OPERATING SYSTEMS II
CS444 FALL 2017

NOVEMBER 28, 2017

OPERATING SYSTEM FEATURE COMPARISON

MEMORY MANAGEMENT

BY

COREY HEMPHILL

Abstract

This document examines, compares, and contrasts low level operating system kernel memory management implementations in Windows, FreeBSD, and Linux operating systems.

I. Compare and Contrast FreeBSD and Linux

In the FreeBSD virtual memory system, every process is given its own private, protected 32 or 64-bit address space– depending on the CPUs architecture. Address spaces cannot be accessed by other processes, and are divided into three segments: text, data, and stack. When a process begins, it is mapped into the systems virtual address space. A process may begin execution even if part of the processs data is not present in the systems main memory; if not present, the system will page the necessary data into memory so that the process can access it. A process can map a file to anywhere in its address space, and can create a shared mapping of a file which can be accessed by other running processes. Shared mapping allows the system to minimize on RAM usage. When system memory becomes scarce, FreeBSD uses swapping and demand paging in an effort to continue running processes. Demand paging is where the system only retrieves pages from disk and places them in RAM when its necessary. FreeBSD will also take unused resources from other processes if they have not been recently used and attempt to redistribute those resources to processes that need them. Furthermore, the system may begin swapping a given processs context to a secondary storage place, usually a hard disk. Memory management in FreeBSD is geared toward multi-process context switching, where there are a large number of running processes and limited resources, and fairly sharing these resources amongst processes [1] [2].

In comparison, Linux and FreeBSD are extremely similar in terms of memory management, which makes sense because they are both, at their core, UNIX based operating systems. Linux systems support both 32 and 64-bit address spaces, and they also utilize a virtual memory system that provides a protected address space to processes, as well as paging, swapping, and shared virtual memory. Again, this makes a lot of sense, and if the method isnt broken, why fix it? Since RAM is expensive and a normal hard disk isnt, memory must be managed in a clever manner by the operating system to achieve an optimal pace. There are a few small differences between Linux and FreeBSD memory management. One such difference in FreeBSD is the system will reallocate unused resources to higher priority processes, whereas Linux relies on demand paging and swapping alone [3] [4].

II. Compare and Contrast Windows and Linux

In the Windows virtual memory system, every process is given its own private, protected 32 or 64-bit address space– depending on the CPUs architecture. Processes in Windows are provided 4 GB of private virtual address space. When a process begins, it is mapped into the systems virtual address space. A process may begin execution even if part of the processs data is not present in the systems main memory; if not present, the system will page the necessary data into memory so that the process can access it. A process can map a file to anywhere in its address space, and can create a shared mapping of a file which can be accessed by other running processes. Is this all sounding familiar? Thats because, conceptually, almost everything thats true for memory management in FreeBSD is also true for Windows, with very few exceptions. Windows utilizes a method called prefetching, in which files from the hard disk are retrieved and placed into physical memory before they are actually needed by the kernel. Windows also uses a method called lazy allocation, in which memory will not be allocated until is it absolutely necessary. When system memory becomes scarce, Windows will begin copying out 4 KB address space pieces from RAM, and move them to the hard disk into the Pagefile.sys file. Pagefiles live in the root of a given partition and the size of a pagefile can be configured by the user. Furthermore, a pagefile can store larger pages that

dont fit into physical memory. Unlike FreeBSD, Windows will not take unused resources from idle processes. Windows systems will also use swapping when RAM is too full. Memory management in Windows, similar to FreeBSD, is designed for multi-process context switching, where there are a large number of running processes and limited resources, as well as sharing its limited resources equally amongst all running processes [5] [6].

For the most part, it appears that Windows and Linux have more similarities than differences when it comes to how they both approach memory management–which is actually not very surprising for some reason. Virtual memory management, paging, use of swap memory, shared virtual memory; its all the same in concept. The details regarding how these concepts are implemented under the hood, so to speak, may be different, but the ideas behind the how and the why are essentially the same. Which of these operating systems is faster or more efficient in their memory management is arguable. The best answer to that question is, it depends on what youre trying to do. [3] [4].

III. Conclusion

The underlying theme of it all seems to be that RAM is a lot more expensive than a mechanical hard drive, so system memory management has to be economical by design. Most modern operating systems memory management is predicated on the assumption that most computers will have more hard disk space immediately available than they will RAM at any given time; this seems to be a pretty solid assumption, at least for now. The core idea behind virtual address spaces, shared virtual memory, paging, and swapping is to make data available to the processor as quickly as possible. Since RAM is very expensive, and a hard disk drive is vastly cheaper, there is usually far more hard disk to work with than there is RAM space. Therefore, memory must be managed in a clever manner by the operating system in order to make up for this disparity in the availability of these resources. These very slightly varied management schemes are far from perfect, but it is interesting to see that, for the most part, Windows, FreeBSD, and Linux are in general agreement on how to approach the management of memory resources. One day this may change, though.

REFERENCES

[1] "Memory management," https://www.freebsd.org/doc/en/books/design-44bsd/overview-memory-management.html, FreeBSD, [Online; accessed 25-November-2017].

[2] R. N. M. W. Marshall Kirk McKusick, George V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*. Microsoft Press, 2012.

[3] "Memory management," http://www.tldp.org/LDP/tlk/mm/memory.html, The Linux Documentation Project, [Online; accessed 25-November-2017].

[4] R. Love, *Linux Kernel Development*. Pearson Education International, 2010.

[5] "Ram, virtual memory, pagefile, and memory management in windows," https://support.microsoft.com/en-us/help/2160852/ram--virtual-memory--pagefile--and-memory-Microsoft, [Online; accessed 25-November-2017].

[6] "Virtual address space," https://msdn.microsoft.com/en-us/library/windows/desktop/aa366912(v=vs.85).aspx, Microsoft, [Online; accessed 25-November-2017].