Laboratory Exercise 2

cpe 453 Winter 2023

Due date TBA.
The Laboratory Exercises may be done with a partner.

Laboratory Exercises: A MINIX Scavenger Hunt

Well, sort of. It is designed to introduce you to working in the MINIX environment. It will consist of a series of tasks to familiarize you with processes from logging in to transfering data to and from a MINIX system. The collective purpose is to provide the foundation necessary for doing serious kernel hacking in the future.

Performing some of these tasks will require a certain amount of sleuthing: reading of man pages and other documentation (starting with the rest of this assignment).

The deliverable will be a lab report (see below) describing approaches taken, problems encountered, solutions developed, and lessons learned.

Note: I'm allowing partnerships for this lab because (a) it's more fun and (b) it prevents many bonehead mistakes. Be sure, however, that each partner has a full understanding of everything that is accomplished.

Also note: There are many useful footnotes in this document. Actually read them.

The List

1. **Get a MINIX system.**

   There are two options you have for getting your very own personal MINIX system on which to work:

   (a) Install it on a real partition on your personal computer. (**dangerous**)

   > **Only do this if you are sure you know what you are doing. You risk losing all data on your hard drive if you make a mistake.**

   That said, installing MINIX is fairly painless. If you have a free partition of at least 100MB, follow the instructions in Appendix A of the Tanenbaum textbook and you'll be ready to go.

   If you choose this option, it would be a good idea to work with someone who has previously installed a Unix operating system.

   (b) Install an x86 architecture simulator: (**preferred**)

   With a simulated machine, you can have as many MINIX installations as you like safely isolated from your own machine.

i. First, get a simulator. You can use `VMWare` on the CSL machines, or you can install your own simulator on your own machine. Since `VMWare` is a commercial, non-free, program, there are three free options I can recommend in addition:

VirtualBox, is a free x86 architecture simulator (`https://www.virtualbox.org/`) produced by Oracle. You can download pre-built versions for Linux, OSX, and Windows. It understands the .vmdk disk format[1].
This is my current personal favorite combining "free" with "configuring the network usually works."

VMWare emulates the entire PC machine, so that MINIX (or any other operating system for that matter) will behave exactly as if it were on a bare machine. It's installed on the CSL machines. The only problem with using it at home is that it's not free.
If you choose VMWare, you can also use the extra space in the `/vm` partition of the machines in 232a. This is a large workspace, but you'll have to choose a particular machine since it's local to that machine.

VMWare Player is a free version of VMWare's Workstation PC emulator. This is a reliable product available for download from
`http://www.vmware.com/products/player/`
The only disadvantage is that, when I last checked, VMWare Player can *only* run in Intel CPUs. If you have a one, you're in luck. If not, keep reading.

QEMU is free open-source x86 architecture simulator that runs under Linux, Solaris, Windows, Mac OS, and just about anything else for which you can compile it (and if you're lucky you won't have to compile it; You may manage to just get it directly using `yum(1)` or equivalent).
QEMU is available for download from `http://www.qemu.org/` as source or binary packages.
QEMU has the advantage that it understands the compressed disk image format used by VMWare so you can use the pre-built images designed for VMWare, as well as build your own.
I've installed a version that should run on the CSL machines in `~pn-cs453/QEMU`, called, would you believe "`qemu`."

Bochs, like the others, is a free x86 architecture simulator that will run anywhere you can build it.
The main web site for bochs is `http://bochs.sourceforge.net/`. Precompiled versions of bochs as well as prebuilt disk images for various operating systems are available at `http://sourceforge.net/project/showfiles.php?group_id=12580`. (Unfortunately the MINIX images are out of date.) Of equal importance is the bochs users manual—complete with installation and configuration instructions—available at
`http://bochs.sourceforge.net/doc/docbook/user/`.
Go to the above site and get a copy of bochs. If you can't find a copy pre-compiled for your favorite platform, read the users manual entry on your platform to see how to build it. I have built bochs from source under Linux[2] and Solaris with no

[1] Just be sure that your image has read/write permissions. I just spent an hour and a half chasing that one...

[2] **Linux note:** installing the X windows version of some versions of bochs requires you to install the vga fonts. This is easily done if you follow the instructions in the users manual. (You *did* read the manual before installing it, didn't you?). If you install these fonts globally as root, be careful that the font file, `vga.pcf.gz` is world readable and that the recompiled `fonts.dir` files are, too. This is not a problem if you install them in a local (user) directory.

problems. The Windows version is a self-extracting zipfile.
Note: The bochs clock tends to run fast. You may need to change the configuration file to reflect the real speed of your machine. You can either set the instructions per second (ips) field to an appropriate value, e.g.:

```
ips:  67500000
```

or use experimental synchronization feature. Comment out the ips line and add:

```
clock:  sync=realtime
```

> **Note**
> If you are repeatedly failing to get minix to run on whichever emulator you try, you may need to enable virtualization in your computer's BIOS. If that's turned off (as is is by default on many machines) you're doomed.

ii. **Now, get** MINIX
To make this easier, I have provided an image[3] of version 3.1.8 linked from the class web page. This isn't quite the same as the book version, but has better support for networking, etc.
To use this version with VirtualBox, create a new machine, and select "Linux" as the type, and "Other Linux (32-bit)" for the version. When asked to create a disk, choose "Use an existing virtual hard disk file" and select the `minix_3_1_8.vmdk` file you downloaded. All the other defaults should work.
To use this version with VMware is a little trickier since it doesn't offer the option to provide your own disk. To use my `minix_3_1_8.vmdk` with vmware it's something like:

A. create a new VM in vmware
B. when it asks what sort of disk, select the "single file" option
C. tell it you'll install the OS later.
D. shut down the VM
E. Find where vmware put the virtual machine's files (it'll vary depending on the system).
F. In there somewhere will be a file named "something.vmdk". This is the disk image
G. Replace "something.vmdk" with my minix disk image

If you really, really want to install your own (why?), keep reading this section...
At this point, create a virtual disk for your simulated PC and follow the instructions for installing MINIX on any machine. For VMWare or VMWare Player, you can find these instructions at:
`http://wiki.minix3.org/doku.php?id=usersguide:runningonvmware`
The images you need you will find at `http://www.minix3.org/download` or `http://www.minix3.org/previous-versions/`. The version in the book is 3.1.0. Newer versions have more features, but they have also changed more from the version described in your text.
Note: Resist the urge to download the latest and greatest alpha versions of the operating system. These are alpha for a reason and are often unstable. You want

[3]Compressed. Be sure to `gunzip` first.

to be working on as stable a platform as you can get (so you can know whether any strangeness is your fault).

iii. **And you're off**
Run your simulator, load the MINIX configuration file, and hit "power on"...
To run the provided version with `qemu`, the command would be:

```
~pn-cs453/QEMU/bin/qemu-system-i386 -rtc base=utc -net user -net nic -m 256 -hda minix_3_1_8.vmdk
```

To run it using VMWare or VirtualBox, start the emulator as appropriate.

Regardless of the approach you choose, be sure to read the following man pages:

usage(8)	MINIX configuration and usage guide. This is also included on the CD-ROM in `MINIX/INSTALL.TXT` so you can read the installation instructions *before* installing the system.
monitor(8)	describes the MINIX boot monitor process
boot(8)	describes the MINIX boot procedure

2. **Log in**

Once you've started up your system, boot it and log in. Your first login will probably be as `root`, but in general you don't want to work with superuser privileges.

Be sure to read the following man pages:

getty(8)	the program that waits for a user to log in
login(1)	the program that does the actual logging in

3. **Create a user account**

In general, it is inadvisable (read "dangerous") to work all the time with superuser privileges. One slip of the keyboard can delete half the system and radically change your weekend plans. The proper way to work is to do everything possible as an ordinary user and only become root (using `su(1)`) temporarily for the few tasks that must actually be performed with privilege.

For this task, create a user account for yourself.

Be sure to read the following man pages:

passwd(1)	change the password on a user account. See also passwd(5) which contains information on the format and contents of the passwd and group files.
adduser(8)	a program to make the process of adding user accounts easier
su(1)	the Set User program

4. **Create a MINIX disk image and use it to store data**

In this section you'll create a virtual disk drive, create a filesystem on it, and mount it on your MINIX system. We're going to emulate a floppy drive because that interface is simpler (there's usually no partitioning), but there's no reason you couldn't make a second hard disk.

The idea here is that we're going to tell the guest operating system that there's a disk in the first floppy drive. To do this, the emulator will need someplace to store the data that's supposed to be on the pretend floppy. For this, we provide a file in the host's filesystem to hold the data.

VirtualBox will ask you if you want to create the file when you configure it to have a floppy controller and floppy drive (see below). For the others you'll have to create the file.

- First, create an empty disk image file. It doesn't matter how you do this, but it needs to be a 1.44MB file. An easy way to do this under *nix is:

  ```
  dd if=/dev/zero of=testfloppy bs=1024 count=1440
  ```

  This file goes in the host filesystem—outside of MINIX—since it's the storage space that the emulator is going to use for the simulated floppy disk. (VirtualBox and VMWare insist on the filename ending with "`.img`" or "`.flp`." Do it their way.)

- Configure your "floppy" drive:

  Note, the configuration file that comes with the MINIX disk image does not have any floppy drives configured. To use a floppy drive, you will need to tell your simulator where it is.

  In all cases below, the simulated floppy can either be the device name of the real floppy ("`/dev/fd0`" or "`A:`")—c'mon, you don't really have a real floppy—or the name of a 1.44Mb file to be used as the storage space for a pretend floppy.

  - VMWare and VMWare Player: You should be able to do it from the menus.
  - QEMU:
    This is traditionally done with a simple command-line option: "`-fda testfloppy`"
    In the very latest version, this has become the truly inaesthetic:
    "`-drive file=testfloppy,interface=floppy,format=raw,index=0`"
    but the old one still works.
  - VirtualBox: under "**Settings→Storage**", click on the add controller icon at the bottom and select "Add Floppy Controller". Once you've done that, you can select it, click the icon that looks like a floppy and create a new floppy disk. VirtualBox seems to insist that your floppy file end in "`.img`" or possibly "`.flp`".
  - Bochs:
    To enable bochs to use your floppy you will have to tell it where it is by adding a line to the configuration file:

    For Windows: `floppya: 1_44=A:,status=inserted`
    For Linux: `floppya: 1_44=/dev/fd0,status=inserted`

    It is also possible to attach the bochs floppy drive to a file if you wish by replacing "`/dev/fd0`" with "`testfloppy`".

- Now create a filesystem on it so you can use it[4].

  Be sure to read the following man pages:

format(1)	describes how to format a disk device to prepare it to take a filesystem.
mkfs(1)	describes how to create a MINIX filesystem on a device
fd(4)	More information about floppy devices under MINIX
mount(1)	the command to mount a disk on the filesystem
umount(1)	the command to unmount a portion of the filesystem

[4]The benefit of knowing how to create a filesystem will become evident when you want to test your filesystem readers later in the quarter.

Now, using `format(1)`[5] and `mkfs(1)`[6] create a MINIX filesystem and `mount(1)` it as part of the filesystem. The mount point must be a directory. The customary place to mount filesystems temporarily is on subdirectories of the `/mnt` directory. For example, mounting a temporary floppy disk on `/mnt/floppy` while you are using it.

Once you have created a floppy filesystem and mounted it, it's part of your MINIX system (until you remove it). Do be sure to `umount(1)` it before removing it from the drive. *Be sure never to remove the media for a mounted filesystem without umounting it: parts of the filesystem may still be in cache, leaving the filesystem on the disk incomplete or inconsistent.*

5. **Accessing your data from outside** MINIX

   There are various options you can take for getting data on or off of your MINIX system:

   (a) Reading the filesystem directly.

   If you are lucky enough to have a linux system with the MINIX filesystem module installed and you have root access, you can mount it directly. If not, keep reading.

   With your MINIX filesystem in hand (in disk?), go to a Linux box, either at home or in the lab, and mount it there. The Linux version of mount supports the MINIX filesystem, but it is necessary to tell it what kind of filesystem it is. Something like:

   mount -t minix /dev/fd0 /mnt/floppy

   Of course, that won't work if you don't have a real floppy drive, and you don't. To do it with a floppy disk image file via the loopback device (loop(4), if interested), you do:

   mount -t minix TestImage /mnt/floppy

   This is a fairly simple way to get data on and off MINIX systems[7].

   If you're using a native MINIX system on a machine without a floppy, good luck.

   If you are using bochs with Windows as the host operating system, you might want to look into Mintools, a set of utilities for accessing Minix filesystems from a DOS system. For more information, see `http://minix1.hampshire.edu/faq/mintools.html`.

   (b) **What if you can't mount it?**

   If you are unable to mount your MINIX floppy image on a linux host due to permissions, I have made a preview of a MINIX filesystem reader assignment available in `~pn-cs453/demos`. This consists of two programs, `minls` and `minget`:

   - `minls [ -v ] [ -p num [ -s num ] ] imagefile [ path ]`
     List files on a minix filesystem.

```
Options:
       -p       part    --- select partition for filesystem (default: none)
       -s       sub     --- select subpartition for filesystem (default: none)
       -h       help    --- print usage information and exit
       -v       verbose --- increase verbosity level
```

   - `minget[ -v ] [ -p num [ -s num ] ] imagefile minixpath [ hostpath ]`
     Read a file from a minix filesystem.

[5] If you didn't listen and installed a version other than 3.1.8, you'll find that there's an error in the floppy driver that causes `format` not to work. `mkfs(1)` still does. After all, you're "formatting" a pretend disk, not a real one.

[6] Called `mkfs.mfs` on 3.1.8

[7] Or set up the network. See item 5c.

```
Options:
       -p       part    --- select partition for filesystem (default: none)
       -s       sub     --- select subpartition for filesystem (default: none)
       -h       help    --- print usage information and exit
       -v       verbose --- increase verbosity level
```

An example of using `minls` and `minget` can be seen in Figure 1.

```
-bash-4.2$ ls -l TestImage
-r--r--r--. 1 pn-cs453 domain users 1474560 Jan  9  2015 TestImage
-bash-4.2$ minls TestImage
/:
drwxrwxrwx       384 .
drwxrwxrwx       384 ..
-rw-r--r--     73991 Other
drwxr-xr-x      3200 src
-rw-r--r--        11 Hello
-bash-4.2$
-bash-4.2$ minget TestImage Hello
Hi, there.
-bash-4.2$ minget TestImage Hello foo
-bash-4.2$ cat foo
Hi, there.
-bash-4.2$
```

Figure 1: Using `minls` and `minget` to read a filesystem image

(c) **Or set up networking**

If your emulator emulates a network card that MINIX understands, you're in luck. Just use `ssh` and `scp`. The image provided on the class web page has OpenSSH installed[8]. When run with the command line arguments given above it should be possible for you to simply `ssh` or `scp` to any system you can name.

The command to configure networking on MINIX is `netconf`, which must be run as root.

(d) **As a last resort: (Dirty tricks)**

As you know, the MINIX operating system has just been updated from version 2.*X*.*X* to version 3.*X*.*X*. What you may not know is that the filesystem has changed and most of the machines that used to be able to mount version 2 of the filesystem have not yet been updated to handle version 3.

Here's where we step outside the filesystem and cheat. A disk, like a file, is simply a collection of data and it can be used to store data without having a filesystem on it. To do this, we will use `tar(1)`, the Unix Tape ARchive program, to write an archive to a raw disk image and read it on another machine. This can be done with either a disk image file or a real disk. Bear in mind that this will destroy any other information on the disk.

i. Insert a disk or attach a file, as above, but do not mkfs or mount it.

ii. From MINIX use `tar(1)` to create an archive on the raw disk:

```
% tar cvf /dev/fd0 file0 ... filen
```

[8]To find out how to install other packages, man `pkgin(1)`

iii. From the host machine (or another machine altogether) use `tar` to read the archive:

```
% tar xf diskimage
```

If your host machine is a Windows machine, some windows archiving programs speak the tar format. You may have to name your disk image `image.tar` for them to recognize it.

6. **Clean up and shut down**

   The last remaining task is to shut your system down cleanly. A unix-type operating system cannot be safely shutdown by simply powering it off[9]. It needs time to flush caches and leave the system in a consistent state for the next time. The proper way to shut down a unix system is using the `shutdown(8)` command.

   Be sure to read the man page for

shutdown(8)	the polite way to shut down or reboot a running unix system. For example, "`shutdown -h +10`" to halt the system in 10 minutes, or "`shutdown -r now`" to reboot immediately.

   then shut the system down.

What to turn in

For the Laboratory Exercises: a written laboratory report describing approaches taken, problems encountered, solutions developed, and lessons learned (as described below).

If working with a partner, turn in only one lab report, but be sure both partner's names are on it.

Report Format

The laboratory report should describe what you did and what you learned for each of the above tasks. For each task, you should include a section containing:

Approach This section should describe how to perform the given task.

Problems encountered This section should describe any particular problems encountered while implementing the plan described above.

Solutions This section should describe solutions to the problems described above.

Lessons learned when all is said and done, what were the lessons learned in this task.

Given the different natures of the various tasks, the task reports may vary widely. I expect there will be much more to explain for data transfer than for logging in.

In addition to the individual task reports, you may include any other introduction or conclusion material you deem necessary or appropriate.

Turn in one report per partnership, and be sure that both partners' names appear on it.

General expectations:

- The report should be well-written. That is, proper English grammar, spelling, punctuation, etc., are significant.

[9] Stay away from the little 'X' button in the window frame. This means you.

- The report should be legible. Unless you are an accomplished calligrapher, please type.

  If you want to use this as an excuse to learn LaTeX(1), come talk to me, or see: Lamport, Leslie. *LaTeX: A Document Preparation System (2nd edition),* Addison-Wesley, 1994.

- The report should contain all the pertinent specific details, e.g. the type of platform you are running (native vs. bochs vs. VMWare), etc.

- Never forget that the overall goal of the report is to convince me that you know what you are doing. Fulfilling the letter of the requirements for form but failing to convince the reader of competence is a failure.

A sample report section appears in Figure 2.

Laboratory Exercises

What to turn in

For the Laboratory Exercises: a written laboratory report describing approaches taken, problems encountered, solutions developed, and lessons learned.

If working with a partner, turn in only one lab report, but be sure both partner's names are on it.

2. Logging in

This does feel a little silly for this task, because instant success is expected, but it serves as an example.

Approach

The login procedure is to enter the username at the "`login:`" prompt followed by the password at the "`password:`" prompt. The username for the user account I created is "`pnico`". The password is not recorded here for obvious reasons.

Problems Encountered

I was distracted and hit too many returns to wake up the terminal, ended up entering the login name at the password prompt, and typing my password at the following login prompt:

```
login:
password:  pnico

login:  <password withheld>
password:
```

Solutions

The solution was to re-enter my login name and password at the appropriate prompts. I also checked to see if the bad login had been logged to `/usr/adm/log`, because that would have been a security risk. On this system bad logins are not logged, so there was no breach.

Lessons Learned

In this section I learned to pay attention while logging in to the machine. I also learned how hard it is to come up with bogus problems having to do with logging in. (Forgetting that UNIX is case sensitive is more likely. Still more likely is "none" for this particular task.)

Figure 2: A sample report section for task 2, logging in.