CS-321: Continuous Learning Essay 1

Due on 2/2/2024

Prof. Arghya Kusum Das (Argo), Spring 2024, 2/2/2024

Kaleb Burris

On the functionalities between Linux and Windows

Linux and Windows offer incredibly different experiences that empower different types of workflows and appeal to different demographics.

Linux is an open-source, lightweight family of operating systems that offers extreme customizability, and this has led to many different Linux distributions catering to different needs and purposes. This leaves Linux in a much less user-friendly position as standardization is difficult and many things are configured in a technically demanding manner, however, this makes Linux much more favorable to the technically minded; often being the preferred platform for software development.

Windows, on the other hand, is a family of closed-source operating systems that follows a linear versioning from Windows 1.0 to the current Windows 11. Windows focuses much more on user-friendly software that is often included with the default Windows installation. Most tasks in Windows come with a GUI that makes user interaction simple and easy.

Technically speaking, Linux is a UNIX-like operating system while Windows is based on a proprietary architecture. This means that for Linux, scheduling is done through processes using `fork()` and `exec()` to create a new process and schedule it, respectively. This is done under the Native POSIX Thread Library, which ties the idea of a thread and process together - they are not differentiated between by the Linux scheduler. Windows is more thread-based in its approach but shares the thread/process agnosticism that Linux has. In Windows, processes are data that threads can access and execute based on their priority, which is a simpler approach than Linux's scheduler. It is important to note that Windows tends to have more complicated processes for the sake of security.

Indeed, Linux and Windows share more functionality under the hood than one might initially assume; even their System Calls follow a similar scheme in that they act as the only interface to their respective kernels.