Samba (SMB) and CIFS Sharing

# Samba Introduction

The Samba server runs on Linux; it allows Windows users to access files and printers on a Linux host. We will use other CIFS tools from the Samba project to mount a Windows share on a Linux VM.

*What is Samba?*

*Samba is a suite of Unix applications that speak the Server Message Block (SMB) protocol. Microsoft Windows operating systems use SMB to perform client-server networking for file and printer sharing and associated operations. By supporting this protocol, Samba enables computers running Unix to get in on the action, communicating with the same networking protocol as Microsoft Windows and appearing as another Windows system on the network from the perspective of a Windows client. A Samba server offers the following services:*

* *Share one or more directory trees*
* *Share one or more Distributed File System (DFS) trees*
* *Share printers installed on the server amongst Windows clients on the network*
* *Assist client with network browsing*
* *Authenticate clients logging onto a Windows domain*
* *Provide or assist with Windows Internet Name Service (WINS) name-server resolution*

*The Samba suite also includes client tools that allow users on a Unix system to access folders and printers that Windows systems and Samba servers offer on the network.*

*Samba is the brainchild of Andrew Tridgell, who currently heads the Samba development team. Andrew started the project in 1991, while working with a Digital Equipment Corporation (DEC) software suite called Pathworks, created for connecting DEC VAX computers to computers made by other companies. Without knowing the significance of what he was doing, Andrew created a file-server program for an odd protocol that was part of Pathworks. That protocol later turned out to be SMB. A few years later, he expanded up his custom-made SMB server and began distributing it as a product on the Internet under the name “SMB Server.” However, Andrew couldn’t keep that name – it already belonged to another company’s product – so he tried the following Unix renaming approach:*

*$ grep -i ‘^s.\*m.\*b’ /usr/share/dict/words*

*And the response was:*

*Salmonberry*

*Samba*

*Sawtimber*

*Scramble*

*Thus, the name “Samba” was born.*

SysAdmins often use Samba because, frankly, we don’t want to pay for a bunch of Microsoft licences.

# Samba Installation

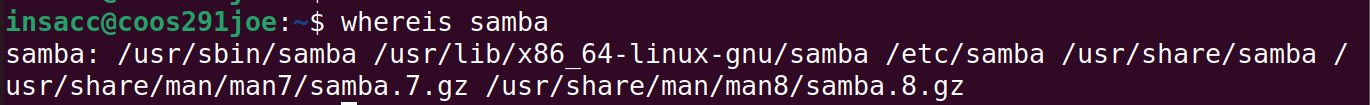
Install Samba via the following install command: **sudo apt install samba**

Text

Description automatically generated

To confirm installation, we use run the following command: **whereis samba**

We have multiple locations returned:



# Setting up Samba Shares

The Samba configuration file is located in **/etc/samba/smb.conf** This file uses the same structure as Windows .ini files.

We’ll edit that file soon and add a share.

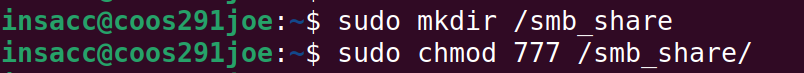
First, we need to create a directory to share and give full access to that folder.

Create a directory to share from in Linux:

**sudo mkdir /smb\_share**

and give full access to it:

**sudo chmod 777 /smb\_share** 🡨 777 is good for testing and lab purposes, never done in a production environment!



Next, we’ll edit the **/etc/samba/smb.conf** and add the following at the bottom:

|  |
| --- |
| [smb\_share]  path = /smb\_share  comment = A Place for Sharing and Caring  writable = yes |

To do this, run the following command:

**sudo vi /etc/samba/smb.conf** (you can make a backup beforehand: sudo cp smb.conf{,.bak})

Text

Description automatically generated

NOTE: *the spacing around the “=” is important here.*

After we save and close, we can test the configuration by running the following:

**sudo testparm**

Text

Description automatically generated

Once this is complete, we need to set up an account(s). We can’t connect yet, because it will ask for a password. **Samba has a separate set of passwords – but the account must already exist. (Also, login with a starting \ to remove any domain specification.)**

To create a user just for SMB. Run the following command: **sudo adduser --force-badname smbUser**

Text

Description automatically generated

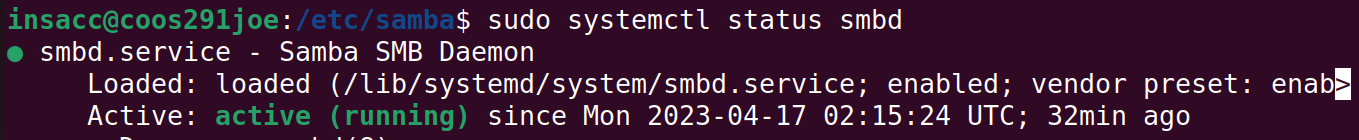
Next run the following command to set the SMB password for the smbUser: **sudo smbpasswd -a smbUser** 🡨 Note: the -a argument means “add,” and is only used the first time when setting a samba password for a specific user; if we want to change the password later on we would leave it out.

Use **Polytech123!** for the password. Note that there are tools to keep the Samba and Linux passwords sync’d which we are not covering in this class.

Text

Description automatically generated

From here, let’s confirm the service has started: **sudo systemctl status smbd**



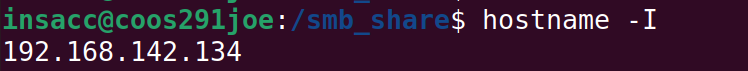
Add a file in the /smb\_share directory so we can confirm on Windows that everything is good:

Text

Description automatically generated

# Mount the Linux Share in Windows

Obtain the IP address for the Samba Server:



“Map network drive:”

A screenshot of a computer

Description automatically generated with medium confidence

Once complete, we should have an S: drive that contains the /smb\_share files from our Linux machine. We can edit, create, delete or whatever permissions we’ve set on the /smb\_share folder in Linux.

A screenshot of a computer

Description automatically generated with medium confidence

Let’s edit the file to make sure the changes are replicated.

Graphical user interface, text, application

Description automatically generated

Saving it, then going back to Linux should have that file reflect the changes. However, we don’t have edit permissions on that file:

A screenshot of a computer

Description automatically generated

So let’s go back to linux and edit them:

Text

Description automatically generated

Now we should be able to make changes.

A screenshot of a computer

Description automatically generated with medium confidence

Looks like it’s working correctly. We can create / copy files on the Windows end and have them appear on Linux as well:

A screenshot of a computer

Description automatically generated

Text

Description automatically generated

Notice the user **smbUser** and group **smbUser** when we created new files on the Windows end. This is here because we’re connecting to the Linux SMB share via the **smbUser**. Linux permissions are important!

# Accounts and Authentication

What happens if we change the permissions to disable write? Samba uses your filesystem permissions.

What if we changed writable = no in the smb.conf file? Samba won’t let you write, even if the Linux permissions say you can. The end result is always the MOST restrictive of the Samba config and Linux permissions.

We can look at *valid users* in the samba config for permissions. We can also use **@** to give access via group permissions, for the group named students we would use **valid users = @students**.

**Share-level access options (These are additional directives for your smb.conf file)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Option** | **Parameters** | **Function** | **Default** | **Scope** |
| admin users | string (list of usernames) | Users who can perform operations as root | None | Share |
| valid users | string (list of usernames) | Users who can connect to a share | None | Share |
| invalid users | string (list of usernames) | Users who will be denied access to a share | None | Share |
| read list | string (list of usernames) | Users who have read-only access to a writable share | None | Share |
| write list | string (list of usernames) | Users who have read/write access to a read-only share | None | Share |
| max connections | Numeric | Maximum number of connections for a share at a given time | 0 | Share |

You can check out smb.conf examples online for all kinds of different setup configurations.

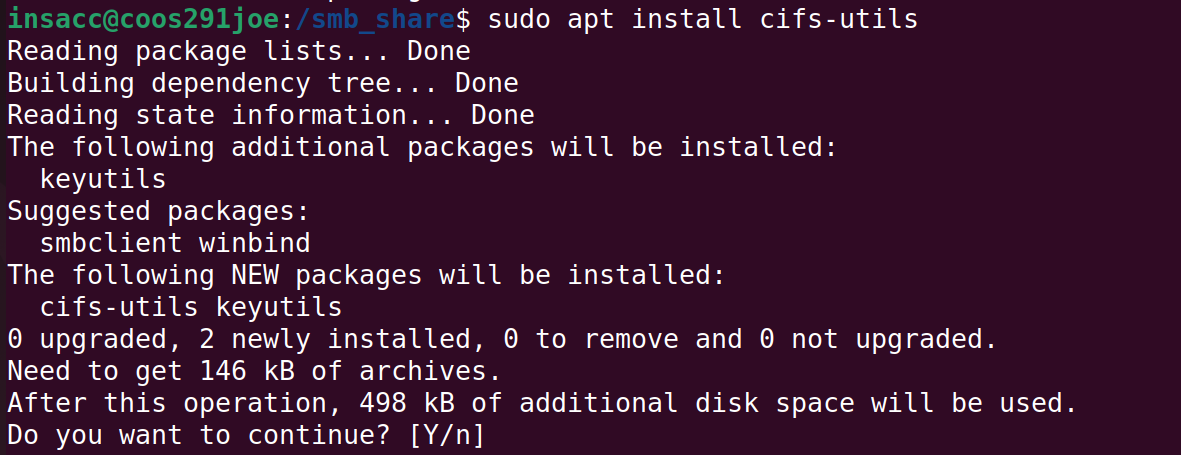
The easy way to set up users and accounts is, of course, to just mirror your Windows users in Linux. If you’re logged in as **cst201** user on Windows, it will try to link up with that same username when logging in to the Linux share. If they are different, you can use the **username map** directive to set up equivalents.

We *could* get linked up with the Windows domain to use their networks, but we’d need a couple of domain admins to help us do that, and that’s not going to happen. That’s set with a security option – **security = domain**. Instead we’re using security = user, which is the default, and means we use our Linux accounts.

# Linux Client for Windows SMB (CIFS) Server

To mount shared Windows folders on a Linux client machine, we’ll need to install the CIFS utilities, then make sure we have a Windows folder shared, and finally mount that share via the mount command.

To install CIFS, run the following command: **sudo apt install cifs-utils**



Next, we’ll create a mount point for our Windows share (ie: create a directory in Linux that we can link to)

Run the following command: **sudo mkdir /win\_share**



Next, let’s create a directory on our Host (Windows) and share it. We will also need a local account (non-domain account) for a demonstration to connect.

Keep in mind that we are able to hook into domain accounts, but we haven’t joined our Linux machine to the domain so we’ll not be covering this.

Next, let’s create a directory on our Host (Windows) and share it. We will also need a local account (non-domain account) for a demonstration to connect.

Keep in mind that we are able to hook into domain accounts, but we haven’t joined our Linux machine to the domain so we’ll not be covering this.

Graphical user interface

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated

Once this is done (Creating a share folder, creating a local user, and adding permissions), we can this folder in our Linux machine using the UNC path specified: ([\\win­share](file:///\\winshare))

Before mounting, our df -h command output:

A picture containing text

Description automatically generated

To mount it in Linux, run the following command:

**sudo mount -t cifs -o username=insacc //10.28.52.16/win\_share /win\_share**

Notes:

* -t cifs
  + This mounts the filesystem as a CIFS filesystem
* username=insacc
  + This is the Windows account to connect via
  + We can also specify password and domain
  + IE: [username=herbertj@saskpolytech.ca,domain=SIAST](mailto:username=herbertj@saskpolytech.ca,domain=SIAST)
* //10.28.52.16/win\_share
  + The path (UNC) to the folder on our Windows machine that is shared.
* /win\_share
  + The folder in our Linux environment to mount it to.

And now on Linux we can see that win\_share is mounted.Text

Description automatically generated with medium confidence

Browsing to that folder in Windows, I can create and edit files on the Windows Shared folder.

Graphical user interface, table

Description automatically generated

And in Linux, we see the same:

Text

Description automatically generated