Relationship between Operating System, Computer Hardware, Application Software and Other Software

John O. Ugah[1], Sunday C. Agu[2], Felix Elugwu [3]
[1] Department of Computer Science, Ebonyi State University, Abakiliki, Nigeria
[2]Department of Computer Science, Madonna University, Elele, Nigeria
[3] Department of Computer Science, Delta State Polytechnic, Otefe-Oghara, Nigeria

Abstract

This paper studied the relationship between operating system (OS), computer hardware, application software and other software, the functions of OS to computer hardware and application software, the types of OS and the application programs they manage. It also studied the OS abstraction of physical memory to curb the security issues on processes that may arise as a result of multiple programs residing concurrently in memory, where a process may read or worse, write some other process's memory, thus explaining the concepts of memory visualization and principle of isolation. Moreover, it looked into how OS manages various I/O devices, taking an application I/O request and sending it to the physical device, and taking the response back from the device to the application, thus exploring the three approaches available through which the CPU communicate with the Devices: Special Instruction I/O, Memory-mapped I/O and the Direct memory access (DMA).

Keywords - *Operating System, Hardware, Software, Abstraction, Address Space, I/O Devices.*

I. INTRODUCTION

Computer hardware is the physical parts or components of a computer, such as the monitor, keyboard, computer data storage, graphic card, sound card and motherboard. By contrast, software is instructions that can be stored and ran by hardware. Hardware is directed by the software to execute any command or instruction. A combination of hardware and software forms a usable computing system [1].

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes system calls to an act independently of one another [2]. For a computer to effectively manipulate data and produce useful output, its hardware and software must work together. Without software, computer hardware is useless. Conversely, computer software cannot be used without supporting hardware. Similarly, computer software has to first be loaded into the computer's hardware and then executed.

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, etc. An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response back from the device and send it to the application [3]. Operating System – I/O software is often organized in the following layers: User Level Libraries, Kernel Level Modules and the Hardware. A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance.

Application software is a type of computer program that performs a specific personal, educational, business and other functions in different areas of human endeavor. Each program with the help of OS and hardware support is designed to assist the user with a particular process, which may be related to productivity, creativity, and/or communication [4].

Computer software is basically programs and procedures intended to perform specific tasks on a system from the lowest level assembly language to the high level languages [5]. Computer software systems are classified into three major types namely system software, programming software and application software operating on a particular hardware platform.

A computer system consists of two major elements: hardware and software [6]. Computer hardware is the collection of all the parts you can physically touch. Computer software, on the other hand, is not something you can touch. Software is a set of instructions for a computer to perform specific operations. An OS is a software program that enables the computer hardware to communicate and operate with the computer software [7].

Over the years, OS has implemented different techniques such as multiprogramming, timesharing to enhance memory and make processes available to computers users at a given time. In timesharing, OS allows multiple programs to reside concurrently in

memory which makes protection an important issue; where a process may read or worse, write some other process's memory [8].

This paper studied how OS abstracts physical memory known as the address space which is the running program's view of the memory. It also studied the OS I/O hardware and software and explores the basics and functions of OS, hardware and software.

II. OPERATING SYSTEM, COMPUTER HARDWARE AND APPLICATION SOFTWARE

An operating system (OS), in its most general sense, is software that allows a user to run other applications on a computing device. While it is possible for a software application to interface directly with hardware, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details. The operating system manages a computer's hardware resources, including: (a) Input devices such as a keyboard and mouse. (b) Output devices such as display monitors, printers and scanners. (c) Network devices such as modems, routers and network connections. (d) Storage devices such as internal and external drives.

The OS also provides services to facilitate the efficient execution and management of, and memory allocations for, any additional installed software application programs. It consists of many components and features. Which features are defined as parts of the OS vary with each OS. However, the three most easily defined components are: (a) Kernel: This provides basic-level control over all of the computer hardware devices. Main roles include reading data from memory and writing data to memory, processing execution orders, determining how data is received and sent by devices such as the monitor, keyboard and mouse, and determining how to interpret data received from networks. (b) User Interface: This component allows interaction with the user, which may occur through graphical icons and a desktop or through a command line (c) Application Programming Interfaces: This component allows application developers to write modular code [3].

Hardware refers to the physical elements of a computer. This is also sometime called the machinery or the equipment of the computer. Examples of hardware in a computer are the keyboard, the monitor, the mouse and the central processing unit. However, most of a computer's hardware cannot be seen; in other words, it is not an external element of the computer, but rather an internal one, surrounded by the computer's casing (tower). A computer's hardware is comprised of many different parts, but perhaps the most important of these is the motherboard. The motherboard is made up of even more parts that power and control the computer [9].

An application program (app or application for short) is a computer program designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user. Examples of an application include a word processor, a spreadsheet, an accounting application, a web browser, a media player, an aeronautical flight simulator, a console game or a photo editor. The collective noun application software refers to all applications collectively. This contrasts with system software, which is mainly involved with running the computer. Applications may be bundled with the computer and its system software or published separately, and may be coded as proprietary, open-source or university projects. Apps built for mobile platforms are called mobile apps [10].

A. Functions of Operating System to Computer Hardware and Application Software

According to [2], an operating system performs the following functions to either the computer hardware or application software:

- Booting: Booting is a process of starting the computer. The operating system starts the computer to work. It checks the computer and makes it ready to work.
- Memory Management: The memory cannot be managed without operating system. Different programs and data execute in memory at one time. If there is no operating system, security can be compromised and the system will not work efficiently.
- Loading and Execution: An application program is loaded in the memory before it can be executed. Operating system provides the facility to load programs in memory easily and then execute it.
- Data security: Data is an important part of computer system. The operating system protects the data stored on the computer from illegal use, modification or deletion.
- Disk Management: Operating system manages the disk space. It manages the stored files and folders in a proper way.
- Process Management: CPU can perform one task at one time. If there are many tasks, operating system decides which task should get the CPU.
- Device Controlling: operating system also controls all devices attached to computer. The hardware devices are controlled with the help of small software called device drivers.
- Printing controlling: Operating system also controls printing function. If a user issues two print commands at a time, it does not mix data of these files and prints them separately.
- Providing interface: It is used in order that user interface acts with a computer mutually. User interface controls how you input data and instruction and how information is displayed on screen. The operating system offers two types of the interface to the user:
  i. Graphical-line interface: It interacts with of visual environment to communicate with the

computer. It uses windows, icons, menus and other graphical objects to issues commands.

Command-line interface: it provides an interface to communicate with the computer by typing commands.

B. Types of Operating Systems and the Application Programs they manage

- Single and Multi-tasking OS: A single-tasking operating system can only run one program at a time, while a multi-tasking operating system provides the ability to run more than one task at once. Multi-tasking may be characterized in preemptive and co-operative types. In preemptive multitasking, the operating system slices the CPU time and dedicates a slot to each of the programs. Unix-like operating systems, such as Solaris and Linux—as well as non-Unix-like, such as Amigos—support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions of both Windows NT and Win9x used preemptive multi-tasking [11].
- Single- and multi-user OS: Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem. A multi user operating system let more than one user access the computer system at a time. Access to the computer system is normally provided via a network, so that users access the computer remotely using a terminal or other computer. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users [12].
- Distributed OS: This s a situation whereby the OS is decentralized. In this case each process in a unit undergoes a complete execution. It does not wait for a process to be executed at a particular unit and hence, it is termed Distributed Processing. In an OS, distributed and cloud computing context, tinplating refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in virtualization and cloud computing management, and is common in large server warehouses.
- Embedded OS: are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.
- Real time OS: are systems that respond to input immediately. This category includes operating systems designed substantially for the purposes of controlling and monitoring external activities with timing constraints. They are used for such tasks as navigation, in which the computers must react to a steady flow of new information without interruption. A real-time operating system may be single- or multi-tasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved.

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries and composed with the application and configuration code to construct a unikernel: a specialized, single address space, machine image that can be deployed to cloud or embedded environments

III. OPERATING SYSTEM ABSTRACTION

According to [8], to curb the security issues that may arise as a result of multiple programs residing concurrently in memory Fig. 1 where a process may read or worse, write some other process's memory, OS create an easy to use abstraction of physical memory Fig. 2. This abstraction is called the *address space*, and it is the running program's view of memory in the system. Hence, the address space of a process contains all of the memory state of the running program.

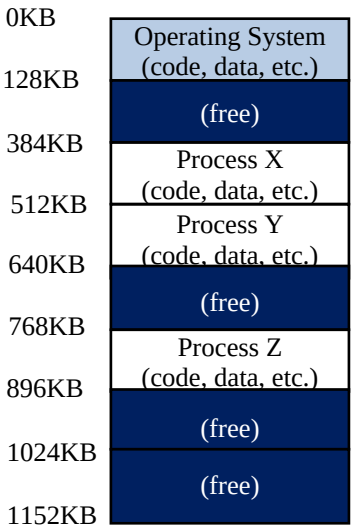

Fig 1: Three Processes sharing memory [1].

For instance figure 3.2 shows that the program code lives in memory and as it runs, it uses a stack to keep track of where it is in the function call chain as well as to allocate local variables and pass parameters and return values to and from routines. Lastly, the heap is used for dynamically-allocated, user-managed memory, such as that you might receive from a call to malloc() in C or new in an object oriented language. Address space also contains other components apart from code, stack and heap such as statically-initialized variables).

In Figure 3.2, the address space is only 32KB. The program code lives at the top of the address space

(starting at 0, and is packed into the first 2K of the address space). Code is static. It is placed at the top and it does not require extra space as the program runs. The heap and the stack are placed at opposite ends of the address space because each can grow. The heap begins after the code at 2KB and grows downward such as when more memory is requested by a user through malloc(). The stack begins at 32KB and grows upward such as when a procedure call is made by a user. The placement of stack and heap could be arranged in a different style as when multiple threads co-exist in an address space but the placement used in figure 3.2 is a convention.

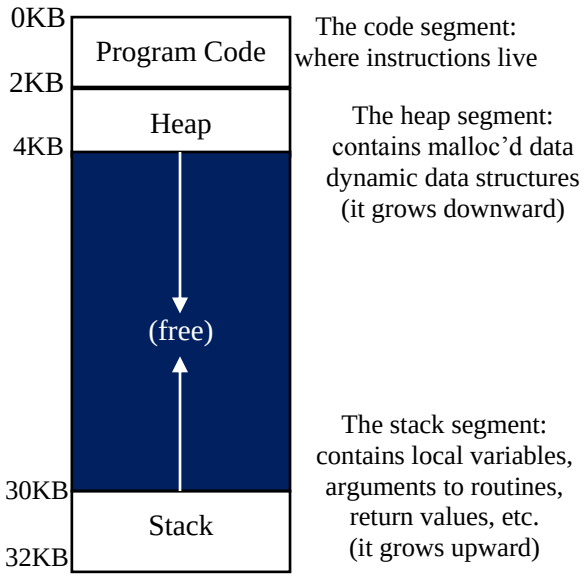

Figure 3.2: An Address space [1].

In address space which is the abstraction that the OS is providing to the running program, the program really is not in memory at physical addresses 0 through 32KB. Rather it is loaded at some randomly selected physical addresses. Examine processes A, B, and C in Figure 3.1; each process is loaded into memory at a different address which poses the security problem:

A. Memory Virtualization

The act of OS abstraction of physical memory is referred to memory virtualization because the running program thinks it is loaded into memory at a particular address (such as 0) and has a potentially very large address space (such as 32-bits or 64-bits). The reality is quite different [8]

When, for example, process A in Figure 3.2 tries to perform a load at address 0 (which we will call a virtual address), somehow the OS, in tandem with some hardware support, will have to make sure the load doesn't actually go to physical address 0 but rather to physical address 320KB (where A is loaded into memory). This is the key to virtualization of memory, which underlies every modern computer system in the world

B. The Principle of Isolation

Isolating two objects means that the shutting down of one of the objects would not affect the other. OS ensures that processes are isolated from each other thus avoiding one from damaging the other [8]. Additionally, this technique ensures that the running programs cannot affect the operation of the underlying OS. Recent OS barricade pieces of the OS from other pieces of the OS. Such micro kernels thus may provide greater reliability than typical monolithic kernel designs

IV. OPERATING SYSTEM – I/O HARDWARE

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc [3]. An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application.

A. Device Controllers

Device drivers are software component that are used in OS which enable a particular device to work. Device drivers assist OS to take charge of all I/O devices. A Device Controller serves as a contact point between a device and a device driver. As a contact point, its primary function is to convert serial bit stream to block of bytes and perform error correction as necessary. A device controller and a corresponding driver must be available for each device to communicate with the Operating Systems.

B. Communication to I/O Devices

There are three methods through which the CPU communicate with the Device: Special Instruction I/O, Memory-mapped I/O and the Direct Memory Access (DMA) [3].

Special Instruction I/O: make use of specific CPU instructions that allow data to sent to or read from an I/O device

Memory-mapped I/O: in this approach, the memory and I/O devices share the same address space hence, application data can be transferred to or from I/O devices and memory without passing through CPU. This method is beneficial because an I/O device can be operated by every instruction that can access the memory. Majority of high-speed I/O devices like disks and communication interfaces use memory mapped IO.

Direct Memory Access (DMA): DMA unit controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred. OS uses DMA to reduce the time it uses in handling the interrupts to the main CPU generated by either slow devices such as

keyboard or fast devices such as disk after each byte is transferred by these devices. DMA requires DMA controller (DMAC) that manages the data transfers and enables access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles figure 4.1

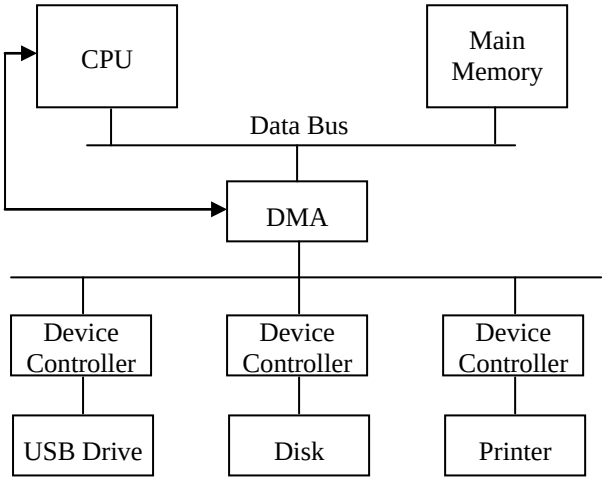

Figure 4.1: Direct Memory Access [3]

V. CONCLUSION

Hardware and software are mutually dependent on each other. Both of them must work together to make a computer produce a useful output. A driver software component provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details about the hardware being used. Software cannot be utilized without supporting hardware and vice versa

OS creates address space, an abstraction of the physical memory which is the running view of the memory, to curb the security issues that may arise as a result of multiple programs residing concurrently in memory. This act is referred to as virtualization because the running program thinks it is loaded in memory at a particular address. Moreover, OS uses isolation principles the separate processes to ensure that the operation of one process does not affect the other and the operations of the underlying OS.

DMA offers better advantage over the other methods through which the CPU communicate with the devices as the OS used DMA to reduce the time it uses in handling the interrupts to the CPU. In addition, DMA controllers are programmed with (1) source and destination pointers which indicate where to read / write data and (2) counters which track the number of transferred bytes.

REFERENCES

[1] Keizer, G. Microsoft gets real, admits its device share is just 14%, Computerworld. IDG, [Microsoft's chief operating officer], 2014.

[2] Sophia T. (2018) The Relationship Between Hardware and Software. [Online]. Available: https://www.sophia.org/tutorials/the-relationship-between-hardware-and-software.

[3] Tutorial Point. (2018) Operating System – I/O. [Online]. Available: https://www.tutorialspoint.com/operating_system/os_io_hardware.htm

[4] Custer, H. Generations of Computer, Microsoft Press, USA. Pp 78, 1993.

[5] David, C. Introduction to General Purpose Computing (Part One), CNET Prints, New York. Pp 31, 2001.

[6] Campbell, K. and Aspray, W. Computer: A History of the Information Machine, New York: Basic Books. Pp 34-36, 2006.

[7] Anand, L. (2010) The Xbox One - Mini Review & Comparison to Xbox 360/PS4. [Online]. Available: anandtech.com.

[8] Arpaci-Dusseau. (2014) The Abstraction: Address Spaces. [Online]. Available: http://pages.cs.wisc.edu/~remzi/OSTEP/vm-intro.pdf..

[9] Ceruzzi, Paul E. History of Modern Computing, Cambridge, Mass.: MIT Press. Pp 67, 2000.

[10] Mike, N. Fundamental of Computer, Windows Team Blog. Microsoft. 2008.

[11] Ulrich, W. (2013) Application Package Software: The Promise Vs. Reality, Cutter Consortium. [Online]. Available: https://www.cutter.com/article/application-package-software-promise-vs-reality-39387.

[12] Gassée, Jean-Louis (2012) The Silly Web vs. Native Apps Debate. [Online]. Available: https://web.archive.org/web/20160415200141/http:/www.thisurlisfalse.com/the-silly-web-vs-native-apps-debate/.