Advanced Exercises Set 1
Setting Up a Highly Customizable Linux Distro

EECS 201 Fall 2020

Submission Instructions

This assignment is an "online assignment" on Gradescope, where you fill out your answers directly. These questions are reflected in the "Questions" section of each section.

Preface

This advanced assignment is not for the faint of heart. This may require additional reading on the Arch wiki and other research. I will record a supplemental lecture explaining the more general concepts.

1 Set Up an Arch Linux Virtual Machine (7 pts)

Arch Linux is a GNU/Linux distribution with a rolling-release model, where its package repository keeps up with the latest stable updates. Unlike many distributions which try to be user-friendly, Arch strives to be simple and minimalist upon installation, allowing the user to pick and choose exactly what packages they want. For example, after a typical Ubuntu installation you will have a graphical environment, office software, and other utilities, whereas after a baseline Arch installation you will be greeted with a non-graphical command line and a handful of core nix utilities, notwithstanding the highly involved installation process. With this minimal install process, however, comes a great deal of customizability. You can read more about the Arch Linux philosophy on its ArchWiki.

For this advanced assignment, I want you to install Arch Linux on another VirtualBox VM. The idea behind this is to introduce you to all the bits and pieces that go into Linux systems that most people take for granted when using more user-friendly distributions.

Follow the Installation Guide. **Do not follow online tutorials for this as they tend to be outdated. The ArchWiki's installation guide is the canonical way to install Arch.** When you go through the installation process, be mindful of times the guide offers suggestions: you will have to answer questions about certain decisions you made. When in doubt, follow the links in the installation guide and **read**.

For Windows hosts, in my testing, it seems that Hyper-V does indeed run into issues here and there with Arch, namely with PGP signature checking for packages, resulting in package installations failing. If you do run into issues, try turning off Hyper-V as was described in Basic 1.

1. Create a new VirtualBox VM (you could use another hypervisor or even dual boot if you wish). Arch Linux has lower baseline system requirements than Ubuntu, but you can still go with the guidelines from Basic 1.
2. By default VirtualBox VMs use a typical BIOS (i.e. they aren't UEFI). When you encounter instructions that differ based on BIOS vs UEFI, go with the BIOS instructions (if you do choose to go UEFI, follow the UEFI instructions).
   I personally use UEFI on VMs as UEFI has superseded BIOS on recent computers
3. As with most operating systems, you'll need to get the ISO and then insert it into your VM's CD drive to boot and run the installer. Only this time, the installer is completely command line :)
4. You don't have to really bother with checking the signature of the ISO or setting the keyboard layout
5. With VirtualBox, the internet should work out of the box; configuring the network connection *for the installer* should not be necessary.

6. Partitioning a disk involves dividing a disk into independent sections. A disk is exposed as a device such as `/dev/sda`, with its partitions being exposed as `/dev/sda1`, `/dev/sda2`, etc. There are two partition table layouts which hold the partitioning information on a disk. MBR is the older table used by MS-DOS with several quirks (such as having a max of 4 partitions) while GPT is a newer table. For more details see https://wiki.archlinux.org/index.php/Partitioning. Note that "MBR" is also referred to as "DOS" when it comes to partition tables. For this assignment, MBR is probably the easier to work with BIOS while GPT is easier to work with UEFI.

7. As for partitioning scheme, you can set aside partitons to be for directories such as `/var`. *However a "single root partition" can suffice.* See https://wiki.archlinux.org/index.php/Partitioning#Partition_scheme for advice.

8. After partitioning the disk, you'll have to *format* the partitions. Partitioning splits a disk into independent sections, but we still need a system to perform book-keeping for files. This is where a *file system* comes into play. Formatting a disk sets up all the structures and metadata needed to track files and directories. Windows uses NTFS, mac OS uses APFS, while Linux has many options, with ext4 often being the "default". If you're interested, you can look at https://wiki.archlinux.org/index.php/File_systems for other file systems to use.

9. Mounting a partition puts it into the directory tree, allowing it to be accessed via `cd` and the like.

10. After all this disk business, you are ready to perform the installation proper. I suggest adding `base-devel` (for tools like `gcc`, `make`, and `sudo`) and your preferred text editor (`nano`, `vim`, etc.), and `dhcpcd` for networking to the `pacstrap` invocation. You can also add whatever other packages you want to be installed from the get-go.

11. Once you're chroot-ed into the system, you are effectively "logged-in" as the root user on your installation, having used the installer as a launching point. Note that the package manager is called `pacman` (how quaint) in case you need to install any packages that you've forgotten.

12. The next several steps are fairly straightforward. When it comes to the timezone, our region is "America", and our closest city is "Detroit". **Make sure to configure your network connectivity, lest you be unable to connect to the internet after you reboot.** `dhcpcd` **(which will have to be installed via** `pacman` **if not included with** `pacstrap`**) is a decent choice on VirtualBox. You can enable** `dhcpcd` **with** `$ systemctl enable dhcpcd`.

13. The boot loader step offers a bunch of choices depending on whether you're using BIOS or UEFI. If you are using BIOS, I recommend GRUB. Do note that GRUB has different instructions depending on whether you're using BIOS or UEFI and MBR or GPT. If you are using UEFI, `systemd-boot` can suffice.
    I've taken to rEFInd on my dual-boot setups.

14. Once you've reached post-installation, you should create a user for yourself, as logging in only as root is rather insecure. If you don't have any users created, you can log in as "root", using the root password set during installation, to perform this step. **The user you create should have sudo privileges.** For info on user administration see https://wiki.archlinux.org/index.php/Users_and_groups#User_management.

15. A note on `visudo`: by default it will try to use `vi`, which may or may not be installed. You can change it by setting the `$EDITOR` environment variable to `nano` or your preferred editor: `$ EDITOR=nano visudo`.

16. When you're done doing your setup and user management, exit the chroot, and shut down your VM. Remove the installation ISO.

17. Start your VM. If you have successfully performed the installation as well as set up the bootloader, you should see Arch booting up and then presenting you with a login prompt.

    (a) If things don't seem to be right, you can always get back into that nice chroot environment from earlier. This is the beauty of those installer ISOs: besides installation duties, they provide a live environment for you to perform repairs with. All you have to do is the mounting steps and then chroot, then you can try to fix things up.
    (I have had to do this a few times on real machines).

18. Hooray! You now have a bare-bones, highly customizable Linux distro! Hope you enjoy staring at that blinking cursor!

Questions:

- ☐ What sort of partitioning scheme did you use and why?
- ☐ What filesystem did you use and why?
- ☐ Log into your user. What is the output of `$ whoami`?
- ☐ What is the output of `$ uname -r`?
- ☐ How did you like the installation process compared to Ubuntu's?

2 Let There Be Light (3 pts)

Being stuck in a purely command line environment is a bit of a drag. Let's make the installation more home-y by getting a graphical user interface (GUI) going.

1. You can find some info about this in General Recommendations.
2. The backbone of graphics on *nix systems is a display server. X11 is probably the most popular one, though Wayland is also an option. In either case, you'll have to install a display server before anything else, though often-times the GUI you install already has this as a dependency that `pacman` will resolve.
3. Install your preferred GUI, be it a full on desktop environment or minimalist window manager. GNOME and KDE Plasma are popular fully featured options, but are relatively heavyweight. If you prefer something a bit lighter, LXQt might be of interest.
   I showcased KDE Plasma in Lecture 1.
   By the way, I used to use KDE Plasma but have since moved onto i3 window manager.
4. There's various different ways to access the GUI. Choose and set up an option that suits your fancy (e.g. display manager, xinit, etc.).
5. Install a terminal emulator, which is a graphical program that presents a terminal. One might have been already installed if you went for a fully featured desktop environment. Here is a list of terminal emulators: `https://wiki.archlinux.org/index.php/List_of_applications/Utilities#Terminal_emulators`
   My daily driver is `urxvt`.
6. You can also install the VirtualBox Guest Additions if you want. (You may want to enable the Guest Services if you do...)

Questions:

- ☐ What graphical user interface did you pick and why?
- ☐ Upload a screenshot of your graphical environment, showcasing the taskbars and various GUI elements.
- ☐ Open your preferred terminal emulator. Install `archey3`. Run `archey3` inside the terminal emulator and upload a screenshot of it.

If you've completed this assignment, pat yourself on the back! Installing Arch is no easy feat!