2-2 Journal: Operating System Shell Commands

Christopher Gray

IT: 600: Operating Systems

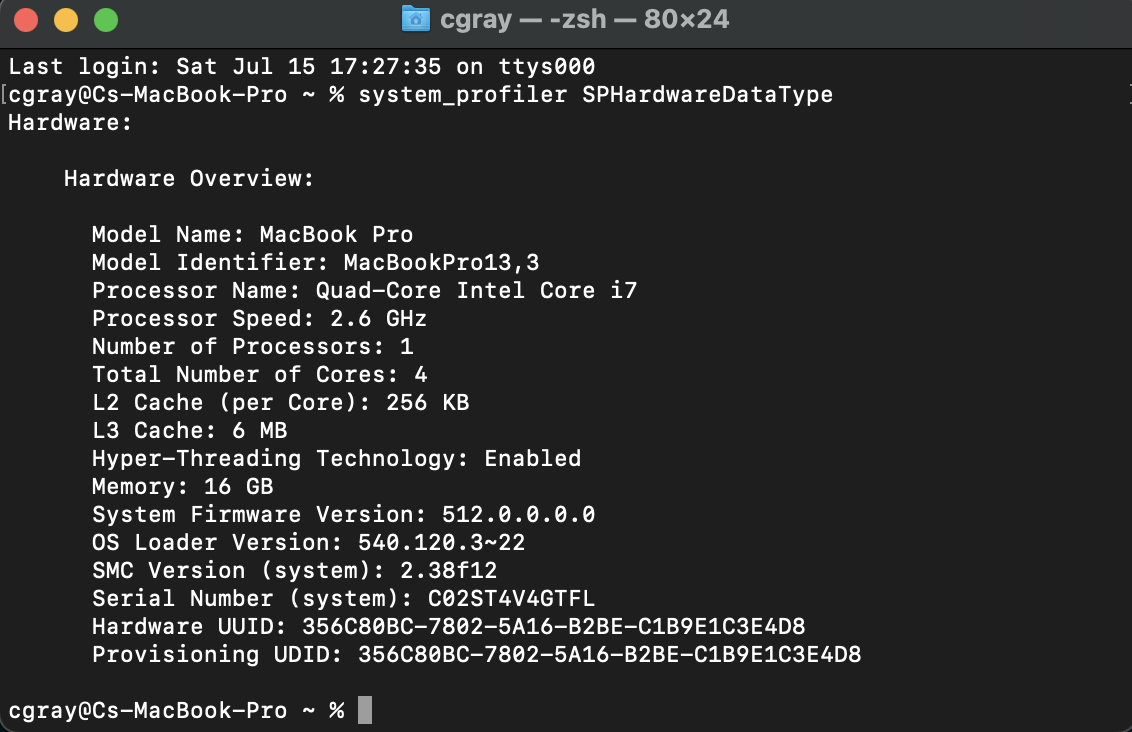
Southern New Hampshire University

Professor Michael Gallaher

September 9, 2023

Journal Entry #1:

Today, I conducted the organizational profile necessary for my journal entry by examining the architecture of the operating system on my computer. I used the command "system\_profiler SPHardwareDataType" on my Mac OS to obtain the system information. In order for me to use the system call I pressed “cmd + space” to bring up the spotlight, where I then typed in “terminal” to bring up the terminal application in the utilities folder. Here are the details I gathered:



I initially went about obtaining this information the way in which I knew how, which was to click on the Apple logo in the upper left-hand corner and then selecting the “About This Mac” button, and then selecting “System Report”. This is the information I retrieved doing it this way as well:



Upon examining the information, I confirmed that my MacBook Pro is equipped with a Quad-Core Intel Core i7 processor running at a clock speed of 2.6 GHz. It has a single processor with a total of four cores, and each core has a 256 KB L2 cache. The system also features a 6 MB L3 cache. Hyper-Threading Technology is enabled, allowing for improved multitasking capabilities.

In terms of memory, my computer has 16 GB of RAM, which should provide ample resources for running various applications simultaneously.

One interesting piece of information I found is the hardware UUID (Universally Unique Identifier). This unique identifier is assigned to my MacBook Pro and can be used for various purposes, such as software provisioning or identifying the hardware in a networked environment.

Considering the system call that was executed to generate this output, the "system\_profiler SPHardwareDataType" command triggered a system call specific to Mac OS. While the exact system call used may vary, it would likely involve gathering information from the underlying hardware and system components, such as the processor, memory, and cache. This information is then formatted and presented to the user as the system profile.

The process management category includes system calls that deal with the creation, termination, and replacement of processes. The fork() system function piqued my interest since it generates a child process that is identical to the parent process. It enables the execution of numerous operations in parallel, potentially enhancing system efficiency. Moreover, the waitpid() system function stood out because it allows a parent process to wait for a child process to end before advancing. This synchronization mechanism guarantees that processes are executed in the correct order. A process can also use the execve() system function to replace its core image with a new application. This system call is required for starting and running various applications within a process.

In conclusion, this exercise provided valuable insights into the architecture of my operating system and the underlying hardware. Understanding the specifications of my computer will be crucial for my final project, as it will help me optimize the performance and compatibility of the software.