Assignment 2

This report is based on two parts, that include two operating systems. One of these operating systems is Windows, which is the operating system that belongs to Microsoft, whereas the other operating system is MacOS, which is an operating system that belongs to Apple. The differences and similarities of Windows and MacOS will be inspected respective to the process and memory management techniques throughout this report. Within the operating system families Windows and Macintosh, the ones that will be referred in this report will be the latest versions of Windows 10 and MacOS. As both operating system families are widely used, graphical user interface-based systems for personal computers since 1985, hypothetically, comparison between their architecture would be clear as they are designed to provide same services, at the same time (as version).

A process is private in respect to other processes, and both as in code and data, it has its individual virtual address location. Every process may contain single or multiple threads, that each can work individually to execute application code. Moreover, a single thread can also create new independent processes and threads, as well as managing communications and synchronization between the new and already existing threads. Process management investigation part of this report aims to compare how Windows and MacOS configure a process and their threads. For a long time, Windows has supported multiprocessor systems, which enables created threads to be allocated to different processors. A process is created by a function which has ten parameters that can be simply default values. This function returns two outputs, one for process and one for thread. Initially, a process begins with a single thread, which then can create more threads. Each created process and thread are identified with a number and an ID. Windows kernel-mode handles the execution of a process’ all threads to make sure the system operates properly and there are no zombie threads left behind. The system uses synchronization to prevent different processes to use the same resource. Current versions of Windows use preemptive scheduler that assigns priorities to the threads. This priority then can be edited by kernel based on the I/O and CPU usage. MacOS is a multitasking operating system, process wise, where a process may control a single or multiple thread. The scheduling is preemptive, provided by kernel. The complete set of processes of Process Manager run by a task that is specialized for multiprocessing, whose name is “blue task”. Blue task schedules processes based on round robin algorithm. Moreover, Thread Manager is used to schedule threads of each process. Thread Manager uses a multilevel feedback queue to schedule threads based on four bands of priorities; normal, system high priority, kernel mode only and real-time. Both systems are multitasking, so a process starts with a single thread and can have multiple later. Scheduling of processes and threads in each system is preemptive. However, the scheduling algorithms are different. Windows seems to prioritize the usage of resources when scheduling threads, unlike MacOS. In conclusion, both systems are designed to handle multiple processes and threads, doing so with different scheduling algorithms, which hypothetically causes Windows to handle heavier tasks easier than MacOS.

Memory in computing refers to data storage for immediate use by CPU. Memory is split to different ranks based on speed and size ratio, as primary storage or secondary storage. As memory is one of the resources that is used by processes and threads, its management is vital, to allocate portions of memory and free it after it is no longer needed for reuse in other tasks. Nowadays, most operating systems use virtual memory managers, which separate the memory addresses from physical addresses to allow processes to have required memory amount to carry out their tasks, even though the total size given to processes is beyond the actual available amount of RAM. To increase the given memory size to processes, techniques as paging or swapping is used between primary memory and secondary memory. The second investigation of this report is to find out how is the memory management is dealt and which techniques are used in both Windows and Mac OS. Windows OS has two different memory architectures as 32-bit and 64-bit. The virtual addresses enable addressing up to 4 gigabytes and 8 terabytes of memory, respectively to architecture type. All process’ threads are able to access this virtual memory space but they are not allowed to access memory that is used up by another process to prevent corruptions in the processes. The memory manager of Windows creates two memory pools which are paged and nonpaged. Both memory pools are located in the system reserved address space and they are in the virtual address space of processes. The difference between two is that, nonpaged pool contains the virtual memory address that does not require paging to physical memory. This is guaranteed if the corresponding kernel objects are allocated in the said physical memory. The paged pool can contain virtual memory that will be paged in and out occasionally. Each process has a default heap provided to them by the system. Heap function is useful for both memory security, as a heap can be private, and memory management. If a process has a private heap, only that process is able to reach to the contents of the heap. Furthermore, using a heap allows the system to free the memory after the process is done with the task, releasing the pages of the heap object. Also, a file mapping system is used by the system, so processes can reach to the files through virtual memory. A long time ago, Mac OS began the memory management with problems as the computer hardware did not support virtual memory addressing. The first problem this caused was fragmentation because of the continuous allocations of memory through the pointer it used. This was handled by implementation of heap. In fact, two types of heap, system heap and application heap. However, these heaps were not private, that is why there was a memory protection issue. When the company introduced a 32-bit system, they used pointer and handle with flags, such as “locked”, “purgeable”, or “resource”. The reason was to make the unused bytes useful. But when they changed the CPU they use in the further models, these flags caused addressing errors. The architects then had to implement a new way to use the flag system without having system crushes. However, once they implemented 64-bit processes, the limit of virtual addressing went up to 18 exabytes. This was achieved by mapping everything virtually within the system. The system used default pager and vnode pager to be able to map all the files in virtual memory. The kernel keeps comparing the total amount of free physical pages with a threshold value and if the number is below the threshold value, then the kernel swaps inactive pages out of memory to reclaim physical pages for the free list. To compare Windows and Mac OS memory management, including the past, Windows has a more reliable process of development and a more secure system, where Mac OS had many problems in implementation, the system is not as secure yet, but its power can go beyond Windows’.

In the conclusion of this report, both operating systems have a long past and it brings an insightful development process. In terms of process management, Windows has a secure and reliable system. Both systems use efficient scheduling algorithms, but Windows seems to be able to handle heavier work. As for memory management, Windows has a reliable process, to finally improve the system to be secure and efficient. MacOS on the other hand, had many problems in the process of development due to computer hardware. However, the system is now more efficient than Windows, even though it is not as reliable.

References:

* Bovet, D. and Cesati, M. (2007). *Understanding the Linux Kernel*. 3rd ed. Sebastopol: O'Reilly Media, Inc.
* Hart, J. (2010). *Windows system programming*. 4th ed. Upper Saddle River, N.J: Addison-Wesley.
* Lewis, R. (2000). *MAC OS in a nutshell*. 1st ed. Sebastopol, CA: O'Reilly.
* En.wikipedia.org. (2019). *Mac OS memory management*. [online] Available at: https://en.wikipedia.org/wiki/Mac\_OS\_memory\_management [Accessed 9 May 2019].
* En.wikipedia.org. (2019). *Macintosh operating systems*. [online] Available at: https://en.wikipedia.org/wiki/Macintosh\_operating\_systems [Accessed 9 May 2019].
* En.wikipedia.org. (2019). *MacOS*. [online] Available at: https://en.wikipedia.org/wiki/MacOS [Accessed 9 May 2019].
* Docs.microsoft.com. (2019). *Memory Management - Windows applications*. [online] Available at: https://docs.microsoft.com/en-us/windows/desktop/Memory/memory-management [Accessed 9 May 2019].
* Scribd. (2019). *MEMORY MANAGEMENT mac os x | Operating System | Kernel (Operating System)*. [online] Available at: https://www.scribd.com/doc/23120100/MEMORY-MANAGEMENT-mac-os-x [Accessed 9 May 2019].
* En.wikipedia.org. (2019). *Microsoft Windows*. [online] Available at: https://en.wikipedia.org/wiki/Microsoft\_Windows [Accessed 9 May 2019].
* En.wikipedia.org. (2019). *Process management (computing)*. [online] Available at: https://en.wikipedia.org/wiki/Process\_management\_(computing) [Accessed 9 May 2019].
* Russinovich, M. (2008). *Windows internals*. 5th ed. Microsoft Press.
* En.wikipedia.org. (2019). *Scheduling (computing)*. [online] Available at: https://en.wikipedia.org/wiki/Scheduling\_(computing) [Accessed 9 May 2019].
* Docs.microsoft.com. (2018). *Windows Kernel-Mode Process and Thread Manager - Windows drivers*. [online] Available at: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-process-and-thread-manager [Accessed 9 May 2019].
* Encyclopedia Britannica. (2019). *Windows OS | History & Versions*. [online] Available at: https://www.britannica.com/technology/Windows-OS [Accessed 9 May 2019].