#### COEN 177: Operating Systems

**Lab 6: Minix Operating System**

**Objectives**

##### To setup a virtual machine and install Minix as a guest operating system

##### To understand Minix and Minix scheduler and modify and re-build Minix kernel

##### To modify the Minix scheduler and observe effects.

##### 

##### **Guidelines**

In this lab, you will be working on Minix operating system. Minix (mini-Unix) is a Unix-like operating system based on a microkernel architecture. Andrew S. Tanenbaum have created a number of MINIX versions for educational purposes[[1]](#footnote-1). These include Minix 1.0 in 1987, Minix 1.5 in 1991, Minix 2.0 in 1997, and Minix 3 in 2005. Minix 3 [[2]](#footnote-2) is not specifically educational but rather a highly reliable and self-healing microkernel OS. Minix 3 is a free and open-source software distributed under the BSD permissive free software license.

In this lab, you will install, run, and rebuild Minix. You will need a virtual machine on top of which Minix will run. Minix uses a multi-level priority queue scheduler, and modification of the scheduling algorithm, particularly changes to high priority queues, may make the system unstable. You will need to experiment with what changes can be done safely.

**Getting started with Minix on the ECC Systems**

1. [40 points] Install and run Minix on the ECC Systems, as described below, then write a description:

**$ setup vmware**  
This command prepares vmware to run. You should only need to run it once, until the next time you log into your machine. To obtain a copy of a Minix system image, you can use “minix-get-image”

**$ minix-get-image (can tab complete)**  
Please note that this will COMPLETELY ERASE any old image (including any changes you may have made which are not saved outside the Minix system). You must do this once, the first time you are setting up vmware. After that, you should only do this if you need a new image, such as if a modification you make causes your VM to break and become unresponsive. Next, you should run vmware.

**$ vmware &**

This may take a little while to boot up, so be patient. Don’t assume that it is not working.  
Once it boots up, if you don’t see the “minix” option on the left, go to:

• Open a Virtual Machine -> “vm images” subfolder -> “minix3” subfolder -> minix3.vmx

When you launch the minix system, you will eventually be promoted for a username. The username for the minix system is “root” (with no password, at least until you set one).

##### **Alternative Approach: Installing Minix under VirtualBox**

To install Minix on your own system, without using the prebuilt image provided, you can follow the instructions below instead. VirtualBox is a popular and freely available option, which is Open Source Software under the terms of the GNU General Public License.

1. Download VirtualBox from <https://www.virtualbox.org> and install on your computer to support virtualization.
2. Download Minix (minix\_R3.1.7-r7256.iso.gz) from <https://wiki.minix3.org/doku.php?id=www:download:previousversions>, decompress, and save as a .iso image file. Using this older version rather than 3.3.0 can hopefully help us avoid having to debug before we even start working.
3. Configure your Virtual Machine to boot from the downloaded ISO image with RAM and hard drive (**do not use the default settings from VirtualBox!).** Then login as root (no password required) and run the setup script. You most likely will use all default settings for the setup script.
4. Close the VM and delete the ISO image. Reboot from the installed Minix and then it should be ready for use. You may setup a password for the root user and create other users!
5. No need to use pkgin or Git. Simply **$cd /usr/src** to start work (after deleting ISO or file system will be read-only).

**Important Note:** the reboot command may not always work immediately to implement your changes after **$make world**. If this is the case, use **$shutdown**, power down the VM and start it back up to successfully boot with your changes. After that, when you use **$reboot** again, it should be with your changes.

When you are familiar with Minix, demo to the TA your Minix platform, disk size, memory available, utilities and applications, bootup process, etc. Plan your next steps to begin kernel hack and rebuild of a modified kernel: <https://wiki.minix3.org/doku.php?id=developersguide:rebuildingsystem>

When you modify OS modules (CPU scheduling, memory management, etc.) make sure you make a clean copy of the kernel source, by typing:

**$cp -r /usr/src /usr/src.clean**

In this case, you can retrieve the clean copy at any time.

**File Transfer**

To copy files between the Minix system (running in the virtual machine), and your local system, you may use FTP. Inside of the VM, do the following:

If you have not already, type “**passwd**” to give the system a password. Make sure it’s a password you’ll easily remember (ask yourself, how secure does this particular system need to be? What would it take for someone to access it?) (**Note:** If you wipe your minix image, you will need to do this again.)

Type the following command (again within the Minix system): **tcpd ftp /usr/bin/in.ftpd &**  
This will launch the FTP daemon in the Minix system, allowing the local system to connect to it through FTP.

Type “**ifconfig**” to find out what the IP address for the VM is.

Now get out of the VM (with Ctrl+Alt) and open a terminal on the host machine. In that terminal, type:  
   
**ftp <the IP address you got from ifconfig>**

This will launch FTP.

When FTP launches, it will take a few minutes at most to connect; once it connects, you’ll see something like this:

(....*yourname…)*:

**Type in your username and hit enter; as** a reminder, your username is root.

Then you’ll be prompted **to type in the password** you created when you set up your MINIX; if you didn’t prepare a password, go back and reread the start of the instructions.

In case you are unfamiliar with FTP commands:

*For Remote Navigation*

**ls** - show contents of remote directory   
**pwd** - show current location in VM   
**cd** - change directory on the VM

*For Local Navigation*

**lcd** - show current location on host machine   
**lcd <directory>** - change location to <directory>

*File Transfer*

**get <file>** - copy <file> from current location in VM to current location on host.   
**put <file>** - copy <file> from current location on host to current location in VM.

DO NOT work on files inside of the VM. You should use FTP to transfer files from the VM to the host, work on them there, then transfer them back. This is so that you keep all of your progress if you should make a mistake which crashes the VM and corrupts the bootable OS.

The files you must search in this lab can be found beneath the /usr/src directory.

The grep utility can be very helpful for finding which files contain particular strings, and should be very helpful in this lab.

**Recompilation**

Once you have modified a file and wish to see the effects of your changes, do the following:

Return to the /usr/src directory.

Type **make world**. This will recompile Minix, and may take some time, so be patient. When that finishes, type **reboot**. This will reboot the VM with your modifications in effect.

If your VM freezes/crashes during/after a reboot and you wish to restart with a fresh image and setup, start with the first instructions above to get a fresh copy (and note that it will overwrite your existing copy).

**Changing the Kernel**

1. [20 points] Locate the kernel source file that prints out the messages you see at system bootup. For example, the copyright statements that are printed. Then add your message, and rebuild the kernel, reboot the system, and see whether your changes took effect. Explore the source code of Minix and familiarize yourself with the main directories under /usr/src. This is your first step into modifying the Minix kernel!

##### **The Minix Scheduler**

1. [30 points] To gain experience modifying an operating system kernel, e.g., scheduling algorithm, modify the queue selection algorithm to skew the priority scheduling. Specifically, the current selection is based on a pure priority order. Include a random selection of a lower-level priority job. Note that if you modify the priority queue imprudently, the operating system will cease to function. Your goal is to achieve the following:

* upon attempting to select the next job, modify the selection to add a random possibility of choosing from a different level (the lower the probability you choose for this, the more consistent with the current selection mechanism you are, some experimentation may be required to select a reasonable value).

**Observing a change**

1. [10 points] Identify any observable change in behavior between the unmodified and modified schedulers. To achieve this, you may need to write a simple test program (e.g., a simple hello world program that identifies which process is running). *Hint:* the function called to **pick** the process - or **proc** - to run next is named logically, and the utility **grep.** The Minix documentation can be useful in finding it.

**Additional Resources:**

* Minix Wiki: https://en.wikipedia.org/wiki/MINIX
* Minix user guide: https://wiki.minix3.org/doku.php?id=usersguide:start
* Minix installation guide: https://wiki.minix3.org/doku.php?id=usersguide:doinginstallation

**Requirements to complete the lab**

1. Show the TA your running Minix system.
2. Write a description of your steps. Imagine you are writing a guide for a class-mate unfamiliar with setting up a Minix system image and provide instructions guiding them to the point where they can also modify the boot-up messages and rebuild a Minix system.
3. Write a description of your steps, which files you modified, and how you observed a change in the scheduler.

1. https://en.wikipedia.org/wiki/MINIX [↑](#footnote-ref-1)
2. http://www.minix3.org [↑](#footnote-ref-2)