File System

Daniel R. Cender

Grand Canyon University: CST-221-O500

Feb 15, 2020

**Exploring the Linux File System**

**Base Directories**

Below is a table of descriptions for the most common base directories in a Linux file system.

|  |  |
| --- | --- |
| **Directory** | **Purpose** |
| / | This is the root directory. It holds every other folder listed below and a handful of others, depending on the distribution of Linux. |
| /bin | This directory mainly holds executable programs. They may be native to the kernel, or a part of an extended command-line program or utility. Programs like the bash or zsh (Z-shell) shells are held here, whereas programs like Firefox or Chrome would be stored in /usr/bin (Hoffman, 2016). |
| /dev | The utilities found in this directory correspond to devices, both real and abstract. This might store a program that interfaces with a SATA drive, or a programming utility like /dev/null/, which discards all output piped into it from other programs (Hoffman, 2016). |
| /etc | The files here are configuration files. They configure settings on a system-wide basis, separate from user-level settings, which are set in the usr/ directories (Hoffman, 2016). |
| /lib | The */lib* directory contains libraries needed by the binaries in the */bin* and */sbin* directory (Hoffman, 2016). User-specific libraries are held in the */user/lib* directory instead. |
| /boot | This is a crucial directory, since it holds all the needed files for booting up the operating system. All GRUB boot loader files sit in here, as well as the Linux kernel(s) (Hoffman, 2016). |
| /home | This directory holds the home folders for every user on the Linux system. The corresponding user has “write” access to this folder, so they must login as the “root” user to augment other system files outside this folder (Hoffman, 2016). |
| /mnt | This directory holds temporary mount points. Although you can mount temporary file systems anywhere in the system, this is where administrators will oftentimes do so. |
| /proc | This directory is similar to the */dev* directory in that it holds files for running kernel and program processes. Not traditionally edited directly by the user. |
| /tmp | This directory is where apps store temporary files, which can be deleted when the system is rebooted or shut down. Open applications seem to use this space to store data that it doesn’t need immediately in RAM but wants available in a smaller indexed space than the entire hard drive. |
| /usr | This directory contains all user-specific programs and settings. It may hold user-specific binary executable programs which aren’t essential to kernel operations. It is also where user-level programs, like a game installed from the internet, might be installed to (Hoffman, 2016). This folder is read-only. |
| /var | This is the writable counterpart to the */usr* directory. Here, programs may write logs or cache files during ordinary execution (Hoffman, 2016). |
| /sbin | This directory is similar in nature to the */bin* directory. Except, the binaries included here are intended for use by system administrators logged into the root user. |
| /kernel | The only */kernel* directory to be found was under */sys/kernel*, which seems to hold configuration files and peripheral program files that may not be essential to ordinary kernel operations. |

**Root Directory**

The directions for this section were to list all the directories under */root* and give a short explanation for the purpose of each. However, on my MacBook, running OSX, the directories under root only mirror the Linux file system to an extent.

**C Compiler Flow Chart**

Below in Figure 1 is a flow chart which diagrams the high-level process of compiling a C program. The two main components of the process, sections *A* and *B,* are segmented away from the main process line. *A* shows the flow of creating a temporary directory to hold the tokens generated from parsing out the user’s C program. *B* illustrates the process of linking the temporary token file with a required system library, *system.out*, so a final executable file may be generated in the user’s current directory. Of course, this chart overly simplifies the complexity of compiling C programs. Just the error checking and handling alone warrants many larger flow charts than this.

A close up of text on a white background

Description automatically generated

*Figure 1.* High-level diagram of basic C compiler process

References

Hoffman, C. (2016, September 22). The Linux Directory Structure, Explained. Retrieved February 14, 2020, from <https://www.howtogeek.com/117435/htg-explains-the-linux-directory-structure-explained/>

Tanenbaum, A.S. & Bos, H. (2015).*Modern Operating Systems.*Chapter 4.