Lab 04: Exploring your OS

Team Members: **Ilyas Nur, Alan Chen, Sang hwa Lee, Sumaya Shameem**

BACKGROUND INFORMATION

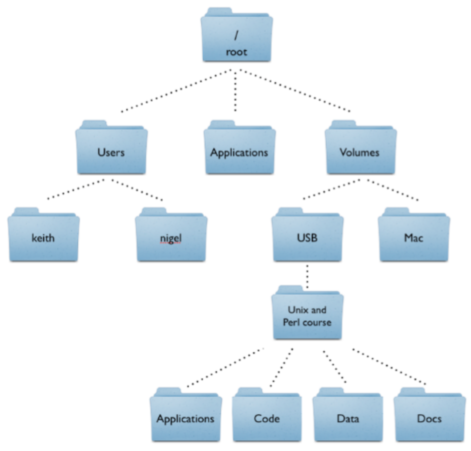
As information professionals, we use computers running some form of operating system nearly every day. We use the features of our operating systems so much, that they have become background noise that we ignore. In this lab, you will explore the features of your operating system to identify the services that the operating system provides and how to take advantage of them.

Activity 00: Exploring the Filesystem

A recent article on the Verge titled "File Not Found" (Chin, 2021) identifies the trend that with the rise of cloud-based systems and mobile devices, students often do not understand the concept of a filesystem. The success of modern user interfaces has made it unnecessary for end-users to know the filesystem. However, programmers, data scientists, and other information professionals frequently interact with the filesystem to work with their data.

Instructions

Download this file onto your computer. You may collaborate with your team by working in a Cloud-based solution like Office 365 or Google Docs, but we will be using this file as a part of the activity, so you will need to have it downloaded.



The *root* of your filesystem is the starting point for all *paths* on your computer. In Windows the *root* will usually be C:\ and on a Mac or Linux computer the *root* will always be / .

Directories (sometimes called folders) are named containers for holding *files*. We use *directories* to organize our *files*. Think back to when you were younger and were not so diligent in putting your clothes away (Because you are adults now and put your clothes away, right?). The *directories* are like the drawers and each piece of clothing is like the *files*. You don't want to dig through a pile of clothes to find a matching sock, you would rather open a drawer and find a pair of socks. *Files* and *directories* are a similar concept.

The filesystem is also *hierarchical* meaning that we can put as many *directories* as we want in a directory and those directories can also contain directories. Figure 1 shows a simplified filesystem. Even though the image does not show any, each of these *directories* likely contain many *files*.

A *path* is a description of where a file is located in the directory structure. There are two types of *paths*. An *absolute path* will start from the *root* of the filesystem. A *relative path* will describe the location of a file or directory based on its relative location to something else. For example, in Figure 1 the *absolute path* to the directory named "nigel" would be written as /Users/nigel. If we wrote the *relative path* starting from within the /Users directory, then it would be nigel.

Windows uses a different slash than Mac and Linux to separate items in a *path*. Figure 1 is using the Mac and Linux character called slash or /. In Windows, we would have to use the backslash or \ character.

Questions

1. **What is currently the *absolute path* to this file on your computer?**

   /Users/ilyasnur/Downloads/Lab04-Exploring_the_filesystem.docx

2. **What is currently the *relative path* to this file on your computer when starting from the Users folder?**

   /ilyasnur/Downloads/Lab04-Exploring_the_filesystem.docx

3. **We often need to refer to files when we are programming. When do you think it would be more advantageous to use *relative paths* rather than *absolute paths*?**

   The concept of portability is important because that decides which path one should use. There are certain advantages to using both paths, however it would be more advantageous for one to use a relative path when both users know where the code is being executed from.

   Relative paths are also advantageous if one moves the entire website from one domain to a different domain altogether.

Activity 01: Reflecting on COBOL

In this week's readings, Dr. Hicks talks about the "scapegoating" of COBOL that is a recurring theme in the news (Hicks, n.d.). COBOL is a language designed to be easy to learn, but these legacy applications run on legacy systems not modern hardware. These systems are large mainframe systems running proprietary operating systems like IBM z/OS. In this activity, I would like you to think about the challenges that would face an aspiring COBOL programmer today and what lessons we might be able to take away to apply to our professional practice.

Questions

4. **Beyond learning the COBOL programming language, explain some issues that someone looking to get a COBOL programming job might experience.**
   Some issues that they might experience is the lack of support and funding for COBOL that companies may have, since it is potentially getting outdated and modern programming languages are being utilized instead. There could be scalability issues with COBOL that companies just avoid working with. For people with knowledge of only COBOL, the job security is not too good in that case and COBOL jobs could only be available in certain niches.

5. **What are some lessons that we can learn from COBOL's success, reduced importance, and the problems that keep bringing it into our awareness?**

   Some lessons we can take from COBOL's success is that simple is sometimes golden. Although COBOL is more than 60 years old, it's still widely used in businesses, financial institutions, and in administrative settings. Another lesson we can learn is that sometimes upgrading can be helpful as new computing languages allow for features such as recursion while COBOL cannot process such demanding computations.

6. **Can we make better decisions when designing, building, and maintaining systems to avoid similar issues? Why or why not? Explain.**

   We can make better decisions when designing, building, and maintaining systems to avoid similar issues. Some of these include being able to handle traffic issues, and learning multiple programming languages if there's ever the need to make the big switch from a certain primary language to another.

Citations

Chin, M. (2021, September 22). *Students who grew up with search engines might change STEM education forever*. The Verge. https://www.theverge.com/22684730/students-file-folder-directory-structure-education-gen-z

Hicks, M. (n.d.). *Scapegoating COBOL? | Mar Hicks*. Retrieved September 15, 2020, from https://marhicks.com/videos/cobol.html