically run by fcron - is ***create_network_cache.sh***

```
#!/bin/sh
if [ -z "$TMPDIR" ]; then
   if [ ! -d /tmp ]; then

 exit

   else

 TMPDIR=/tmp/networkcache
```

```
    fi

fi
if [ ! -d $TMPDIR ]; then
    install --directory $TMPDIR

fi
if [ ! -f /etc/networkcache.conf ]; then
    do_log "The file /etc/networkcache.conf is not found."
    exit

else
    source /etc/networkcache.conf

    if [ -z "$BASE_NETWORK_CACHE_DIR" ]; then

 do_log "The variable BASE_NETWORK_CACHE_DIR is not set."
 exit

    fi

    if [ -z "$BASE_NETWORK_CONF_DIR" ]; then

 do_log "The variable BASE_NETWORK_CONF_DIR is not set."
 exit

    elif [ ! -d $BASE_NETWORK_CONF_DIR/run.d ]; then

 do_log "The directory $BASE_NETWORK_CONF_DIR/run.d not found."
 exit

    fi
```



```
fi
for script in $(ls $BASE_NETWORK_CONF_DIR/run.d/system/*.sh); do
    if [ -x $script ]; then

 eval $script

    fi

done
for script in $(ls $BASE_NETWORK_CONF_DIR/run.d/session/*.sh); do
    if [ -x $script ]; then

 eval $script

    fi

done
```

This script is run by fcron. The entry in the tab for user root looks like:

```
@bootrun(true) 10 /etc/networkcache/create_network_cache.sh >> /dev/null 2>&1
```

You can edit the tab by the command:

```
fcrontab -u root -e
```

(assuming you're using fcron.)***Configuration of the usersession of Autofs***

The various settings for starting and stopping the automount program I've put in***/etc/autofs.usersession.conf***. It looks like:

```
#
# directory where the autofs daemons will mount the various filesystem per user per service
#
# the actual mounts will go in a subdirectory user/%USER%/%SERVICE%
BASE_AFSUS_MOUNT_DIR=/mnt/autofs
#
# directory for the usersession information is
#
BASE_AFSUS_CONFIG_DIR=/etc/autofs
#
# directory where the auto.master and related files for this user are kept
#
# the actual files are in a subdirectory %USER%/%SERVICE%
# (the service is for example smb or ssh)
#
BASE_AFSUS_FILES_DIR=/var/run/autofs/user
#
# directory where the generalised templates for the auto.master and related files are kept
#
# the actual files are in a subdirectory %filesystem%
# the filesystem is for example cifs or smbfs
BASE_AFSUS_TMPL_DIR=/usr/share/autofs/usersession/filesystem
#
# Name of the directory in the homedirectory where all the different networks (smb,nfs,ftp,..) will go
#
GLOBAL_NETWORK_NAME="Global Network"
```

I'm using here the name AFSUS. It stands for AutoFSUserSession. The meaning of the variables should be self explanatory.

The configurationdirectory ***/etc/autofs*** looks like:

```
/etc/autofs/
        autofs.usersession.conf -> ../autofs.usersession.conf
        service
            smb
               usersession.conf
```

The configurationfile ***usersession.conf*** looks like:

```
#
# name of the network
#
SMB_NETWORK_NAME="Windows Network"
#
#filesystem to use
#
FILESYSTEM="cifs"
```

Configuration of KDM

This construction needs extra scripts which are run when the session starts and ends. Already explained in chapter/page 8 this is easy. In my case the directory with all the scripts looks like:

```
/etc/session.d/
               kdm
              reset
              10stop_usersessions.sh -> ../../scripts/stop/stop_usersessions.sh
              30dbus.sh -> ../../scripts/stop/dbus.sh
           startup
              10dbus -> ../../scripts/start/dbus-session.sh
```

```
   20start_smb_usersession.sh -> ../../scripts/start/start_smb_usersession.sh
scripts
       misc
        loganddebug.functions
   start
        dbus-session.sh
    start_smb_usersession.sh
   stop
        dbus.sh
    stop_usersessions.sh
```

Here also I'm using symbolic links to the real scripts, with the same reason. Further I'm not discussinghere the scripts to start and stop the session daemon of dbus, and the file loganddebug.functions. The latest is a file with functions to write log and debug information to a logfile.

Some disadvantages

The construction is working very good. When entering the directory representing the share, the share is automatically mounted. But this does not only happen when entering the share. It does happen also when youare listing all the shares (and not entering):

```
mount -t cifs
```

```
//LFS20060812/bononline on /mnt/autofs/user/sbon/smb/LFS20060812/bononline type cifs (rw,mand)
```

```
ls -l ~/Global Network/Windows Network/CWWERKGROEP/ROUTER
```

```
lrwxrwxrwx 1 root root 37 2007-08-06 12:16 ftp -> /mnt/autofs//user/sbon/smb/ROUTER/ftp
lrwxrwxrwx 1 root root 40 2007-08-06 12:16 public -> /mnt/autofs//user/sbon/smb/ROUTER/public
lrwxrwxrwx 1 root root 38 2007-08-06 12:16 sbon -> /mnt/autofs//user/sbon/smb/ROUTER/sbon
```

```
mount -t cifs
```

```
//LFS20060812/bononline on /mnt/autofs/user/sbon/smb/LFS20060812/bononline type cifs (rw,mand)
//ROUTER/ftp on /mnt/autofs/user/sbon/smb/ROUTER/ftp type cifs (rw,mand)
//ROUTER/sbon on /mnt/autofs/user/sbon/smb/ROUTER/sbon type cifs (rw,mand)
//ROUTER/public on /mnt/autofs/user/sbon/smb/ROUTER/public type cifs (rw,mand)
```

This happens when listing the contents of the "hosts contents" on the commandline as well when using the default filemanager in KDE. It's easy to explain why this happens. When checking the contents of the "host", symlinks are found. The normal behaviour is to check every target a symlink is pointing to: does is exist? When doing so, autofs is activated and performs the mount. The explanation is simple, solving is harder. I do not have a clue. I'll post this issue on the maillist of autofs.

Other filesystems: extending the construction

I've build this construction to automate the discovery and mounting of smb shares. But this is not only possible for smb only. There are others I can think of as well like mounting NFS or Netware shares. With all the various modules available for Fuse, allmost anything is possible:

- ftp hosts with ***CurlFtpFs*** or ***Fuseftp***
- webdav shares with ***Fusedav*** or ***wdfs***
- ssh connections with ***SshFS***
- http directories and pages with ***httpfs***
- git repositories with ***GitFS***
- cvs contents with ***CvsFS***

More information about Fuse and projects are on: ***Fuse project site on SourceForge***.

Note that these Fuse modules can't be used directly. Autofs expects a mount.%FILESYSTEM% (and a simmular umount) utility when accessing the share with %FILESYSTEM%. For example you want to use the Fuse module CurlFtpFs to mount ftphosts. It comes with with the utility ***curlftpfs***. The ***mount.ftpfs*** will look like:

```
#!/bin/bash

curlftpfs $1 $2 -o allow_other
```

and the umount.ftpfs will look like:

```
#!/bin/bash

fusermount -u $1
```

Note futher that the autofs.* for this service look also different. To mount ftphosts only two autofs files are neseccary, not three.

Conclusion: in essence it's not hard to extend this construction with other filesystems/services, as long as the utility to mount and to umount the filesystem is available. I've build this construction with this possible extending in mind.

Other ways of detection

Here I've used nbtscan to discover all the smb hosts in my network, and smbclient to find all the shares on a single host. There are other ways to detect services available:

- detection via ***Avahi*** and/or ***OpenSLP***
- other servicerelated utilities
- manually/static configuration

One server maintaining the cache

In stead off letting every host detect the services available on the network, let one server do all the scanning, and export that information to every other host. Note that the access to a share is host and user related, so this can never be maintained on a central place. What is possible are for example all the smb hosts and the services they are offering (compare: a WINS server does something like that in a network with Windows hosts)

Integration with dbus

This construction makes use of several scripts, some of them are system related, some are session related. It's working very good on my computers, but when using this on a greater scale, integration here with dbus is inevitable in my opinion. When the cache is changed, the changed should be communicated through some signal/method. Maybe communicating with autofs also through dbus.....

Requirements

NBTSCAN

The building of the cache relys on the program nbtscan. It's not a well known, common program.
So, probably you'll have to install it. Maybe you'll find a rmp or deb or .. package, I as a LFS user prefer installing from source. The source is not hard to find (look at:***http://www.inetcat.net/software/nbtscan.html***).

Building and installing it:

```
tar -xvf nbtscan-1.5.1.tar.gz

cd nbtscan-1.5.1

./configure --prefix=/usr

make

make install
```

SMBFS or CIFS

For your operarting system to mount smbfs (or cifs) shares, it has to be supported. This means that:

- smbfs (or cifs) support in the kernel
- the mount.smbfs (or mount.cifs) and the umount.smbfs (or umount.cifs) are present

AUTOFS

Of course autofs has to be present on your system. At this moment [summer 2007] there is a new version available (5), but this to work autofs version 4 is sufficient. Required is:

- autofs support in kernel
- autofs tools and programs

As LFS user I've installed it from source:

http://www.linuxfromscratch.org/blfs/view/stable/postlfs/autofs.html

It's not necessary to install it from source, it's a very common package, so it will probably be available for your system.

Running scripts/programs when a usersession starts and stops

In the previous chapters I've shown the scripts to start and stop the automount program. The next question is how to run these scripts. The desktop of mychoice [KDE] has the ability to run scripts when a session start and stops. The scripts go in $KDEDIR/env and $KDEDIR/shutdown. (and in the ~/.kde/env and ~/.kde/shutdown). The disadvantage here is that the scripts are executed with the account of the user logging in, while root privileges are necessary to mount the autofs and the cifs filesystems. It is very possible to give the user enough rights (with Sudo for example), but this is not really necessary. Using KDM for this purpose is a better choice: it's very simple when users are logging in with KDM (the loginagent for KDE). Then it is very easy to create a construction which does what is needed here:run scripts when a session begins and when it ends, with root privileges. This possibility already exists in KDM (and otherloginmanagers like GDM and XDM).

KDM uses the following files to start and stop:

. Xstartup
run as root, after a user succesfully logs in.

. Xsession
runs with permissions of the authorized user, to start the desired session (KDE).

. Xreset
run as root, after the user session has ended.

Where Xstartup is the place to start things up, Xreset is the place to undo these commands.

For more information about these files look at the ***handbook of KDM***.

```
#! /bin/sh
# Xstartup - run as root before session starts
. /etc/session.d/scripts/misc/loganddebug.functions
do_log " "
do_log "-----------------------------------------------------"
do_log "WELCOME $USER at $(uname -n)."
```

```
do_log "------------------------------------------------------"
do_log " "
do_log "Running startup scripts...."
if [ -d /etc/session.d/kdm/startup ]; then
    for script in /etc/session.d/kdm/startup/*.sh; do

 if [ -x $script ]; then

     eval $script $USER kdm $LOGPRIORITY

 fi
   done

fi
do_log " "
do_log "Ready."
do_log " "
```

and the code to the Xreset file:

```
#! /bin/sh
# Xreset - run as root after session exits
. /etc/session.d/scripts/misc/loganddebug.functions
do_log "."
do_log "Running stop scripts...."
if [ -d /etc/session.d/kdm/reset ]; then
    for script in /etc/session.d/kdm/reset/*.sh; do

 if [ -x $script ]; then

     eval $script $USER
```

```
 fi

   done

fi
```

Create the directories where the scripts go:

```
install -m755 -d /etc/session.d/kdm/startup

install -m755 -d /etc/session.d/kdm/reset
```

The files in these directories must be accessible for every ordinary user:therefore the permissions are 755.

All scripts in these directories should have the same permissions: 755.

Every user should be able to execute the script, but only root is able to modify them.

Futher, important is that the directory to store start/stop scripts is ***/etc/session.d/scripts***. By making a symlink from the kdm directories the scripts will run when a KDM session starts en ends.Compare the symlinks in the runlevel directories in ***/etc/rc.d/rc?.d*** pointing to the base directory***/etc/rc.d/init.d***. The main reason for me to use this construction is the ability to run the scripts in an particular order. ***Notes***

One of the programs which in my system is started is the session part of dbus for the authorized user.(here I have to make use of sudo, because it has to be started with the privileges of the authorized user)It's I think important to start it here, because dbus communication should be turned on as soon as possible. When starting dbus in $KDEDIR/env for example should be too late.***Organisation of files***

Name of scripts: Xstartup and Xreset

Description: runs scripts when a session starts and when it ends. It will search for scripts in the directories ***/etc/session.d/kdm/startup*** at startup and in ***/etc/session.d/kdm/reset*** when it ends.

Directory: ***/opt/kde-3.5/share/config/kdm***