SYMBIAM OPERATING SYSTEM

ITP 51

OPERATING SYSTEM

**CHAPTER 1**

**Introduction and History**

A new chapter in the history of mobile technology began in the late 1990s when leading industry players worked together to develop a single, flexible mobile operating system. This massive project yielded Symbian OS, a ground-breaking platform that shot to prominence in the early 2000s and will go on to become a defining feature of smartphone history. Symbian Ltd. was founded in 1998 by a group of significant players in the mobile industry, including Nokia, Ericsson, Motorola, and Psion. Creating a standardized operating system that could power the upcoming generation of smartphones was the main goal. This cooperative project launched Symbian OS and set the stage for a revolution in technology. With its roots in Psion's EPOC operating system, which was initially created for PDAs, Symbian OS inherited a solid platform for mobile computing. The creation of an advanced and flexible operating system that would surpass the constraints of conventional mobile platforms was made possible by this inheritance.

**Chapter objectives:**

* Discover the background of the development of the Symbian operating system.
* Analyze the market dominance of the Symbian operating system and how its came to an end.

**Symbian Operating System History:**

In the 1990s, a partnership between Scion and phone makers like Ericsson, Motorola, and Nokia gave rise to Symbian.

In the beginning, Symbian had a 67% global market share for smartphones since it had no serious competition due to Palm OS and Windows phones' limited availability. Notwithstanding its widespread use, Symbian proved to be a challenging platform to develop for, primarily due to the intricacy of the OS itself and the native programming languages OPL and Symbian C++. Furthermore, small developers could not afford the cost of IDEs and SDKs. Due to the discouragement of third-party developers, the native app ecosystem was unable to develop at the same pace as the Android Google Play Store and the later Apple App Store. On the other hand, because of their manageable complexity, iOS and Android had a simpler design and welcomed contributions from outside developers. Included features like graphics and multitasking to accommodate demands from customers in the future. Nokia purchased Symbian Ltd. in June 2008. The goal of the Symbian platform was to make it open source and royalty-free through the use of the Symbian OS and related user interfaces. In February 2010, the open-source code for the Symbian platform was released.

Finnish businessman Fredrik Idestam (1838–1916) was a co-founder of Nokia, which went on to become one of the top technology and telecommunications firms in the world. Idestam, who was born in Orimattila, Finland, on June 28, 1838, was a key figure in the early development of Nokia and helped lay the groundwork for its success.

Nokia then took the lead in contributing to the Symbian code because it had resources for both the user interface and the OS source code. The Symbian Foundation became a licensing-only organization in November of 2010. It was declared by Nokia that it would assume responsibility for the Symbian platform. In the third and fourth quarters of the 2010 fiscal year, Symbian's market share decreased from 39% to 31% as it was overtaken by iOS and Android. On February 11, 2011, Nokia and Microsoft announced that they would be switching from the Symbian to the Windows platform for smartphones. As a result, Symbian's market share decreased even further.

Nokia and Accenture reached an agreement for an outsourcing program on June 22, 2011. Accenture committed to offering Nokia software development and support services based on Symbian operating systems until 2016. Beginning on January 1, 2014, Nokia stopped providing software development and maintenance support for Symbian.

**The market dominance of the Symbian operating system:**

Nokia became a major proponent of Symbian OS, incorporating it into its range of smartphones. Symbian-powered devices saw a meteoric rise in the early 2000s, with Nokia dominating the market. Classic phones such as the Nokia 7650 and Nokia N95 demonstrated the potential of Symbian OS, solidifying its leadership in the rapidly changing field of mobile operating systems.

The 1982 release of the Mobira Senator marked the debut of Nokia's first mobile phone. Nokia made a significant technological advancement with the Mobira Senator, which marked the company's debut in the mobile phone market. Although the Mobira Senator was Nokia's initial mobile phone, the company's success in the consumer mobile phone market really began in 1992 when the company released handheld devices such as the Nokia 1011, the first GSM mobile phone that was commercially available. Nokia went on to become the industry leader in mobile phones after this, a position it held for a long time.

The year 1988 marked a significant milestone for Nokia with the introduction of the highly anticipated Nokia 6100 series, a product line that would redefine mobile phone technology and catapult the company into the global spotlight. The Nokia 6100 series was a testament to Nokia's innovation and commitment to delivering cutting-edge mobile experiences. Outstanding sales numbers helped Nokia overtake rivals including Motorola and establish the company as the world's top maker of mobile phones in 1998. Nokia became a world leader in mobile phone technology thanks to its dedication to innovation and the popularity of the 6100 series.

In the context of mobile phones, Nokia is indeed the **king of the early digital era.** Nokia was a dominant force in the mobile phone industry and helped shape the digital landscape during the late 20th and early 21st centuries. The brand's influence can still be seen today, and Nokia is still remembered as a significant player in the mobile technology industry. Even though Nokia had difficulties in the smartphone era, its influence in the early digital era cemented its place as the King of that revolutionary time.

The rise and fall of the Symbian operating system is a story of industry dominance, technological challenges, and strategic decisions that shaped the trajectory of mobile technology. Symbian, which began in the 1980s and was formalized in 1998, quickly rose to the top of the mobile OS landscape, backed by major manufacturers such as Ericsson and Motorola, with Nokia emerging as its most prominent supporter.

Nokia and Symbian had a mutually beneficial relationship, dominating the cell phone market in the early 2000s. The operating system reached its pinnacle, remaining the world's best-selling smartphone operating system until the end of 2010. However, change was in the air, fueled by the rapid evolution of technology and shifting consumer preferences.

**Chapter 2**

**Process Management**

**Creative Concept and Teamwork-Based Growth**

In the following half of the 20th century, Symbian OS was designed as a specific mobile platform. Its innovative spirit at the outset established the foundation for a strong framework for process management. A collective pool of expertise was created through cooperative development efforts involving major players in the industry, such as Motorola, Ericsson, and Nokia. The management approach was characterized by an emphasis on flexibility, acknowledging the special requirements of mobile devices. This early focus on joint innovation served as a foundational element for process management in Symbian OS.

**Strategic Alliances and Market Dominance**

The strategic partnerships that Symbian OS forged, especially with Nokia, helped it to become the market leader. The OS and Nokia devices have a symbiotic relationship that necessitated careful process management. A versatile and effective process management system was needed to meet the wide range of computational requirements posed by the diversity of mobile hardware. Symbian OS's extensive adoption can be attributed in large part to its proficiency in coordinating various processes to maximize performance on a variety of devices.

**Mobile Environment Adaptation**

Symbian OS demonstrated an acute awareness of the challenges posed by the mobile environment. Its process management strategies were designed to address the limitations of battery life, processing power, and intermittent connectivity. The operating system efficiently managed background processes, balancing responsiveness and power efficiency. This adaptability to the mobile landscape distinguished Symbian OS at a time when mobile devices were evolving from simple communication tools to sophisticated computing platforms.

**Legacy and Acquired Knowledge**

Beyond its years of operation, Symbian OS has left a lasting legacy in process management. The design and administration of contemporary mobile operating systems are still influenced by the knowledge gained from its inventive and cooperative methodology. As mobile technology develops further, the importance of flexibility, resource optimization, and strategic partnerships does not go out of style.

To sum up, the management of Symbian OS processes involved a dynamic interaction between creativity, teamwork, and flexibility. Being a driving force in the mobile space, Symbian OS established the foundation for contemporary mobile operating systems and left a legacy that emphasizes how crucial effective process management is in the rapidly changing field of mobile computing.

**Features of a Procedure**

**A process possesses the following qualities:**

**Process Id**: An individual number that the operating system assigns

**Process State**: The state of the process could be running, ready, etc.

**CPU registers**: Similar to the Program Counter (when a process is switched in and out of the CPU, CPU registers need to be saved and restored).

**Details about accounts**: CPU usage for process execution, time constraints, execution ID, etc.

**Information about the I/O status**: For instance, open files, devices assigned to the process, etc.

**CPU scheduling details**: Priority, for instance (differing processes may have varying priorities; a shorter process might be given a high priority in the shortest job first scheduling).

**Cycle of Process Life**

The state of the process at any one time is called a process state. It also details the present status of the process.

**A process may exist in any of the following five states at any given time.**

**Start:** This is the stage in which a process is first initiated or created.

**Ready:** The process is awaiting the assignment of a processor. Processes that are ready to run are awaiting a processor assignment from the operating system. This state may be reached by the process either during startup or operation, but in reality, the scheduler may interrupt it to give CPU time to another process.

**Running:** Once more, the process state is set to running and the processor starts executing commands when the OS scheduler assigns the task to a CPU.

**Waiting:** A process goes into the waiting state when it needs to wait for a resource, like user input or a file, to become available.

**Terminated or Exit**: A process is moved to the terminated state and waits to be erased from memory space when it finishes its job or is stopped by the operating system.

**Chapter 3**

**CPU SCHEDULING**

The special problem faced by Symbian OS is striking a balance between real-time demands and general-purpose functionality. Symbian OS is an operating system for mobile phones that integrates features from other operating systems. In order to address these different needs, Symbian OS uses a scheduling strategy that is covered in this section. Real-time operating system (RTOS) implementation is used for Symbian OS because of its real-time requirements. It's called a soft real-time system even though it runs on various phone platforms without specific hardware. Implementing mobile protocol stacks such as GSM and 3G requires this soft real-time nature. Time slices are used in conjunction with Symbian OS's static, monotonic scheduling strategy. The processes are arranged according to the shortest deadline first in the static, monotonic strategy. By introducing time slices, priority scheduling can be used to assign time slices to processes that have the same deadline or no deadline at all. 64 priority levels are supported by the OS, providing for flexible process management.

Soft real-time performance depends on a predictable execution time. By taking into account a number of system parameters, such as interrupt handling latency times, process information retrieval times, and thread movement times between queues and the CPU, Symbian OS is able to achieve predictability. For effective process selection, the operating system makes use of 64 distinct queues, each of which represents a priority level, and a 64-bit mask. Symbian OS uses a design with 64 distinct queues and a matching mask to forecast how long it will take to find the highest-priority thread. Because of this design, the operating system can detect queues containing running processes more quickly, which improves scheduling process predictability. The complexities of CPU scheduling, with a focus on the needs of sharing a computer's CPU between processes. It started with defining the word ‘sharing’ and providing standards for evaluating how successful sharing tactics are. The goal of this Cpu scheduling was to give the impression that a single processor could support several processes running at once. It covered the methods and difficulties of effectively allocating CPU time.

**CHAPTER 4**

**MEMORY MANAGEMENT**

Memory Management in the Symbian Operating System the Symbian operating system was once a prominent player in the mobile phone industry, particularly during the early 2000s. Known for its efficiency and resource optimization, Symbian employed a unique approach to memory management that allowed it to deliver a smooth and responsive user experience on devices with limited hardware capabilities. In this essay, we will explore the memory management techniques employed by the Symbian operating system and their significance in ensuring optimal performance. Symbian OS was designed to run on a variety of mobile devices, ranging from basic feature phones to more advanced smartphones. As such, it had to make the most efficient use of the limited memory resources available. The operating system employed a combination of virtual memory, dynamic memory allocation, and a sophisticated memory management system to achieve this goal. One of the key components of Symbian's memory management was the concept of "chunks." Chunks were fixed-size blocks of memory that could be allocated and deallocated by applications. These chunks provided a means for applications to request and manage memory resources dynamically. The use of fixed-size chunks allowed for efficient memory allocation and reduced fragmentation. To facilitate the management of chunks, Symbian introduced the concept of "handles." Handles were lightweight references to chunks that were used by applications to access and manipulate memory. Handles allowed for efficient memory access and minimized the overhead associated with memory management. Symbian also employed a demand-paged virtual memory system. This system allowed the operating system to swap memory pages between physical RAM and secondary storage, such as the device's internal flash memory. By swapping out less frequently used memory pages, Symbian could effectively increase the available memory for running applications, thereby allowing devices to execute more complex tasks without running out of memory. The Symbian operating system utilized a priority-based approach to memory management. Each application and system component was assigned a priority level that determined its access to memory resources. Higher priority tasks were given preferential treatment in terms of memory allocation, ensuring that critical system components and foreground applications had sufficient resources to operate smoothly. Furthermore, Symbian employed a garbage collection mechanism to reclaim memory that was no longer in use. The garbage collector periodically scanned the system, identifying and releasing memory that was no longer referenced by any active process. This approach helped to prevent memory leaks and ensured that memory was efficiently utilized. Overall, the memory management techniques employed by the Symbian operating system played a crucial role in its success. Through the use of fixed-size chunks, virtual memory, priority-based allocation, and garbage collection, Symbian was able to maximize the utilization of limited hardware resources. This resulted in a responsive and efficient user experience on a wide range of devices. However, it is worth noting that the Symbian operating system has gradually lost its prominence in the mobile phone industry, primarily due to the rise of more advanced operating systems such as iOS and Android. These newer platforms offered enhanced capabilities and a more extensive ecosystem of applications, eventually leading to the decline of Symbian. In conclusion, the memory management techniques employed by the Symbian operating system were instrumental in its ability to deliver a smooth and efficient user experience on devices with limited hardware resources. The use of fixed-size chunks, virtual memory, and priority-based allocation allowed Symbian to optimize memory utilization and provide a responsive platform for running applications. While Symbian's dominance has waned over time, its memory management techniques remain a significant contribution to the field of operating system design.

**CHAPTER 5**

**STORAGE MANAGEMENT**

Storage is the third essential pillar in the complex ecosystem of computer systems, after central computer operation and device I/O. The foundation of modern computing is the capacity to store and retrieve data; without storage, a system would not be able to execute programs because modern systems are essentially dependent on stored programs.

The 'fetch-execute' cycle, which is the core computing cycle, emphasizes how heavily storage is dependent upon it. This loop, which was a crucial component of the design envisioned by the visionary John von Neumann, coordinates the retrieval of instructions from memory (storage), their arrangement in registers (additional storage), the execution of instructions—which may require retrieving additional information—and the eventual saving of the execution results in memory (even more storage). This architecture has become the structural foundation of computing systems, with roots in sequential computer memory and external storage devices.

Computers without disk storage do exist, of course, but even these devices have memory for storing data, demonstrating that all computer systems have storage—if only in the form of memory or registers that the CPU can access. These basic elements are where most systems start when creating their storage hierarchy.

Storage Hierarchy

* Registers -The fastest and priciest memory that a computer system can have is represented by those at the top of the hierarchy. Registers are crucial temporary storage for the CPU, even though they are volatile and mainly used for hardware purposes.
* Storage Caches-which are more costly but provide faster storage, are placed in a buffer between main memory and rapid register storage. Caches, which can be categorized into tiers such as 'L1' and 'L2', offer a dynamic resolution that eliminates hardware waiting for main memory operations.
* Main Memory-storing program code while it is being executed by the CPU and acting as the hub for general-purpose temporary storage. Located outside of the CPU and occasionally available to users for modification, it serves a crucial function in the temporary storing of data during program execution.
* Secondary Storage-holds enormous amounts of data permanently in a slower extension of the main memory. Using the polarity of magnetic fields to indicate 1s and 0s, magnetic disks a popular type of secondary storage store bits. Secure digital cards and compact flash cards are examples of electronic disks that are quicker options.
* Tertiary Storage-or archival storage, is intended for long-term preservation and infrequent access. Examples are optical storage like CD-ROMs and magnetic tape, which are placed between secondary and tertiary storage because of their comparatively fast access times.

This storage hierarchy incorporates a number of fundamental ideas that affect how operating systems communicate with different types of storage. From the earliest days of permanent storage, the access model has remained constant, treating data as files with specific purposes. To handle the abundance of files, organizational structures such as directories and folders have been created, resulting in what is called a file system. In order to implement secure storage, the idea of access rights is essential. Certain systems link access permissions to particular users or processes, while others demand identification before granting storage access. Usually, the owner of a storage unit establishes access rules, which calls for a way to identify users or processes.

The idea of caching has developed as a way to protect devices from slower storage as storage access speeds drop down the hierarchy. In order to maintain pertinent data while effectively managing data flow between various storage levels, cache management becomes essential. The concept of virtual storage surpasses the tangible constraints of every tier within the structure. Larger or more feature-rich virtual storage is implemented as an extension on lower layers of the system. When necessary, main memory extends into disk space to serve as virtual cache storage. Efficient transitions between storage layers are made possible by the careful organization required by the dynamic interplay of virtual storage.

**CHAPTER 6**

**I/O SYSTEM**

An operating system's control over input and output is just as vital as its supervision over other computer resources. Managing input and output, however, presents special difficulties because different devices have different functions and speeds. A variable number of I/O devices may be connected or disconnected, in contrast to the CPU of a computer, which is always the same. Consider devices that differ in terms of speed and functionality, such as a printer, a CD-ROM drive, and a mouse. Notwithstanding these variations, they are all dependent on coordinated administration by a single operating system.

One of the most important components of Symbian OS that makes it easier for user code to interact with system-protected resources and enable hardware communication is the input and output (I/O) system. Device drivers function as kernel-privileged code in this operating system and act as the hardware components' software interface. Device drivers are handled by Symbian OS using a two-tiered architecture that consists of Logical Device Drivers (LDD) and Physical Device Drivers (PDD). The LDD ensures uniformity among a particular class of devices by providing a standardized interface to the higher layers of software. Conversely, the PDD differs for every individual device and engages directly with the hardware. The default PDD for common or standard hardware may be provided by Symbian OS. Using a serial device as an example, the user-side API for accessing serial devices is defined by the generic serial LDD (ECOMM.LDD) included in Symbian OS. The PDD, which oversees buffering, flow control, and device-specific operations, interfaces with the LDD. A single LDD can connect to multiple PDDs thanks to this modular approach, which facilitates flexibility and simplicity of use.

In Symbian OS, kernel extensions are unique because they act as device drivers that are loaded during the boot process. These extensions support platform-specific device drivers without requiring kernel recompilation and are essential for features like DMA services, LCD management, and bus control. They also comply with the microkernel design philosophy. An example of the use of abstraction is found in Symbian OS's Hardware Abstraction Layer (HAL). It is made up of variables and functions that retrieve system properties and configuration, offering a standardized API for hardware device management. The HAL is composed of multiple groups that handle hardware devices as well as platform parameters in general. An essential component of Symbian OS device drivers is Direct-Memory Access (DMA), which facilitates effective data transfer between memory and devices. The DMA service uses a layered architecture that consists of a kernel extension that interfaces with DMA hardware, a software DMA layer, and platform-independent and platform-dependent layers. Under Symbian OS, media drivers—a type of PDD—manage storage media devices only for the file server. The file server supports removable and fixed media, and with its standard LDD and interface API, it can handle up to 26 drives at once. The Symbian OS presents active objects, which are lightweight threads managed by a single scheduler, as a solution to blocking I/O. The operating system can manage blocking I/O calls more effectively thanks to these objects, which improves system responsiveness. Socket objects, device drivers, and software controllers are used in Symbian OS to implement removable media. Device drivers, set up as active objects, react to removable media events, which cause state changes, allowing media card insertions and removals to be handled seamlessly.

**CHAPTER 7**

**FILE SYSTEM**

To bridge the divide between unorganized storage and systematic organization, file systems provide a structured hierarchy of files and directories. This chapter delves into the foundational aspects of file systems, exploring their implementation across various scenarios and their relevance in the realm of mobile devices, using Symbian OS as a case study. Additionally, we examine the security risks associated with the implementation of file systems.

Breaking down the components of file systems, we begin by studying their basic building blocks: names, attributes, and operations. These elements play a crucial role in creating and managing files and directories, offering insights into the transformation of raw bits and bytes into an orderly structure. Moving beyond theoretical concepts, the chapter explores the practical execution of file systems. By providing examples in various formats, including those from popular file systems in mainstream operating systems, we demystify the process of translating abstract ideas into tangible structures. Real-world examples illustrate the adaptability and versatility of file system designs. In the context of mobile devices, file systems face new challenges and considerations. This section presents a Symbian OS viewpoint on file systems for portable electronic devices, offering a case study on how file systems can be tailored to the specific needs of mobile platforms. Examining the evolution of file systems within the language of mobile devices provides a clearer understanding of their responsiveness to changing technological paradigms, addressing aspects from storage management to user interactions.

Addressing the essential aspect of security, the chapter explores potential flaws and challenges in file system implementation. A comprehensive examination of file system security, encompassing data integrity and access control, emphasizes the importance of robust security measures in safeguarding user data and system resources.

Mobile phone operating systems need to fulfill similar file system criteria as desktop operating systems. The majority operate within 32-bit environments, provide users with the capability to rename files, and manage a substantial volume of files that require organization. This underscores the preference for a file system structured around hierarchical directories. While designers of mobile operating systems have various file system options, an additional factor influences their choices: most mobile phones come equipped with storage media that can be shared with Microsoft Windows environments.

If mobile phone systems lacked removable media, any file system could be employed. However, in systems utilizing flash memory, unique considerations come into play. Typically, block sizes range from 512 bytes to 2,048 bytes, and flash memory follows a specific process: it must erase before writing. Moreover, the unit of erasure is relatively coarse, requiring entire blocks to be erased at once, as opposed to individual bytes. The erase times for flash memory are relatively extended.

To adapt to these characteristics, optimal performance is achieved with specially designed file systems for flash memory. These file systems distribute writes across the media and address the prolonged erase times. The fundamental concept involves writing a new copy of changed data to a fresh block, remapping file pointers, and later erasing the old block when feasible.

One of the earliest flash file systems, Microsoft's FFS2, emerged for use with MS-DOS in the early 1990s. The approval of the Flash Translation Layer specification by the PCMCIA industry group in 1994 allowed flash devices to mimic a FAT file system. Linux also boasts tailored file systems, such as the Journaling Flash File System (JFFS) and the Yet another Flash Filing System (YAFFS).However, as mobile phone platforms share their media with other computers, ensuring some level of compatibility is essential. Frequently, FAT file systems are chosen, with FAT16 often preferred for its shorter allocation table (compared to FAT32) and reduced need for long files. Given its status as a mobile smartphone operating system, Symbian OS necessitates implementing at least the FAT16 file system. While providing support for FAT16 and employing it for most storage media, the Symbian OS file-server implementation is structured on an abstraction akin to UNIX’s VFS. Object orientation facilitates the integration of objects implementing various operating systems into the Symbian OS file server, allowing for the utilization of diverse file-system implementations, even coexisting within the same file server.

The inception of the FAT file system can be traced back to the early days of personal computing when Microsoft developed the first operating system for IBM hardware. In 1977, FAT made its debut on IBM PCs using the Microsoft Disk Basic system. The FAT file system, named after its utilization of a file-allocation table, has evolved over the years, persisting in modern versions of Microsoft Windows and finding application in mobile media storage.The evolution of the FAT file system encompasses various versions, starting with FAT12. The initial iteration was simplistic and restrictive, lacking support for hierarchical directories and limiting disk size to 32MB. With subsequent releases like MS-DOS 2.0 in 1983, FAT12 embraced hierarchical directories, enabling more efficient storage. The introduction of FAT16 in 1988 marked a significant advancement, expanding disk address sizes to 16 bits and allowing partitions up to 2 GB. However, both FAT12 and FAT16 were constrained by the 8.3 filename limitation.

The advent of VFAT, associated with Microsoft Windows 95, addressed the filename limitation by introducing support for longer filenames. This variant of FAT16 became instrumental in accommodating modern naming conventions. The FAT32 file system, introduced in 1996, marked a milestone with its 32-bit disk addressing and cluster sizes. Although theoretically supporting media sizes up to 2 TB, practical limitations in Microsoft utilities restricted it to 124 GB. FAT32 retained compatibility with Microsoft Windows and implemented long filenames from VFAT.Each variant of the FAT file system shares common characteristics, employing a file-allocation table with linked clusters to form file pieces. The generic format includes a boot sector, optional reserved sectors, duplicate file-allocation tables, a root file directory, and the remaining storage medium for file block storage.

In the context of mobile phone operating systems like Symbian OS, which often share storage media with Microsoft Windows environments, the implementation of FAT file systems becomes imperative. Most notably, FAT16 is widely utilized due to its shorter allocation table and reduced demand for long files.

Symbian OS, being a mobile smartphone operating system, integrates support for FAT16 and employs it for a significant portion of its storage media. However, the file-server implementation in Symbian OS exhibits an abstraction akin to Unix's VFS (Virtual File System), allowing flexibility for various file-system implementations. This design enables the coexistence of different file-system implementations within the same file server, showcasing the adaptability and versatility of the Symbian OS architecture.

**References**

[**https://www.scaler.com/topics/process-management-in-operating-system/**](https://www.scaler.com/topics/process-management-in-operating-system/)

[**https://www.25yearsofprogramming.com/guide/fat-and-vfat-file-systems.html**](https://www.25yearsofprogramming.com/guide/fat-and-vfat-file-systems.html)

[**https://www.25yearsofprogramming.com/guide/file-systems-on-mobile-phones.html**](https://www.25yearsofprogramming.com/guide/file-systems-on-mobile-phones.html)

[**https://www.bcg.com/publications/2015/telecommunications-technology-industries-the-mobile-revolution**](https://www.bcg.com/publications/2015/telecommunications-technology-industries-the-mobile-revolution)

[**https://historydraft.com/story/symbian-os/timeline/557**](https://historydraft.com/story/symbian-os/timeline/557)

[**https://symbianfoundation.org/all-about-the-symbian-os/**](https://symbianfoundation.org/all-about-the-symbian-os/)

[**https://symbianfoundation.org/all-about-the-symbian-os/**](https://symbianfoundation.org/all-about-the-symbian-os/)

[**https://en.wikipedia.org/wiki/Linux\_kernel**](https://en.wikipedia.org/wiki/Linux_kernel)

[**https://pubsonline.informs.org/doi/abs/10.1287/opre.26.1.127?journalCode=opre**](https://pubsonline.informs.org/doi/abs/10.1287/opre.26.1.127?journalCode=opre)

[**https://1library.net/article/scheduling-in-symbian-os-real-time-strategy.yjv80l5y**](https://1library.net/article/scheduling-in-symbian-os-real-time-strategy.yjv80l5y)

[**https://www.semanticscholar.org/paper/Evolving-an-Open-Ecosystem%3A-The-Rise-and-Fall-of-West-Wood/80f83dbee5a5c99d5ff2f93668ec6f6785930a9e**](https://www.semanticscholar.org/paper/Evolving-an-Open-Ecosystem%3A-The-Rise-and-Fall-of-West-Wood/80f83dbee5a5c99d5ff2f93668ec6f6785930a9e)

[**https://www.tandfonline.com/doi/full/10.1080/00076791.2019.1593964**](https://www.tandfonline.com/doi/full/10.1080/00076791.2019.1593964)

**https://en.wikipedia.org/wiki/Fredrik\_Idestam**