Linux Notes

Tommy Bui

10-10-2022

1 High-level View of Unix Environment

- To best understand how an operating system works, keep in mind the concept of abstraction.
  - Abstraction focuses on the basic purpose and operation of an object.
- There are many times for an abstracted subdivision in software. In these notes, the term **component**
- This chapter provides a high-level overview of the components that make up a Linux system.

1.1 Levels and Layers of Abstraction in Linux

- Abstraction helps break down the Linux operating system into easy-to-understand components.
- We arrange components into layers or levels, classifications or groupings of components according to where the components lay between the user and hardware.
  - i.e. Web browsers, games, etc. are at the top layer; the bottom layer consists of the memory in hardware which is composed of 0's and 1's.
- A Linux OS consists of 3 main levels:
  - The base consists of hardware:
    - Hardware includes the memory as well as the Central Processing Unit(s) to perform computation or RD/WR to memory.
    - Devices such as disks and network interfaces are also part of the hardware.
    - Examples of Hardware: CPU, main memory (RAM), Disks, Network ports, etc.
  - The next level up is the kernel:
    - The kernal is consider the software within memory that tells the CPU where to look for its next task.
    - As a mediator, the kernal manages hardware (i.e. main memory) and is the primary interface between hardware and any running program.
    - Linux Kernal contains: System calls, Process Management, Memory Management, and Device Drivers
  - Processes:
    - Running programs that are managed by the kernal, make up the system's upper level known as **user space** (i.e. all web servers run as **user processes**).

    - User Processes include GUI, Servers and Shell
  - The main difference between how the kernal and the user processes run is that the kernal runs in kernal mode and user processes run in user mode
  - Code running in kernal mode has unrestricted access to the processor and main memory. This can be powerful but is a dangerous privilege that can cause the kernal to easily corrupt and crash the entire system.
  - Memory area that the only the kernal can access is **kernal space**
  - Unlike kernal mode, user mode is restricted to a subset of memory and safe CPU operations
    - The Linux kernal can run kernal threads, which are similar to processes but have access to kernal space (i.e. kthreadd and kblockd)
  - *User space* refers to the parts of the main memory that user processes can access. If a process were to crash, the consequences are limited and can be repaired by the kernel
    - i.e. If your web browser crashes, it won't stop the scientific computation background processes that has been running for days
    - In theory, a user process gone haywire can't damage the majority of the system. However, user processes may affect other parts of your system
    - i.e. With the correct permissions, a user process can damage data on a disk

1.2 Hardware: Understanding Main Memory

- In it's rawest form, main memory is a giant storage of bits.
- All input & output from peripheral devices flows through main memory is also in form of bits.
- The CPU operates on memory; it reads instructions & data from the memory and writes data back to the memory.
- The term state in reference to memory, processoes, the kernal, etc. refers to the particular arrangement of bits.

1.3 The Kernel

Nearly everything the kernel does revolves around the main memory. One of the kernel's task is to partition memory into subdivisions and it must maintain certain state information about those subdivisions at all times. Each process gets its own share of memory & the kernel manages each process' memory.

The kernal is in charge of managing tasks in four general system areas:

- **Processes:** The kernel is responsible for determining which proccess can use the CPU.
- **Memory:** The kernel needs to keep track of all memory; Memory can be shared between processes even if allocated to a particular process.
- **Device drivers:** The kernel acts as an interface between hardware & processes. The kernel usually operates the hardware.
- **System calls & support:** Processes normally use system calls to communicate with the kernel.