Mobile phones have become part and parcel of our life. Today’s mobile phones not only offer traditional voice services but also other applications like SMS, MMS, internet, video player, radio etc. All these advancements is made possible by using Symbian OS in mobiles. Just like PCs have an operating system like Windows, Symbian is the O.S for mobile phones. But unlike PC design, mobile phone put constrains on a suitable O.S. The operating system has to have a low memory footprint, and low dynamic memory usage, and efficient power management framework and real time support for communication and telephony protocols. Symbian OS is designed for the mobile phone environment. It addresses constraints of mobile phones by providing a framework to handle low memory situations, a power management, and a rich software layer implementing industry standards for communication, telephony and data rendering.

Symbian OS development was the result of extensive research carried out by world’s leading mobile phone companies like Nokia, Motorola, Sony Ericsson. The first open Symbian OS phones are the Sony ericsson’s P800 smartphones, the Nokia 9200 series and the Nokia 7650. Today two versions, Symbian OS v7 and v6 has evolved and extensively used. With improved picture resolution and flexible user interfaces, today’s mobile phones are capable of replacing PDAs and even the Palmtops.
INTRODUCTION

Just like PCs have an operating system like Windows, Symbian is the O.S for mobile phones. But unlike PC design, mobile phone put constrains on a suitable O.S. The operating system has to have a low memory footprint, and low dynamic memory usage, and efficient power management framework and real-time support for communication and telephony protocols. Symbian OS is designed for the mobile phone environment. It addresses constraints of mobile phones by providing a framework to handle low memory situations, a power management, and a rich software layer implementing industry standards for communication, telephony and data rendering.

Symbian OS is designed for the specific requirements of open, advanced, data enabled 2G, 3G mobile phones. Compact enough to fit in the memory of a mobile phone, Symbian OS was planned from the beginning to be a full operating system in terms of functionality. Symbian OS is already available in Ericsson R380 smart phones, Nokia 9200 communicator series, the Nokia 7650 and Sony Ericsson P800.

Key features of Symbian OS are:

- **Rich suit of application engines** – Including contacts, schedules, messaging, browsing, office, utility and system control
- **Browsing** – fit for the purpose of browsing engine, for full web browser support and WAP stack for mobile browsing
- **Messaging** – multimedia messaging using MMS, picture messaging with EMS and text messaging using SMS
- **Multimedia** – shared access to screen, keyboards, phones and bitmaps; audio recording and play back, and image related to functionality (support for all common audio and image format)
SYMBIAN

- **Communication protocol** – wide area networking stacks including TCP, IP version 4, IP version 6 and personal area networking stacks including Blue tooth, and USB.

- **Software development** – 3 main programming and content development options: C++, Java, WAP.
Symbian OS development was a joint venture of world’s leading mobile phone companies like Nokia, Ericsson, and Motorola. After the introduction of first generation mobile phones, these companies realized that adding new functionalities to their handsets was a key market sector. Thus back in 1998, they jointly developed an operating system suited for mobile phone environment.

The first open Symbian OS phones are the Sony Ericsson’s P800 smartphones, the Nokia 9200 series and the Nokia 7650. Today two versions, Symbian OS v7 and v6 has evolved and extensively used. With improved picture resolution and flexible user interfaces, today’s mobile phones are capable of replacing PDAs and even the Palmtops.
Mobile phone hardware can be divided into three logic layers:

1. The CPU core.
2. The system on chip (SoC).
3. The phone PCB.

Figure 1: Mobile phone hardware can be divided into three logical layers: The CPU core, the SoC and the PCB. Symbian OS also conforms to this layering. This enables easy porting of Symbian OS as the code for particular CPU core or SoC can be reused in many products.
**CPU CORE**

Central to data enabled mobile phones is a fast, low power, low cost CPU core, which has a compact code and can be highly integrated with peripherals. The family of ARM architecture RISC processors are usually used. The CPU core has a memory management unit (MMU) for efficient usage of valuable memory space, and a cache for speedy access of data. The CPU core must be capable of operating in various privileged access modes to handle interrupts and exceptions. It is here, all the logic calculations and decisions are being carried out.

**SYSTEM ON CHIP (SoC)**

The CPU core is placed in an SoC. SoC also contains timers, interrupt controllers, DMA controllers and other vital peripherals for the effective functioning of phone operating system. These SoCs are often commercially available and are sometimes custom build by handset manufacturers. Some of the SoCs developed by Intel are Intel strong ARM processor series and Intel X-scale series.

**PHONE PCB**

Phone PCB is the motherboard of mobile. The system on chip (SoC) is placed in the phone PCB. The I/O systems like LCD, keypad, audio codec, radio, etc. are connected to this PCB. It also contains some memory.
MEMORY MANAGEMENT UNIT (MMU)

MMU is a hardware unit which performs the task of converting linear addresses, as they appear as outputs from a program, into physical addresses that access a physical memory location placed anywhere within the memory system. MMU has a memory paging mechanism, which allows much physical memory location to be assigned to any linear address. The linear address is defined as the address generated by a program. With a memory paging unit, the linear address is invisibly translated into any physical address, which allows an application written to function at a specific address to be relocated through paging mechanism.

MMU also protects process data from access by other processes, enforces protection of application and kernel codes and isolates the hardware from the application codes. The MMU is a crucial component in the design of the protected mode system, which enhances both the security and stability of the platform. A standard two-level page tabled MMU allows small 4 KB pages for efficient memory usage.

CACHE

A temporary and a fast storage system are called cache memory. If CPU needs information, it will first check whether it is in cache. If it is, CPU takes the information directly from the cache; if it is not, we take the information from the main storage system, placing a copy in the cache under the assumption that there is a high probability that it will be needed again. In a microprocessor, internal programmable registers such as index registers provide a high-speed cache for the main memory. The main memory can be viewed, as a fast cache for secondary memory, since data on secondary storage must be copied into main memory for use and data must be in main memory before being moved to secondary storage for safe keeping.
The data transfer from cache to CPU and registers is usually a hardware function, with no hardware intervention. On the other hand, transfer of data from secondary memory to primary memory is controlled by the operating system. Caches can hold both data and instructions. For acceptable performance of Symbian OS, we must use both data and instruction cache.

**DMA**

A form of I/O is called DMA (Direct Memory Access). The DMA I/O technique provides direct access to memory while the microprocessor is temporarily disabled. This allows data to be transferred between memory and the I/O device at a rate that is limited only by the speed of the memory components in the system or the DMA controller. A programmable DMA controller is 8237-A.

**TIMERS**

On-Chip timers provide the real time Clock for the system tick time, and millisecond scale timers are needed for the use with hardware drivers.

**MEMORY**

Memory is for safe keeping all the data and instructions. In mobile phones, while some memory will be available on the chip most will be provided off chip. This off chip memory has three major functions: storage of symbian image; persistence of user data in a file system; and storage of processes data at runtime. Speed of memory access, cost of memory chips and persistence of the data must be considered when choosing the memory for each of these three functions.
SYMBIAN

One option is for the operating system image to be located in ROM. ROM is cheap but cannot be reprogrammed. Any change in software at a later stage would have to be performed by reorganizing the MMU. Flash is more versatile than ROM, allowing the image to be placed in RAM and marked as read-only by the MMU, however this doubles the amount of memory needed and increases the time to boot, as a boot loader has to copy the image from ROM to RAM at boot time. Flash is the most commonly adopted solution for mobile phones.
SOFTWARE

OPERATING SYSTEM

An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs. The two main goal of an operating system is to make the computer system convenient to use, and secondary to use the hardware in an efficient manner.

APPLICATION PROGRAM

The programs which are executed using computer hardware like the CPU, memory, IO devices etc are called the application programs. For example the compilers, databased systems, games, business programs etc.

KERNEL

Kernel is the most vital part of an operating system. When a computer is switched ON, only the kernel part of OS is transferred to the memory from the disc. The program running at all times on the computer, all else being application program is what we call kernel.

LIBRARIES

Library files are collection of procedures that can be used by many different programs. Libraries allow common procedures to be collected into one place so that they can be used by many different applications.
SYMBIAN OS IMAGE

Symbian OS image is a compact collection of executable code and various data files. The image consist of dynamically linked libraries (DLLs) and other required data, including configuration files, bitmaps, fonts and other file resident resources.

The image code is execute-in-place. However speed is critical, libraries in the image can be marked to be executed from RAM and hence will be copied there before being used. Also, by having almost all the code as DLLs, there is only a single copy of each library required regardless of the number of applications linked to it.

HARDWARE INTERFACES

The Symbian OS kernel is a compact pre-emptive multitasking operating system with very little dependence on peripherals. The core kernel executable of less than 200 KB fully supports the multi-threaded operating system. Peripheral hardware integration is added to the kernel in several ways. Hardware support is usually implemented in separate DLLs associated with particular hardware to allow the easy insertion and removal of hardware and to facilitate code reuse.

The MMU is configured so that all hardware registers can only be accessed in privileged mode. The kernel always executes in privileged mode and hence has access to all the hardware registers. Applications interface to kernel services through an API provided by the User library. Because all applications run in unprivileged mode, operations that require hardware access must either switch momentarily into privileged mode while maintaining the context of the application or issue a request to the kernel server, which involves a switch in context to that of the kernel process.
The kernel library includes support for all peripheral hardware that is resident on the chip (e.g., the ASIC or SoC) and that is essential to the operating system. The peripheral hardware includes such things as timers, DMA engines, interrupt controllers and UART serial ports. The kernel library is customized for a particular chip. Applications are not permitted to access peripheral hardware directly. Instead, applications must link to the User library whose functions may invoke peripheral control through the kernel.

Peripherals associated with user input can be packaged as a separate DLL, called a kernel extension. User input simply provides events that are consumed by the kernel. Different kernel extensions can be written for keyboard, keypad, digitizer, and navigation button and wheels. The appropriate kernel extensions are added into the image, where the kernel detects their presence at boot time and initializes them. The kernel itself has no dependency on the extensions, and no kernel extension functionality is accessible to applications.

![Diagram](image)

**Figure 2:** All access to hardware occurs from, or through the kernel. There are several software frameworks that support hardware access, direct kernel access for vital hardware, kernel extensions for hardware associated with user input, and device drivers for further peripheral hardware exposed to applications and server.
Device drivers expose API to applications to allow control of hardware that is not essential to running the operating system. Device drivers can be loaded and unloaded at any time. A device driver consists of two parts: a library providing the device's API to which applications can link; and one or two libraries, running in privileged mode, kernel-side, to access the hardware.

The kernel side library is often split into two libraries: a logical (LDD) and a physical (PDD) device driver DLL.

The LDD encapsulates the logical functions of a device e.g. on and off, and read and write. The PDD carries out the functions on a specific device. The LDD contains all the complexity of typical device usage usually in the form of a state machine. If the choice of hardware part is changed in prototyping a phone or the progression of a product line, only the PDD needs to be replaced. For example, the media server uses a standard application-side API and a logical device driver so only a physical device driver has to be provided for a particular codec chosen for a phone.

Both logical and physical kernel-side device drivers are polymorphic DLLs. The device drivers must be loaded by the application side before access to the hardware can be obtained. The application-side library uses a message passing mechanism to communicate to the hardware libraries through the kernel. The devices can be used synchronously or asynchronously, though asynchronous use is preferred, wherever possible, as it is more CPU efficient.

Support for file system media is also provided through a device driver. A file system driver consists of two components, a file system and a media driver. The file system is typically FAT, but not necessarily if the system is internal to the phone and would benefit from a different format. The media driver is a
physical device driver performing all the functions that the file server expects. By constructing drives with these two components, the file server conforms to a plug-in model where new media drivers and file systems can be plugged into the OS without affecting the core code in the file server. If the need arises for a ROM patch, device drivers are used to implement the patch because they have access to the MMU, which has been protected by the kernel.

The only exception to this model of peripheral access controlled by the kernel is the screen buffer, which is copied via DMA to the LCD display. The screen buffer is usually given read/write permissions to all threads, for applications to provide fast drawing routines through a graphics API. This increases speed as no switch to privileged mode and back is required.

Figure 3: The rich layer of system software for industry standard support is part of Symbian OS.
Symbian OS contains an extensive and rich collection of libraries to implement many industry standards. This layer of system software, in Version 6, includes support for networking (TCP/IP, PPP, TSL, SSL, IPSec, FTP), communications (Bluetooth, IrDA, Obex), Security (DES, RSA, DSA, DH), messaging (POP3, IMAP4, SMTP, SMS, BIO), browsing (HTML, HTTPS, WAP, WML), telephony (GSM, GPRS, fax), graphics, multimedia (WAV, AU, WVE, JPEG, BMP, MBM, GIF) and many more.

Access to these services and resources is coordinated through a standard client-server framework. Servers run as unprivileged threads. Any application thread can be a client connecting to a server by name and passing messages through a standard interface imposed by the kernel. The framework is constructed through inheritance from server and session classes. The kernel support for the client-server framework is optimized for low memory use and speed; it also keeps a record of objects in the system such that any thread death results in all the memory being recovered. The uses of the client-server architecture in the Symbian OS include the file server, media server, telephony server and many more. The media server is a good example of a hardware resource, accessed through a device driver, which can be synchronized through the client-server framework.

Figure 4: Most standards are implemented as system servers. The servers can make use of device drivers to access the hardware required for particular services. For example the multimedia server can be used to play standard audio clips.
The top layer of Symbian OS provides support for applications. This includes application engines for common Symbian OS phone applications: contacts (an address book), agenda (a diary application), and jotter (a document producing application). This layer also includes skeleton support for graphical user interface (GUI) components, however handset manufacturers provide the actual user interface. This allows the phones to maintain the unique look, feel and branding of each manufacturer.
ADVANTAGES

- **Integrated multimode mobile telephony** - Symbian OS integrates the power of computing with mobile telephony, bringing advanced data services to the mass market.

- **Open application environment** - Symbian OS enables mobile phones to be a platform for deployment of applications and services (programs and content) developed in a wide range of languages and content formats.

- **Multi-tasking** - Fully object-oriented and component-based, Symbian OS includes a multi-tasking kernel, middleware for communications, data management and graphics.

- **Low memory usage** – Symbian provides a framework designed for programming when memory is limited. This framework consists of a clean up stack onto which any partially constructed objects are placed until their construction is over. If the phone runs out of memory it is possible to delete the objects on this clean up stack avoiding a memory leak and enabling the process to continue.

- **Robustness** - Symbian OS maintains instant access to user data. It ensures the integrity of data, even in the presence of unreliable communication, and shortage of resources such as memory, storage and power.
SYMBIAN

CONCLUSION

Symbian OS is a robust multi-tasking operating system, designed specifically for real-world wireless environments and the constraints of mobile phones (including limited amount of memory). Symbian OS is natively IP-based, with fully integrated communications and messaging. It supports all the leading industry standards that will be essential for this generation of data-enabled mobile phones. Symbian OS enables a large community of developers. The open platform allows the installation of third party software to further enhance the platform.
REFERENCES

- ELECTRONICS FOR YOU MAGAZINE
- OPERATING SYSTEMS BY SILVERSHWTS
- MATRUBHUMI NEWS LET
- WWW.SYMBIAN.COM