File- and printersharing on the local network

Suppose you are living in a household with Windows and Linux computers scattered all over the place. How are you going to share all the data across those Operating Systems? Running up the stairs with USB sticks is only so much fun.
Here is where Linux shines (yeah I know you can setup a Windows server, but you'll have to pay the price for that). In this article, I will try to explain how to setup a file- and print server where Windows and Linux clients (right… and Mac OSX too!) can connect and store their valuable stuff.
This will not be a complete hand-holding experience - I will assume you've got at least a basic knowledge of Linux, Samba and (possibly) NFS. This article helps you get things right, when you tried and failed before.

Having said that, let's delve a little deeper. There are two basic methods of accessing files on filesystems located on a networked machine. When you're a Windows users, filesharing amounts to *Windows File Sharing* using the SMB or CIFS protocols. These protocols are built into Windows, so why let them go to waste? I'll describe how you setup a Samba server on your central Linux box and all your Windows clients will be happy. Looking at Linux users, the horizon broadens. Here, we face several more alternatives, Samba and NFS being the two that will be available to virtually every Linux user. NFS has the advantage of being able to map all the user IDs on the server to the same user IDs of the client. With Samba, an exported share can be mounted on the client (provided you have sufficient access rights) using another user ID or group ID than that of the original files.

The rest of the article will describe how to set up a Samba server as well as a NFS server, and enhance the network experience of your fellow computer users (or perhaps it's only you that will be using Samba). Setting up networked printers will also be discussed. An article on setting up CUPS printing is not in scope of this article, but we do need a configured CUPS server if we want to print from our Windows clients.

Setting up Samba on Slackware

The exercise is meant to setup a Samba server where you make available several shares, some username/password protected, others that don't need a password. People will be able to connect to their homedirectory on the server. The printers that were configured in the CUPS printserver will be available as shared network printers through Samba, and Windows printer driver files can be automatically downloaded from the Samba server by Windows clients. So far, sounds nice, doesn't it? The last section of this paragraph contains an example smb.conf file that is based on what I'm about to tell.
Here is the skinny:

Basic setup

- Install Samba and CUPS packages
  Strictly speaking, we're talking about setting up a file server here. But since fileserving and printserving are closely related issues, and the Samba package depends on the CUPS package anyway, we will proceed to install both. If you decide not to use the CUPS printserver, that's up to you. If you do a full install of Slackware, both packages will have been installed.

- Create a Samba guest user account (for anonymous aka public shares)

```
groupadd smbguest
useradd -g smbguest -m -d /home/smbguest -s /bin/false -c "Samba guest
user" smbguest
smbpasswd -a smbguest -d
passwd smbguest -l
```

  This will make sure that the account *smbguest* can not be used for direct logons to your Linux box. Only Samba will be able to use it.

- Make sure these lines are in `/etc/samba/smb.conf` so that Samba can use it for the public shares:

```
    guest account = smbguest
    map to guest = bad user
```

- This should be in `/etc/samba/smb.conf`, it defines the password backend Sabma will use (I will not discuss the alternative LDAP backend here, I hope to write another article on that all of it's own):

```
    passdb backend = tdbsam:/var/lib/samba/private/passdb.tdb
```

  NOTE: when you enable *tdbsam* on an already configured and running Samba server, and run `killall -HUP smbd` to let it re-load the configuration files, you might have to re-add the 'smbguest' user to Samba.
  In the past, the default to use was the *smbpasswd* file but since Sa,ba 3.4 the tdbsam database became the default.
  So, again run:

```
smbpasswd -a smbguest -d
```

- Create samba directories (as root) that don't yet exist (but we need them):

```
    # General purpose:
    mkdir -p /var/log/samba/messages
    mkdir -p /usr/local/samba/bin
    mkdir -p /usr/local/samba/netlogon
    # A shared directory where you can dump stuff temporarily:
    mkdir -p /usr/local/samba/share
    chmod 1777 /usr/local/samba/share
    # These are for the network printers:
    mkdir -p /usr/local/samba/printers/{W32ALPHA,W32MIPS,W32X86,WIN40}
    chgrp -R wheel /usr/local/samba/printers
    chmod -R g+w /usr/local/samba/printers
```

- Make sure you have configured printers in CUPS if you want to use printing in Samba. For Linux, you will use a CUPS printer configuration that is using the printer-specific PPD file, but for Windows clients, you will have to setup additional printer queues that use *RAW* printing (i.e.

CUPS does not mess with the incoming printer data and passes the data on to the printer unaltered). The CUPS server does not know about raw printer data by default, so you will have to uncomment a couple of lines.

1. In the file `/etc/cups/mime.types` uncomment the line `application/octet-stream`
2. In the file `/etc/cups/mime.convs` uncomment `application/octet-stream` `application/vnd.cups-raw 0 -` And then restart CUPS daemon using

```
/etc/rc.d/rc.cups restart
```

- It is now time to fire up our Samba server. But we will test the configuration first by running the command `testparm`. It will show us if anything went wrong while editing the `smb.conf` file. If everything seems allright, we will procede with making the Samba start script executable (so that it will still start when we boot our server) and then running the script:

```
chmod +x /etc/rc.d/rc.samba
/etc/rc.d/rc.samba start
```

Check if your server is up and running by using either of these two commands:

```
smbclient -L localhost
```

The result should be a dump of the visible shares that are defined in Samba, plus a list of computers in the network that also talk the *Windows File Sharing* protocol (this list will be small or empty at first, it may take up to 15 minutes for machines in the network to become aware of each other).

```
smbstatus
```

which will give an overview of the activity on the Samba server (a verbose list of connected shares and open files).

Was it really that easy? Yes! It really is that easy. We have of course not yet messed with networkprinters, or password-protected shares, but the basics are there. You can take a Windows machine and (if you followed my example to the letter) try to connect to `\\BOB\PUBLIC` and see your Linux ftp server directories.

Advanced setup

There are some topics I have to cover. These are: password-protected access, and Windows clients.

Windows clients

Ever since Windows 98 and NT4SP3, windows clients exchange encrypted passwords with the server. The Samba server is configured for encrypted passwords, so this will cause no problems. Only DOS, and Windows 95 clients will *not* be able to access our Samba server. People with these old Operating Systems in their network will need to disable the use of encrypted passwords on their other OS'es like

Windows 2000/XP and Samba. The line to add to your `/etc/samba/smb.conf` is

```
  encrypt passwords = no
```

With Windows XP, the authentication mechanism changed somewhat, due to MS Active Directory (AD) support. A Windows XP client can only connect and authenticate against a Samba server after a small registry modification (the *requiresignorseal* hack). This is the registry key - save this in a file with extension `.reg` and double-click on it. XP will ask if you want to merge the files's key into the Windows registry. You want that.

```
Windows Registry Editor Version 5.00

;
; This registry key is needed for a Windows XP Client to join
; and logon to a Samba domain. Note: Samba 2.2.3a contained
; this key in a broken format which did nothing to the registry -
; however XP reported "registry key imported". If in doubt
; check the key by hand with regedit.

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters]
"requiresignorseal"=dword:00000000
```

You can find this file (`/usr/doc/samba-3.0.xx/docs/registry/WinXP_SignOrSeal.reg`) and several others in the Samba documentation directory on your Linux box.

Passwords in Samba

The Samba server will use your local Linux accounts (the ones in `/etc/passwd`) but with encrypted passwords enabled (see above) Samba must have a way of storing these passwords somewhere. They are incompatible with the Linux account passwords that you find in `/etc/shadow`. Samba keeps a database with these encrypted passwords and the trick is to keep these synchronized with the Linux passwords. The lines in `smb.conf`

```
# Synchronize Samba and Unix passwords
   passwd program = /usr/bin/passwd %u
   passwd chat = *password* %n\n *password* %n\n *changed*
   unix password sync = Yes
```

will take care of that.
There is one caveat: for every Linux account that wants to use the Samba server, you will have to add the entry in the Samba password database manually. You do this as root, and it's only needed once for every user:

```
smbpasswd -a <some_username>
```

The `smbpasswd` command can be used later on to change the Samba password *and* the Linux password together by running

```
smbpasswd <some_username>
```

A Windows user can use the <CTRL><ALT><DEL> sequence to change the password!

Samba printers

If you have a configured and running CUPS server that has at least one queue setup for *RAW* printing, we can now proceed with integrating this CUPS printer queue with our Samba server, so that Windows clients can automatically download their printer drivers from the Samba server. This is of course more convenient than accessing each and every Windows PC with a printer driver CD and manually configuring the printer.

Using the directions of the previous sections and the smb.conf example of the last section, you have everything in place already, server-side. You will now have to take a Windows XP workstation, and logon to a Samba share using an account that is known to Samba as a *printer admin*. In our setup, that means: everyone who is a member of the Linux group wheel.

🔧 Fix Me! *to be completed* 🔧 Fix Me!

The Linux client setup

On a linux client computer, it is the `smbmount` command that lets you mount a Samba (or a Windows!) share on the local filesystem. You can run the command manually in a console like this:

```
mount -t cifs //192.168.0.1/public /mnt/samba/bob/public -o
rw,uid=0,gid=10,fmask=664,dmask=775 -U <some_special_user>
```

which will mount the share called *public* on our server called *bob* which has the IP Address `192.168.0.1` in this example. The mountpoint `/mnt/samba/bob/public` must of course be created as a directory before. I chose `/mnt/samba/bob/public` arbitrarily, I like looking at the mount point's name and be able to guess what it is all about. You are of course free to take another mount point.
The command mounts the share as user *<some_special_user>* and it's up to you who that user account is, as long as it has the necessary access rights to the share. If the account has a password associated with it, you will be asked for it.
The *-o rw,uid=0,gid=10,fmask=664,dmask=775* part means that the remote share will be mounted locally read/write, seemingly owned by user root:wheel (uid=0, gid=10) and with file- and directory masks that make the share's files and directories read/only for non-root, non- wheel users. Having to type all this in order to mount the share is a tedious effort, so we take the easy way and add the following line to `/etc/fstab`:

```
//192.168.0.1/public /mnt/samba/bob/public  cifs
rw,uid=0,gid=10,fmask=664,dmask=775,credentials=/etc/bob.cred  0 0
```

We store the username and password that we need for gaining access to the *public* share, in a file called `/etc/bob.cred` which we protect from prying eyes by removing read access for all but the root user:

```
chmod 600 /etc/bob.cred
```

The contents of that file are like this:

```
username = <some_special_user>
password = <the_secret_word>
```

Having this in our client computer's `/etc/fstab` will cause the samba share to be automatically mounted when the computer boots.

The older "`smbfs`" kernel module is outdated while the "`cifs`" kernel driver is well-maintained and supports higher versions of the SMB protocol than `smbfs`. Samba implements the CIFS protocol (a dialect of the SMB protocol) and it supports SMB2 and parts of the SMB3 protocol extensions.

The default CIFS protocol changed from SMB 1.0 to SMB 3.0 in kernel 4.13. If you are running an older Samba server and try to mount its shares on a recent Linux computer this changed default breaks the mounts. In order to force the mount to use SMB 1.0 you need to add "`vers=1.0`" to the mount options

Mixing protected and passwordless shares

A note on the sometimes unexpected consequences of using a mix of passwordless shares (like the *PUBLIC* share in my example) and protected shares, where you have to type a username and password to access the data (like the *HOMES* share in the example).

Windows will not allow you to logon to a Samba (or even a real Windows) server using more than one set of credentials. This means that if you start with a connection to a passwordless share, you actually are using the "guest" credentials. Once you've done that, there is no way you can map your homedirectory for instance - Samba will of course ask you for a valid account/password combination, and you can actually enter those in the Windows dialog box that will appear, but after clicking on *OK*, Windows will complain that it can only use a single set of credentials and will refuse to map to your protected Samba share.
This is annoying, and on some occasions it will be sufficient to close the Windows Explorer, open another, and then remove the lingering "guest" network connection through the "Extra" menu of the Windows Explorer. After doing so, Windows will accept your credentials and map you to your Samba home directory.
Things can become more difficult once you've connected to a *printer share* that is exported by the Samba server. If printing does not require a password (because that is quite convenient) and you have printed anything, it will be *very* hard to get rid of the "guest" connection to the Samba server because the Windows printing subsystem will keep opening that connection. In many such cases, you will have to reboot the Windows PC in order to map to a protected share. Or, use the IP address or internet host name of the Samba server for your password-protected share access (like, try to connect to `\\192.168.0.1\alien` instead of the usual `\\bob\alien`). This will force the protocol to the more modern CIFS instead of SMB, and then you won't have this problem.

A sample smb.conf

```
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# For a step to step guide on installing, configuring and using samba,
# read the Samba HOWTO Collection.
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not made any basic syntactic errors.
#
#======================= Global Settings
=====================================
[global]

# workgroup = NT-Domain-Name or Workgroup-Name, eg: LINUX2
   workgroup = SLACKWARE

# The server's NETBIOS name
   netbios name = BOB

# server string is the equivalent of the NT Description field
   server string = BOB File Server

# Security mode. Defines in which mode Samba will operate. Possible
# values are share, user, server, domain and ads. Most people will want
# user level security. See the HOWTO Collection for details.
   security = user

# Specify the debug  level for multiple debug classes.
# The  default will be the log level specified on the command line,
# or level zero if none was specified.
   #log level = 2 passdb:5 auth:3 winbind:2
   log level = 1

# Don't log anything at all in the syslog.
   syslog = 0

# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the "loopback" interface. For more examples of the syntax see
# the smb.conf man page
```

```
   hosts allow = 192.168. 127.

# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes

# you may wish to override the location of the printcap file
   printcap name = cups

# It should not be necessary to specify the print system type unless
# it is non-standard. Currently supported print systems include:
# bsd, cups, sysv, plp, lprng, aix, hpux, qnx
   printing = cups

# Commands with which to control the printer queues
   print command = lpr -oraw -r -P'%p' %s
   lpq command = /usr/bin/lpq -P%p

# Members of the wheel group should be able
# to add drivers and set printer properties
# root is implicitly a 'printer admin'
   printer admin = @wheel

# Uncomment this if you want a guest account, you must add this to
/etc/passwd
# otherwise the user "nobody" is used
   guest account = smbguest

# Bad User - Means user logins with an invalid password are rejected,
# unless the username does not exist, in which case it is treated as a guest
# login and mapped into the  guest account.
   map to guest = bad user

# User name remapping
;  username map = /etc/samba/smbusers

# this tells Samba to use a separate log file for each machine
# that connects
   log file = /var/log/samba/%m.log

# Put a capping on the size of the log files (in Kb).
   max log size = 50

# Backend to store user information in. New installations should
# use either tdbsam or ldapsam. smbpasswd is available for backwards
# compatibility. tdbsam requires no further configuration.
   passdb backend = tdbsam guest

# Synchronize Samba and Unix passwords
   passwd program = /usr/bin/passwd %u
   passwd chat = *password* %n\n *password* %n\n *changed*
```

```
   unix password sync = Yes

# Most people will find that this option gives better performance.
# See the chapter 'Samba performance issues' in the Samba HOWTO Collection
# and the manual pages for details.
# You may want to add the following on a Linux system:
#         SO_RCVBUF=8192 SO_SNDBUF=8192
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192

# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must list them
# here. See the man page for details.
;   interfaces = 192.168.12.2/24 192.168.13.2/24
   interfaces = eth0 lo
   bind interfaces only = yes

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
# browser on your network. Otherwise the normal election rules apply
;   local master = no

# OS Level determines the precedence of this server in master browser
# elections. The default value should be reasonable
;   os level = 33

# Domain Master specifies Samba to be the Domain Master Browser. This
# allows Samba to collate browse lists between subnets. Don't use this
# if you already have a Windows NT domain controller doing this job
   domain master = yes

# Preferred Master causes Samba to force a local browser election on startup
# and gives it a slightly higher chance of winning the election
   preferred master = yes

# if you enable domain logons then you may want a per-machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
;   logon script = %m.bat
# run a specific logon batch file per username
;   logon script = %U.bat

# Where is a user's home directory and where should it be mounted at?
   logon drive = H:
   logon home = \\%N\%U

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable it's WINS
Server
   wins support = yes

# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names
```

```
# via DNS nslookups. The default is NO.
   dns proxy = no

# Message command action
# (whenever a client user types "net send SILAS 'some message text'")
   message command = sh -c 'echo ""|cat %s >> /var/log/samba/messages/%m.msg
-; rm -f %s' &

#============================ Share Definitions
==============================
[homes]
   comment = Home Directories
   hide dot files = yes
   browseable = no
   writable = yes
   valid users = $S
   create mode = 0664
   directory mode = 0751
   invalid  users = root nobody smbguest


# NOTE: there is no need to specifically define each individual printer
[printers]
   comment = All Printers
   path = /var/spool/samba
   browseable = no
   # Set public = yes to allow user 'guest account' to print
   public = yes
   writable = no
   printable = yes
   printer admin = @wheel
   # See
http://us3.samba.org/samba/docs/man/Samba-HOWTO-Collection/printing.html#id2
553131
   # Try to work around crashing Explorer.exe ("function address 0x606xxxx
   # protection fault" messages) when installing a printer driver
   # (e.g. for the HP3820C). Set 'default devmode' to 'yes' and reboot the
   # client PC that experiences the explorer crash ; then install the driver
   # onto the server share and re-set this parameter to 'no':
   default devmode = no

# This one is useful for people to share files
[tmp]
   comment = Temporary file space
   path = /usr/local/samba/share
   read only = no
   public = yes


# A publicly accessible directory, but read only, except for people in
# the "wheel" group
```

```
[public]
   comment = Public Stuff
   path = /home/ftp/pub
   public = yes
   writable = yes
   printable = no
   write list = @wheel


[print$]
   comment = Printer Drivers
   path = /usr/local/samba/printers
   guest ok = yes
   #browseable = yes
   browseable = no
   read only = yes
   ; since this share is configured as read only, then we need
   ; a 'write list'.  Check the file system permissions to make
   ; sure this account can copy files to the share.  If this
   ; is setup to a non-root account, then it should also exist
   ; as a 'printer admin'
   write list = @wheel,root
   create mode = 0664
   directory mode = 0775
   force group = wheel

# A publicly accessible directory, read/write to all users. Note that all
files
# created in the directory by users will be owned by the default user, so
# any user with access can delete any other user's files. Obviously this
# directory must be writable by the default user. Another user could of
course
# be specified, in which case all files would be owned by that user instead.
[pool]
   comment = Pool
   path = /home/ftp/pool
   public = yes
   only guest = yes
   writable = yes
   printable = no
   force group = users
   create mode = 664
   directory mode = 775

## Example: share with mp3 file, accessible without a password (read-only)
## And only someone in the 'wheel' group can write here if he mapped the
drive
## using his account and password.
#[mp3]
#   comment = MP3 files
#   path = /data/mp3
```

```
#   level2 oplocks = True
#   write list = @wheel
#   create mode = 775
#   directory mode = 775
#   public = yes
#   writable = yes
#   follow symlinks = yes
#   wide links = yes
```

Setting up NFS on Slackware

Actually, setting up a NFS for Slackware is even easier than Samba.

NFS Server

You setup an NFS server by creating or editing the file `/etc/exports`. That file has a man page (man exports) and I encourage you to read that if you want more than my simple example. But basically, this file can look like this:

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other computers.
# It is used by rpc.nfsd and rpc.mountd.
/home                192.168.0.0/24(rw,async,no_root_squash)
/var/www/htdocs      192.168.0.0/24(rw,all_squash,anonuid=99,anongid=99)
/home/ftp/pub        192.168.0.0/24(ro,sync,insecure,all_squash)
```

This creates three exports, all accessible by any client with an IP address in the range `192.168.0.0/24`. I'll discuss them in reverse order:

1. the ftp server's 'pub' directory aka the anonymous ftp area. This export will be available as read-only (the 'ro' parameter) with as safe as possible settings
2. your webserver's DocumentRoot (/var/www/htdocs) which will be available as writable, but on the server side, all writes will appear to originate from the user with the userid:groupid of `99:99` which is actually the “nobody” user. If you let the DocumentRoot tree be owned by this account (a configuration you often see), then the Web Server's CGI or PHP scripts can write files in these directories
3. the server's `/home` directory tree which can be mounted writable (the 'rw') using asynchronous transfers (faster but with a chance of data corruption in case of a server crash - 'sync' is safe but slower). User ID's (*uid*) and group ID's (*gid*) will be mapped 1-on-1 (even for user 'root' - the 'no_root_squash' option). This means, if the server knows a user 'alien' with a “uid:gid” pair of `1001:100`, then alien's files in his homedirectory will appear with this uid:gid number pair on the NFS client side as well! So, if the NFS client PC also has an account 'alien' with the same “uid:gid” number pair `1001:100`, this alien will be able to use the files on the server as they were his own.

You see why it is important to create users on your LAN with the same UID (and GID)

on *all* computers if you ever intend to install a NFS server.

NFS client

On a NFS client, you should enable the *portmapper*. To do so, activate the rc script and then run that script -once- manually (saves you a reboot to make it work)

```
chmod +x /etc/rc.d/rc.portmap
/etc/rc.d/rc.portmap start
```

Should you ever forget this, you will still be able to mount a NFS export, but it will take *forever* for the mount command to return to the prompt (if you run the mount command in a console).

Next, you should add a line to your `/etc/fstab` for the NFS export that you want to mount on your NFS client. Suppose your NFS server has IP Address `192.168.0.1` and it exports `/home` you could add this line to the fstab file:

```
192.168.0.1:/home  /mnt/nfs/home  nfs  auto,rsize=8192,wsize=8192,hard,intr
0   0
```

This NFS export will then automatically be mounted by Slackware when booting up. The manual mount command would be:

```
mount -t nfs -o rsize=8192,wsize=8192,hard,intr 192.168.0.1:/home
/mnt/nfs/home
```

Note, that I expect you to create the mount point (`/mnt/nfs/home` in the example, but you may pick your own of course) in advance...!

I hear you thinking... how do I find out the export list of my NFS server? This is easy: run

```
showmount -e <NFS_servername>
```

to obtain a list. This is what the output will look like:

```
# showmount -e bob
Export list for bob:
/home                          192.168.0.0/24
/var/www/htdocs                192.168.0.0/24
/home/ftp/pub                  192.168.0.0/24
```

Note that this specific NFS server also exports the webserver's DocumentRoot and the ftp server's 'pub' directory. What you *don't* see how those exports are configured (access restrictions and such, apart from the allowed IP address range).