# FLEXOS'S MUSCLE

*FlexOS may look like MS-DOS, but that is where the similarity ends*

*Ben Smith*

PART Six

FlexOS, from Digital Research (remember CP/M?), is a real-time, multiuser, multitasking operating system particularly suited for point-of-sale applications. The basic utilities look and behave just like their MS-DOS counterparts (e.g., COPY, CHDIR or CD, and TREE), but the underlying functionality of the operating system is from an alternate reality.

FlexOS is rich with features for both applications developers and users (e.g., X/GEM, the multitasking version of the GEM graphical user environment). Concurrent processes can share the same program image. Files have read, write, execute, and delete privileges for three user classes: Owner, Group, and World. The list of advanced operations is far greater than the list of elements of DOS compatibility. FlexOS with all its options is a solid, attractive, and familiar world for any real-time application.

## At the Core

Unlike MS-DOS, FlexOS is a protected-mode operating system that takes advantage of 286 and 386 processors. It uses DOS media for its files. It recognizes DOS file systems, making the exchange of DOS and FlexOS files totally transparent. The 386 version even has a DOS applications environment, allowing you to run DOS applications in protected mode under FlexOS. The core of FlexOS consists of the *supervisor* and the *resource managers*. The supervisor controls the flow of requests from application programs. It handles CPU and memory-related requests with its kernel and passes device-related requests on to the resource managers as appropriate (see the figure).

The resource managers provide the administrative glue between the supervisor and the device drivers. There are separate resource managers to control the disk drives, pipes (interprocess communications), network facilities, the console, and a collection of miscellany, including communications with serial terminals and parallel printers.

In some ways, the process management is similar to Unix: Each process has a process ID and a family ID. Spawned processes retain the family ID of their parent. Besides spawning a new process (which runs concurrently with the parent), you can create a subroutine process (running in series with the parent). In this latter case, a new process ID is not created, and the calling process sleeps until the subroutine is finished. FlexOS also allows you to chain processes together; in other words, the calling process becomes the called process. All child processes can share memory with their parent processes.

At the heart of the supervisor kernel is the event-driven *dispatcher*, which responds to interrupts and schedules processes in the following way:

- All asynchronous service routines are run to completion, one at a time.
- Each standard process is run for its slice of time. These are run in the order of the priority levels in the process table.

Unlike Unix, FlexOS yields both system and user processes to interrupts. This way, FlexOS supports real-time asynchronous I/O in its application programming and device driver interfaces. An interrupt service routine is allowed to take control once the CPU state has been saved.

Once the time-critical operations are complete, the interrupt service routine can pass its activities on to an asynchronous service routine, which has a priority like the user processes. At this point in interrupt handling, the control returns to the dispatcher. FlexOS also provides a polling mechanism for non-interrupt-driven device drivers.

Applications for FlexOS must be polite about memory usage, because the operating system does not swap user process space out to disk. If a program isn't using some of its allocated memory, it should free it so that other processes can use it.

The FlexOS memory model treats system processes differently from user processes. System processes (e.g., supervisor, resource managers, and drivers) run without parameter-bound checking and have direct access to system hardware; memory

addressing is direct. A user process runs with full protection and is isolated from any direct access to hardware.

## For the Developer

The primary applications development language for FlexOS is C—specifically, the MetaWare High C compilers. This is supplemented with Digital Research's RASM-86 relocatable code assembler and LINK86, its corresponding linker.

The kicker for the FlexOS development environment is the FlexView windowed symbolic/source code debugger. A good symbolic debugger can make the difference between painful applications development and fast, efficient applications development—particularly for programmers who are new to the operating system and libraries.

The basic programming interface to FlexOS is provided by supervisor calls, the counterpart to Unix system calls. (Table 1 is a summary of the SVCs.) For example, to change the priority of a process from 200 to 100 (a higher priority), you would SET the `prior` field of the process table entry for a process. (When the COMMAND SVC creates a user process, the default priority is 200.) Many of the fields in a system table (see table 2) are read-only. In fact, the `prior` element is the only field that you can change in the process table.

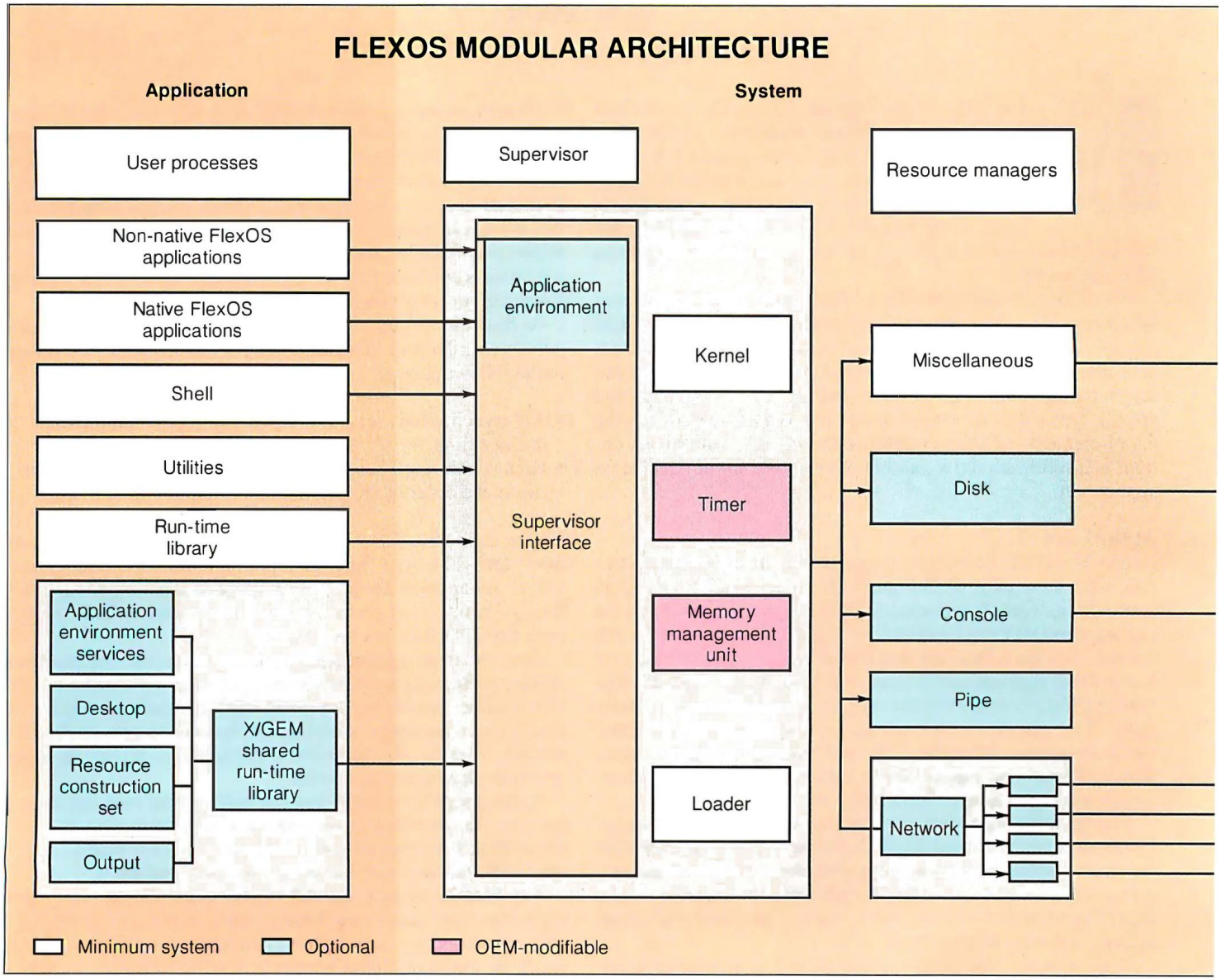
*The system comprises application services, a supervisor, resource managers, and their associated drivers and subdrivers.*

## Control from the Command Line

Many of the facilities that are available to the programmer through SVCs are also available to the user through shell commands and utilities. For example, the information of the process table is available through the PROCESS command. Unfortunately, you can do little more than view entries in the process table and kill processes. There is no way at this level of FlexOS to change the priority of a process or to move a background process into the foreground.

Although there are far more commands available to the FlexOS user than there are for the MS-DOS user, the script language has the same flow control and syntax as DOS .BAT files, with one notable extension: BATCH. The BATCH command allows nested scripts, meaning a .BAT file can be used as a subroutine of another .BAT file.

The interactive command interpreter has a command his-



tory. You cycle through the history list using a search string. As any user of the Unix Korn shell will tell you, this is a great time saver when you are doing system administrative chores. Another convenience is the ability to load device drivers without rebooting the system.

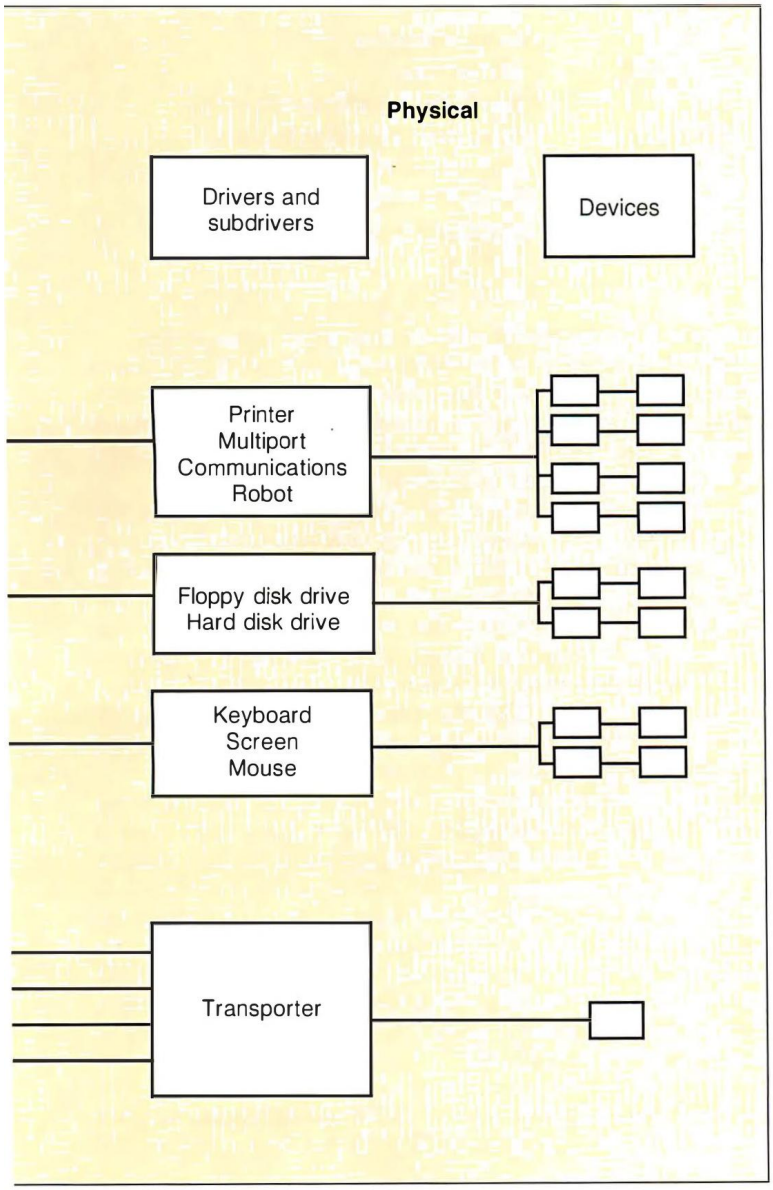
Several Unix utilities are part of FlexOS, including `pr`, `grep`, `split`, and `paste`. The impression you get at the command level is that FlexOS is MS-DOS with Unix extensions for multiuser, multitasking operation.

### Valuable Modules

Most of the alternative operating systems that we have covered in this series have provided only functionality; they have left aesthetics as an exercise for the developer. This is not the case with FlexOS.

Digital Research has ported its valuable GEM graphical user interface (GUI) to FlexOS. This version is called X/GEM but still looks the same as the MS-DOS version, including a window manager, a file manager, and several utilities. (X/GEM requires more memory than your basic 640K-byte system.)

The value that X/GEM adds to the GEM of MS-DOS and the Atari ST comes from the use of the underlying multitasking of FlexOS and support for sophisticated intelligent graphical



## FLEXOS SUPERVISOR CALLS

**Table 1:** *The basic programming interface for FlexOS is provided by supervisor calls.*

| SVC | Purpose |
|---|---|
| | **File management** |
| CREATE | Create a file. |
| DELETE | Delete a file. |
| OPEN | Open a file. |
| CLOSE | Close a file. |
| READ | Read a file. |
| WRITE | Write to a file. |
| RENAME | Rename or move a file. |
| DEFINE | Define logical name for a path. |
| LOCK | Lock/unlock an area of a file. |
| SEEK | Modify or obtain current file pointer. |
| | **Console file management** |
| COPY | Copy one screen rectangle to another. |
| ALTER | Alter a screen rectangle. |
| XLAT | Specify keystroke translation. |
| | **Event management** |
| STATUS | Get status of an asynchronous event. |
| RETURN | Get return code of completed event. |
| WAIT | Wait for multiple events. |
| CANCEL | Cancel asynchronous event. |
| | **Process management** |
| COMMAND | Perform shell command. |
| CONTROL | Control a process for debugging. |
| OVERLAY | Load overlay from command file. |
| TIMER | Set and wait for timer interrupt. |
| ABORT | Abort specified process. |
| EXIT | Terminate with return code. |
| ENABLE | Enable software interrupts. |
| DISABLE | Disable software interrupts. |
| SWIRET | Return from software interrupts. |
| EXCEPTION | Set software interrupts on exceptions. |
| MALLOC | Allocate memory to heap. |
| MFREE | Free memory from heap. |
| | **Device management** |
| SPECIAL | Perform special device function. |
| DEVLOCK | Lock/unlock device for user/group. |
| INSTALL | Install, replace, and associate drivers. |
| | **Table management** |
| GET | Get table values. |
| SET | Set table values. |
| LOOKUP | Scan and retrieve tables. |

## ITEMS DISCUSSED

**FlexOS** .....(Contact vendor for pricing)
Digital Research, Inc.
OEM Sales
4401 Great America Pkwy., Suite 200
Santa Clara, CA 95054
(408) 982-0700
**Inquiry 1008.**

## FLEXOS SYSTEM TABLES

**Table 2:** *The key to system control in FlexOS programs lies in system tables, data structures that contain information about the status of the system. To operate on system tables, you use the supervisor calls GET, SET, and LOOKUP (see table 1).*

| Table | Contents |
|---|---|
| PROCESS | Process information |
| ENVIRON | Process environment |
| TIMEDATE | System time of day |
| MEMORY | System memory use |
| PIPE | Pipe information |
| SHMEM | Shared memory information |
| DISKFILE | Disk file information |
| DISK | Disk device information |
| VIRCON | Virtual console information |
| SYSTEM | Global system information |
| FILNUM | File numbers table |
| SYSDEF | System logical name table |
| PROCDEF | Process logical name table |
| CMDENV | Command environment |
| DEVICE | Device information |
| PATHNAME | Full path name |
| SRTL | Shared run-time libraries information |
| MOUSE DT | Mouse driver table information |
| VDIPRN | VDI printer device information |
| PORT | Port device information |
| SPECIAL | Special device information |

hardware. It is compatible with existing GEM applications.

There is an optional network extension to FlexOS that allows several FlexOS systems to share disks, printers, and communications pipes. The underlying transport and low-level protocol is not defined. Digital Research distributes drivers for IBM Token Ring Adapter and TCP/IP on Ethernet.

Nearly everything in FlexOS is modular. Not only are the device drivers, networking, and the GUI and application programming interface modules, but the basic system is broken into submodules that can be installed as needed. The whole system can be stripped down to the point that it can be put into ROM, a requirement for factory-floor and point-of-sale devices. FlexOS data-entry devices and "cash registers" can boot and run entirely from ROM; the transactions are sent across the network to another device that has mass storage and appropriate administrative programs.

FlexOS was not designed as a DOS clone, nor is it marketed as one; it is for multitasking and multiuser applications. Nor is it designed as a simple Unix substitute; it is a real-time operating system that necessarily lacks much of the complexity of Unix. Digital Research has been in the business of making operating systems since the beginning of the microcomputer era, and in many ways it is responsible for early acceptance of the microcomputer. It has also been making multiuser operating systems from those early days. The company's experience is quite apparent in FlexOS. ■

*Ben Smith is a technical editor at BYTE and the author of* Unix Step by Step *(Howard Sams, 1990). He can be reached on BIX as "bensmith."*