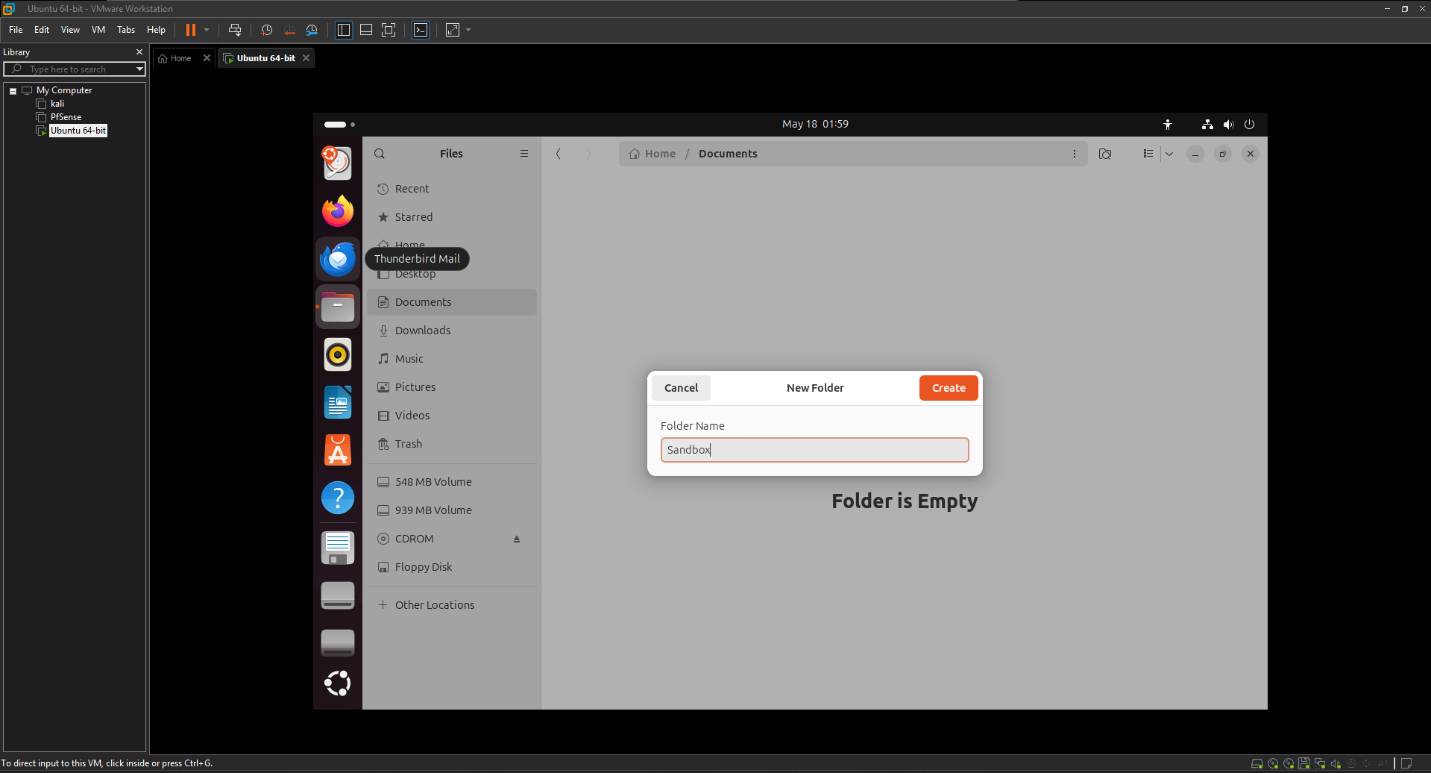
**Encryption**

I’m working on encryption, and I will work through each step I take to complete it. In completing my encryption running, I attempted to directly manipulate files (encrypted files with GPG), use GPG plain to encrypt and decrypt files, and hash in MD5. A Linux GUI Desktop (Ubuntu), and I am the admin. Let's get started.**1. Setting Up My Environment**

I begin by opening the file manager on my Ubuntu system. I navigate to the Documents directory, where I right-click and select “New Folder” to create a new directory. I named it Sandbox.



Next, I right-click on the Sandbox folder and choose “Open in Terminal.” This opens a terminal with the working directory set to ~/Documents/Sandbox. To perform the lab’s tasks, I need root privileges, so I type: *sudo -i*

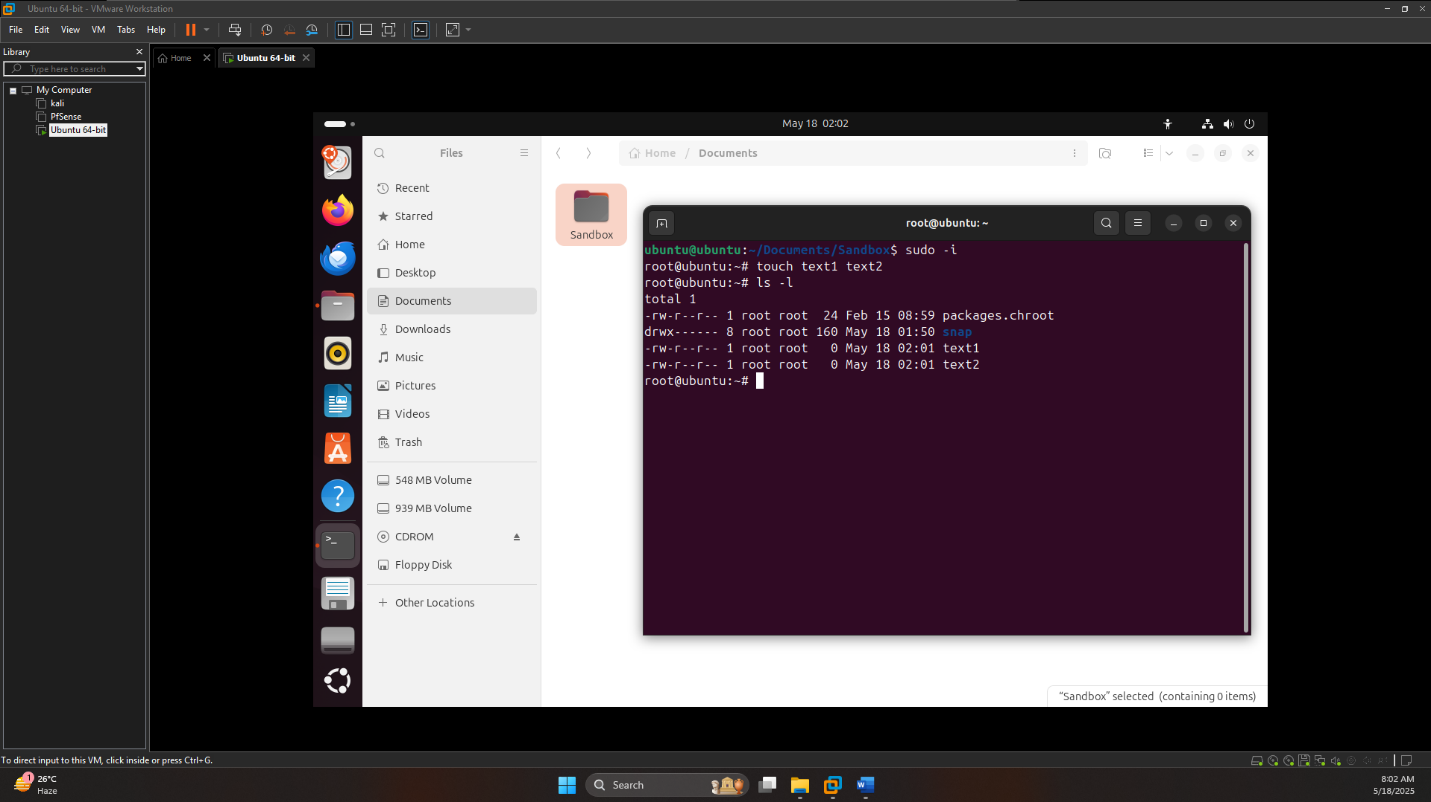
I enter my root password when prompted, and I notice the terminal prompt changes to #, confirming I’m now in “super user” mode as root@ubuntu.

**1. Creating and Editing Files**

Now, I have created two empty files named text1 and text2 by running: ***touch text1 text2***

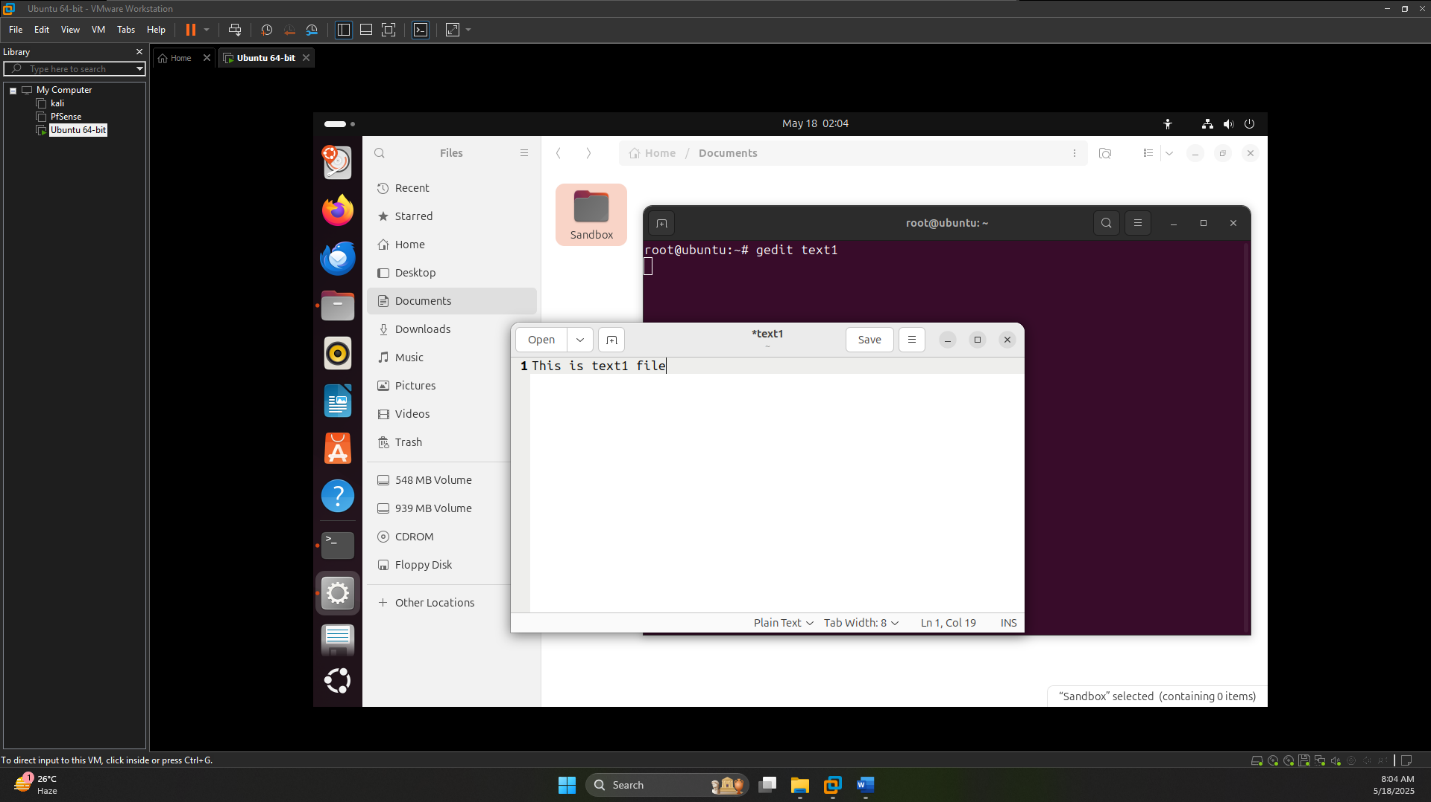
To confirm the files, exist, I list them with: *ls -l*

I see the output:



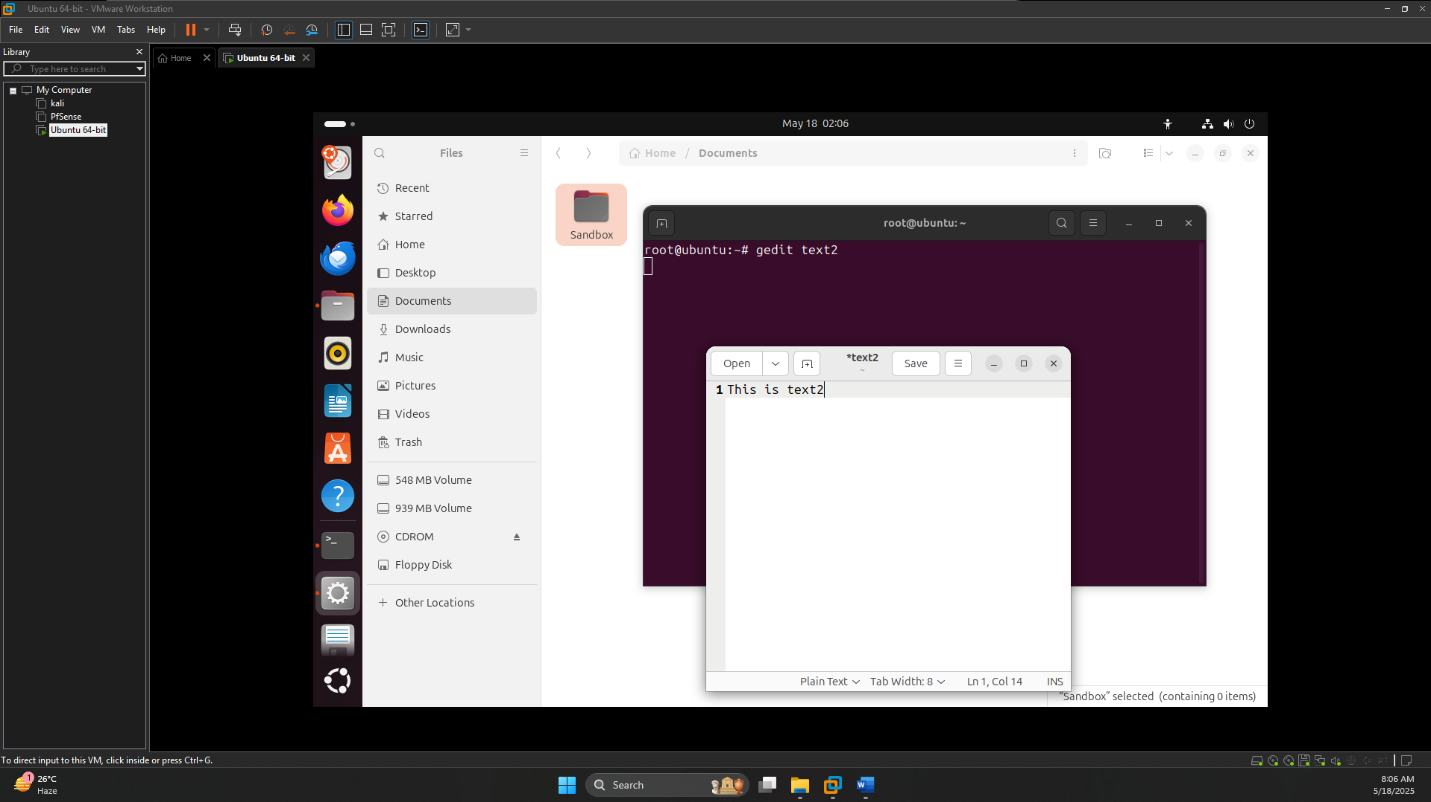
Both files are empty (0 bytes), as expected. Next, I need to add content to them. I open text1 in the gedit text editor by typing: *gedit text1*

In gedit, I type exactly: This is text1 file



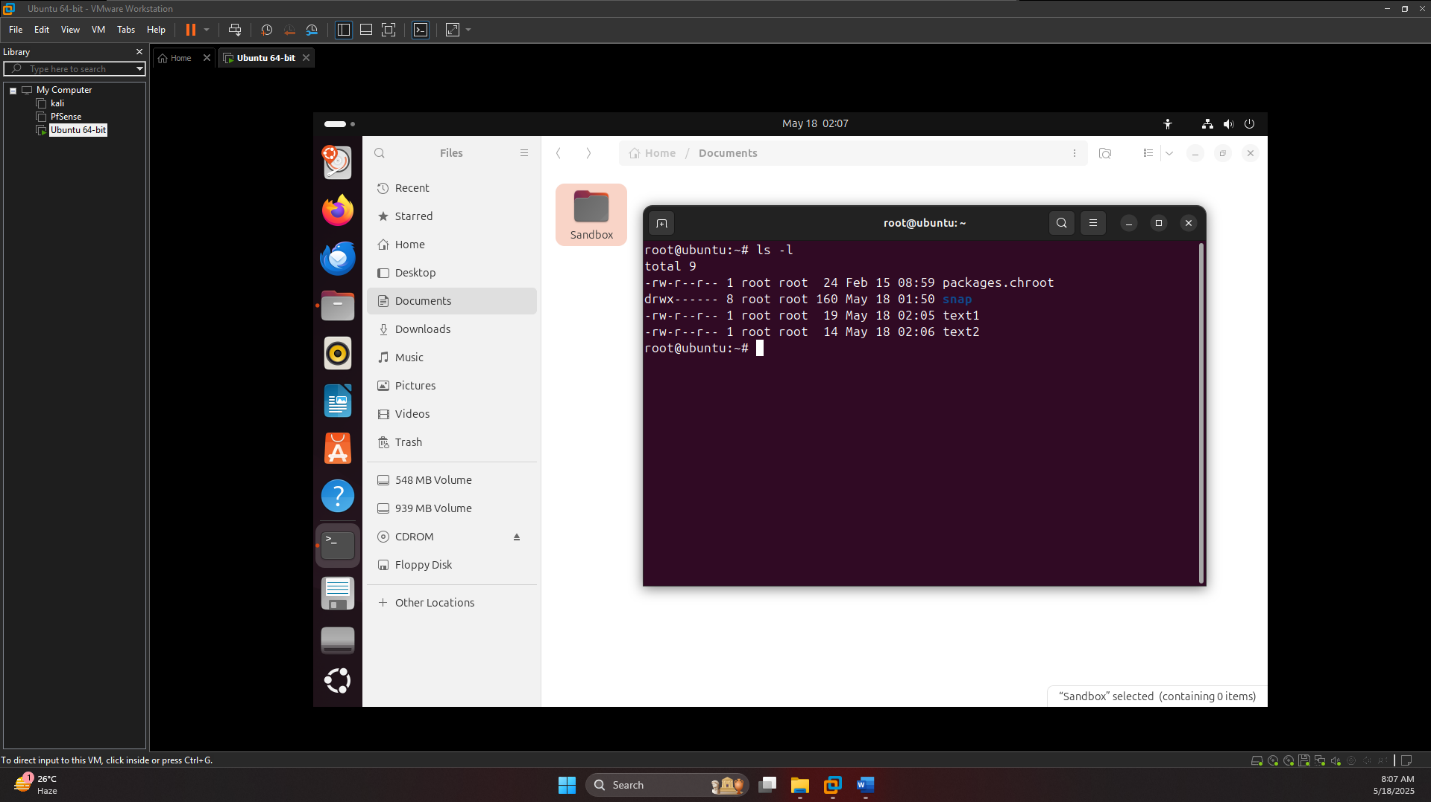
I saved the file and close gedit. Then, I do the same for text2: *gedit text2*

I typed: This is text2



I save and close gedit again. To verify the files now have content, I checked their sizes: *ls -l*

The output shows:

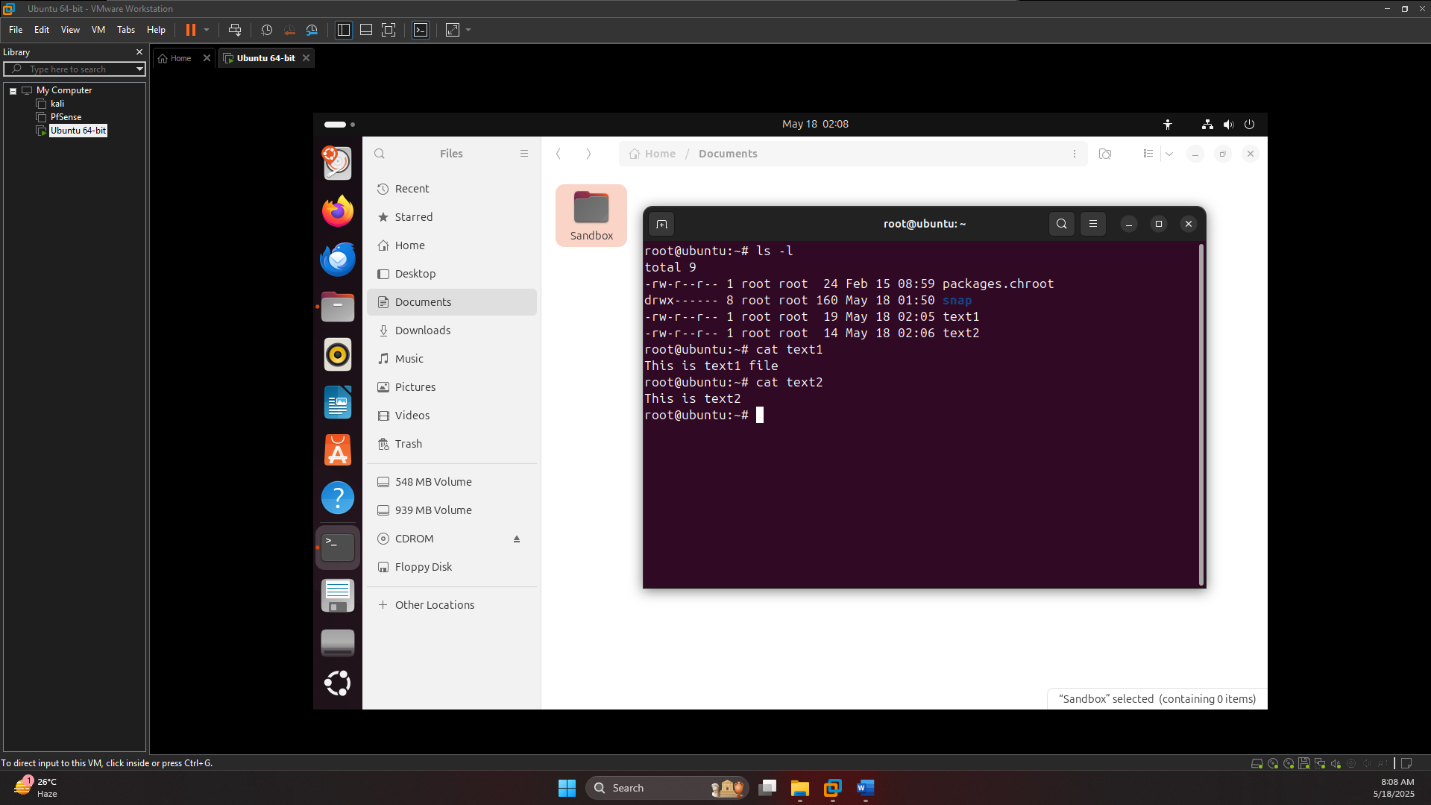


Confirming I added the text correctly. I also view their contents to double-check:

*cat text1*

*cat text2*

I see:



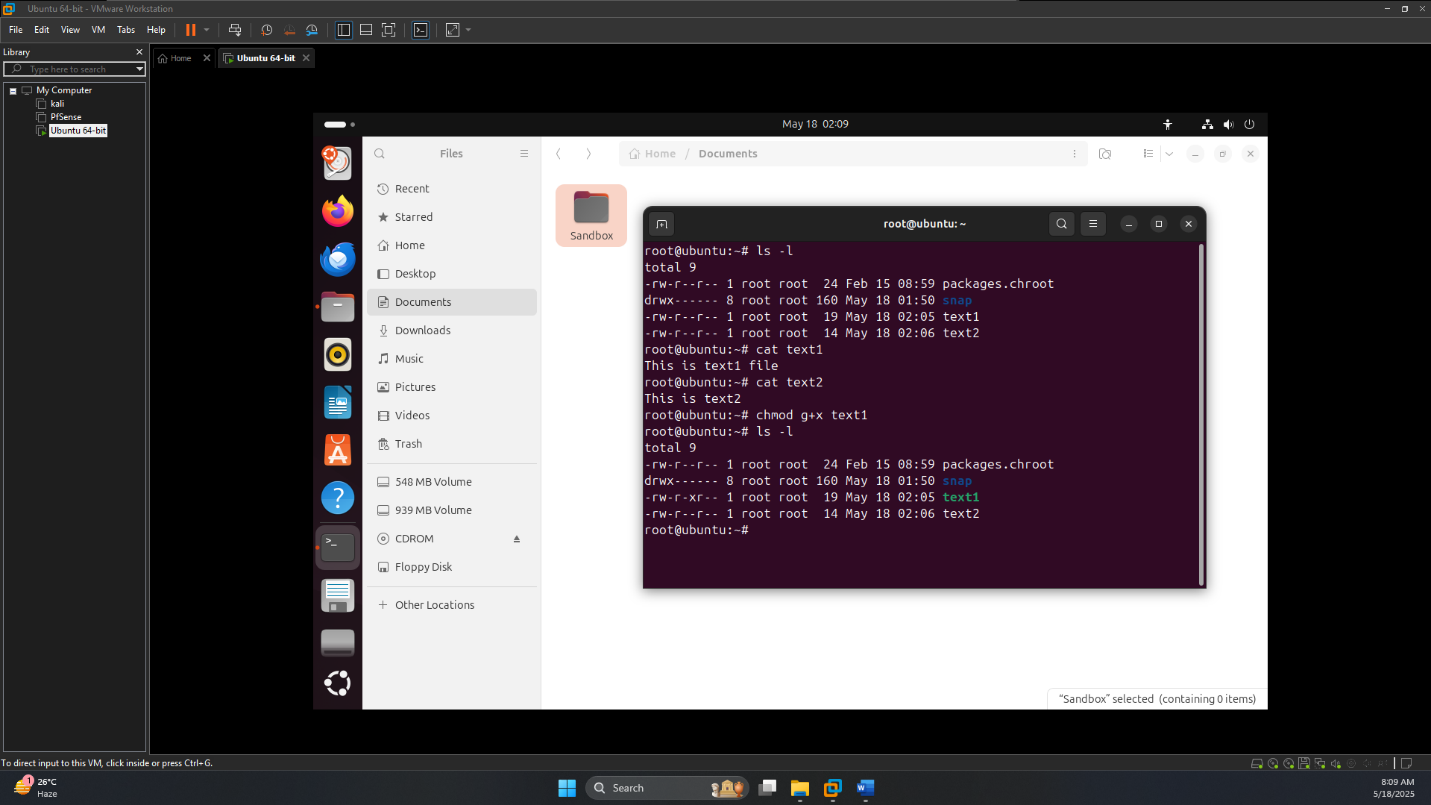
**2. Changing File Permissions**

The lab asks me to modify the access control for text1 by giving the group execute permissions. I know text1 isn’t an executable file, but this is just an example. I run:

*chmod g+x text1*

To check the change, I list the files again: *ls -l*

The output is:



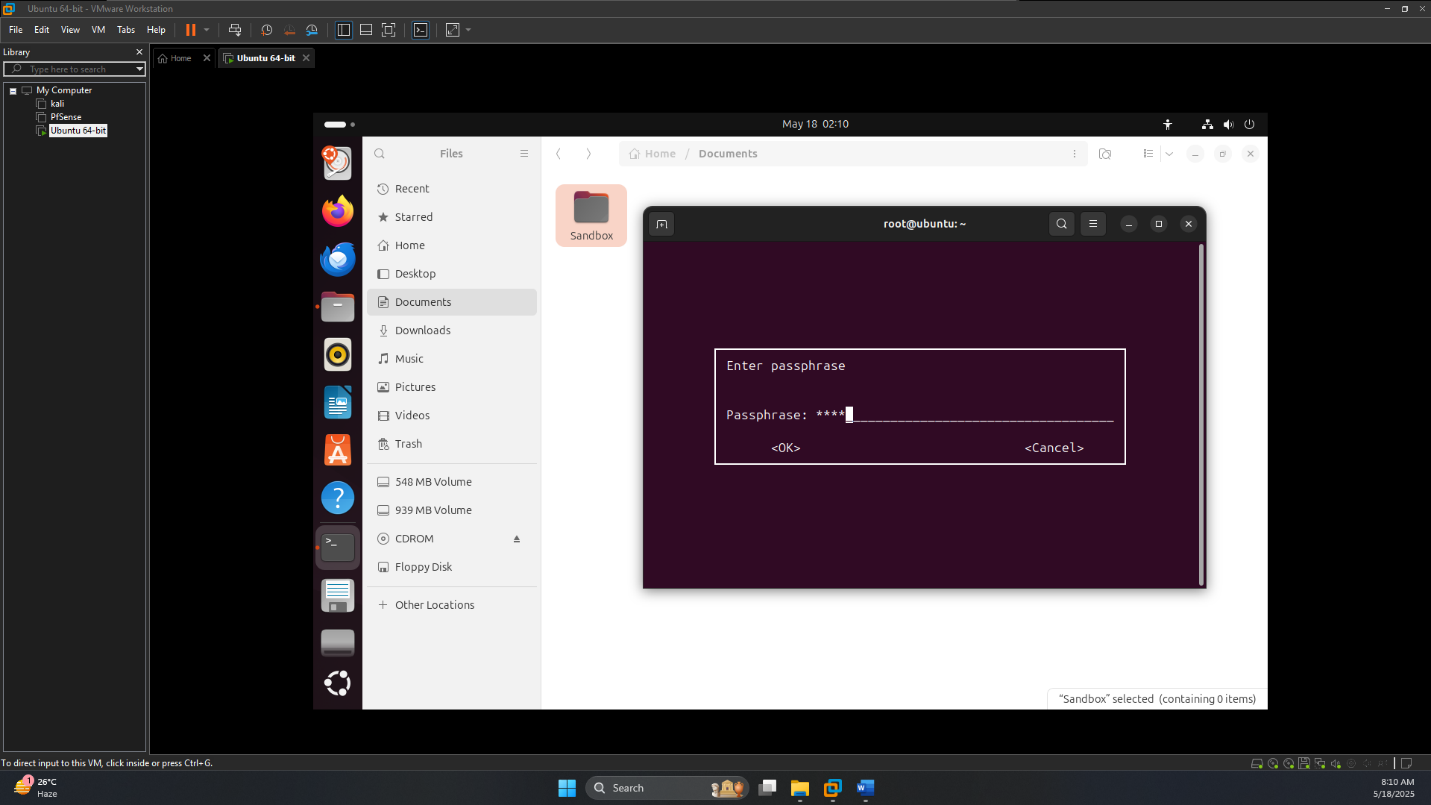
I notice the group permissions for text1 now include x (r-xr), indicating execute permission. In my terminal, text1 appear in a green color to show it’s “executable,” as the lab mentions.

**3. Encrypting and Decrypting with GPG**

Now, I am moving on to encryption using GNU Privacy Guard (GPG). First, I explore the gpg command by checking its manual: *man gpg*

I skim through to understand the -c option (for symmetric encryption) and the -d option (for decryption). I exit the manual by pressing q.

I’m ready to encrypt text2. I use the following command: *gpg -c text2*



I’m prompted to enter a passphrase. The lab suggests using “password,” so I type password and confirm it by entering it again. This creates an encrypted file called text2.gpg. I check the directory: *ls -l*

I see text2.gpg is 94 bytes, larger than text2 due to encryption overhead. To confirm it’s encrypted, I tried to view its contents: *cat text2.gpg*

The output is unreadable gibberish, as expected, since it’s encrypted binary data.

Next, I decrypt text2.gpg to verify I can recover the original content: *gpg -d text2.gpg*

I enter the passphrase password when prompted, and I see:

A computer screen shot of a computer screen

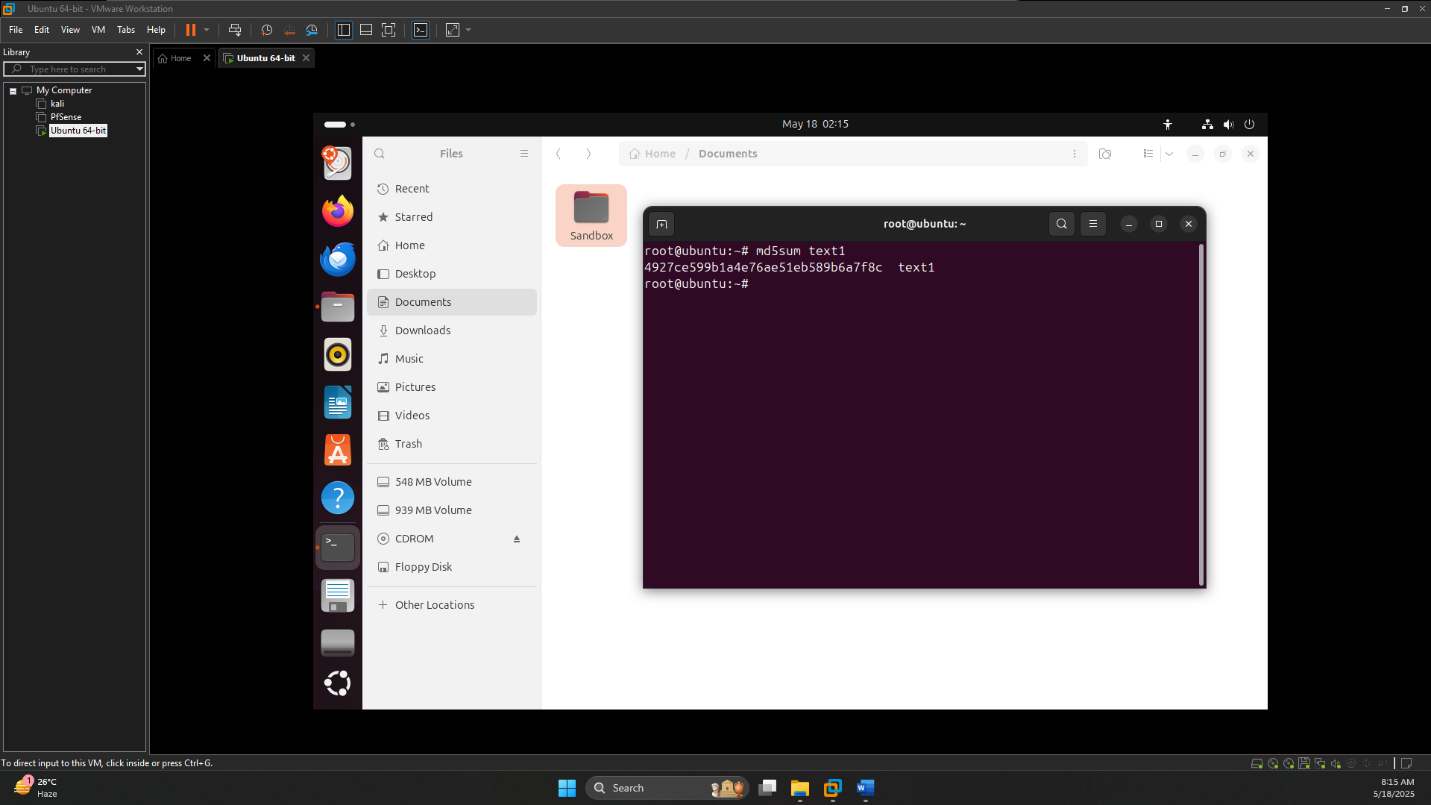
AI-generated content may be incorrect.

The decrypted content displays correctly, confirming the encryption and decryption worked.

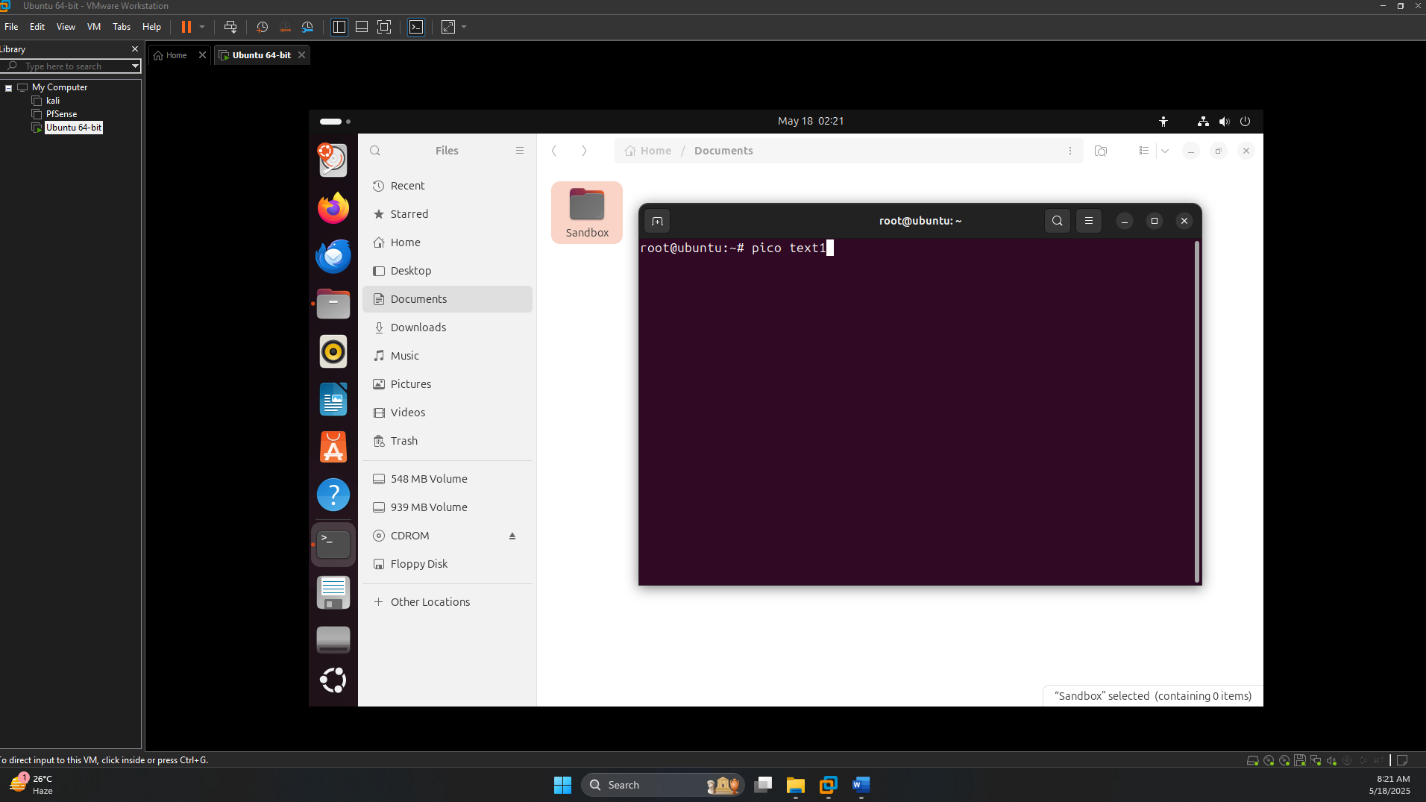
**4. Hashing with MD5**

For the hashing part, I calculate the MD5 hash of text1: *md5sum text1*

I get:



Now, I modify text1 by adding a period at the end of its content. The lab suggests using the pico editor, so I open text1: *pico text1*



I changed the content to:

This is text1 file.

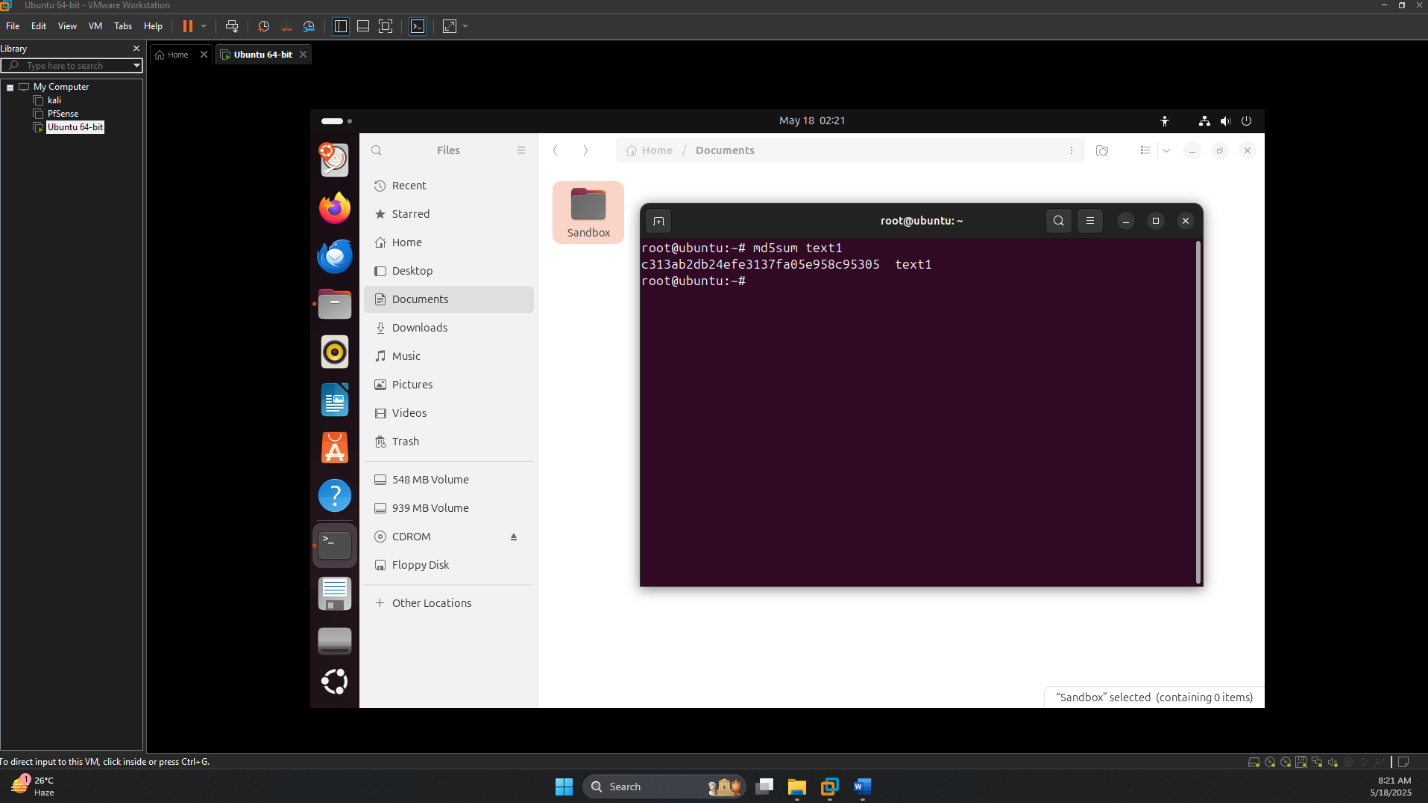
A computer screen shot of a computer

AI-generated content may be incorrect.

I saved the file (Ctrl+O, Enter) and exit (Ctrl+X). To see the effect of this change, I recalculate the MD5 hash:

md5sum text1

The hash is different now, showing that even a small change alters the hash.



I looked at the Bugs section and noted that MD5 is considered cryptographically broken due to collision vulnerabilities, meaning it’s not secure for critical tasks.

**Answering Lab Questions**

* **Q1: What permissions do the user have to the two files?**  
  As the user (root), I have read, write, and execute permissions (rwx) for text1 after the chmod g+x command, since the user permissions remain rw- and I’m root. For text2, I have read and written permissions (rw-).
  + text1: read, write, execute
  + text2: read, write
* **Q2: What permissions do group members have to the two files?**  
  For text1, I gave the group read and execute permissions (r-x) with chmod g+x. For text2, the group has only read permissions (r--).
  + text1: read, execute
  + text2: read
* **Q3: Were you able to decrypt the file?**  
  Yup, I blasted text2.gpg with gpg -d text2.gpg and got this “This is text2” on the screen from content.
* **Q4: What encryption algorithm was used?**  
  When I decrypted the file, the output showed me “AES256 encrypted data.”
* **Q5: Is this symmetrical or asymmetrical encryption?**  
  I used gpg -c as I needed symmetric encryption (one passphrase for both encryption and decryption). That is why symmetric encryption.
* **Q6: Briefly explain the benefits of using a hash.**  
  Hashing was found to be my Hash does indeed return unique, fixed length strings based off data which I can use for file integrity (like I edited text1) It also serves to verify downloaded files and safe password saving securely. It verifies the integrity of data by detecting changes, confirming file authenticity and hash passwords.
* **Q7: Briefly explain the difference between hashing and encryption operations.**  
  Hash is a one-way process to create irreversible hash for verification md5sum. Encryption (again, like my GPG task) is reversible. I can encrypt and decrypt data using a key or passphrase. Hashing is one-way like a fingerprint; you cannot reverse it for verification, and encryption is reversible, which encrypts the data with a key.
* **Q8: Examine the output of the man md5sum command. Should this type of hashing be used? Why or why not?**  
  When I read man md5sum, the Bugs part advised me that MD5 is broken because collisions exist → the same hash value can be generated from different inputs. I believe it could be dangerous for security-critical things such as digital signatures, though I guess fine for basic checksums. MD5 shouldn't be used for anything security-related because it is broken and can be used to produce collisions, but it may be OK for simple integrity checks.