CHAPTER 1

OS-9 - THE OPERATING SYSTEM

1.1 WHAT IS AN OPERATING SYSTEM?

OS-9 is an operating system. The name is an abbreviation of "Operating System for the 6809 microprocessor". OS-9 was originally written by Microware under contract to Motorola, who wanted to demonstrate that the 6809 microprocessor was sufficiently powerful to be the heart of a true computer. OS-9/6809 is therefore owned jointly by Microware and Motorola. OS-9/6809 was very popular in its own right, and was used on a number of different high-volume home and educational computers. Microware later rewrote OS-9 for the 68000 family of microprocessors, and it is in this form that OS-9 has reached its current very wide popularity on the industrial market. But what is an operating system?

Depending on your viewpoint as a user, you might see it as one or more of the following:

- The kernel of software that gets the computer going.
- The user interface and command interpreter.
- A provider of a basic I/O interface for the user.
- A set of functions for applications programmers.
- A core of standards and documentation for applications programmers.
- A set of philosophies for programmers.

An operating system will (usually) provide all of these things. But these are features as seen from the point of view of a user. To appreciate what an

operating system does we must look at it from a system designer's viewpoint. A computer consists of electronic hardware - central processing unit (CPU - the processor), memory, disk drives, displays, keyboards, printers, and so on. The user wants to run **applications programs** to carry out specific tasks, such as a word processor, spreadsheet, or language compiler. Somewhere there must be software that controls and manages these "hardware resources". This software *could* be in the application program. However, that has very important disadvantages:

- The software must be present in every program.
- Programs are not "portable" to computers with different hardware.
- The computer cannot run multiple programs concurrently, as they will clash in their use of hardware resources.

An alternative approach is to use an operating system. The operating system is a body of software that provides functions to allocate and manage the hardware resources. It divides up the processor time between multiple programs running concurrently, allocates memory to programs as they need it, and arbitrates between programs trying to use the same input/output (I/O) device.

In addition, the operating system provides software to implement commonly required functions, such as disk file management. This provides standards for programmers, and significantly reduces the work required to write applications programs.

The operating system is not a program in itself. It provides functions for use by programs. But the operating system vendor will usually also provide a set of commonly required programs for general system maintenance, such as file copying, deleting, and display. These programs are known as "utilities".

1.2 WHY IS AN OPERATING SYSTEM IMPORTANT?

Many companies have in the past elected to write their own operating systems, in the belief that this gives them greater control over the functionality of the operating system, and a fuller understanding of all the functions.

In reality, however, real time kernels, executives and operating systems written in house "re-invent the wheel" (creating techniques and software already available for purchase), take a great deal of support and documentation, and are vulnerable to the departure of key personnel.

The learning curve for this type of project is usually very long, unless programmers with particular "systems programming" experience are hired. An operating system requires very different programming techniques and philosophies from those of applications programming.

There are other advantages to using a bought-in, widely used operating system. A known, documented, and fixed environment simplifies large projects, and new generations of applications. Additional functionality to simplify the applications programming is likely to be available, because a generally accepted operating system available on a wide range of hardware promotes the development of third-party software products and documentation. It also gives long-term confidence to manufacturers and users.

1.3 WHAT IS MULTI-TASKING?

A multi-tasking operating system is one which provides for the concurrent running of multiple programs. As the processor of a computer can only run one program at a time, multi-tasking is achieved by running one program for a short time (perhaps 20 milliseconds), stopping it (saving its state so it can be restarted later), starting the next program, and so on. Provided the time interval - known as a "time slice" - is short enough (whether it is will depend on the application), this gives the illusion of running the programs concurrently.

There are two (very different) uses of a computer for which multi-tasking is very important. The first is a multi-user system. This is a computer system which has multiple users working on it (on different terminals!) at the same time. Each user is running his own program independently of the others - the application programs do not interact with each other.

The second use is a multi-tasking application. Here multiple programs work together to implement a single application. This is how most computer-based industrial and domestic products work. For example, one program handles the operator interface, while a second program collects data from sensors, and a third program communicates with a central factory computer. This multi-tasking approach is very important. It allows each program to concentrate on a single job, without having to "poll" (check) frequently to determine whether other jobs need doing - they will be done concurrently by the other programs.

In a multi-tasking application the programs must work together to perform the overall job. This requires the passing of data between the programs. It

also requires synchronization between the programs. For example, one program must not continue its job until another program has collected data for it. These functions of synchronization and data transfer are known as inter-process communication. Correct use of these functions is essential to the writing and working of a multi-tasking application.

1.4 PROGRAMS IN ROM

Computer-based products vary widely in their complexity and cost. A product may have a powerful processor, with disk drives, high-quality displays, and a networking connection to other computers. Or, it may be a low cost, "embedded" product, with no operator interface or disk drives.

If the system has no disk drives (or other suitable storage medium), the operating system and all of the programs must reside in ROM. Therefore the capability for the operating system to be ROMmed (placed in ROM), and to support programs in ROM, is essential to low-end applications.

Traditionally this group of applications has been serviced by "real time kernels" (not to be confused with the OS-9 "kernel" module!) or "executives". These differ from an operating system in that the programs and real time kernel are all linked together to make one software package, which is then typically ROMmed. In general, new programs are not dynamically loaded and run. Also, traditionally a real time kernel does not automatically share out the time of the processor between the multiple programs of the application. Instead, programs are assigned a priority, and the highest priority program runs until it asks to be suspended.

This approach produces a final software package that is (in general) somewhat smaller, and (because the real time kernel offers less functionality) perhaps a little faster. However, development and debugging are more difficult, and more complex applications cannot easily be implemented. So a ROMmable operating system is attractive, allowing one programming environment to be used for a very wide range of applications.

1.5 THE FUNCTIONS OF AN OPERATING SYSTEM

In the light of all the above "desirable features", we might therefore hope that an operating system would provide the following functionality (the terms used are explained in later chapters):

☐ **Management of system resources**

- Memory
- processor time
- I/O devices
- Co-processors
- External events
- Inter-process communication mechanisms
- Inter-user protections

☐ **Provision of commonly required functionality**

- Disk file handling
- Input line editing
- Program loading
- Date and time
- Terminal functions
- Process debugging
- I/O device control
- Multi-user management
- Utilities

☐ **Application program portability (hardware independence)**

- Complete set of system functions (so the programmer never needs to access hardware directly)
- Standardized error numbering
- Device-independent I/O functions

☐ **Ease of system customization**

- Modular operating system structure, especially I/O
- ROMmable operating system and programs
- Isolation of areas requiring customization
- Ease of new system build
- Independence of filing systems from hardware functions

☐ **Standardization of software and documentation**

- Well-defined operating system functions
- Modular approach to system customization and extension

1.6 A COMPARISON OF OPERATING SYSTEMS

Having drawn up a long wish-list of functions for a hypothetical operating system, it is interesting to compare different well-known operating systems for their functionality - see figure 1 on the next page.

1.7 THE MAIN PROPERTIES OF OS-9

The list of features in figure 1 makes OS-9 appear very attractive. This is perhaps not so surprising. OS-9 was not written in a hurry. It was developed over a period of about three years by a small group of programmers at Microware under contract to Motorola (who therefore jointly own OS-9/6809). The programmers carefully considered existing operating systems (particularly UNIX), building on the good features of existing technology, but inventing new techniques wherever they saw the need.

This makes OS-9 one of the very few operating systems developed through a long-term project as a commercial product outside of a hardware manufacturer. As a result OS-9 implements almost all of the functions that might be wished of an operating system, in an elegant, straightforward way. Indeed, it is a good model for the academic study of a broad-spectrum, multi-tasking, real time operating system. This section summarizes the main properties of OS-9, with a brief description of how each is implemented.

1.7.1 Multi-tasking

Using a hardware timer (whatever is available on the particular system) to generate "tick" interrupts, OS-9 performs automatic time-slicing between any number of programs. In addition, OS-9 has several mechanisms to allow the advanced programmer to modify this "scheduling", even to change it to the non-automatic priority-only scheduling expected in a real time kernel.

1.7.2 Real Time

The term "real time" is often abused or misunderstood. It means that real world events are being processed as they happen. A real time system is one that must respond to an external event within a specified time. For example,

	OS-9	UNIX	MS-DOS
Multi-tasking	Yes	Yes	No
Multi-user	Yes	Yes	No
Real time	Yes	No	No
Modular	Yes	No	No
Broad spectrum applicability	Yes	No	No
Large systems	Yes	Yes	No
Single-user workstations	Yes	Yes	Yes
Personal computers	Yes	No	Yes
Home computers (diskless)	Yes	No	No
Embedded industrial products	Yes	No	No
Processors available	Motorola Intel	Many	Intel
Memory limitation	None	None	640k
Requires Memory Management Unit?	No	Yes	No
ROMmable (and can operate diskless)	Yes	No	No
Device-independent I/O system	Yes	Yes	No
User-customizable I/O device drivers	Yes	No	Yes
User-customizable I/O file management	Yes	No	No
User-customizable kernel	Yes	No	No
Dynamically customizable	Yes	No	No
Robust disk filing structure	Yes	No	No
All versions generally compatible	Yes	No	No
Associated with a hardware manufacturer	No	Yes	Yes
Accepted by major manufacturers	Yes	Yes	Yes
Wide third-party software/documentation	No	Yes	Yes
Next generation standard	Yes	Yes	No

- **Figure 1 - A comparison of operating systems**

an image processing system must process the image of one part before the next part on the conveyor belt comes along. If it cannot, then it has failed its function. Therefore the real time response time will vary from application to application, and between external events in the same application. It could be as much as five minutes, or as little as 500 nanoseconds.

A non-real-time system may also be specified to respond within a certain time, but it is not fatal if it does not. For example, an accounts computer may be specified to respond to an operator input within two seconds, but if it takes three seconds on occasion that is only an annoyance to the operator.

Computers respond to external events by polling or by interrupts. Polling - repeated testing for a condition - is relatively slow and uncertain - a program may take a while to perform a particular task before it is ready to poll again. An interrupt is a hardware signal to the processor. It causes the processor to suspend execution of the current program, and execute a separate function (called an "interrupt handler"). The interrupt handler carries out any tasks that require "immediate" attention, and/or sends a software signal to the program that wants to know about the external event. This would typically wake up the program, which would be in a suspended state waiting for the event.

Response by interrupt is therefore very much more efficient and reliable than response by polling, although it requires somewhat more initial learning and programming. A real time operating system must offer a way of adding new interrupt handlers, and mechanisms for interrupt handlers to communicate with programs. OS-9 has both. Also, interrupts are not very useful without multi-tasking, as the single program could not sleep, waiting for the interrupt - it would still need to poll to see if the interrupt had occurred. Therefore a real time operating system must also be multi-tasking (which OS-9 is).

1.7.3 ROMmable

The unique memory module mechanism of OS-9 (described later) makes the operating system and application programs inherently ROMmable. Also, there is no need to provide any information as to where in ROM the modules are - at startup the OS-9 kernel scans all the ROM areas to find all modules present in ROM, and builds a module directory indicating where they are. So modules can be placed in ROM in any order, and with gaps between them if desired. Once entered in the module directory, a module is located by its name, in a similar way to named files on a disk.

1.7.4 Modular

The operating system itself is separated into several memory modules. This allows very easy customization. If certain functionality (such as disk filing) is not required, that module is simply omitted. If additional functionality is required - even dynamically at run-time - additional modules (such as the Internet Support Package for Ethernet networking) can be loaded, and their functionality is immediately available.

1.7.5 Unified Device Independent I/O System

A device independent input-output (I/O) system is one in which the same basic functions - such as "open", "read", and "write" - are used by a program to access all types of I/O devices. This simplifies programming, and permits "redirection" - a program written to write to a terminal can have its output sent to a disk file without being modified in any way.

An operating system with a unified I/O system is designed to allow the I/O system to be easily customized, extended, or reduced. All I/O calls from programs go to the operating system kernel, the core functionality of the operating system which is present in all configurations. The kernel provides a common environment to manage the call, and passes it on to the appropriate subroutine within the operating system.

OS-9 has a particularly well structured I/O system. I/O calls go to the kernel, which then calls a file manager appropriate to the class of device (disk drive, tape drive, terminal, network, and so on). The file manager has the job of handling the logical data manipulation, such as the file handling on a disk. The file manager does not know how to access the physical device, and calls a device driver specific to that device whenever physical I/O is required (for example, reading and writing of sectors on the disk).

The kernel, file managers, and device drivers are all separate memory modules, so new device drivers can easily be added for new physical devices, and new file managers can be written for new classes of device.

1.7.6 Inter-process Communication Functions

OS-9 has several different inter-process communication functions available for use by application programs. They are discussed in detail in the chapter on "Inter-process Communication", as they are very important to the effective generation of multi-tasking applications.

1.7.7 High Performance

OS-9 was designed with execution speed and code size very much in mind, and so was written in assembly language. This makes it very well suited to even small, cost-sensitive products.

However, because OS-9 is written in 68000 family assembly language, it cannot be adapted ("ported") to other processor families. The greatly increased power of microprocessors, and reduced cost of memory, since OS-9 was first written has reduced the need for the operating system to be as small and fast as possible. Therefore Microware has written a companion operating system - OS-9000 - in C, which can readily be adapted to different microprocessors. In particular, OS-9000 is available for most 80386 and 80486 IBM PC compatible computers.

1.7.8 Adaptation to New Hardware

The modular arrangement of the I/O system and other parts of the operating system makes OS-9 very adaptable to new hardware, without the need for any of the source code of hardware-independent parts, such as the kernel.

The user can also customize the operating system, by adding or removing memory modules. This applies to all parts of the operating system - even the kernel can be adapted or extended, using "kernel customization modules".

1.7.9 Complete Set of Functions

It is very important that the programmer can always work within the operating system environment. He must not need to bypass the operating system and access hardware directly because of limitations of the operating system, as this reduces portability of the application, and can destroy the multi-tasking capability. OS-9 provides a full set of functions for the management and allocation of all resources, but it is not unnecessarily complex. Therefore the programmer can learn how to work within the operating system environment without too much effort (so he is much more likely to do so!).

Although OS-9 can be reduced for low-end applications, all of its functions are sophisticated enough to support top-end applications as well. For example, the disk file manager provides a full hierarchical directory structure, with almost unlimited file size, long file names (28 characters), and true record locking. Microware has additional file managers for a wide range of applications, such as graphics and networking, all equally sophisticated.

1.7.10 Broad Spectrum of Applications

OS-9 is suitable for a broader range of applications than perhaps any other operating system available. This comes from its modular, ROMmable construction, its full set of sophisticated functions (including multi-user support), and its relatively small size. OS-9 can be (and has been) used in small, diskless, embedded products, on large multi-user systems, and on everything in between, such as personal computers and home computers.

Microware is a wholly private company, and owes no allegiance to any hardware manufacturer. This makes OS-9 unusual, as most other popular operating systems are partly or wholly owned by a hardware manufacturer. So Microware's only aim is to develop and support OS-9 as a commercially attractive operating system. It cannot be coerced into making OS-9 less accessible, or into bending it to be more suited to particular uses and less suitable to others. And it cannot be shut down at the commercial whim of a parent company.

This gives much greater confidence to companies who commit to using OS-9. The operating system is the environment for the whole product. It is much easier to change hardware than to change operating system!

Perhaps surprisingly, given its relatively low market profile, OS-9 is one of the most popular and widely used operating systems in the world, and has sold many hundreds of thousands of copies. It has been used in high volume products - such as Fujitsu and Tandy home computers - and in high-tech products - such as the Space Shuttle control station.

1.7.11 The Future

OS-9 is already the most widely used real time operating system on Motorola microprocessors. The advent of Compact Disc Interactive (CD-I) may well make it the most popular operating system in all fields. CD-I - launched at the end of 1991 in the USA and Japan, and early in 1992 in Europe - is a standard developed by Philips in conjunction with Sony to use compact disc for the storage of audio, video, and computer data, as well as programs. The operating system in the player is OS-9 (under the name CD-RTOS), and all CD-I applications run under OS-9.

Philips, Sony, and most of the other major Japanese domestic electronics manufacturers are or will be producing CD-I players, and are hoping that CD-I will be as big as, or bigger, than audio compact disc. If so, every field will be open to OS-9, which could conceivably become the standard

operating system in industrial products, home computers, personal computers, and workstations.

1.8 THE PARTS OF OS-9

OS-9 is a sophisticated operating system made up of several parts, and often supplied with accessory programs. This section summarizes these software components. Microware, and third-party software vendors, also supply a wide range of complementary software products.

1.8.1 Utilities

Microware license OS-9 in a number of guises, containing more or less of the operating system components and companion products. The programmer is most likely to be using Professional OS-9, which comes with a large set of utility programs, a C compiler, and the **umacs** screen editor. These are not part of the operating system itself, which is an environment for programs to run under. They are the basic programs that every programmer is likely to need in order to develop his software. Microware also sell additional utility programs for particular purposes, such as the C Source Level Debugger.

1.8.2 Language compilers

Professional OS-9 comes with a C compiler. Microware also have Pascal, Fortran, and Basic compilers. Third-party software suppliers offer compilers for other languages, such as Modula-2.

1.8.3 Kernel

The kernel module is the core of the operating system. It contains all of the system-independent functions, such as multi-tasking support and memory allocation. The kernel also provides the environment for all I/O calls to be serviced by the I/O system (which is composed of the file managers and device drivers).

1.8.4 File managers

The file manager modules perform the logical data processing part of an I/O call. For example, the maintenance of the filing structure on a disk, or input line editing on a terminal. In general, each file manager is suitable for a class of devices, such as disk drives, tape drives, or a network. The file managers

do not know how to physically transfer data through an I/O interface. For this they make calls to the device drivers. OS-9 can support any number of file managers, and each file manager can work through any number of device drivers. This gives OS-9 a "tree structured" I/O system.

1.8.5 Device drivers

Each device driver module has the functions to control a particular I/O interface device, such as a disk controller chip, or a serial communications chip. The device driver does not know why it is requested to perform the I/O, and (in general) performs no data manipulation or interpretation - that is the job of the file manager that is calling the device driver.

This functional split between file managers and device drivers simplifies new implementations and customizations of OS-9. To add a new type of I/O interface which fits an existing class it is only necessary to write a new device driver. The device driver writer does not need to have any understanding of how the filing system works.

1.8.6 Device descriptors

Somehow the operating system must know what device a program is referring to, and must be able to select the appropriate file manager and device driver to manage the I/O request. OS-9 does this using memory modules known as device descriptors. These modules contain only data. They give the name of the device (equal to the name of the device descriptor module), the name of the file manager module, the name of the device driver module, and information about the hardware configuration (such as the memory address of the interface chip).

1.8.7 Program support modules (trap handlers)

Programs can access memory modules by name. A program makes a call to the operating system, passing the name of the desired module, and the operating system (having searched the module directory) returns the memory address of the module. Modules can therefore be used to store data, or sets of subroutines for use by programs.

OS-9 provides another way by which a program can call subroutines in a separate module, without having to determine the address of the module. These modules are known as trap handlers. The calling program informs the operating system of the names of the trap handler modules it wishes to use.

Then, when it wants to call one of the subroutines in the trap handler, it makes another operating system call, specifying the trap handler and the number of the function within the trap handler.

1.8.8 Customization modules

OS-9 is very customizable. It is even possible to customize the kernel. The user supplies an additional module containing functions to add to or replace the existing system calls. The name of the additional module (or modules - any number are permitted) is placed in the configuration data module **init**. On startup, the kernel finds all such modules, and calls their initialization functions. This allows the module to add new system calls or replace existing ones, so extending or modifying the functionality of the kernel.

1.9 SPECIAL FEATURES OF OS-9

OS-9 is an unusual operating system in many respects. It uses advanced techniques to allow it to address a very wide spectrum of applications effectively, and yet remain small. OS-9 also makes *no* demands regarding the hardware configuration it runs on. It can run programs with nothing more than a processor and some memory (although without some I/O a system cannot do anything useful!).

☐ OS-9 modules

The OS-9 memory module concept is central to much of the flexibility of OS-9. It allows the operating system to be dynamically configurable, for programs and operating system components to be in ROM, and for programs to be held in memory even when not running. Programs can also use memory modules as common data pools, for inter-process communication.

☐ Relocatable, re-entrant, ROMmable operating system and programs

Microware specifies that all programs (and operating system components) must be written relocatably (or "position independent"). This means that the program does not use any absolute program addresses. Instead, program accesses such as calls to subroutines are done relative to the current program counter, so the program module can be placed anywhere in memory without needing modification.

The 6809 and 68000 family processors are specifically designed to support this mode of programming, so it is not a restriction. As a result, OS-9 does not need any memory management hardware to translate memory addresses (this is done in other operating systems so that the program always logically lives at the address for which it was written).

Microware also specifies that programs must access their data memory "register indirect". That is, when a program is started the operating system sets one of the processor's registers to point to an area of memory that the operating system has allocated for the data memory of the program. The program then accesses its variables relative to this register. This allows programs to live in ROM, while having variables in RAM. It also permits multiple "incarnations" of the program to run concurrently. For example, several users can be using the **umacs** editor at the same time. Only one copy of the program exists in memory, but OS-9 allocates separate data memory for each incarnation (or "process"), so each functions independently of the others.

This type of program (where data accesses are all register indirect) is known as a re-entrant program. Again, the Motorola processors are designed to support this kind of addressing.

The language compilers all automatically generate position-independent, re-entrant code.

☐ Tree-structured I/O system

The OS-9 I/O system is separated into file managers, device drivers, and device descriptors. This fragmentation of the I/O system into completely separate modules makes the OS-9 I/O system very customizable.

☐ Dynamically modifiable I/O system

OS-9 is very unusual - its I/O system is dynamically modifiable while the system is running. New file managers, device drivers, and device descriptors can be loaded at any time. An I/O interface can be used in one way, and then used for a completely different purpose by loading a new file manager and/or device driver. A manufacturer of an I/O board can supply a device driver with it that can be loaded whenever the user wishes to use the board.

Not only does this add to the flexibility of the system, but it makes debugging new I/O system components very much easier, as the system does not have to be rebooted to test each revision.

☐ Customization hooks throughout

OS-9 is intended for a very wide range of applications, many of which will have unique requirements. Microware have therefore made almost every aspect of OS-9 customizable - often in more than one way - by providing well defined mechanisms to modify or extend the operating system.

☐ User interface is very similar to UNIX

Microware modelled the user view of OS-9 very much on UNIX. Users with experience of UNIX are therefore rapidly at home with OS-9 - although there are differences!

The programmer's view of OS-9 is also very similar to UNIX, especially at the C programming level. Most UNIX programs can be ported (at the source code level) to OS-9 with little or no modification.

☐ Very regular utility command line syntax

The utilities provided by Microware all conform closely to the same command line syntax. Options are preceded by a '-', and can be in any order anywhere on the command line. All utilities respond to the '-?' ("help") option, and common options are the same on different utilities - for example, '-z' is used to indicate that file names will come from standard input.

This significantly improves the user-friendliness of the operating system, and encourages third-party software suppliers to use the same syntax conventions.