Mach DOS Update

Gerald Malan
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue Pittsburgh, Pennsylvania 15213

1. The Mach DOS Software Support

Approximately a year and a half ago we began work here at Carnegie Mellon on a project to implement DOS as a user level application based on a Mach 3.0 kernel. All of this work on the Mach DOS software was done by Gerald Malan, Rick Rashid, David Golub, and Robert Baron. To date we support DOS as a multithreaded Mach Task running in conjunction with either the 4.3 BSD Unix Server or the OSF/1 Single Server. The Mach DOS software support consists of machine dependent kernel support, and a multithreaded emulator task which provides virtual devices, BIOS emulation, support for common DOS extensions, emulation of many DOS system calls, and software for integration with the Unix server. This work was outlined in an article presented at the Second Usenix Mach Symposium and is detailed in the Symposium's proceedings pages 27 through 40.

The system supports DOS versions 3.1 to 5.0 and runs Microsoft Windows 3.0 in real-mode. We have successfully run over 100 DOS applications ranging from business software such as Lotus 123 and Microsoft Word to entertainment software such as Wing Commander, Space Quest IV, Populous, and the Microsoft Flight Simulator.

The Mach DOS software does not supplant DOS, but provides an environment in which it is executed. We load the DOS bootstrap image from a DOS floppy or hard disk partition, initialize a special thread's instruction pointer to the bootstrap code, and resume it's execution. This brings DOS into the Mach DOS task's address space and begins execution of the DOS command line interface. This special thread executes in a mode called V86 mode, which interprets instructions as an 8086 chip would. Whenever a DOS program generates an exception, like a DOS system call or enabling the system's interrupt flag, the Mach 3.0 exception handling mechanism catches the instruction and either handles it inside the kernel or passes it to a thread in the user level Mach DOS task.

The Mach 3.0 kernel contains the machine dependent emulation code which handles many of the DOS V86 thread's exceptions. The balance of support between the kernel and user space was decided after several iterations of testing. Currently the Mach 3.0 kernel contains support for V86 mode exceptions pertaining to interrupt flag management. These exceptions account for approximately 50 to 90 percent of all V86 mode exceptions, depending on the DOS application. All other V86 mode exceptions are passed from the kernel to the Mach DOS task's monitor thread.

The Mach DOS software's user level support is broken down into two major areas: a monitor thread that handles the V86 exceptions passed from the kernel, and several input handling threads. The monitor thread receives exception messages from the kernel and determines the type of system calls to which they refer. The input handling threads gather input from the keyboard, mouse and timer chip and either queue the input or generate external interrupts to the V86 thread.

The monitor thread receives the V86 exceptions that the kernel doesn't emulate and determines which kind of exception was generated: a BIOS or a DOS system call. The Mach DOS software emulates many of the BIOS calls, but only a few of the DOS system calls. The BIOS is a collection of standard device drivers that are contained in the ROM on DOS machines. The Mach DOS software intercepts the BIOS calls related to the disk drive, the timer chip, the keyboard, and the mouse. The only DOS system calls that the monitor thread emulates are those that do output to the console. All other BIOS and DOS system calls are redirected back into the 8086 code in the Mach DOS task's address space. The old version of the Mach DOS software emulated the all of the video BIOS, but recently we removed this code and now redirect the video BIOS calls back into the ROM.

Currently there are three input handling threads, one for the keyboard, mouse, and timer devices. These threads are all similar in the fact that they gather input for the executing DOS program, but are very different in operation. This difference in operation stems from the fact that DOS programs break down into two categories: the well-behaved DOS program, and the ill-behaved DOS program. Well-behaved DOS programs use the BIOS and DOS system call interface to acquire all of their data. Ill-behaved DOS programs supplant the input device's driver with their own and redirect the device's hardware interrupts into their own code. In order for the Mach DOS software to support such programs, the input handling threads must be able to determine which kind of program is executing and supply the correct style of input: request/supply system call return values or interrupt style direct i/o procedures. For detailed examples of these operations, refer to the Usenix Mach Symposium proceedings.

In an effort to integrate the Mach DOS and Unix environments we added the ability to access the Unix File System (UFS) as a DOS network drive, and the ability to pause the Mach DOS software and return to the parent unix shell. The ability to use the UFS as a DOS disk drive enabled us to access AFS and RFS from within the DOS environment. The pausing feature allows the user to create multiple Mach DOS instances and the ability to switch between them at random.

In December of 1991, working with the OSF team in Grenoble, CMU made changes to the Mach DOS software so that it would work with the OSF/1 Single Server. Most of the changes were made to the code that mapped the machine's physical memory into the Mach DOS task's virtual address space. The OSF/1 Single Server was modified very slightly so that the keyboard and mouse thread could do a blocking select for input. After these modifications were made, the Mach DOS software was ready to show at the OSF RI Symposium this last February. In the corridor outside of the presentation hall, an HP Vectra was set up running this code on top of the OSF/1 Single Server.

Can the performance of DOS as a Mach 3.0 application beat that of native DOS? During the OSF RI Symposium last February, I presented figures on the performance of Mach DOS versus native DOS. The current implementation of the Mach DOS support is in almost every aspect slower than native DOS. Although Mach DOS is slower, the magnitude of this decrease in speed is not the same for every DOS program. In fact the differences vary widely, depending on which strategies a DOS program employs. For example, a directory listing is approximately 150% slower under Mach DOS than native DOS, while launching Windows 3.0 is only 7% slower. Our analysis of the performance data shows DOS programs that are optimized to run fast on top of native DOS continue to run well under Mach's emulated DOS. In contrast, programs which run slowly on native DOS run even slower on emulated DOS. The reason for this is that non-optimized DOS programs use safe DOS interfaces that cause multiple exceptions to be generated for a single operation, whereas the optimized DOS programs only cause a single exception or none at all.

Currently the work being done on the Mach DOS project centers around polishing up the version that is dependent on a Unix Sever, and preparing the groundwork for moving to a stand alone Mach DOS server. As mentioned above, we removed the video BIOS support from the Mach DOS code. This was done so that SuperVGA support could be added without the complexity of writing SuperVGA specific code for each video BIOS routine. In MK73, kernel changes were made so that the setup and installations procedures are radically simpler. The current version of the Mach DOS software is available license free from CMU, either via sup or ftp.