CS312 :: Lab Week 4 :: Installing Arch Linux on VirtualBox

In this lab, we'll install and configure Arch Linux in a VirtualBox Virtual Machine. Arch Linux is very small, very light, and includes nothing but what you manually specify. It's a hobbyist's dream, and you'll like the tiny distro it produces.

Supplies needed

- Personal laptop with VirtualBox installed
- The Arch Linux DVD .ISO available on our Canvas page (556 MB)
- The VM will take up an additional 1.4 GB

Perform the Tasks and Problems in order, as given below.

Procedure

To get Arch installed, we need to create a Virtual Machine to hold it.

Create Virtual Machine

On your laptop, create a normal VirtualBox VM with these settings:

- Type: Linux
- Version: Arch Linux (64-bit)
- 2GB RAM
- Dynamic-sized HD with an 8GB size limit, normal .VDI format

Once the VM has been created (but before the Arch image is booted for the first time, set these settings:

- 2 CPUs
- Video Memory 128 MB

Install Arch

1. In the Settings controls for your new VM, on the Storage tab, insert the .ISO into the Optical Drive.

2. Close out the Settings controls, and start the VM.

3. Choose the first option, "Boot Arch Linux". Note that this startup will take all of about ten seconds to finish! This will get you into a virtual console (i.e. a RAM-based Linux install called Archiso) that you can use to install Arch on the hard drive. Note that your mouse will likely be "captured" by this VM. You'll need to press the "Host key" on your keyboard to get it back. The Host Key is listed in the bottom-right corner of the VM window. For example, on Windows, this is usually the Right CTRL key.

4. Because you didn't change the network settings for the VM, the settings are configured to NAT this VM, which means it piggybacks on the networking of your Host laptop. We can check to make sure we have an internet connection like this (note that the # prompt traditionally means that you are in a root shell - as always, you don't type this when entering the command):

   ```
   # ping archlinux.org
   ```

   If you do not have an internet connection (i.e. this ping fails), then verify that the VM network adapter is enabled and "Attached to" has been set to "NAT", and that your laptop itself has an internet connection.

5. Now we need to partition the hard drive. Let's do this assuming we will use 8GB of data.

a. First let’s start up our partitioning program “parted”:

```
# parted
```

b. You’ll now be at the parted prompt. Inside parted, lets list the existing disks we have:

```
(parted) print devices
```

c. You should see a lovely 8GB disk and our ~600MB Arch install image. Let’s select that disk so we can work with it:

```
(parted) select /dev/sda
```

d. Next, let’s create a partition table to hold entries about our partitions: This is an old-school MBR partition table, not the newer GPT format, because MBR is easier to configure in VirtualBox and we have a tiny hard drive:

```
(parted) mklabel msdos
```

e. We now need to create a partition for the partition that will hold the actual Linux files. This goes from the 1MiB position to the 6GiB position on the drive:

```
(parted) mkpart primary ext4 1MiB 6GiB
```

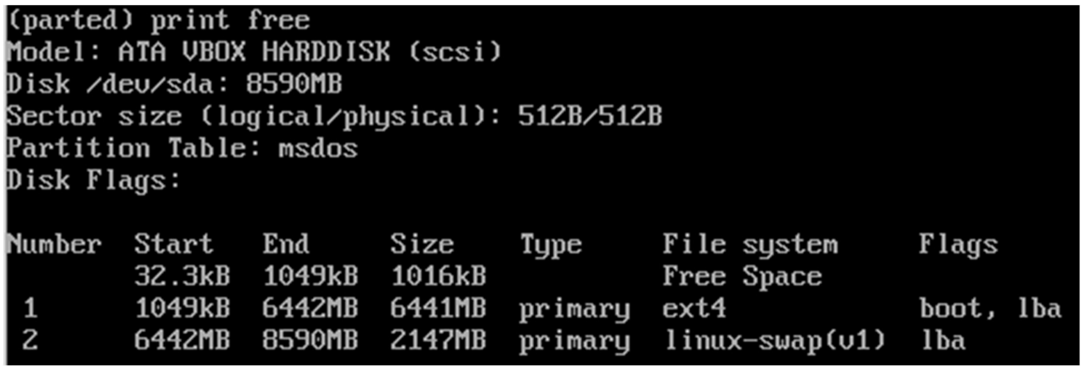
f. Now we need to make this partition be bootable. We do this bysimply running a command to make it so:

```
(parted) set 1 boot on
```

g. We’ll create one more partition which will hold swap space. This is a virtual memory storage location Linux will use when RAM overflows. Windows does this with a swapfile located at “C:\”, as you may have seen. For our purposes, we create this as a separate partition of 2GB in size, extending from where the previous partition left off at the 6GiB mark, filling the rest of the drive:

```
(parted) mkpart primary linux-swap 6GiB 100%
```

h. If you receive any errors, you can always blow away the entire partition table to start over. Just start at step c, above, and try again! You can check your status to see if you got it all right by using the “print free” command, which should return this result:



```
(parted) print free
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
        32.3kB  1049kB  1016kB           Free Space
 1      1049kB  6442MB  6441MB  primary  ext4            boot, lba
 2      6442MB  8590MB  2147MB  primary  linux-swap(v1)  lba
```

i. We can quit out of parted by just typing:

```
(parted) quit
```

6. We've now got partitions: physical spaces to install files on. These partitions need to be formatted with the appropriate filesystem before we can actually put files onto them.

    a. First, let's format the root partition as a EXT4 file systems:

    ```
    # mkfs.ext4 -F /dev/sda1
    ```

    b. Now, we'll mount the root directory file system in our large 6GB parition so that we can create directories in it and copy files to it - we'll be positioning this at /mnt:

    ```
    # mount /dev/sda1 /mnt
    ```

    c. Finally, we'll simply create swap space, and then turn on swap activity for our swap partition which we created as /dev/sda2:

    ```
    # mkswap /dev/sda2
    # swapon /dev/sda2
    ```

7. With the disk all nicely partitioned (Windows this for us practically automatically - do we appreciate that enough?), we can install Arch. We'll start the installation with the dedicated tool called "pacstrap" to install Arch into our /mnt directory, and we'll install only the base files. Note that this uses an internet connection to download files, so make sure you have a network connection with ping, first, then run:

    ```
    # pacstrap /mnt base
    ```

    IMPORTANT: If this fails with a note that indicates invalid signatures (potentially around the 5-minute mark), then we can make a change to the way pacstrap runs. Essentially, we can disable signature checking, which means that pacstrap won't validate that the packages are secure, genuine, etc. This is a security hazard, but since we're in a dedicated VM, this is acceptable for testing and teaching purposes.

    To make this change, edit `/etc/pacman.conf` in your VM and change the line that says "`SigLevel = Required Database Optional`" to "`SigLevel = Never`". Then, rerun the command above. It will pick right back up where it stopped.

    NOTE: This is going to take about 15-30 minutes total to finish! There are about 134 packages to download and install (watch for the big Linux kernel itself). While it's running, let's go ahead and do some research.

    QUESTION: Go answer Questions 1 and 2 now. These will take some online research.

8. We now need to create a way for the OS to keep track of all the mount points and IDs (the -U switch, below) created in the installation process that we just went through. This is called a "file system table", or fstab. Create it like this:

    ```
    # genfstab -U /mnt > /mnt/etc/fstab
    ```

9. In order to configure our new installation, we need to create a "chroot jail". This is essentially a mapping that makes it so the changes we are about to make are applied to the OS we're installing, not the OS that's actually currently booted (which is the one from the .ISO image). It also prevents commands from being used anywhere outside of this /mnt directory: the /mnt will become our new root directory:

    ```
    # arch-chroot /mnt
    ```

    Verify that we're in the jail by printing out the current working directory. Note that it now thinks we're in "/":

    ```
    # pwd
    ```

10. We need to set some locale settings, next.

    a. First, we need to copy the configuration of the "Los Angeles" area to where the OS looks for timezones. Then, check the time: it should be the current time in our timezone:

    ```
    # cp /usr/share/zoneinfo/America/Los_Angeles /etc/localtime
    # date
    ```

    b. Now we need to generate a set of locales the system can use. We'll pickle away the language and character set we want to use, and lastly make them available to the system while this install process is happening:

    ```
    # locale-gen
    # echo LANG=en_US.UTF-8 > /etc/locale.conf
    # export LANG=en_US.UTF-8
    ```

11. Now we are going to install the bootloader that boots Arch Linux when the system turns on. We've installed Arch the OS, but it currently won't be booted when the VM is started! Let's fix that by installing a bootloader called Grub.

    a. Here's how we use the Arch package manager "pacman" to install grub (this will take a minute or two):

    ```
    # pacman -S grub
    ```

    b. Now we need to tell grub to position itself in our root partition /dev/sda:

    ```
    # grub-install --recheck --target=i386-pc /dev/sda
    ```

    c. Finally, grub needs to write its own configuration file so it knows how to operate:

    ```
    # grub-mkconfig -o /boot/grub/grub.cfg
    ```

12. Arch is installed! Now we're going to do some configuration on it to make it more secure and usable. First, we're going to edit /etc/pacman.conf in order to allow 32-bit software to be installed onto our 64-bit OS.

    a. Open up the file (here we'll use vi, but you could use nano, too):

    ```
    # vi /etc/pacman.conf
    ```

    Remove the #'s:

    Now that you're in the file, find these lines, and remove the **#** comment symbol from in front of them:

    ```
    #[multilib]
    #Include = /etc/pacman.d/mirrorlist
    ```

    Save and exit the file (In vi, this is `:wq`).

    b. Re-sync pacman with its new configuration (takes 1 or 2 minutes):

    ```
    # pacman -Syy
    ```

13. Now we're going to install sudo. Up until now, we've been operating solely as the superuser "root": this is handy for administrative work, but dangerous in production. This sudo tool allows us to run one-shot configuration commands as if we were root (or another user) while remaining as a lesser-privileged user.

    a. Install sudo like this:

    ```
    # pacman -S sudo
    ```

b. We now need to modify the sudo config file to allow a security group called “wheel” to be authorized to use sudo. With this done, any user in the “wheel” group will be set up to use sudo from all terminals, acting as any user, and run any commands with it.

   This config file is edited in a strange way. Instead of editing the config file directly, we use a command that loads an editor to go work on the file:

Remove the # and space:

```
# visudo
```

   When you run this, it will tell you that vim is not installed. That’s correct, it’s not, but sudo (when it was installed), assumed that you had vim, and asks you to configure it with vim. To change this configuration, we can edit the environment variable that sudo is looking at, and change it to vi. Then, we can run our editor command:

```
# export VISUAL=vi
# visudo
```

Remove the # and space:

   As before, find this line and remove the leading # symbol and the space:

```
# %wheel ALL=(ALL) ALL
```

   Then save and exit the file.

14. Now we’re going to add a new user: this will be who we want to log in as normally.

    a. Create the user with the following command, which will create the home directory (-m), add it to a group (-g) called “users”, add it to the additional group (-G) “wheel” which will give it sudo access, set the shell (-s) to bash, and call it “archuser”:

    ```
    # useradd -m -g users -G wheel -s /bin/bash archuser
    ```

    b. Then, set the password for this new user (here, I use the amazing temp password “password”, note that you have to type it twice):

    ```
    # passwd archuser
    password
    password
    ```

15. Lastly, in our installation phase, we want to set the password for the root user. As a demonstration, we’ll simply set it to “password”:

    ```
    # passwd root
    password
    password
    ```

16. Now, we can reboot!

    a. Exit out of the chroot jail like so:

    ```
    # exit
    ```

    b. Eject the .ISO out of the Virtual Machine by clicking on the Machine menu of the running VM, and then choosing Settings. Inside Settings, click on Storage on the left side, then click on the “archlinux..” CD in the middle. On the right side of the dialog box is a little CD icon with an arrow: click that, then click on “Remove disk…”. Click OK to exit Settings.

c. Reboot the VM:

```
# reboot
```

Configure Arch

17. You are now going to see a LOT of errors, crashes, and warnings about *everything* but you are going to just let the countdowns do their thing, and allow the reboot to finish. If all goes well, it should go to an OS selection screen where it will automatically choose Arch after 5 seconds, and then finally present you with a login prompt!

    Go ahead and log in. You'll find that many, many, most, many, practically all things are not installed. That's Arch! For example, you probably don't have a network interface turned on, so ping will fail. Continue on!

    FYI, if you'd like to shutdown the VM, you can do so with this command (note that we're using the bash shell now, so the prompt symbol is the traditional $):

    ```
    $ sudo shutdown now
    ```

18. Let's get networking working.

    a. The first thing we need to do is create a *hostname* for our system. This name is used to identify this machine on the network, and is required for it to get an address. We'll need to create a file in this location with this name:

    ```
    $ sudo vi /etc/hostname
    ```

    Inside that file, put just a word, no spaces. Consider "RefArch", for example, which we'll use below. Save and exit.

    b. Now we need to configure the networking hosts file to know about our name. This file tracks several memorized network locations, including our own. Edit it:

    ```
    $ sudo vi /etc/hosts
    ```

    Add to this the following three lines:

    ```
    127.0.0.1  localhost
    ::1        localhost
    127.0.1.1  RefArch.localdomain RefArch
    ```

    c. Next, we need to list the interfaces to see what we have:

    ```
    $ ip a
    ```

    This will return all interfaces. The first one, "lo", is the loop device, or the loopback identifier - this isn't used for actual communication. The second one is the actual ethernet connection we care about. Traditionally this would be called "eth0" or similar, but because we're doing this in VirtualBox, it'll show up with a name like, "enp0s3". If yours is slightly different, just substitute that below.

    We can query just the interface we care about, to see its state, with this:

    ```
    $ ip addr show dev enp0s3
    ```

    As you can see, the state is DOWN. That's right: Arch's networking starts in a down state so that it doesn't automatically reach out and join the network. That allows you to be explicit about when you're ready to connect. Let's enable our interface with this:

```
$ sudo ip link set enp0s3 up
```

Check it again:

```
$ ip addr show dev enp0s3
```

You should see that it now says UP.

d. Let's now start and enable (which means it starts on boot) the DHCP client, which will allow our system to get an IP address from the DHCP server being NAT'd by VirtualBox:

```
$ sudo systemctl start dhcpcd.service
```

Let's check our interface to see if it got an address:

```
$ ip link show dev enp0s3
```

Look in the "inet" field: it should show an IP address like `10.0.`*`X`*`.`*`Y`*`/24`. If so, then you've got a connection! Try pinging:

```
$ ping archlinux.org
```

QUESTION: Go answer Question 3 now.

e. Lastly, let's permanently enable the DHCP client:

```
$ sudo systemctl enable dhcpcd.service
```

QUESTION: Go answer Questions 4 and 5 now.

19. You're done! Once you're done answering the remaining **questions** below, shut down your VM using the command given in step 17, above.

Questions

1) What are some differences between GPT and MBR-style partition tables? *(3 points)*
2) What are some differences between the EXT4 file system we partitioned our primary with, and the FAT32 filesystem that used to prevail? *(3 points)*
3) Get the TAs initials, showing that you can ping a website or server on the internet. *(25 points)*
4) What is the IP address of your VM? *(3 points)*
5) What is the name of your ethernet adapter? *(3 points)*
6) What are three types of platforms other than "`i386-pc`" that `grub-install` can target? *(3 points)*

You're done! To receive credit for this lab, you must turn in your answer sheet with your names listed.