Writing Assignment One

Brennan Giles

CS 444 Oregon State University

Fall 2018

Abstract

In todays day and age, it is easy to take operating systems for granted. Many dont understand what they are, and even more dont think there are any beyond mac and windows. Despite this, entire communities have worked together to create thriving albiet less popular operating systems that often outperform their more famous counterparts. Operating systems like Linux and FreeBSD have the luxury of being able to drop superfluous processes and graphical interfaces in favor of excellent speeds, processes, and security. In this paper, we will examine the advantages and disadvantages of Linux, Windows, and FreeBSD in order to understand their differences and learn more about what makes each Operating system great.

1 Windows Overview

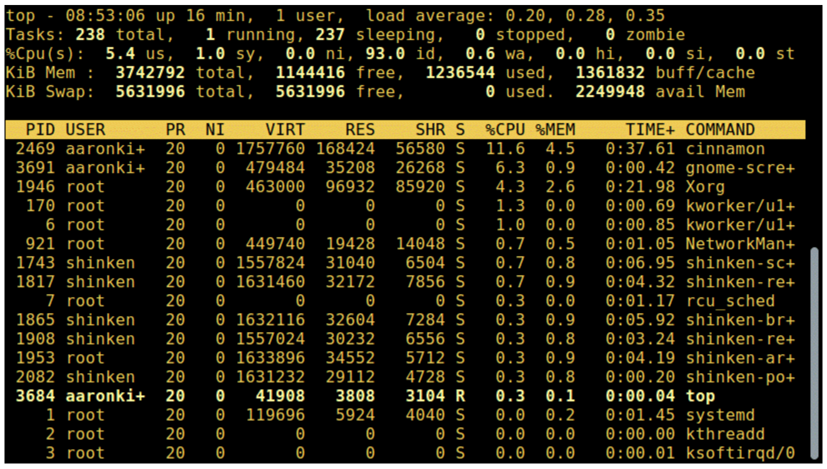
Windows is an extremely mainstream and widely used operating system that assigns priority to threads instead of processes based on the ascertained importance of those threads. According to the Microsoft Dev Center Manual, A process has a virtual address space, executable code, open handles to system objects, a security context, a unique process identifier, environment variables, a priority class, minimum and maximum working set sizes, and at least one thread of execution [1]. Windows utilizes CreateProcess in order to create new Processes with process handles. These handles can be passed on to child processes if necessary. CreateProcess also Assigns ProcessID, which can be used to manually access any process through OpenProcess. A more accessable way of seeing and manipulating these processes can be done through Task Manager, which displays these processes by name, priority, resource usage, and more.

More important to how the kernel operates however are the threads that processes use. All Threads of a process share its virtual address space and system resources. In addition, each thread maintains exception handlers, a scheduling priority, thread local storage, a unique thread identifier, and a set of structures the system will use to save the thread context until it is scheduled. The Thread context includes the threads set of machine registers, the kernel stack, a thread environment block, and a user stack in the address space of the threads process [1]. In Windows, preemptive multitasking is supported which allows simultaneous execution of multiple threads from multiple processors, and the system will execute as many threads as possible at the same time. Threads take turns occupying processor time based on priority. Windows processes threads depending on priority and not according to the process the threads are associated with; If process A has 3 threads and process B has 2 then the CPU gives each thread 1/5 of its time as opposed to giving half of its time to A and half to B.

2 Linux Overview

Linux is the most commonly used operating system in the entire world and according to the IDC dominates the smartphone market as well, controlling over 85 percent of the entire worlds smartphones [2]. Linux is a multiprocessing operating system that like windows, strives to use all of the CPU for processes at all times. Each process is represented by a task-struct data structure, and by default the maximum number of processes in the system is limited by the task vector 512 entries. Processes are kept track of by the ”Process ID”, or PID. Unlike Windows, processes share lots of information about themselves with eachother such as their state, identifier, links, timers, and more. Processes cannot interrupt other processes and usually run until they need to wait for some sort of system call or exceed their Time Slice of usually about 200ms. [3] Which process gets to take over next is usually handled by the Low level scheduler, which looks for the highest priority thread waiting and runs it. Next, the high level scheduler takes over and assigns all the other runnable threads that are still waiting priority. Once assigned priority, these threads are placed in a run queue and the low level scheduler takes over again.

```
Dynamic real time view of running system accessed through the command "top"
```



```
top - 08:53:06 up 16 min,  1 user,  load average: 0.20, 0.28, 0.35
Tasks: 238 total,   1 running, 237 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.4 us,  1.0 sy,  0.0 ni, 93.0 id,  0.6 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3742792 total,  1144416 free,  1236544 used,  1361832 buff/cache
KiB Swap:  5631996 total,  5631996 free,        0 used.  2249948 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2469 aaronki+  20   0 1757760 168424  56580 S  11.6  4.5   0:37.61 cinnamon
 3691 aaronki+  20   0  479484  35208  26268 S   6.3  0.9   0:00.42 gnome-scre+
 1946 root      20   0  463000  96932  85920 S   4.3  2.6   0:21.98 Xorg
  170 root      20   0       0      0      0 S   1.3  0.0   0:00.69 kworker/u1+
    6 root      20   0       0      0      0 S   1.0  0.0   0:00.85 kworker/u1+
  921 root      20   0  449740  19428  14048 S   0.7  0.5   0:01.05 NetworkMan+
 1743 shinken   20   0 1557824  31040   6504 S   0.7  0.8   0:06.95 shinken-sc+
 1817 shinken   20   0 1631460  32172   7856 S   0.7  0.9   0:04.32 shinken-re+
    7 root      20   0       0      0      0 S   0.3  0.0   0:01.17 rcu_sched
 1865 shinken   20   0 1632116  32604   7284 S   0.3  0.9   0:05.92 shinken-br+
 1908 shinken   20   0 1557024  30232   6556 S   0.3  0.8   0:03.24 shinken-re+
 1953 root      20   0 1633896  34552   5712 S   0.3  0.9   0:04.19 shinken-ar+
 2082 shinken   20   0 1631232  29112   4728 S   0.3  0.8   0:00.20 shinken-po+
 3684 aaronki+  20   0   41908   3808   3104 R   0.3  0.1   0:00.04 top
    1 root      20   0  119696   5924   4040 S   0.0  0.2   0:01.45 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/0
```

Courtesy of tecmint.com

Linux is so popular due to its efficiency, adaptability, and simplicity. As stated before, it dominates the smartphone market because it is extremely lightweight; it doesnt even need a graphical user interface or the bloated features that Windows (10) suffers from. It gets the job done reliably and efficiently and can be configured an infinite amount of ways in order to tailor to any creators needs.

3 FreeBSD Overview

FreeBSD is an operating system derived from BSD, a version of UNIX developed at the University of California Berkeley. It offers advanced networking, performance, security and compatibility features and does a particularly good job as an internet or intranet server [4]. FreeBSD is extremely similar to Linux and shares all the same aspects of process and thread handling. FreeBSD is more commonly used in commercial settings but is available for personal use. Unlike Linux, FreeBSD uses tcsh instead of bash as its default shell, which is generally considered not as scripting friendly. One important distinction between Linux and FreeBSD is that Linux is merely a kernel while FreeBSD is an entire Operating system.

4 Comparison

By simply reading the descriptions of each a neat picture of the differences and similarities between the different Systems is made pretty clear, but there are a few more key aspects to discuss. Firstly, closing a parent process in Linux kills all children processes whereas in Windows the children stay alive. Another big difference is that Windows processes dont share very much information between parent and child while our UNIX friends greatly intertwines the information between the processes. Lastly, Linux in general is considered to have better process handling and creation with fork() compared to windows CreateProcess() based on speed and efficiency.

Windows, Linux, and FreeBSD all use priority to choose which processes first, and all use something called processor affinity this enables the binding and unbinding of a process or a thread to a central processing unit (CPU) or a range of

CPUs, so that the process or thread will execute only on the designated CPU or CPUs rather than any CPU. [5]

For every job there are different ways to accomplish it, and the same goes for operating systems. Modern technology allows us to exceed in all areas of our lives and it is thanks to the similarities and differences in operating systems that we can choose exactly what we need for success.

References

[1] Windows Dev Center. About Processes and Threads May 30 2018.
`https://docs.microsoft.com/en-us/windows/desktop/procthread/about-processes-and-threads`

[2] Malissa Chau, Ryan Reith, Kathy Nagamine. Smartphone OS Marketshare October 2 2018.
https://www.idc.com/promo/smartphone-market-share/os

[3] David A. Rusling. Chapter 4 - Processes September 22 1999.
`https://www.tldp.org/LDP/tlk/kernel/processes.html`

[4] The FreeBSD Project. About FreeBSD January 23, 2018.
`https://www.freebsd.org/about.html`

[5] TMurgent Technologies. White Paper - Processor Affinity - Multiple CPU Scheduling November 3, 2003
`http://www.tmurgent.com/appv/en/`