CPE 453: Operating Systems
Lab 2: Setting Up Minix
Advika Deodhar and Daniel Gutierrez

Executive Summary: In this lab, we were able to set up the Minix operating system on our machines through a Virtual Machine. We then simulated using a physical floppy disk to transfer data from our host machines to the Virtual Machine that is running Minix.

Task 1: Getting a Minix System Set Up

Approach:

- The first step of the lab was to get a Virtual Machine set up so that we could run Minix on our respective computers without damaging our own files and OS.
- To do so, we both took the route of QEMU, which is an open-source x86 simulator.
- Danny had the VM downloaded already on his machine, the INSERT MACHINE
- Advika completed the lab on her Mac, so to get QEMU, she went on homebrew and typed the command "`brew install qemu`" which installed it onto her computer.
- Once we had our environment set up, we had to download the actual operating system Minix
- A "disk image" of Minix was provided on the class website. We downloaded it onto the computer in a file titled CS453Minix.
- For QEMU, the next step to boot up MINIX was to type in the command: "`/qemu-system-i386 -rtc base=utc -net user -net nic -m 256 -hda minix 3 1 8.vmdk`"

Problems Encountered:

- Was a little confused about what a disk image was, and also about how the floppy disk image would help communicate with the OS

Solutions:

- Did a little research on these terms to clear up conceptual confusion.

Lessons Learned:

- A "disk image" is a single file that represents all the contents of any storage device.
- It contains an exact replica of that device on it, which is why it is called an "image"
- In the Minix context, the Minix system was stored as a disk image file, or a .vmdk, virtual machine disk, and included the entire operating system, the expected Minix file structure, and instructions for booting the OS.

Task 2/3: Logging In and Creating a User Account

Approach:

- Once Minix booted up on our VM, we had to log in and make an account because the default user upon start is root, and it is not safe to always operate as root.
- We used the functions adduser(), and passwd() to create a user. We also read the man pages for these commands to learn how they worked. Here is what we gathered:

  login()
    - Allows for a user to log in as someone else
    - You do not have to log out first
    - If there is a password needed, login will ask for it

  passwd()
    - Used to change password
    - Asks for the old password then the new
    - Will ask for the new password twice

  adduser()
    - Adds a new user to the system
    - Makes new entries by creating a new home directory
    - Copies the contents of the template home directory into it

  su()
    - Used to temporarily run a shell under the identity of the superuser or another user
    - Calls that need a password are logged
    - Default user is root
    - Activities are logged through syslog

Problems Encountered:

- None, pretty straightforward

Solutions:

- N/A

Lessons Learned:

- N/A

Task 4: Create a Minix Disk Server and Use it to Store Data

Approach:

- Since Qemu does not have the option to create a floppy drive like VirtualBox, we had to make this file on the host's file system.
- In the same folder that we downloaded the Minix disk image, we created another disk image file and named it "testfloppy" by using the command provided in the lab spec `"dd if=/dev/zero of=testfloppy.img bs=1024 count=1440"`
- Next, we had to tell Qemu where to find the floppy disk file, since it does not recognize files on the host computer without being told.
- We needed to "configure" the floppy drive, and tell Qemu that the file is a floppy drive so that Minix will see it as a real floppy drive and be able to mount/read/write to it.
- Entered the command given in the spec to configure the file `"-drive file=testfloppy,interface=floppy,format=raw,index=0"`

Problems Encountered:

- Was a little confused about how the floppy drive would interact with the host OS and Minix
- When configuring the floppy drive, my Mac terminal did not recognize the commands -drive and -fda for the two commands we were supposed to type in.

Solutions:

- Did some reading on how floppy disks used to work and what functionality we are trying to replicate
- Got around the commands not being recognized by providing the full path to my testfloppy.img

Lessons Learned:

- Floppy disks in older computers were used to transfer data between computers. You would plug in the floppy disk, then the OS of that computer would "mount" it, and make it so that the floppy disk has access to the file system. You could then write to or read from the disk, "unmount" it when you are done and take out the floppy disk.
- We essentially are doing the same thing but instead of transferring data between two separate devices, we are using a floppy disk image to replicate the physical floppy disk. The host OS and Minix are both on the same device but are mounting and unmounting the floppy disk image to communicate.

Task 5: Mounting it on the Minix Side

Approach:

- First, we read the man pages for the calls format, mkfs, fd, mount, and umount, and gathered the following:

  format()
    - Allows a user with read-write permissions to format a floppy disk
    - The size of the floppy is specified on the command line

  mkfs()
    - Makes a file system
    - Builds the whole file system
    - Prototype file which directories and file to copy into this system
    - Max size is 1 GB for a version 2 file system
    - 64mb for version 1 file system
    - First couple of entries show the name that will be on the new file system
    - The mode is next, -dbc is for regular files directories and block special files and character special files.
    - Next characters are for SETUID and SETGID bits

  fd()
    - Floppy disk
    - Refers to the floppy disk driver
    - A diskette is an array of 512-byte sectors
    - Driver is configured for 2 floppy disk drives, fd0 and fd1

  mount()
    - Mount a file system
    - The file system is mounted on file.

  unmount()
    - Unmount a mounted file system
    - Already knows where to unmount it from do not need to specify a location

- We needed to create a filesystem on the floppy disk so that Minix knew how to read and write data on it.
- We called format to prepare the disk, then mkfs.mfs to make the file system on there, and specified the driver to be fd0.
- We then called "`mount /dev/fd0  /mnt/floppy`" to mount the floppy disk on the Minix side
- Once successfully mounted, we were able to create text files within that file system, and add text to them. Advika created a text file named random.txt with the message "hello world" within it.

- Once we put text inside the floppy drive, we needed to unmount it to make sure that data did not get stuck in the cache or corrupted.
- We used the command "`umount /dev/fd0`" to unmount.

Problems Encountered: encountered some issues when trying to mount the file system

- Tried doing `mount/dev/fd`
- Mount `/dev/fd0 mnt/floppy`
- Got the output that "`Can't mount /dev/fd0 on /mnt/floppy: no such file or directory`"
- `Mkfs.mfs /dev/fd0` kept giving us the error not found

Solutions:

- `Mkdir -p /mnt/floppy` - tried that from stack exchange
- format `/dev/fd0 1440`
  `Mkfs.mfs /dev/fd0`
  `sudo mount -t minix ./testfloppy /mnt/floopy/`
  `mount /dev/fd0/mnt/floopy`
- ^ That ended being the correct order
- Realized we needed to be in root to execute these so we had to use sudo
- `Mkfs /dev/fd0` was not working, so we tried logging out of root and logging in again and it worked

Lessons Learned:

- The hint was in the lab footnotes to include .mfs. Must read the footnotes closer!

Task 6: Accessing the Data from the Host

Approach:

- To read the data that is on the floppy, we needed to mount it onto the host computer so that we could read from it.
- For Danny, this mounting process worked through his terminal since he is on a Linux distro.
- For Advika, she ended up logging into the unix servers and copying her floppy disk file onto there, and then running the programs minget and minls to read the file.
- Did "`scp testfloppy unix3:/home/amdeodha`" to copy over the floppy disk to the Unix servers
- Then on the Unix servers, did "`./minls testfloppy`" and "`./minget testfloppy random.txt`" and was able to see the "`hello world`" that was written in there through Minix.

Problems Encountered:

- Advika could not mount from her host terminal at all
- Kept getting the error "`failed with 72`"
- Tried to use /Volumes since we read that MacOs does not have the /mnt folder as its mount point
- Danny kept getting the error "`mount: /mnt/floppy: wrong fs type, bad option, bad superblock on /dev/loop15, missing codepage or helper program, or other error`"
- While trying to figure out the networking, it enabled some "dhcp networking" on the Minix side. Could not boot up Minix anymore because it was taking forever and getting stuck.
- Would not run minls and minget on the host computer since they are Unix commands

Solutions:

- Mounting did not work directly through the MacOs terminal, so Advika tried to do it by following the networking steps on the Lab Spec.
- Went to office hours to ask about the networking error, and was told to either wait for it to time out or to try another way. Ended up taking a different route of using the programs miget and minls.
- Uninstalled Qemu and reinstalled it, also deleted the Minix disk image and redownloaded another version. That ended up working
- Instead of copying over the programs minget and minls from the Unix servers, had to copy my floppy disk onto the Unix servers since the midget and minls programs could not be run in the MacOs host terminal.
- Danny fixed the mount errors by running through the mount commands again from scratch.

Lessons Learned:

- MacOs does not have the "mount" command at all since that is a Minix command

- MacOs cannot recognize the commands “minget” and “minls” because those are also Unix commands