Journal of Technology and Informatics (JoTI)
Vol. 6, No. 1, October 2024
P-ISSN 2721-4842
E-ISSN 2686-6102 DOI: 10.37802/joti.v6i1.637

Operating Systems (OS): An Insight Investigative Research Analysis and Future Directions

Zarif Bin Akhtar[1]
[1]Department of Engineering, University of Cambridge, Cambridge, United Kingdom
e-mail: zarifbinakhtarg@gmail.com[1]
* Corresponding author: E-mail: zarifbinakhtarg@gmail.com

Abstract: *In the realm of technological computing, the pivotal interface of operating systems (OS) governs the orchestration of machinery, orchestrating seamless human interactions with the swiftly advancing array of device peripherals. Over decades, the intricacies of computing have undergone a profound metamorphosis, embracing monumental leaps facilitated by the progressive proliferation of operating system distributions. From the erstwhile colossal processing units to the present-day intricately crafted nano-fabricated microcontrollers, motherboards, and chipsets, all human-computer interactions gravitate towards the nuanced tapestry of OS distributions and intricately woven source-coded programming. This comprehensive research endeavors to undertake a meticulous exploration of the myriad typologies of operating systems, intricately dissecting their distinctive functionalities and performance metrics, with a discerning focus on aligning specific user profiles with the most fitting OS distributions. Moreover, this investigation seeks to unravel the labyrinthine landscape of OS distributions, illuminating the optimal pathways for both seasoned users and neophytes alike.*

Keywords*: Computing; Computer Architecture; Computation Processing; Information Technology; Open-Source; Operating Systems (OS); Security; Technological Computing; System Distributions.*

INTRODUCTION

In the dynamic landscape of the 21st century's tech sector, groundbreaking innovations have significantly shaped the industry's trajectory. From the game-changing advent of the iPhone and Android to the meteoric rise of social media giants like Facebook and Twitter, this era has witnessed a seismic shift in the technological paradigm. The introduction of curved displays and foldable smartphones marked a pivotal moment in addressing the inherent limitations of traditional flat-screen displays. This research investigations precisely delves into the transformative potential of these novel technologies, illuminating how they have revolutionized the technological market since their inception. Notably, tech conglomerates such as Samsung, Honor, and Xiaomi have channeled significant financial resources into the development and deployment of these cutting-edge technologies, propelling the industry towards a new era of innovation and consumer engagement. As of now, Samsung has emerged as the frontrunner, overshadowing its Chinese counterparts, but the relentless investments by various companies are poised to intensify the competitive landscape, promising an expansive array of choices for discerning customers. A key aspect of this research is the comprehensive analysis of the various types of OS shortcomings [1-7], offering valuable insights into which type of device might be most suitable for users. Furthermore, it delves into prevailing consumer trends, providing a forward-looking perspective on the future trajectory of this technology segment [8-11]. By inspecting the merits and demerits of these innovative devices and distribution systems, the research aims to equip users and industry stakeholders with the knowledge necessary to navigate this ever-evolving landscape effectively.

This comprehensive investigation also examines computer operating systems (OS) and offers insightful analysis of emerging trends in this dynamic field [12,13]. OS functionality, serving as an intermediary interface between computer hardware and users, is a critical cornerstone in modern computing systems [14-18]. This research aims to discern the trajectory and projected evolution of OS, unraveling the intricate concepts underpinning its architecture and development over time. The investigative research analysis delves into the intricate components of OS, shedding light on the intricacies of its underlying structures and providing comprehensive insights into security issues associated with various types of OS architectures. Moreover, the exploration offers a compendium of best practices intended to bolster OS security, advocating for a dual-layered security approach [19]. This approach entails fortifying the OS with robust security policies while simultaneously embedding stringent security protocols within the hardware architecture of the OS [20-26]. A significant revelation of the research pertains to the identification of contemporary OS trends, which encompass an array of cutting-edge developments such as IoT OS, Cloud OS, AI-powered OS, Blockchain OS, Hybrid OS, and Container OS [27,28,29]. The analysis also precisely weighs the strengths and weaknesses of these major OS categories, enabling readers to gain a nuanced understanding of their relative merits and limitations. Furthermore, the research espouses a forward-thinking approach, advocating for the conception of a universal OS framework that accommodates diverse architectures, fostering a scalable environment for potential expansion. In an effort to address sustainability concerns, the research highlights the integration of green computing technologies within the design architecture, aimed at

mitigating power consumption issues and fostering environmentally responsible computing practices. By precisely addressing the intricacies of OS architectures and their evolving landscape, this research offers a holistic perspective that not only outlines the current state of the field but also provides invaluable insights into the potential pathways for future advancements and innovation in the realm of operating systems for modern computing systems.

METHODS AND EXPERIMENTAL ANALYSIS

The methodology for this research follows a systematic approach to investigate the impact of operating systems (OS) on distribution systems and computing. Initially, a comprehensive iterative background research explorations with available knowledge investigations was conducted to gather existing knowledge and identify research gaps. Data collection was performed using specific methods such as surveys, experiments, and data mining techniques to ensure relevance and accuracy. All the collected data underwent pre-processing, which included cleaning, normalization, and validation, to maintain quality and relevance. The performance and visualization techniques were evaluated using specific metrics, including system throughput, latency, scalability, resource utilization, and reliability. These metrics provided a benchmark to compare the evaluated techniques against traditional and existing computing approaches. This comparison highlighted the effectiveness and efficiency of the Operating Systems (OS) in enhancing technological computing performance. Results and findings were analyzed and interpreted in line with the research objectives. The analysis discussed the implications of OS advancements on technological computing performance and efficiency, providing insights into the near future developments. The findings demonstrated how OS could enhance computing, influencing the interconnected world of digital device peripherals. Finally, the research summarized the findings, acknowledged the limitations encountered, and suggested future research prospects. These suggestions focused on areas that could benefit from further investigation to enhance the understanding and application of OS in distribution systems and computing. By employing these specific methods and metrics, this methodology ensures a comprehensive exploration of the potential enhancements OS can bring to computing, contributing to improved performance and paving the way for accelerated innovations in the field.

BACKGROUND ITERATIVE RESEARCH AND AVAILABLE KNOWLEDGE

An operating system (OS) is an essential system software that manages computer hardware and software resources, providing fundamental services for computer programs. OSs play a critical role in facilitating communication between applications and the hardware components of a computer system, including memory allocation and input/output functions.

They are mostly found across a wide range of devices, from smartphones and video game consoles to web servers and supercomputers. In the personal computer market as of September 2023, Microsoft Windows maintains a dominant market share of around 68%. macOS by Apple Inc. holds the second position with approximately 20%, and Linux variants, including ChromeOS, collectively account for about 7%. On the other hand, in the mobile sector, Android leads with a share of 68.92%, followed by Apple's iOS and iPadOS with 30.42%, while other operating systems collectively hold 0.6%. In the server and supercomputing sectors, Linux distributions dominate, while other specialized operating systems exist for embedded systems, real-time processing, and security-focused applications. Different types of operating systems cater to varying computing needs. Single-tasking systems can handle only one program at a time, while multi-tasking OSs allow the concurrent execution of multiple programs. This is achieved through time-sharing, where the processor time is divided among different processes. Multi-tasking can be preemptive or cooperative. Single-user systems support only one user but can execute multiple programs simultaneously. Multi-user systems facilitate the interaction of multiple users with the system and allocate resources accordingly.

Distributed systems manage a network of computers, making them appear as a single unit by distributing computations among connected computers. Embedded systems which operate in small, resource-constrained machines like PDAs, embedded OSs are designed for efficiency and compactness. Windows CE and Minix 3 are examples of embedded operating systems. Real-time systems are the OSs that ensure processing of events or data within specific time constraints. They can be single-tasking or multi-tasking, utilizing specialized scheduling algorithms to maintain deterministic behavior. Library operating systems are such systems which provide OS services in the form of libraries, composing with application and configuration code to form a unikernel—a specialized, single address space machine image deployable to cloud or embedded environments [12,13].

The history of operating systems dates back to the 1950s when early computers were programmed for specific tasks. As hardware evolved, basic OS features were developed, leading to the emergence of modern and complex operating systems in the 1960s. Early electronic systems were programmed using mechanical switches and plugboards, with no operating systems. The introduction of high-level languages and machine libraries in the late 1950s paved the way for the modern concept of operating systems. Notable early examples include GM-NAA I/O and the SHARE Operating System [14,15,16]. Mainframe computers in the 1950s pioneered key operating system features such as batch processing, multitasking, and spooling. IBM's OS/360 in the 1960s laid the foundation for a single OS spanning an entire product line.

Other significant mainframe OSs include Burroughs MCP, UNIVAC EXEC, General Electric's GECOS, and the Multiplexed Information and Computing Service (Multics) developed by Bell Labs, General Electric, and MIT. With the advent of microcomputers, minimalistic operating systems like CP/M and MS-DOS were developed in the 1970s and 1980s. The introduction of the Intel 80386 CPU chip in 1985 enabled microcomputers to run multitasking OSs.

The GNU Project, led by Richard Stallman, aimed to create a complete free software replacement for proprietary UNIX. Linus Torvalds's release of the Linux kernel in 1991 marked a significant milestone, leading to the development of the popular Linux OS. Microsoft Windows, initially built on top of MS-DOS, emerged as a dominant family of operating systems, gradually transitioning to the Windows NT kernel. Windows remains widely used, especially on personal computers, although it faces competition from Linux and BSD in the server market [17,18]. In addition to the major operating systems like Unix, Linux, macOS, and Windows, various other systems were once significant but have now become obsolete or niche, including AmigaOS, OS/2, classic Mac OS, BeOS, and others. Some systems like z/OS, OpenVMS, and IBM i continue to be actively used and developed, catering to specific enterprise needs. Academic environments also use specific systems such as MINIX and Singularity for educational and research purposes. The development of these various operating systems has significantly shaped the landscape of modern computing [30].

The operating system, as a crucial layer between user applications and computer hardware, comprises several fundamental components that enable the efficient functioning of a computer system. The kernel, at the core of the operating system, serves as the bridge connecting application software to the hardware components. With the assistance of firmware and device drivers, the kernel exerts control over the computer's hardware devices, managing memory access, allocating resources, and organizing data storage with file systems. It further regulates the CPU's operational states for optimal performance, ensuring the smooth execution of various processes and applications. The execution of application programs involves a systematic process facilitated by the operating system. This process includes the creation of a process by the kernel, where memory space and resources are assigned, and program binary code is loaded into memory. The operating system also sets the priority for the process in multi-tasking environments, ensuring efficient utilization of computing resources. Through this mechanism, application programs interact with users and hardware devices, adhering to predefined rules and procedures incorporated into the operating system [31]. Interrupts, both hardware and software, play a crucial role in the responsiveness and coordination of the operating system. Hardware interrupts enable the CPU to handle asynchronous events efficiently, allowing I/O devices to signal completion without requiring continuous CPU polling.

On the other hand, software interrupts serve as messages to processes, informing them of specific events or errors, such as time slices, error conditions, or user-initiated interruptions. These interrupts ensure the synchronization of processes, aiding in the seamless execution of multiple tasks within the system. Interrupt-driven I/O, a mechanism triggered by user inputs such as keystrokes or mouse movements, facilitates immediate responses to user actions, ensuring real-time interaction between users and the system. Additionally, direct memory access (DMA) allows high-speed data transfer between devices like hard disk drives and memory, circumventing the need for CPU intervention for each data transfer. The operating system manages these processes efficiently, enabling the seamless exchange of data and information between hardware devices and memory. The various components of the operating system work in tandem to provide a stable and efficient platform for the execution of applications, the management of hardware resources, and the seamless coordination of input and output operations. This collaborative functionality ensures a smooth and responsive user experience while harnessing the full potential of the computer's hardware capabilities [32]. The concept of operating systems is integral to the functioning of modern computers. Operating systems facilitate the interaction between the user and the hardware, providing a range of services including memory management, multitasking, disk access, networking, and security. Two key operating modes, user mode and supervisor mode, govern the level of access to resources, with the supervisor mode providing unrestricted access to machine resources and the user mode setting limits on instructions and direct access. Memory management is critical to ensure that programs don't interfere with one another, with memory protection serving as a key mechanism to restrict a process's access to the computer's memory [33,34]. Techniques such as memory segmentation and paging enable the kernel to control the memory accessed by different programs. Virtual memory, a crucial concept, allows the kernel to manage memory effectively by temporarily storing less frequently accessed memory on disks or other media. This makes space available for other programs, giving the illusion of a larger RAM capacity. Multitasking is another vital function that allows for the simultaneous execution of multiple independent computer programs, commonly achieved through time-sharing. Early models of multitasking were cooperative, allowing programs to execute for as long as they wanted, which could potentially lead to system crashes. Modern operating systems implement preemptive multitasking, ensuring all programs receive regular time on the CPU by utilizing a timed interrupt [35].

File systems enable the organization and management of files on a computer, with a hierarchical structure of directories or folders. Various file systems and their characteristics, such as naming conventions and access permissions, present challenges for implementing a single interface for all file systems.

To ensure compatibility, most operating systems provide support for widely used file systems and often require third-party drivers for others. Device drivers play a critical role in enabling interaction with hardware devices. They act as a mediator between the operating system and hardware devices, translating operating system calls into device-specific commands. Networking support in operating systems allows computers to share resources across a network, enabling functions like file sharing and remote access. Security measures within operating systems include authentication, authorization, and auditing, crucial for protecting sensitive data from unauthorized access [36,37]. Operating systems provide user interfaces to interact with computers, ranging from command-line interfaces to graphical user interfaces (GUIs). The evolution of GUIs has seen significant advancements, with many modern systems incorporating GUIs for better user experiences. Real-time operating systems are designed for applications with fixed deadlines, commonly used in embedded systems and industrial control. Operating system development as a hobby has led to the creation of unique systems independent of existing ones, often driven by individuals or small groups with shared interests. Ensuring software portability across various operating systems often requires adaptation or the use of software platforms like Java or Qt, which can minimize the costs of supporting diverse operating systems. Standardization efforts such as POSIX and OS abstraction layers have aimed to reduce the complexities of porting applications across different operating systems. To provide an idea concerning the perspective of the matter figure 1 provides an illustrative representation of the retrospect.

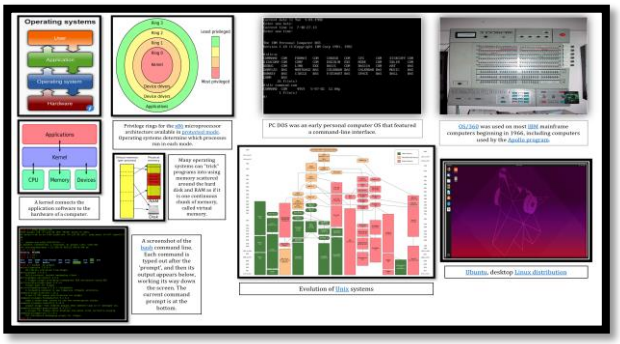

FIGURE 1. An overview of OS and its associated device integrations with distributions

THE DIFFERENT TYPES OF OPERATING SYSTEMS (OS)

Operating systems play a critical role in the functioning of computers and devices, providing the necessary software framework for managing hardware and software resources. Several types of operating systems have been developed to cater to different computing requirements and contexts. Each type has its own unique characteristics, benefits, and drawbacks, catering to a diverse range of applications and user needs.

Batch operating systems, typically utilized in the early days of computing, facilitated the processing of a series of similar jobs grouped together into batches. Despite their effectiveness in managing large workloads and allowing multiple users to share systems, batch systems posed challenges in terms of debugging, job failure impact, and cost, making them less favorable in contemporary computing environments. Distributed operating systems, a recent technological advancement, enable the connection of multiple independent computers through a unified communication channel, providing benefits such as fault tolerance, load distribution, and increased scalability. However, their complex software and high setup costs can present significant challenges, particularly in the event of network failures. Multitasking operating systems, also known as time-sharing systems, allow multiple users to efficiently access the CPU by allocating specific time periods for task execution. While offering equitable access and minimal idle time for the CPU, multitasking systems face issues such as data security concerns and potential communication problems, hindering their seamless operation. Network operating systems manage networking functions within a server-based environment, facilitating resource sharing, security management, and remote access for multiple users. Despite their stability and upgradability, these systems entail high server costs and maintenance requirements, making users heavily reliant on centralized operations. Real-time operating systems cater to time-sensitive applications, with hard real-time systems serving critical, time-constrained operations, and soft real-time systems addressing less stringent time constraints. While delivering optimal resource utilization and memory management, real-time systems are limited by expensive resources, complex algorithms, and restricted task capacity, presenting challenges in thread priority management and task switching. Mobile operating systems are designed for smartphones, tablets, and other mobile devices, allowing users to access various applications on the go. While providing user convenience, some mobile operating systems may exhibit limitations, such as poor battery performance and user interface issues, impacting overall user experience.

In addition to the types of operating systems, distinctions can be made based on single-tasking versus multi-tasking functionalities, desktop versus mobile compatibility, and open-source versus proprietary development models, each catering to specific user requirements and preferences. These diverse types and distinctions reflect the continuous evolution and adaptation of operating systems to meet the diverse needs of modern computing environments.

OS FUNCTIONALITIES, POPULARITY AND MARKET SHARE

Operating systems serve as the fundamental software framework for managing hardware and software resources, enabling the efficient functioning of computers and devices.

Their functions encompass critical tasks such as resource allocation, memory management, device management, user interface management, and security management, ensuring smooth and secure operations for users and applications. Resource allocation and management involve the efficient distribution of CPU, memory, and disk space among various applications, prioritizing tasks based on their importance. Memory management ensures optimal memory utilization and efficient sharing among running programs, facilitating seamless performance. Device management handles input and output devices, ensuring their compatibility and functionality within the system. User interface management provides a graphical user interface, enabling user interactions through windows, menus, and other visual elements. Security management safeguards systems and data through user authentication, firewalls, and antivirus software, protecting against unauthorized access and threats. Among the most widely used operating systems, Windows, macOS, Linux, iOS, and Android cater to diverse user preferences and requirements, offering a range of features and functionalities, from user-friendly interfaces to open-source customizability and robust security measures. Operating system evolution has traversed various generations, each marked by distinct technological advancements. From the earliest vacuum tube-based systems and machine language programming to contemporary AI-driven systems and quantum computing, operating systems have continually evolved to accommodate the growing complexities of computing tasks and user demands. The kernel is the fundamental program at the heart of a computer's operating system, exercising complete control over all system operations and hardware management. It serves as the intermediary between the computer hardware and software applications, ensuring smooth and efficient functionality. There are five distinct types of kernels: microkernel, monolithic kernel, hybrid kernel, exokernel, and nanokernel. Each type has unique characteristics and design philosophies, which influence their functionality, performance, and application. A microkernel architecture is designed to contain only the most essential functions of the operating system, such as low-level address space management, thread management, and inter-process communication (IPC). In a microkernel, user services and kernel services are implemented in separate address spaces. This separation enhances system stability and security, as a failure in user services does not directly affect kernel services. However, the design and implementation of a microkernel are complex, requiring more code and effort. Although microkernels are smaller in size and easier to extend, they typically suffer from lower execution speed due to the overhead of frequent context switches and message passing. Examples of operating systems using microkernels include Mac OS X. Contrastingly, a monolithic kernel runs the entire operating system as a single program in kernel mode, with both user services and kernel services sharing the same address space.

This design simplifies implementation and can lead to higher execution speeds because there is no need for context switching or message passing between user and kernel space. However, monolithic kernels are larger and more challenging to maintain and extend. A failure in any component within a monolithic kernel can potentially lead to a system-wide failure, making them less robust compared to microkernels. Debugging is also more difficult due to the integrated nature of the kernel components. Examples of monolithic kernels include Microsoft Windows 95.

A hybrid kernel combines elements of both microkernel and monolithic kernel architectures. It aims to take advantage of the performance benefits of monolithic kernels while maintaining the modularity and resilience of microkernels. This approach allows for a more balanced trade-off between performance and reliability. An exokernel architecture is designed to give application-level software greater control over hardware resources. By minimizing the abstractions provided by the kernel, exokernels allow applications to directly manage resources, potentially improving performance for specialized tasks. Nanokernels are even more minimalistic than microkernels, providing only the most fundamental services necessary to manage hardware resources. They delegate as much functionality as possible to higher-level software, which runs in user space. Understanding the different types of kernels and their respective architectures is crucial for optimizing system performance, stability, and security. Each kernel type offers distinct advantages and challenges, influencing the design decisions for various operating systems. The choice between microkernel, monolithic kernel, hybrid kernel, exokernel, and nanokernel architectures depends on the specific requirements and constraints of the computing environment.

The progression through multiple generations underscores the continual integration of cutting-edge technologies and the potential for future advancements that could revolutionize computing paradigms and interactions, possibly leading to seamless integration between human cognition and computing systems. To better understand the perspective of the matter figure 2 provides the global stat of OS market share and table 1 provides the illustration of 32bit and 64bit OS variations in terms of parameters.

TABLE 1. 32-bit OS vs. 64-bit OS

Parameter	32-Bit OS	64-Bit OS
Data and Storage	The 32bit OS can store and manage less data than the 64bit OS, as its name would imply. It mainly addresses a total maximum of 4,294,967,296 bytes (4 GB) of RAM in more detail.	In contrast, the 64bit OS has a larger data handling capacity than the 32bit OS. It indicates that a total of 264 memory addresses, or as in 18 quintillion gigabytes of RAM, can be addressed.

Compatibility of System	A 32-bit processor system will run only on 32-bit OS and not on 64bit OS.	A 64-bit processor system can run either a 32-bit or 64-bit OS
Application Support	The 32-bit OS support applications with no hassle.	The 64-bit OS do not support many of the older hardware and software applications.
Performance	Performance of the 32- bit OS is less efficient.	Higher performance than that of the 32-bit processor.
Systems Available	These support Windows 7, Windows XP, Windows Vista, Windows 8, and Linux.	These support Windows XP Professional, Windows 7, Windows 8, Windows 10, Windows Vista, Linux, and Mac OS X.

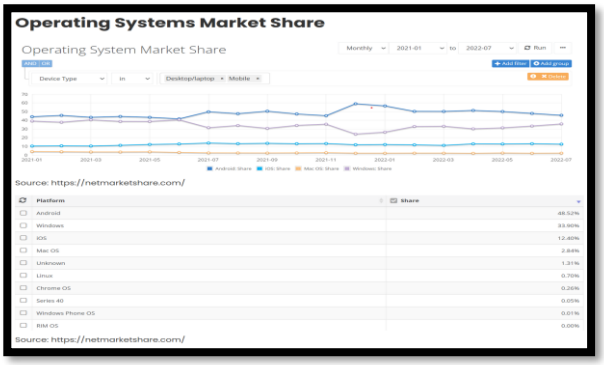

FIGURE 2. OS Market Share around the Globe

OS PROS AND CONS WITH RTOS

Operating systems play a critical role in ensuring the efficient and correct use of a computer's hardware, facilitating the simultaneous operation of various applications, and managing the organization of files and folders. They provide an intuitive user interface, simplifying interactions between users and machines, while also handling security measures to protect against unauthorized access and data breaches. Operating systems efficiently manage system resources, ensuring that hardware components are utilized optimally and that various applications have fair access to necessary resources. Furthermore, they manage printing operations, ensuring that documents and files are printed accurately and efficiently. Operating systems serve as a robust platform for software development, providing a stable and consistent environment for the creation and execution of diverse software applications. Despite their numerous advantages, operating systems are not without their drawbacks. They can be complex and challenging to use, especially for individuals with limited technical expertise, potentially posing barriers to entry for certain users.

The cost associated with purchasing and maintaining operating systems, along with the need for regular updates and maintenance, can impose financial burdens on organizations and individuals. Furthermore, the inherent complexity of operating systems can render them vulnerable to attacks from malicious users, necessitating robust security measures and constant vigilance to safeguard sensitive data and systems from potential threats and breaches.

Real-Time Operating Systems (RTOS) serve as specialized operating systems that execute multi-threaded programs while adhering to stringent real-time deadlines. Unlike the conventional notion of speed, the ***"deadlines"*** in an RTOS pertain to the ability to predict when specific tasks will run before their actual execution. RTOS proves to be an invaluable tool, particularly for complex embedded applications, offering support for task isolation and enabling concurrent operations. Its applications span various critical systems, including defense applications like RADAR, airline reservation systems, systems that demand immediate updating, networked multimedia systems, air traffic control systems, and command control systems. The seamless execution and adherence to real-time constraints make RTOS indispensable in scenarios where timely and accurate data processing and decision-making are of utmost importance.

HOW TO KNOW AND CHOOSE BEST OS

Selecting the most appropriate computer operating system (OS) involves a careful evaluation of several key factors to ensure compatibility and functionality. The price of an OS is an essential consideration, with options ranging from free, like Linux, to paid systems like Windows and macOS. Accessibility is another vital factor, with certain systems, such as macOS and iOS, offering user-friendly interfaces, while others, like Linux, can present a steeper learning curve. The compatibility of an OS with desired applications is crucial, as some systems support a broader range of software compared to others. Additionally, the security features of an OS should be weighed, as certain systems offer stronger security protocols than others.

When selecting an OS, it is critical to prioritize robust memory management, ensuring efficient utilization of system resources. Stability is paramount, especially for individuals relying on their computers for various tasks, whether business-related or for leisure activities like gaming. The OS should be reliable, avoiding frequent crashes and interruptions that could hinder productivity or enjoyment. Support and cost are equally important, with the understanding that paying for an OS does not necessarily guarantee better performance or assistance. Some free OS options offer robust support channels, while certain paid systems may fall short on delivering promised assistance.

Ultimately, the choice of an OS significantly impacts the user experience, influencing the overall functionality and performance of the computer.

Considering factors such as usability, compatibility with preferred applications, and the system's ability to support various tools and customization options are crucial when making this decision. Furthermore, understanding the role of an operating system in organizing and managing files and programs on the computer is fundamental to selecting the most suitable option.

While there may not be a definitive answer to the best OS, careful consideration of individual requirements and preferences will guide users towards making an informed choice and a conclusive decision.

OPERATING SYSTEMS (OS) SECURITY

Security is a critical aspect of any operating system, ensuring the protection of valuable computer resources, including the CPU, memory, disk space, software programs, and data. Authentication is a fundamental component of security, involving the identification of each system user and their association with executing programs. Operating systems employ various authentication methods, such as username/password combinations, user cards/keys, and user attributes like fingerprints or eye retina patterns, to ensure secure access.

One-time passwords enhance security protocols by requiring unique passwords for each login attempt. Implementations of one-time passwords involve the use of random numbers, secret keys generated by hardware devices, or network passwords sent to users' registered mobile or email accounts. Program threats, such as Trojan horses, trap doors, logic bombs, and viruses, pose significant risks by enabling unauthorized access to user credentials, introducing security vulnerabilities, or causing system malfunctions. System threats, including worms, port scanning, and denial of service attacks, can disrupt network performance and compromise system resources, resulting in severe consequences for users and their data.

Computer security classifications, as established by the U.S. Department of Defense Trusted Computer System's Evaluation Criteria, categorize security levels into four types: A, B, C, and D. Each classification represents varying degrees of security measures, with Type A offering the highest level of assurance through formal design specifications and verification techniques. Type B provides a mandatory protection system, while Type C focuses on user accountability and audit capabilities. Lastly, Type D represents the lowest security level, offering minimal protection and often associated with operating systems such as MS-DOS and Windows 3.1. These security classifications serve as vital benchmarks for evaluating and modeling the security of computer systems and their corresponding security solutions.

WINDOWS VS MACOS VS LINUX: AN INVESTIGATIVE ANALYSIS

The comparison of three leading operating systems, namely Windows, macOS, and Linux, reveals distinctive features and suitability for various user preferences and needs. Windows, known for its versatility and widespread usage, is well-suited for general productivity tasks, gaming, software development, and multimedia creation.

With its user-friendly interface and robust security protocols, Windows stands as a popular choice for users who prioritize compatibility, diverse software support, and extensive hardware compatibility. The system's robust security features, including Windows Defender and regular updates, contribute to a secure computing environment, although risks can arise from the installation of potentially malicious software.

MacOS, specifically designed for Apple devices, is celebrated for its seamless integration with the Apple ecosystem, catering to the needs of creative professionals and artists. Recognized for its enhanced security and privacy measures, macOS offers a visually appealing interface and top-notch performance. However, its application compatibility may be limited compared to Windows, and users might encounter challenges in running certain software programs not specifically designed for macOS. The system's superior security measures, such as Gatekeeper and FileVault, contribute to a secure computing experience, safeguarding users' personal data from external threats.

Linux, an open-source powerhouse, offers unparalleled flexibility and customization options, making it a favorite among tech-savvy users, developers, and system administrators. Known for its stability and performance, Linux is widely used for server management, web development, and data analysis tasks. Its strong focus on security and privacy, fostered by its open-source nature, provides users with granular control over system permissions and comprehensive protection against unwanted surveillance. Despite the availability of numerous applications and growing developer support, Linux may present challenges for users accustomed to more mainstream operating systems due to its unique customization options and learning curve for newcomers.

Each operating system offers distinct advantages and serves particular user needs, whether it be Windows' compatibility and user-friendliness, macOS' seamless integration within the Apple ecosystem and enhanced security, or Linux's flexibility, customization, and robust security features.

By understanding the specific requirements and preferences of users, the choice of an operating system can be tailored to individual needs, providing an optimized computing experience and meeting diverse computing demands. For a representative illustration figure 3 provides an insight into the matter.

FIGURE 3. Windows vs. macOS vs. Linux the illustrative representation

RESULTS, FINDINGS, OS CHALLENGES AND FUTURE DIRECTIONS

The realm of modern operating systems is confronted with an array of challenges that predominantly revolve around security, resource management, and device compatibility. Notably, security and privacy concerns have become paramount, as the prevalence of vulnerabilities and exploits, including malware and ransomware, constantly jeopardize the integrity of systems. Additionally, the rising apprehensions regarding user privacy have necessitated the implementation of stringent security measures, in response to issues such as data breaches and unauthorized data collection. Efficient resource management has emerged as a critical facet for optimizing system performance, with memory management techniques such as virtual memory and paging playing a pivotal role in effectively managing limited physical memory. Moreover, the proliferation of hardware devices has posed a significant obstacle for operating systems, demanding the deployment of hardware abstraction layers to ensure uniform management of devices with distinct interfaces. Furthermore, the integration of device drivers has become essential for facilitating communication between hardware and software, while the plug-and-play functionality has streamlined the process of device installation and configuration. Embracing containerization technologies like Docker and Kubernetes has ushered in a new era of lightweight and isolated environments for applications, augmenting scalability and portability within the operating system landscape. The concept of virtualization, encompassing both hardware and software virtualization, has enabled the simultaneous operation of multiple operating systems or instances on a single physical machine, amplifying the versatility of computing environments. Operating systems, functioning as system software, assume the crucial role of managing computer hardware and software resources, while providing fundamental services for computer programs across diverse devices, from cellular phones and video game consoles to web servers and supercomputers.

Notably, Linux distributions have carved a dominant niche in the server and supercomputing sectors, underscoring their versatility and robust capabilities. In light of the burgeoning Internet of Things (IoT) landscape, operating systems are poised to evolve further, necessitating adaptation to accommodate the unique demands of IoT devices. Embedded operating systems, such as FreeRTOS and Zephyr, have garnered significance owing to their prioritization of low resource consumption and real-time responsiveness, enabling seamless integration of IoT devices into networks. As the future unfolds, operating systems will continue to underscore the importance of robust security measures, with secure booting, sandboxing, and secure enclaves emerging as pivotal components in safeguarding system integrity and user privacy. The continued evolution of operating systems is poised to reshape the technological landscape, ushering in more secure, efficient, and adaptable computing environments.

The current structure of modern operating systems, particularly in the Unix/Linux domain, is closely entwined with the C programming language and a shell environment, leading to a complex and often brittle software ecosystem. The weak type systems of both C and the shell, coupled with their reliance on configuration files, make it challenging to verify the correctness of programs and system configurations in advance. Consequently, troubleshooting or configuring a Linux system often involves blindly following online instructions, creating an environment where errors can easily disrupt the system's functionality. This fragility has created an inaccessible hacking environment for many users, contradicting the open-source philosophy that advocates for users to have full control over their machines. The comparison between a program like ***'my_server'*** configured through a file-based approach and one written in a strongly-typed programming language highlights the stark contrast in robustness and predictability. While both scenarios may accomplish the task of running the server, the latter provides a significantly reduced risk of failure and allows for the verification of the program's validity beforehand. This distinction underscores the need for a paradigm shift in system configuration, suggesting a move toward a single declarative-style program for the entire system, akin to what NixOS is pursuing, yet with a stronger emphasis on pre-verification. The notion of content-based addressing and the shift from trusted to untrusted content and infrastructure are also critical elements in the reimagining of a modern operating system. Embracing the principle of identifying programs by the hash of their content fosters a more accessible and streamlined approach to program distribution, promoting the ease of program integration without extensive bureaucratic processes. Leveraging technologies like WebAssembly for cross-platform compatibility and sandboxing programs to isolate them from system resources emerges as a fundamental direction for enhancing security and predictability.

The idea of long-running daemons rather than short-lived programs aligns with the original Unix philosophy of specialized tools accomplishing specific tasks efficiently. This concept emphasizes the need for a shift in perspective, viewing the system as a collection of modules in the context of a programming language rather than isolated programs. Sandboxing technologies, such as Docker containers and virtualization, have arisen as a response to the existing operating system's limitations in isolating programs effectively. However, the challenge remains to ensure a default level of isolation and security within the system, guaranteeing that users can execute programs without jeopardizing the system's stability or their data. Furthermore, redefining the traditional concept of a file system in the context of cloud computing and content-based addressing offers the potential for a simplified and more transparent user experience. Eliminating the dependence on a shared global file system and reimagining the purpose and structure of files could significantly enhance the accessibility and reliability of data management within the system. Ultimately, the pursuit of a modern operating system revolves around creating an accessible, robust, and predictable environment that empowers users to navigate and configure their systems with ease and confidence.

DISCUSSIONS

In the world of operating systems, Windows, macOS, and Linux each come with their own set of specific hardware requirements. While Windows is generally compatible with a wide array of PC configurations, macOS is purpose-built for Apple computers. Linux, known for its broader hardware compatibility, still requires users to verify system requirements before installation. One critical factor to consider is software compatibility. Not all applications are universally compatible across different operating systems. Windows boasts the broadest range of software compatibility due to its widespread use, while macOS caters to a robust selection of creative and multimedia applications. Linux, although increasingly supported by developers, may not have direct alternatives for specialized or exclusive Windows/macOS applications. Cross-platform compatibility is also a consideration. Some Windows software can run on Linux using compatibility layers like Wine or virtualization software, although not all applications function seamlessly. While Linux has made significant strides in user-friendliness, it may still demand a bit more technical know-how compared to the more mainstream Windows and macOS systems. Regarding gaming, Windows is the preferred operating system, providing extensive game libraries and strong driver support. macOS, although offering a smaller gaming selection, still supports several popular titles. Linux has seen substantial progress in the gaming realm, with a growing number of games and compatibility layers like Steam's Proton, enabling gaming on the platform.

In other words, selecting the appropriate operating system depends on individual priorities, needs, and preferences. Windows, macOS, and Linux cater to different user requirements, with each system offering unique features. Windows excels in versatility and gaming capabilities, macOS in elegance and security within the Apple ecosystem, and Linux in flexibility, customization, stability, and security. Therefore, understanding one’s specific requirements is crucial when making an informed decision, as the choice of operating system significantly impacts ones overall digital experience.

CONCLUSIONS

Operating systems play a crucial role in the functioning of computers, managing resources, providing user interfaces, and ensuring security. Windows, macOS, and Linux are three prominent operating systems, each with distinct strengths and weaknesses. Windows stands out for its versatility, extensive software compatibility, and robust gaming capabilities, making it an excellent choice for a wide range of users. macOS, designed exclusively for Apple hardware, offers a seamless, visually appealing interface, robust security, and seamless integration within the Apple ecosystem, making it ideal for creative professionals and artists. Linux, known for its open-source nature, provides unmatched flexibility, customization options, and strong security, making it a preferred choice for tech-savvy users, developers, and system administrators. When selecting an operating system, factors such as price, accessibility, compatibility, security, and support should be considered. Windows enjoys widespread compatibility and a user-friendly interface, catering to diverse user needs. macOS, known for its stringent security measures and seamless integration within the Apple ecosystem, appeals to creative professionals and users valuing elegant design. Linux, with its open-source nature and strong security focus, is suitable for those prioritizing customization, privacy, and stability, particularly in server environments. Each operating system caters to specific user requirements, making it essential to assess individual preferences and needs when making a choice. Considerations such as software compatibility, hardware requirements, and cross-platform usability are crucial. Windows offers the broadest software compatibility, macOS specializes in creative and multimedia applications, while Linux provides a growing selection of software options, often necessitating the use of compatibility layers or virtualization software for cross-platform compatibility. While Windows is widely recognized as the go-to operating system for gaming, macOS and Linux have also made significant strides in gaming support, albeit with a more limited selection of titles. Understanding these factors is essential for making an informed decision, as the choice of an operating system significantly impacts the overall digital experience, productivity, and security of a computer system.

Operating system standards are continually evolving to meet the demands of advancing technologies, security, performance, and user experience. A significant trend in operating system design is the move towards modular and microkernel architectures, aiming to strike a balance between functionality and complexity. While monolithic kernels provide high performance and compatibility, they also raise concerns regarding bugs, crashes, and security vulnerabilities. In contrast, modular or microkernel architectures, as seen in operating systems like Windows NT, Linux, and MINIX, prioritize reducing size, improving reliability, and enhancing security, although they may introduce overhead and communication challenges. Additionally, the increasing prevalence of cloud and edge computing is influencing the future of operating system standards. These technologies facilitate remote data processing and storage, emphasizing scalability and efficiency. Operating systems are required to adapt to the demands of distributed and parallel computing, network and data management, as well as security and encryption. Examples such as Chrome OS, Android, and Azure Sphere demonstrate the integration of these principles. Moreover, the rise of artificial intelligence (AI) and machine learning (ML) with deep learning (DL) is shaping operating system standards by enabling complex pattern analysis, task automation, and optimization. Operating systems must integrate AI and ML capabilities into core functions, providing development tools and frameworks for AI and ML applications. macOS, iOS, and Linux showcase the incorporation of AI and ML features within their systems. User-centric and adaptive design represents another crucial aspect influencing the future of operating system standards. This design philosophy prioritizes improving usability, accessibility, and personalization, catering to changing user needs and preferences. Operating systems achieve this by incorporating features like voice and gesture control, biometric authentication, customization options, and context-awareness. Examples such as Windows 10, Ubuntu, and HarmonyOS exemplify the implementation of user-centric and adaptive design principles. The aspect of security and privacy is of paramount importance in shaping the future of operating systems. Operating systems are increasingly focusing on implementing robust security features such as encryption, authentication, authorization, sandboxing, firewall, antivirus, and update mechanisms. Adherence to data protection regulations such as GDPR and CCPA is also critical. Operating systems like Qubes OS, Tails, and iOS emphasize security and privacy as fundamental components of their standards, ensuring protection against unauthorized access and maintaining user privacy.

ACKNOWLEDGMENTS

The main prospect and the scope for this research was conducted with the idea perspective experimentations along with the manuscript writing was done by the author himself. All the data sources which have been retrieved and resourced for the conduction of this research exploration are mentioned and referenced where appropriate.

REFERENCES

[1] Microsoft. Windows Secure Channel Denial of Service Vulnerability. 2023. Available online: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2023-21813 (accessed on 1 June 2023).

[2] Research, G.S. Linux (Ubuntu)–Other Users Coredumps Can Be Read via Setgid Directory and killpriv Bypass. 2018. Available online: https://www.exploit-db.com/exploits/45033 (accessed on 1 June 2023).

[3] Gorbenko, A.; Romanovsky, A.; Tarasyuk, O.; Biloborodov, O. Experience report: Study of vulnerabilities of enterprise operating systems. In Proceedings of the 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), Toulouse, France, 23–26 October 2017; IEEE: New York, NY, USA, 2017; pp. 205–215. [Google Scholar]

[4] Cheikes, B.A.; Waltermire, D.; Kent, K.A.; Waltermire, D. Common Platform Enumeration: Naming Specification Version 2.3; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2011. Available online: https://csrc.nist.gov/publications/detail/nistir/7695/final (accessed on 2 June 2023).

[5] Vander-Pallen, M.A.; Addai, P.; Isteefanos, S.; Mohd, T.K. Survey on types of cyber attacks on operating system vulnerabilities since 2018 onwards. In Proceedings of the 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 6–9 June 2022; IEEE: New York, NY, USA, 2022; pp. 1–7. [Google Scholar]

[6] Kocaman, Y.; Gonen, S.; Baricskan, M.A.; Karacayilmaz, G.; Yilmaz, E.N. A novel approach to continuous CVE analysis on enterprise operating systems for system vulnerability assessment. Int. J. Inf. Technol. 2022, 14, 1433–1443. [Google Scholar] [CrossRef]

[7] Sonmez, F.O.; Hankin, C.; Malacaria, P. Attack Dynamics: An Automatic Attack Graph Generation Framework Based on System Topology, CAPEC, CWE, and CVE Databases. Comput. Secur. 2022, 123, 102938. [Google Scholar]

[8] Niu, S.; Mo, J.; Zhang, Z.; Lv, Z. Overview of linux vulnerabilities. In Proceedings of the 2nd International Conference on Soft Computing in Information Communication Technology, Taipei, Taiwan, 31 May–1 June 2014; Atlantis Press: Dordrecht, The Netherlands, 2014; pp. 225–228. [Google Scholar]

[9] Kaluarachchilage, P.K.H.; Attanayake, C.; Rajasooriya, S.; Tsokos, C.P. An analytical approach to assess and compare the vulnerability

risk of operating systems. Int. J. Comput. Netw. Inf. Secur. 2020, 12, 1. [Google Scholar] [CrossRef]
[10] Siwakoti, Y.R.; Bhurtel, M.; Rawat, D.B.; Oest, A.; Johnson, R. Advances in IoT Security: Vulnerabilities, Enabled Criminal Services, Attacks and Countermeasures. IEEE Internet Things J. 2023, 10, 11224–11239. [Google Scholar] [CrossRef]
[11] Gorbenko, A.; Romanovsky, A.; Tarasyuk, O.; Biloborodov, O. From analyzing operating system vulnerabilities to designing multiversion intrusion-tolerant architectures. IEEE Trans. Reliab. 2019, 69, 22–39. [Google Scholar] [CrossRef] [Green Version]
[12] Stallings (2005). Operating Systems, Internals and Design Principles. Pearson: Prentice Hall. p. 6.
[13] Dhotre, I.A. (2009). Operating Systems. Technical Publications. p. 1.
[14] "Desktop Operating System Market Share Worldwide". StatCounter Global Stats. Archived from the original on 2 October 2023. Retrieved 3 October 2023.
[15] "Mobile & Tablet Operating System Market Share Worldwide". StatCounter Global Stats. Retrieved 2 October 2023.
[16] "VII. Special-Purpose Systems - Operating System Concepts, Seventh Edition [Book]". www.oreilly.com. Archived from the original on 13 June 2021. Retrieved 8 February 2021.
[17] "Special-Purpose Operating Systems - RWTH AACHEN UNIVERSITY Institute for Automation of Complex Power Systems - English". www.acs.eonerc.rwth-aachen.de. Archived from the original on 14 June 2021. Retrieved 8 February 2021.
[18] Lorch, Jacob R.; Smith, Alan Jay (1996). "Reducing processor power consumption by improving processor time management in a single-user operating system". Proceedings of the 2nd annual international conference on Mobile computing and networking. New York, NY, US: ACM. pp. 143–154. doi:10.1145/236387.236437. ISBN 089791872X.
[19] Akhtar,Z.(2024).Securing Operating Systems (OS): A Comprehensive Approach to Security with Best Practices and Techniques. International Journal of Advanced Network, Monitoring and Controls,9(1) 100-111. https://doi.org/10.2478/ijanmc-2024-0010
[20] Javed, F.; Afzal, M.K.; Sharif, M.; Kim, B. Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review. IEEE Commun. Surv. Tutor. 2018, 20, 2062–2100. [Google Scholar] [CrossRef]
[21] Asghar, A.; Farooq, H.; Imran, A. Self-Healing in Emerging Cellular Networks: Review, Challenges, and Research Directions. IEEE Commun. Surv. Tutor. 2018, 20, 1682–1709. [Google Scholar] [CrossRef]
[22] Li, L.; Zhao, G.; Blum, R.S. A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies. IEEE Commun. Surv. Tutor. 2018, 20, 1710–1732. [Google Scholar] [CrossRef]
[23] Naik, G.; Liu, J.; Park, J.J. Coexistence of Wireless Technologies in the 5 GHz Bands: A Survey of Existing Solutions and a Roadmap for Future Research. IEEE Commun. Surv. Tutor. 2018, 20, 1777–1798. [Google Scholar] [CrossRef]
[24] Mukherjee, M.; Shu, L.; Wang, D. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. IEEE Commun. Surv. Tutor. 2018, 20, 1826–1857. [Google Scholar] [CrossRef]
[25] MacHardy, Z.; Khan, A.; Obana, K.; Iwashina, S. V2X Access Technologies: Regulation, Research, and Remaining Challenges. IEEE Commun. Surv. Tutor. 2018, 20, 1858–1877. [Google Scholar] [CrossRef]
[26] Dunkels, A. Full TCP/IP for 8-bit Architectures. In Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, San Francisco, CA, USA, 5–8 May 2003; pp. 85–98. [Google Scholar] [CrossRef]
[27] Al-Boghdady, A.; Wassif, K.; El-Ramly, M. The Presence, Trends, and Causes of Security Vulnerabilities in Operating Systems of IoT's Low-End Devices. *Sensors* 2021, *21*, 2329. https://doi.org/10.3390/s21072329
[28] Bazuku , R., Anab , A., Gyemerah , S., & Daabo , M. I. (2023). An Overview of Computer Operating Systems and Emerging Trends. Asian Journal of Research in Computer Science, 16(4), 161–177. https://doi.org/10.9734/ajrcos/2023/v16i4380
[29] X. Rong, "Design and Implementation of Operating System in Distributed Computer System Based on Virtual Machine," *2020 International Conference on Advance in Ambient Computing and Intelligence (ICAACI)*, Ottawa, ON, Canada, 2020, pp. 94-97, doi: 10.1109/ICAACI50733.2020.00024.
[30] Mishra, B.; Singh, N.; Singh, R. (2014). "Master-slave group based model for co-ordinator selection, an improvement of bully algorithm". International Conference on Parallel, Distributed and Grid Computing (PDGC). pp. 457–460. doi:10.1109/PDGC.2014.7030789. ISBN 978-1-4799-7682-9. S2CID 13887160.
[31] Hansen, Per Brinch, ed. (2001). Classic Operating Systems. Springer. pp. 4–7. ISBN 0-387-95113-X. Archived from the original on 11 January 2023. Retrieved 19 December 2020.
[32] "Intel® Microprocessor Quick Reference Guide - Year". www.intel.com. Archived from the original on 25 April 2016. Retrieved 24 April 2016.
[33] Arthur, Charles (5 January 2011). "'Windows 8' will run on ARM chips - but third-party apps will need

rewrite". The Guardian. Archived from the original on 12 October 2016.
[34] "Behind the IDC data: Windows still No. 1 in server operating systems". ZDNet. 26 February 2010. Archived from the original on 1 March 2010.
[35] Hyde, Randall (1996). "Chapter Seventeen: Interrupts, Traps and Exceptions (Part 1)". The Art Of Assembly Language Programming. No Starch Press. Archived from the original on 22 December 2021. Retrieved 22 December 2021. "The concept of an interrupt is something that has expanded in scope over the years. The 80x86 family has only added to the confusion surrounding interrupts by introducing the int (software interrupt) instruction. Indeed, different manufacturers have used terms like exceptions, faults, aborts, traps and interrupts to describe the phenomena this chapter discusses. Unfortunately, there is no clear consensus as to the exact meaning of these terms. Different authors adopt different terms to their own use."
[36] PDP-1 Input-Output Systems Manual (PDF). Digital Equipment Corporation. pp. 19–20. Archived (PDF) from the original on 25 January 2019. Retrieved 16 August 2022.
[37] "Reading: Operating System". Lumen. Archived from the original on 6 January 2019. Retrieved 5 January 2019.