MWJ: Mac OS 9

The Weekly Journal for Serious Macintosh[™] Users

" Top of the Special Issue "

• Our Mac OS 9 coverage begins with "Getting Started," a quick look at Mac OS 9's requirements, sensible precautions, where to find software updates you might need, and a couple of cautions that might save you time and trouble, plus a look at "error 119," the most notable of the potential compatibility problems in the new release. The Mac OS 9 Basics, Page 2.

We next look at the release's marquee feature. ٠ Sherlock 2 is an outstanding addition to the Mac OS in almost every important way. Sherlock's original masterstroke was in both expanding file searching to include indexed contents, and in describing Internet search engines with simple text files. Sherlock 2 goes even further, breaking down artificial barriers in Sherlock's file searching and grouping Internet search sites into thematic channels. The channels are slightly more restrictive than in original Sherlock-but they're self-repairing, self-updating, and you can make your own. The program has a few bumps-most notably the atrocious interface and a bad advertising policy-but if you like Sherlock, you'll positively drool for its successor. Sherlock 2 The Next Level, Page 5.

Then comes our look at three of Apple's "Nine ٠ Internet Power Tools"-the three that relate the least to the Internet. Multiple Users doesn't turn the Macintosh into a multi-user operating system, but it's a surprisingly elegant way to share customized environments on one computer with reasonable-though not foolproofsecurity. Voice Verification is a nice password alternate that may foreshadow even more powerful methods in the future. Apple File Security's encryption of files looks great at first, but sadly, fails to withstand scrutiny. Ironically, the only Internet part of any of these three features is the one that Apple's not talking about: digital signatures for downloaded files with thorough Macintosh support, painting enticing possibilities for future self-updating software with reasonable security. Sharing Your Personal Space, Page 17.

• We consider the remaining five of Apple's **Nine Internet Power Tools**: the Keychain, Network Browser 2.0, Software Update, AppleShare over IP, and AppleScript over IP. As usual, the stories behind these technologies may not be quite what you're imagining, but lots has changed since Mac OS 8.6. The "major features" in that revision were sometimes no more than text files. Now we get the real deal, done the right way (though poorly documented at times), and thoroughly integrated into the Macintosh. Join us as we explore exciting new ways to merge the Internet with your desktop. **Living in a Wired World, Page 31**

We next provide examination of features beyond ٠ the most heavily marketed. In Watch This Space, we look at five key technologies that you can't really see today, but they'll become the foundation for new features and greater productivity in the future as Mac OS 9 becomes more of a baseline system. We examine new File Manager APIs granting access to advanced features of the HFS Plus File System, and the quiet but extremely exciting addition of real font management to the Mac OS (yes, it's the real deal), a new Multilingual Text Editor, the HTML Rendering Library, plus an overview of the Carbon and Core Foundation technologies that preview Mac OS X. Join us as we look deep under the hood for signs of future software. Features of the Future, Page 44.

The Nitty-Gritty is the home stretch—a look at all those tiny changes in the new version that affect your daily use but aren't comprehensively documented. The Control Panels section is the largest with three fairly major changes—ColorSync 3.0, FontSync 1.0, and the changes that make all control panels work as separate applications. We also track the splitting of "Monitors & Sound," lots of extensions including the frustratingly-vague Open Transport 2.5, Finder changes, "Package First Aid," and even System file goodies like the Notification Manager QuickDraw enhancements. There's too much in here to summarize-it gets technical at times, so don't be afraid to read it more than once. If you want to know what's really in Mac OS 9, this is your source. The Nitties and Gritties, Page 57.

• *The Weekly Attitudinal*, our right-by-definition opinion feature, wraps it all up with a short (no, really!) analysis of Mac OS 9 answering the question in the Ricky Martin song: do you really want it? **Unguarded Opinion, Page 75.**

Mac OS 9

Getting Started The Basics of "The Best Internet OS Ever"

When Steve Jobs talks about Mac OS 9 at trade shows or in public speeches, he uses the same glib marketing phrase he used a year ago for Mac OS 8.5: "It's a whole new Macintosh for US\$99." It was a bit of an exaggeration then, and it remains so now. However, like Mac OS 8.5, today's US release of Mac OS 9 brings hundreds of small, under-the-hood changes to the guts of your Macintosh. Characterizing a large and wide-ranging OS release is always a challenge, and Apple Computer rarely picks the same emphases we at MWJ would. We're stoked about font and text improvements, a long-awaited overhauling of how the Mac OS deals with files and disks, the precursors for Mac OS X support, and all kinds of tiny changes that make the Macintosh a better solution for personal productivity. Apple's new slogan? "The Best Internet OS Ever." Don't repeat that around UNIX advocates.

Apple, however, long ago learned the danger of marketing to technical wonks like us. Instead, the company's presentations on the new operating system focus on large, user-level features—stuff you could explain to your grandma if need be. We'd hate to try to explain to Granny that "it can now create and manipulate files larger than two gigabytes when using alternate APIs in revised software," but we'd have no trouble at all explaining "it knows who you are when you speak to it."

Before you upgrade, however, you should prepare.

Is It Shipping?

Mac OS 9 is available now in the US. Suggested retail price is US\$99; current owners of Mac OS 8.5 or later are eligible for a US\$20 mail-in rebate with details in the product box. All owners of Power Macintosh G4 and iBook computers, as well as anyone who purchased a Macintosh *computer* on or after October 5 can get Mac OS 9 for US\$20 shipping and handling through the Mac OS Up-To-Date program. Note that this is different from earlier offers: purchases of the retail Mac OS 8.6 product are not covered, just new computer purchases.

Make sure you have the proper equipment: a PowerPC-based Mac OS computer with at least 40MB of addressable RAM. You can use virtual memory to boost your addressable RAM, but the system must have at least 32MB of physical RAM. Virtual memory requires more hard disk activity, and iBook computers ship with only 32MB of RAM, so iBook owners should be prepared for somewhat shorter battery lives. As with other recent Mac OS releases, Apple is only supporting the release on Apple-brand computers, but we know of no reason it should fail on authorized clones.

Is It Compatible?

Oh, yah, you betcha. Pretty much. There's only one *major* problem, and it was exacerbated by early reports from people (violating non-disclosure agreements) more eager to talk about the problem than to learn what's really going on. It led to uninformed speculation and semi-panicked compatibility reports in the weeks preceding release. Here's the real deal.

As MacWEEK.com correctly reported, you'll need new versions of some applications. Some, like SoundJam MP 1.1.1, are already available; others are coming shortly. The most significant problems occur with Aladdin Systems's StuffIt product family, and with Adobe Systems's type products Adobe Type Manager and Adobe Type Reunion.

Type Reunion is a luxury, but most applications require ATM to display PostScript Type 1 fonts properly-and such fonts are in widespread use. Some reports have obscured this by commenting on a "Type 1 font scaler" reportedly present in Mac OS 9. True enough, one exists-but it's an Open Font Architecture scaler and only works for OFA-savvy font clients (see "The Nitty-Gritty" in this issue). Right now, that means Apple Type Services for Unicode Imaging (ATSUI), the successor to QuickDraw GX Typography. Plain old (non-GX) QuickDraw, used by literally every Macintosh application you're ever likely to run, does not use OFA and will only use Type 1 outline fonts with Adobe Type Manager. The only way around this is to use applications with built-in Type 1 rendering code—like all of Adobe's current high-end applications (Photoshop, InDesign, Illustrator, Acrobat-but not PageMaker). In short, you still need ATM to use Type 1 fonts except in very limited circumstances.

What Is The Problem?

The major issue is a new fatal system error, #119. ATM, ATR, StuffIt Engine 5.1.2, DropStuff 5.1, StuffIt Expander 5.1.3, and several other programs stop cold with error 119, returning you to the Finder just like an error of type 10 or type 1 would do. With error 119, however, the Finder helpfully informs you that you need an updated version of your application.

This past spring, we discussed why the Mac OS is limited to 348 open files at once—each open file comes with a 94-byte *file control block* (FCB) that, for backward compatibility reasons, has to fit within a measly 32K table. A partial control block is no good; only entire 94-byte chunks are valid. Only 348 of those fit within the 32,768 allowed bytes (*MWJ* 1999.05.02).

Apple has planned for a long time to fix this in Mac OS 9, but has run into huge compatibility problems. Not the least of these is *glue code*, provided by Apple and built into thousands of third-party applications, that assumes way more about FCB structure than it should (*MWJ* 1999.06.19). Even so, historically Apple has handled this careful compatibility dance fairly well: the old glue code works. Unfortunately, some third-party applications don't use Apple's glue code, and they may wind up reading FCBs directly in a no-longer-supported way. That means reading invalid data and acting on it as if it were valid.

These are the *worst* kinds of bugs to diagnose and fix. It's like picking up the salmonella virus from a single undercooked chicken nugget. You may or may not get sick, and if you do, it may or may not be severe enough for you to seek medical attention. If you do see a doctor, a mild case may not be diagnosed as food poisoning—and even if it is, there is almost zero chance you'll ever trace the infection back to one chicken nugget (out of six, nine, or twenty) that you ate a few days earlier with who knows what other food. You have a problem, but the cause is elusive.

The same thing happens when programs act on invalid data. If the bad data happens to be so wacky that it triggers a bus error, you crash quickly. If not, the program may start wandering through the weeds and eventually crash in a completely unrelated fashion. Or it might not crash at all—it might just corrupt data or a disk. Since the 32K table of control blocks is different with every set of files you open and close, these bugs can be nearly impossible to reproduce. Even if you can, there's no guarantee the developer could reproduce it on his system with the same steps unless *every* file was *exactly* the same. It's maddening.

Apple's engineers knew that applications using the 32K table of FCBs without Apple's glue code *would* fail—the only way they wouldn't is if they were exactly imitating the glue code, and programs just don't do that. Through most of Mac OS 9 development the engineers tried to make it work, but too many applications were using the table. So in August they surrendered—they changed the PowerPC code in the System file for finding the table's address to stop the system with error 119 instead.

If the salmonella virus reacted with a chicken nugget such that it grew to three feet in diameter and turned it shocking pink, you'd know not to eat it. Similarly, error 119 forces the control block issue when you see it, it's immediately clear that a very subtle error was about to happen. The philosophy—and it's correct—is that it's better to stop everything than allow nasty bugs a developer might never find. (This is the same concept behind protected memory, by the way: any code touching memory it's not authorized to use halts the application with a fatal error. Developers and users alike have been clamoring for protected memory for years exactly so major problems will show up sooner rather than later.)

Whose Fault Is It?

If you *insist* on assigning blame, we say it belongs with developers. Apple has warned since 1986 not to directly depend on the 32K control block table or its format. Unfortunately, developers tended ti ignore the warnings, because Apple's glue code did exactly what the warnings said not to do.

The Mac OS has included supported routines for legitimately accessing control blocks since 1994's release of System 7.5. There are only two excuses for not using them—a program might have to work with Mac OS releases earlier than System 7.5, or it's a very low-level program that can't access control blocks through a system routine that might not be available at the time. The low-level developers are in their own world that's not relevant here. Applications, on the other hand, tended to include code to walk the 32K table, and the programmers tended to leave it in place because it worked on all versions of the Mac OS back to 1986.

Apple continued to warn developers against using the FCB table directly-last November, engineer Jim Luther wrote a developer Q&A noting that direct access to the FCB table would soon break (MWJ 1998.11.07). We grant that Apple could have taken more drastic steps. For one, they could have stated flatly that direct FCB access will break in Mac OS 9 instead of talking about the indefinite future-but if compatibility had worked better, developers would have seen that the warning didn't come true and paid even less attention to future warnings. Apple could also have punted on compatibility and introduced error 119 from the beginning-but that's not the attitude Mac OS users and developers have come to expect. But this sort of talk is really beside the point: such extraordinary measures shouldn't have been necessary.

Getting Started

(continued from page 3)

Most developers are forthright in that they've been caught with their hands in the FCB table, but some want to point fingers anywhere but at themselves. In late September, Adobe's Chris Holm posted a note to an Adobe forum, sent as well to several leading Mac-oriented Web sites, confirming that Adobe Type Manager and Adobe Type Reunion would not work under Mac OS 9. We know the problem is direct access to the FCB table, but Holm instead pointed the finger at Apple, saying the company "removed an API" that ATM needed. As you've seen, Apple did no such thing. In fact, the problem is that Adobe's products do not use APIs added in 1994; direct access to the FCB table is a hack to work around the pre-1994 lack of said APIs, and Adobe never removed the hack and embraced the APIs. Claiming Apple "removed" APIs is almost completely disingenuous.

We had thought this an aberration—the statement was removed from the Adobe online forum, and other executives had not stood by it when interviewed on the topic (because, we reasoned, they knew it was not true). Yet late last week, Holm reposted the accusation with a permanent Adobe Forum URL and only slightly changed language. This time, he says "The final release of Mac OS 9 does not include application program interfaces (APIs) required by ATM." This is completely false. Mac OS 9 includes all the APIs ATM needs-the problem is that Adobe isn't using them and has passed on every opportunity since 1994 to correct that shortcoming. The only thing Mac OS 9 is missing is the "let's restrict everyone to only 348 open files until the end of time so Adobe doesn't have to fix its broken code" module. We sympathize with developers who thought they had more time to fix their code, especially since Apple made it clear they would keep direct FCB access from breaking if at all possible. But it wasn't possible, and blaming Apple for "removing" support for a technique Adobe should have ditched five years ago is unsupportable buck-passing.

When you see error 119 in Mac OS 9 and later, you'll know why—nothing else creates error 119, nor will any other condition use that fatal error number. Adobe is not promising new versions of its type utilities before Mac OS 9 ships; Aladdin's replacement utilities will ship with Mac OS 9. Prepare in advance: the changes in Mac OS 9, particularly File Manager changes that increase the number of possible simultaneously open files to 8169 from 348 but also eliminate an old and unsupported method for finding information about open files (*MWJ* 1999.10.09, 1999.10.16), are prompting more OS-related software updates than usual. You've probably already heard *ad nauseam* that Adobe Type Manager and Adobe Type Reunion aren't compatible with Mac OS 9; the Installer moves them to the appropriate "disabled" folders in the System Folder. You also need StuffIt Expander 5.1.4 or later, and StuffIt Engine 5.1.3 or later; both are part of Mac OS 9's "Internet Access" package and are part of the normal installation. Aladdin Systems has instructions for reinstalling the updates if your system didn't get them.

Preparing To The Nines

Be sure to check for updates and incompatibilities in products key to your daily use. We recommend a two-stop information tour: VersionTracker's Mac OS 9 Updates page lists updates for software known to be incompatible with Mac OS 9, updated frequently. And for troubleshooting, the gold-star source is still MacFixIt, where the Troubleshooting Mac OS 9 Report already has lots of reports-but, as is usual with troubleshooting, many are unverified, so read critically. Alsoft has released the Mac OS 9 File Manager Compatibility Checker, a handy utility that attempts to find FCB-related problems in existing applications. It's not foolproof-the utility checks for telltale code sequences that might represent PowerPC instructions to touch the now-forbidden FCB table. It can't detect all problems, and it doesn't even try from 68K code (where problem-finding is trickier), but it's a reasonable first defense.

We're also quite fond of Insider Software's UpdateAgent 3, a utility that scans all the programs on your system and compares them against a database at Insider Software to identify old and outdated versions. The program offers to download any available updaters for you; it also downloads "Read Me" files for commercial upgrades that aren't free online. UpdateAgent is sold via subscription, with an onlineonly version starting at US\$50 per year (with other options including CD-ROM delivery and site licenses). But you can run the program once on one machine for a US\$10 fee. That option could save lots of time in identifying and downloading necessary compatibility updates, and could be well worth the US\$10 in saved time.

And, of course, always always *always* have a full backup of your system before performing major Mac OS upgrades. This time *you* could be the one person in 300,000 for whom something goes tragically wrong, and reconstructing an older System Folder is a chore we wouldn't wish on our enemies.

Startup Significance

Apple has long recommended that you boot the new Mac OS CD before installing the system software. You can still do "live installations" over your active System Folder, but it's a more complicated procedure for the Installer, increasing the odds of problems. Owners of the original Power Macintosh machines (6100, 7100, and 8100 models), however, *must* boot from the Mac OS 9 CD to install the new version. Apple Computer says those machines contain an unspecified ROM bug that prevents installation from working unless you're already running Mac OS 9. Other systems, including the newest ones, have no such requirement.

Don't panic, but the CD might not boot. We confirmed several reports on our own systems: the Mac OS 9 CD may freeze during startup if various USB devices are connected. The exact nature of the problem has so far eluded us, but this advice fits both our experience and what we've seen reported elsewhere: unplug USB devices other than the keyboard and mouse before booting the Mac OS 9 CD. You probably won't need printers, scanners, tape drives, or most of the other whatnot during installation anyway.

.....

Once Mac OS 9 is installed, all seems to work well. We suspect the problem is in the new USB software as burned into the CD-ROM's active System Folder, and probably not as simple as "the CD-ROM doesn't have the USB drivers your peripherals need," so we doubt you'll see the same problem even after a "clean installation." In fact, the Software Updates Control Panel has a pleasant surprise for you the next time you plug in a USB device for which you have no driver installed (see "Living in a Wired World," this issue). In our testing, unplugging a USB hub solved a freezing problem. Installation proceeded normally, and the newly-updated iMac in question has been happily plugging along with Mac OS 9 ever since. Well, ever since we remembered to update incompatible software. We recommend you do that first because we didn't and it wasn't fun.

With a little patience and a minimum of planning, you'll be using those Nine Internet Power Tools in no time flat—so let's commence examining them.

Mac OS 9

Sherlock 2 The Next Level Great New Features With Questionable Interface

No feature of Mac OS 9 is more highly marketed and demonstrated than *Sherlock 2*, the revision to Mac OS 8.5's signature file and Internet searching utility. Credit Apple with realizing what a powerful tool this little program is—late in Mac OS 8.5's development cycle, the executives mandated a name change from "Find 2.0" to "Sherlock" for easier marketing, even though the program's version number (and many of its messages) retained the "Find 2.0" designation. The evolution of "Find", now officially version 3.0.1 in Mac OS 9, has its major revision built into its name: the program's title, evidenced in the "About..." box and title bar, is "Sherlock 2." Rename the file if you want, but the name sticks.

The program retains the same basic features searching file attributes and the contents of indexed files, or querying a set of Internet sites for specified text and returning all the results at once. On the one hand, Sherlock 2 combines some of these familiar features in outstanding ways to make it far more useful even in the human interface. On the other hand, Apple's hell-bent determination to move its marquee software programs to the "burnished metal" interface seen in QuickTime 4 seriously compromises those new features, as well as old ones people have grown to love. Sherlock was an elegant expression of simple concepts, criticized mainly for interface limitations in the "Search Internet" section. Sherlock 2 overcomes some of those and needlessly adds others, even as it extends the feature set in equally elegant ways.

Files, Inside and Outside

Sherlock's ancestry in System 7.5's "Find File" program (originally written by Bill Monk as "Find Pro III") showed in its three-tabbed interface (*MWJ* 1998.10.19). The first tab, "Find File," was virtually identical to the previous utility. The second tab, "Find by Content," allowed searching the contents of files as indexed by Sherlock's companion program, "Find by Content Indexing," controlled from Sherlock's window. In Mac OS 8.6, Apple gave Sherlock the power to use plug-in indexing modules to properly read file formats that were neither built into Sherlock nor accessible through the Macintosh Translation Manager (*MWJ* 1999.05.12).

Figure 11 shows Sherlock 2's main window. The square icon wells across the top are *channels*. They

Sherlock 2 The Next Level

(continued from page 5)

work like "favorites" in QuickTime 4 Player. We'll explore channels more thoroughly later in this article, but right now we're interested in the first channel, the selected one whose outer well is highlighted in aquamarine (regardless of your chosen Appearance colors). This is the built-in *files* channel, a merging of Sherlock's "Find File" and "Find By Content." It works really well—the way Sherlock should have worked in the first place.

Instead of separate tabs for the two functions, they're combined in a single view, though the Command-F and Command-G key equivalents still work. The editable text field contains the text for which you search, while radio buttons determine if you seek it in a file's name or in its contents. The pane of disks from "Find By Content" is now present for all file-finding, a vast improvement on "Find File's" popup menu that let you choose individual volumes or pre-determined sets, like "all volumes except CD-ROMs," but not custom disk sets. "Find By Content's" radio buttons to search either the selected volumes or the Finder selection are gone, but there's now a better way-just drag any folder you want to search to Sherlock 2's volume pane. Although the program incorrectly fails to draw the colored rectangle during dragging indicating it will accept a drag-and-drop operation, it will, with each folder gaining its own checkbox so you can enable or disable it as you please. You can't remove the built-in volumes from the list, but you can get rid of any folder you add by pressing Command-Delete or picking "Move to Trash" from the "File" menu. A new "Toggle" menu item and Command-T key equivalent unchecks all items if any are checked and then serves as a true toggle, checking or unchecking all items with each invocation. All of these options, including custom folders, can be saved in "Search Criteria" documents (as with Sherlock).

But Figure 1 has three radio buttons. After "File Names" and "Contents," a third radio button enables a

Sherlock 2 Sherlock 2 Sherlo									
On	Name	Index Status	<u>غ</u>						
	🙆 Mac OS 9	Volume is not indexed	<u> </u>						
	☑ ☑ Macintosh HD Volume indexed Wed, Oct 20, 1999, 12:02 AM ☑ ☑ ☑ Was OS Read Me Files Volume indexed Wed, Oct 20, 1999, 12:02 AM								
4									
-									
Find iten	ns whose file name contains "R	ead Me", creator is "ttxt", contents include "Apple Computer"							
-									
		Figure 1—Sherlock 2's File Finding							

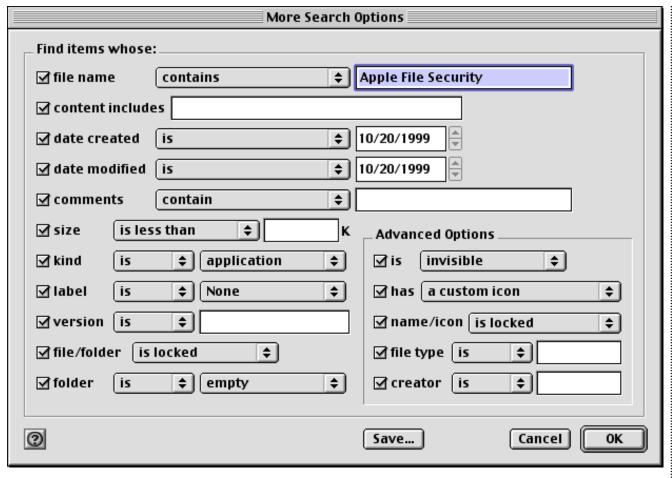


Figure 2—More Search Options

pop-up menu showing a "Custom..." item. If you pick that, or click the "Edit..." button, or pick "More Options..." from the "Find" menu (or press the Command-M key equivalent), you see the dialog box shown in Figure 2. Unlike "Find File's" series of buttons that add extra clauses with "More Choices" and remove them with "Fewer Choices," Sherlock 2 lets you set all of your options in one large but functional dialog box. Select the checkboxes next to the parameters you care about, then set their values as you need. You no longer have to hold down the Option key while selecting a pop-up menu to see "Advanced" options; they're in plain sight but set apart.

Don't worry if the whole thing is too confusing you can control the dialog box by example. Drag any file's icon into the "More Search Options" dialog box and Sherlock 2 sets every option (except "content includes") in the dialog box to match the file or folder you dropped onto it. Just check the parts you want to match and adjust them if necessary. Sherlock's "Find File" section did the same thing, but you had to drag the file to each individual attribute; Sherlock 2 sets them *en masse*.

This large dialog box doesn't follow the "sentence" metaphor of Find File. Even though the levels of choices could be confusing, Find File's interface read as a complete sentence (at least until Sherlock, when the "and" connecting clauses became implicit): "Find items on local disk whose name contains 'read me' and date modified is yesterday." It's a handy reality check, and Sherlock 2 preserves it. Back in Figure 1, you see a sentence-type summary of what you're seeking in the pane underneath the list of volumes and folders to search. Unlike Sherlock, you can't hide or collapse this pane in Sherlock 2.

The "More Search Options" dialog box is complete—except for the disks and folders to search, it contains all the options you could choose from the main Sherlock window. Clicking "Save..." asks you to name your set of search options and adds them to the main window's pop-up menu where you see "Custom..." in Figure 1. Apple ships Sherlock 2 with four pre-configured sets of file searching options: Applications, files larger than 1MB (unfortunately, incorrectly defined as "1,000K"), files modified today, and files modified yesterday. You can start with any of these and customize them in either the main window or the "More Search Options" dialog box. For exam-

Sherlock 2 The Next Level

(continued from page 7)

ple, picking "Modified today" and typing "Personal Journal" in the editable text field searches for all files modified today whose names contain that text. If you want to find files modified yesterday whose contents contain given text, you'll have to visit the "More Search Options" dialog box.

The options you save are regular Search Criteria documents, located in the "Files" folder of your "Internet Search Sites" folder (typically in the System Folder, but mapped to your user folder if you're a non-owner under Multiple Users; see "Sharing Your Personal Space," this issue). You can move other Search Criteria documents to that folder; Sherlock 2 places them in the main window's pop-up menu, but it won't notice changes until you quit and relaunch the application. It notices changes made from within the application immediately. Also note that the Finder doesn't auto-route file Search Criteria to the "Files" channel's folder in "Internet Search Sites"—the files are dropped in the root of the search folder, and you'll have to move them into the "Files" folder by hand to get them into the pop-up menu.

To start the search, click the button with the magnifying glass or press Return or Enter. The icon button changes to a red Stop sign (at least in the US version); you can also halt the search with Esc or Command-Period. Figure 3 shows the results of file searching, presented in the same window. The item list itself is familiar; the pane at the bottom of the window (resizable by dragging the textured "grabber" in the middle of the separator bar) shows how many items were found. If you have exactly one found item selected, the pane switches to show the hierarchical path to the item. If multiple or no items are selected, the pane shows the found item count again.

File Names O Cont	tents 💽	Custom	€ Edit		
Name	Relevance	Kind	Date Modified	Size	
🖥 QuickDraw™ 3D Read Me		SimpleText read-only	3/3/98, 2:00 PM	28K	
Mac OS Tutorial Read Me		SimpleText read-only	9/9/98, 2:00 PM	8K	
🔋 QuickTime READ ME		SimpleText read-only	9/13/99, 12:0	32K	
📑 iMac Read Me		SimpleText read-only	4/13/99,2:00	24K	
Map Control Panel Read Me		SimpleText read-only	7/14/98, 12:0	8K	
🖥 Remote Access Read Me		SimpleText read-only	3/11/99,2:00	20K	
Read Me		SimpleText read-only	5/22/98,2:00	20K	
📅 Read Me		SimpleText read-only	12/8/98, 12:0	20K	
🖥 Read Me - AppleWorks 5		SimpleText read-only	8/28/98,2:00	20K	
					•
					1000
ems Found: 19					

Figure 3—File Finding Results

Merging Attributes and Content

The "More Search Options" dialog box contains fields where you can enter text to find in a file's name or in file contents. "Find File" had this option as well, hidden in the pop-up menus unless you pressed Option while clicking. When you chose "contents" in the first tab, "Find File" did a brute-force search through the contents of every target file that matched other criteria. The pace was tortoise-like, and the results usually not particularly useful. Sherlock removed the option entirely in favor of separate indexbased content searches, but that new functional division no longer allowed searching for files with *both* arbitrary attributes *and* specific content.

Sherlock 2 remedies this deficiency. The "Files" channel is a merger in more than human interface: the "Contents" option in the "More Search Options" dialog box uses the Find By Content indexes. For the first time, you can efficiently search your disks for files containing arbitrary attributes and specific content. Sherlock gave no easy way to find all AppleWorks files containing "1999 Budget" or all files ending with ".html" modified this month containing a given URL. Sherlock 2 has no trouble with such requests.

In various promotional materials, Apple Computer has said that Sherlock 2 indexing is faster and more robust, but such claims are extremely hard to verify and no details about why they might be true are yet available. As with Sherlock, indexing can be frustrating: on hard drives of any size at all (even as small as 4GB), the progress window may estimate indexing to require more than 24 hours to complete. You might as well forget about indexing a drive with more than 6GB or so of files unless you have a long vacation coming up-unless you apply Macintosh skills to the problem. The solution? "Get Info" on the "Find by Content Indexing" application (located in the "Find" folder of your Extensions folder) and give it huge amounts of memory. Indexing requires building large data structures, but the indexer seems to use no RAM outside its own partition, even though it's often the only program doing active work when running. The change can be astounding-providing the program with all the RAM it needs typically reduces estimates for bottom-up indexing by twelvefold. This works with either Sherlock or Sherlock 2 and isn't publicized nearly enough. If you've already done this, don't forget to do it again after installing Mac OS 9, as the new version comes with a default partition of just 2000K.

Unfinished Business

Sherlock 2's file-oriented improvements don't mesh with Apple's "Internet Power Tools" marketing

theme, so the company isn't emphasizing them much. That's a shame, because these improvements are among our favorites in all of Mac OS 9. The merged features are logical and consistent, and the combination of indexed content searches with the traditional "Find File" power is a winner. Sherlock's restrictive choice of searching by content on either entire volumes or a single Finder selection was pretty useless; the new drag-and-drop folders options are the way it should be done.

The "Files" channel needs just two things to make us drop all our other search utilities, and Apple can only implement one of them. Saved Search Criteria behave differently depending on where you save them. Those in the "Files" folder in the "Internet Search Sites" special folder are templates—when you pick one from the pop-up menu, it fills in the "More Search Options" criteria but leaves your volume and folder selections untouched. If you double-click a file saved from "More Search Options," Sherlock 2 warns you that you need to complete your search selections. On the other hand, Search Criteria you save with the "Save..." command include the volume selections in effect when you saved the search. But Sherlock 2 doesn't respect those settings if you drop the document into the "Files" channel folder for use in the pop-up menu, and double-clicking such documents opens a new window and immediately starts searching. What's more, Sherlock 2 still has no concept of stationery.

In other words, Sherlock 2 still has no way to save a complete search template. If you regularly want to search the HTML files in your Web site folder for custom content, you can't automate it. The criteria in the "Files" channel don't include your choices of folders or non-disks, so you'd have to drag them into the window before each search. A saved Search Criteria document contains your choices of folders, but it also includes the text to search for and starts immediately—you'd have to double-click it, stop the search, then retype your text. You can't even do it with AppleScript—Sherlock 2 supports a "search" command that can find text in a file or use saved Search Criteria, but not both. (A third option, searching for files whose contents are similar to one or more documents, is also mutuality exclusive with specifying text or Search Criteria.) Sherlock 2 needs to save everything but the text you want to find in Search Criteria.

Apple can fix that, but they can't fix the problem of context. Once you find a file based on its contents, you typically want to open the document and jump right to the found text. Sherlock introduced a new parameter to the standard "open document" Apple event allowing this—the extra data tells the program what text you were searching for, so the program can

Sherlock 2 The Next Level

(continued from page 9)

then do the right thing (like finding the first instance of any word, or finding them all as closely together as possible). It's slightly complicated—Find By Content's indexing uses partial matches, so low-relevance results may not contain any of the words in the original search criteria. (Search a virgin Mac OS 9 disk for "http://www.microsoft.com" in the content of files and Sherlock 2 returns the Internet Location file for *Apple*'s Web site from the Favorites folder because it contains "http://www" and not much else, making it a relevant partial match. Yow.)

In the year since Sherlock hit the streets, far too few applications have changed to take advantage of this extra information. MVP Solutions's Retrieve It is showing its age, but when you double-click on a file it found, it opens a preview window and searches for the first instance of any search term, continuing through the document each time you press Return. Adobe Acrobat comes with its own Search plug-in for the indexes created by Acrobat Catalog; opening an item from the Search plug-in's results window opens the PDF file to the highlighted first instance of a search term. Acrobat could use similar logic to find the search terms in a file opened through Sherlock or Sherlock 2, but it doesn't-once you get it open in Acrobat, you then have to find what Sherlock thinks was relevant.

From SimpleText to Nisus Writer to Acrobat to BBEdit, no major application takes this information and works with it. If they did, Sherlock 2 would be far more useful—not only would it find files by content, but would also show you what it found through a simple double-click. Without that support, the aging Retrieve It remains more useful for finding text in large collections of files—the convenience of moving right to the information you want outweighs its bugs, clunky interface, and slow searching through nonindexed text. Apple can't solve this problem except by evangelizing developers to do the right thing, but they can make templates work. A combination of those two features, we believe, would obsolete every other Macbased search utility we've seen.

Even so, Sherlock 2's merged file finding capabilities are superb. Searching the insides and outsides of files are no longer separate tasks kept apart by separate code development paths. Eliminating the distinction was logical; the implementation is clear and consistent, and the results are outstanding. For those of us with tens of thousands of files, it could be *the* sleeper story of Mac OS 9.

Internet Searching: New Channels Galore

The genius of Sherlock's original design was the "Search Internet" feature. It wasn't the idea of a metasearch engine, as Web sites that collect the results of dozens of major search engines have existed for years. Neither was it Apple's repeated—and bogus—claim that Sherlock parses "plain English" to find your results: the published specifications make it quite clear that Sherlock passes the text you type in the main window to its plug-ins without a whit of parsing or filtering. It just happens that most major search engines can handle natural-language queries, or at least ignore the common words that might derail such a search. Nor was it even relevance-ranked results: Sherlock does "relevance ranking" only because the Web search engines return their idea of relevance with their results; Sherlock just extracts the numbers and sorts them.

The magic came from the simple text-only design of the plug-ins. They contain no code, so automatic retrieval from the Web and installation onto your system invokes no security issues. The plug-ins have no access to anything on your system other than the text you type into Sherlock's main window, so no privacy problems arise. The concept's utter simplicity amazed us then and now.

A quick review: Every Web search engine automatically returns HTML results using regular, predefined patterns. Within those results, found occurrences, URLs, and relevances are bracketed with predictable text strings. Those patterns form a recognizable structure; knowing and understanding that structure allows you to work with the items in flexible ways. That's exactly what a Sherlock plug-in does: mark a search engine's results, allowing Sherlock to interpret them and display them in a list in a uniform format. We can't believe no one thought of it earlier. (*MWJ* 1998.10.19)

But the implementation wasn't perfect. Sherlock's original "Search Internet" feature used a fixed-size window that quickly constrained those using dozens (or hundreds) of plug-ins. Mac OS 8.6 revised this, allowing the window to grow and adding an "Uncheck All" button to more easily allow starting a search with just a few sites. Long before that, however, third-party developers attacked the need for more intelligent plug-in management. Some provided AppleScript support for moving sets of plug-ins in and out of the "Internet Search Sites" folder; Apple Computer published a few of these. Others wrote original code, with our favorite being Casady & Greene's Baker Street Assistant, renamed from "Sherlock Assistant" after a letter from Apple's legal counsel. Most people clearly wanted the capability built into Sherlock.

Mac OS 9 Marquee Features

Page 11



Figure 4—Sherlock 2's Internet Channel

Channel Expansion

Taking a page from their own QuickTime playbook, Apple decided to look upon Sherlock 2 as some kind of Internet viewer, so they call their sets channels. Figure 4 shows the "Internet" channel in Sherlock 2, with several "built-in" Internet Search Sites and one (CNET's Download.com) "custom" site. Pressing Command-H brings you to this channel-even in the Finder, which has adopted Sherlock's "Search Internet" key equivalent for the same item in its own "File" menu. After you type the text you want to find on the Net, click the magnifying glass (or press Return or Enter) and Sherlock 2 goes out and searches as it did before. The list continues to show all results ranked by your choice of criteria; the pane below that shows a brief description of each item, and the space below that holds a banner advertisement. The main window animates when you switch between files and Internet searches-a small ad pane rises from the bottom, then the list and the ad pane both grow, squeezing the other pane smaller. You can grow the list, but the ad pane can't grow or shrink.

Sherlock 2 comes with seven pre-configured channels:

• The **Internet** channel has built-in plug-ins for AltaVista, CNET, Direct Hit, Excite, GoTo.com, HotBot, Infoseek, LookSmart, and Lycos

• The **People** channel contains new LDAP plug-ins for Bigfoot, Four11, and Yahoo (more on these new kinds of Internet Search Sites later in this article)

• The **Shopping** channel holds four built-in Amazon.com plug-ins (Auctions, Books, Music, and Video), plus Barnes & Noble and eBay (we'll also discuss Shopping plug-ins later)

(see "Sherlock 2 The Next Level," page 12)

Sherlock 2 The Next Level

(continued from page 11)

• The **News** channel holds the plug-ins for CNET News.com, CNN Interactive, ESPN, Motley Fool, and Quicken.com

• The **Apple** channel's built-in sites are for the Macintosh Products Guide, Apple Software Updates (English-North American, at least in the US release of Mac OS 9), the Apple Tech Info Library, and Apple.com itself

• The sparse **Reference** channel hosts Dictionary.com, Encyclopedia.com, and Roget's Thesaurus

• And a default empty channel named **"My Channel**" becomes your plug-in drop box

You might be tempted to refer to channels as "sets" of plug-ins. Resist that temptation. When Macintosh users think of "sets," they usually invoke an image of Conflict Catcher or Extensions Manager, where a "set" of files enables or disables a specific group out of a large set. Baker Street Assistant works that way, telling Sherlock which plug-ins to use in a search based on a set you choose in the add-on program. Channels aren't that way-once you select a channel, *only* the plug-ins in that channel are available for the search. You can enable or disable individual Internet Search Sites within that channel with each plug-in's checkbox, but you can't search all installed plug-ins at once. If you want to search both Amazon.com and AltaVista, you'll have to do some plug-in management on your own.

Channel Management

Fortunately, Sherlock 2 doesn't make this too difficult. When you drop a plug-in on the closed system folder, the Folder Manager and magic routing still direct it to the Internet Search Sites folder, be that in the System Folder or mapped to a user's folder with Multiple Users. Each channel, including the "Files" channel, has its own folder within the Internet Search Sites folder. Opening Sherlock 2 forces it to scan that folder, and it routes all new plug-ins to "My Channel." (You can change this if you're willing to hack on text files, as you'll see later in this article.)

Managing channels is easy—just grab a plug-in from any channel's list and drag it to another channel's icon well. Sherlock 2 does the work for you. Removing a site from a channel should be as easy as removing a folder from Sherlock 2's file-finding display (Command-Delete or "Move to Trash" from the "File" menu), but it's not—you have to drag the plugin from the list to the Trash icon on the desktop.

Making new channels is easy, too: pick "New Channel" from the "Channels" menu. You're asked to name the channel, pick one of four *types* (Searching, People, Shopping, or News), select an icon to represent the channel in the main window, and enter an optional description. That's it—your new channel is listed in the "Channels" menu and gets the next channel well. You can grow the window if there isn't enough room to see all your channels, or use the topmost "grabber" to slide the list downward to reveal more channel wells.

If you want to rearrange plug-ins manually, in the Finder, just copy the plug-ins from one channel folder to another. We say "copy" and not "move" deliberately: if you try to remove a "built-in" plug-in from its intended channel, Sherlock 2 notices it, connects to Apple's Web site, and replaces the plug-in where it should go. You'll save download time by copying the files you wish duplicated in other channels; you can't (easily) remove "built-in" plug-ins from their channels. ("My Channel" has no plug-ins by default, nor do any channels you create, so whack on them to your heart's content.)

Specialized Channels

Those four kinds of channels aren't just eye candy; a channel's type determines how Sherlock 2 displays the results of its plug-ins. Both the *searching* channel type (also called the Internet type in some contexts) and the *reference* type are like the Sherlock plugins Mac OS 8.5 users already know-they display columns for the item's name, its relevance, and its site (URL). A shopping channel, however, displays different columns: name, price, availability, and site. By default, shopping channels sort by price, so the cheapest items matching your query text (numerical value only-US\$1.40 sorts ahead of £1.50) rise to the top. A news channel has columns for name, relevance, date, and site; the default sort is by date so you get the most recent news first. The new *people* channels are the most different of all with three new columns: name (meaning a person's name), E-mail address, and phone number.

You can mismatch plug-ins—Sherlock 2 doesn't stop you from placing a news plug-in on a shopping channel, for example, but it serves little purpose. Display is based on channel type, not plug-in type: a news plug-in is unlikely to find "price" or "availability" in its results, so any matches for the query text will be mostly empty in a shopping channel's display. News and searching plug-ins and channels are close enough to mix well.

Mac OS 9 Marquee Features

Internet Search Site Changes

These new features require changes to the familiar Sherlock plug-in specification, but nothing radical. The basic gist of Internet Search Sites is unchanged: search engines return regularly-formatted pages and plug-ins describe how to find the content within them. They're still just text files.

The first addition is the *routeType* inside the SEARCH block. Sherlock 2 wants to route plug-in files to the appropriate channel, but the Folder Manager's magic routing would require a new file type to each kind of plug-in. That's far too unwieldy and it's not backwards compatible. Instead, Internet Search Sites have the same file type ('issp') as before and they're still routed to the same folder. As noted, Sherlock 2 then places the plug-ins with the appropriate channel the next time it's run. It does this by opening the file and reading the "routeType" field. If it's present, it should contain one of the six standard channel types: "internet", "people", "apple", "reference", "news", or "shopping". Any plug-in without one is routed to the "miscellaneous" channel—"My Channel."

The new columns in the new channel types require plug-ins to describe how to find that information. Sherlock 2 recognizes new delimiters to identify price, availability, date, and a name (in case Sherlock 2's default parsing isn't good enough for the "name" column). Plug-ins can also tell Sherlock 2 how to find the ISO language and country code for results; the program uses that to help interpret the "price" information where digits might use different text encodings.

Plug-ins have one additional option. Each Internet Search Site file has a block named "INTER-PRET" containing all the delimiters Sherlock 2 uses to interpret results. Sherlock only allows one of these blocks, but Sherlock 2 allows as many as a plug-in author wants to write. One set of delimiters per item might not work in all cases; multiple INTERPRET blocks allow the same plug-in to describe alternate ways a search engine might return its information to Sherlock 2.

That's it—other than these changes, plug-in files work just as before. All existing plug-in files are instantly compatible with Sherlock 2, with the caveat that, absent a routeType field, they'll all get put in "My Channel" when you drop them on the System Folder.

People Plug-Ins

Well, there's one exception to this. The new "people" plug-ins are substantially different because they're not based on HTML. Sherlock 2's people searching uses *LDAP*, the Internet-standard Lightweight Directory Access Protocol. Instead of parsing results returned as text through the HTTP protocol, Sherlock 2 takes your typed text and forms an LDAP URL, sending it to the specified LDAP server and waiting for results. LDAP returns pre-defined fields, kind of like a database, so "parsing" isn't necessary.

Sherlock 2 cares about LDAP fields that map to the person channel type: the cn field ("complete name"), the mail field (E-mail address), and the telephoneNumber field. A people plug-in needs only to form the right LDAP URL from the text you type and send it to the server; Sherlock 2 does the rest of the work. People plug-ins tend to be short. Before writing one, you should have a passing familiarity with LDAP and know the format and capabilities of the server you're targeting. If you need interactive help, try playing around with the LDAP URL in your Web browser. Both Netscape Communicator 4.7 and Microsoft Internet Explorer 4.5 have built-in LDAP clients that will format and display results from any valid LDAP URL, giving those writing people plug-ins a good way to debug their attempts. Hopefully Apple will release official people plug-in specifications soon-colleges around the world would love to provide this capability to faculty and students using existing LDAP servers. It's a wonderful idea.

Inside Sherlock Channels

So how do channels themselves work? Very similarly. Plug-ins are written in an HTML-like language called "Internet Search Interface Language," or ISIL. Apple has been kicking itself for this—the best way to go would have been XML. Unfortunately, ISIL isn't a subset of XML, and backwards compatibility doesn't allow Apple to change the rules now without breaking existing Sherlock plug-ins.

The company didn't repeat the mistake with channels. Apple Computer has released no official specifications for channels—and may not, because Sherlock 2 creates them for you on demand, reducing the need for external tools—but you can look at the existing ones and learn a great deal. The XML definition of each channel is found in the file "Channel Data" inside the channel's folder. Start small, with the data for a channel you created in Sherlock 2. There's not much data there: the file contains a single "channel" element with one attribute, a type of "internet". In XML syntax, it looks like this:

```
<channel type="internet">
</channel>
```

Now try one of Apple's default channels. More information, but nothing too complicated. Apple's

Sherlock 2 The Next Level

(continued from page 13)

channels contain a *routeType* field, just like plug-ins may now have, helping Sherlock 2 route plug-ins to the right channels. Each of Apple's channels also contains one *sourceUpdate* element for each plug-in the channel should have by default. The sourceUpdate element has two attributes: the plug-in's file name and the URL where the channel can find that plug-in. (In XML parlance, these elements contain no *content*, just attributes, so they're formatted like empty elements a slash appears before the closing angle bracket, eliminating the need for a closing "</sourceUpdate>" tag.)

This is how Sherlock 2 manages its magic "built-in" plug-ins. When the program opens a channel, it makes sure the folder contains plug-in files with all the specified file names. If any are missing, it uses the URL to go fetch the missing plug-in and install it in the channel. You can't get rid of the built-in plug-ins except by editing the "Channel Data" files themselves.

Copies of the default channels, by the way, are found in 'TEXT' resources inside Sherlock 2's resource fork—if you delete one of the default channels, Sherlock 2 warns you of your carelessness and recreates it. This works well for Multiple Users, as it turns out. Sherlock 2 doesn't deal with plug-in aliases, and Multiple Users doesn't want to copy every plug-in for every user you might create. So it creates an empty Internet Search Sites folder. When each user first opens Sherlock 2's "Search Internet" feature, Sherlock realizes there are no channels and no plug-ins, so it uses the default information to recreate all the channels and download all the plug-ins. Pretty nifty. (Owners can avoid the delays by manually copying the system's Internet Search Sites folder to each user folder.)

The channel element also has its own sourceUpdate attribute, containing a URL that points to the channel data file's home on the Internet. Just as with plug-ins, Sherlock 2 checks channels at specified intervals for updates. If the channel file on the Net is newer, the program downloads and installs it. Consequently, if that newer channel file specifies new plug-ins that you don't have, Sherlock 2 uses the URLs of the plug-ins in the channel file to find them and install them. *Voila!* Self-programming channels (and a reason to be careful about modifying Apple's channel files—newer versions will wipe out your hand-entered changes).

Without a formal definition, it's impossible to say if Sherlock 2 recognizes other elements or attributes in channels, but just understanding this much shows how to make your own self-repairing and self-updating channels. Apple Computer doesn't guarantee any of this, but we know of no reason it wouldn't work. As always, proceed with caution—if you depend on undocumented features like channels, be prepared to switch strategies on a dime if necessary.

Advertising Problems

Unfortunately, not all is well with Sherlock 2's Internet philosophy. Geoff Duncan noticed in *TidBITS* about Sherlock 2 advertising. Duncan discovered that Sherlock 2 does not display banner advertisements from any search sites except those that come with Mac OS 9. He notes that many were appalled that Sherlock allowed sexually-explicit banner ads to pop up on screens that might be in use by children, and Sherlock 2 fixes that problem. "I don't know whether Apple made this change to address issues of objectionable content or whether it simply regards Sherlock's banner area as prime advertising space available only to partners. In any case, Sherlock does not display banner graphics from other sites, instead substituting an Apple banner. This move may help Apple in schools and homes, but may dissuade many sites from developing or supporting Sherlock plug-ins. After all, such sites' banner advertising apparently won't be displayed-even if it's perfectly innocuous-unless they can somehow become an 'approved' site. This could reduce Sherlock's Internet searching capability to a mere bundling opportunity for large Internet services and retailers." After reading this, we confirmed it on our own system.

Our requests to Apple for comment on this issue went unanswered. As advertising-averse as we are, we *do* think this is a big issue.

First, the technical details. You can reverse this procedure on your own system. The key is the "Sherlock Default" file, stored in the "Sherlock Prefs" folder of your Preferences folder (or of any user's Preferences folder if Multiple Users is active). Despite its preferences file type, this is yet another text file, using Apple's Internet Search Interface Language. After some housekeeping information, you'll see a long list of XML-like empty "type1Source" elements, each with a "name" attribute corresponding to an installed Sherlock plug-in. If you add your favorite site as a "type1Source," Sherlock 2 will display that plugin's advertising banners. We do not know what the similar "type3Source" attributes mean. We know that internally, Sherlock also understands a "type2Source," but we're not sure how Sherlock 2 interprets that, either.

If you change the file, you might want to change the "update" interval at the top of the file. Otherwise, if Apple updates the "Sherlock Default" file (to include more "type1Source" sites, perhaps), Sherlock 2 will download a new version and overwrite your changes. If you lose the file altogether, Sherlock 2 recreates it from defaults stored in its 'TEXT' resource of ID 1000.

The problem, of course, is that you don't want to update your Sherlock files to add banner advertisements. Plug-in developers want to do so, and to our knowledge, they can't without asking you to replace the "Sherlock Default" file. (Apple may have built in a hidden tag in plug-ins that says "this is an approved plug-in," but we haven't figured out what it is if it exists.) Web sites use the banner ad revenue from search sites to help pay for the search service; this was Sherlock's great compromise. By displaying each site's banner ads, the Webmasters had little to fear from Sherlock's use of their bandwidth (except that, with some engines, Sherlock didn't show you the other 57 ads placed on each page). If Sherlock 2 uses that search engine but fails to pass through ads that generate revenue, the Webmasters are better served by forcing you to their site in your Web browser. Including Apple's own ads instead of the ones that pay for the server just adds insult to injury.

Web sites *can* block Sherlock 2 from their servers, at the expense of blocking the entire URL Access mechanism Sherlock uses. The "User-Agent" field in the HTTP protocol header sent to each searched site identifies the requestor as "URL_Access/2.0 (Macintosh; PPC)". It's not difficult for most heavy-duty sites to refuse requests from such agents, locking you, Sherlock, and even the Network Browser out of those servers from that point onward. We predict it won't take long for search engines to start doing this if the demand on their servers doesn't produce the advertising exposures necessary to pay for the bandwidth and equipment time.

Sherlock wins over tools like Copernic (*MWJ* 1999.07.03) because non-programmers can create plug-ins with little difficulty, and because the cooperation in advertising banners gives Webmasters little reason to be threatened. Sherlock 2 doesn't merely upset this delicate balance, it destroys it. If Apple isn't careful, Sherlock 2 may wind up even less useful than Copernic because only the approved Web sites will allow searches. That's not what people expect from an "upgrade."

It hasn't happened yet, but we see it on the horizon if Apple does not resolve this issue carefully.

AppleScript Support

The new features in Sherlock 2 required changes in the program's scripting dictionary—what fun would it be to have new features you couldn't automate? The major addition, of course, is channels. Sherlock 2's AppleScript dictionary revises most of the Internet verbs to take optional channel parameters. The "search Internet" and "select search site" verbs both take an optional "in channel" parameter: a string with the name of the channel to use. If you omit a channel, Sherlock 2 defaults to the "Internet" channel, even if another one is selected in the application's main window.

The dictionary adds *channel* as a class, and the normal AppleScript verbs "count", "exists", "get" and "set" all work. Both the channel and the application have an "all search sites" property, but unfortunately neither has a "site" property. You can get all search sites in a given channel, but you can't get "site 2" or "first site" of any channel. Sherlock 2 allows changing the current channel, but not changing any channel contents—sites and names are read-only. You can't create new channels or manage existing ones through AppleScript, save that you can count the sites in a channel and get their names.

Existing verbs from Sherlock still work, but the behavior of some has changed. Sherlock allows you to script an Internet search with a supplied list of sites to search. Sherlock 2 does the same, but ignores any sites in the list that aren't in the current (or specified) channel. The "current tab" property from Sherlock (to change the main window to Find File, Find By Content, or Search Internet) still works, but the appropriate Sherlock 2 interface appears instead. Setting the current tab to "Find By Content" opens the "Files" channel and selects the "content" radio button (as does pressing Command-G, just as it did in Sherlock).

Sherlock 2 also adds a new AppleScript command to index one or more containers in case your hard drive is too large and ponderous to maintain a drivewide index (like Sherlock, Sherlock 2 lacks this option in its human interface, although you can still do it by Control-clicking items in the Finder). Sherlock 2 allows automated indexing of just those folders you wish indexed. Your content-based searches will obviously be restricted to those folders, but that's probably what you want if you pursue this path.

Interface Inanities

All in all, Sherlock 2 is a powerful and logical evolution of the meta-search utility. Yet we can almost guarantee loud complaints from dissatisfied users because of a key Apple blunder—the human interface. Despite vocal complaints from Macintosh owners and human factors designers, Apple has foisted the "QuickTime 4 Player" brushed-metal interface on Sherlock 2. It is a mistake. If the program were not so useful, it might have been a fatal one.

As with QuickTime 4, Sherlock 2's windows have lots of gee-whiz animations and interesting visual

Sherlock 2 The Next Level

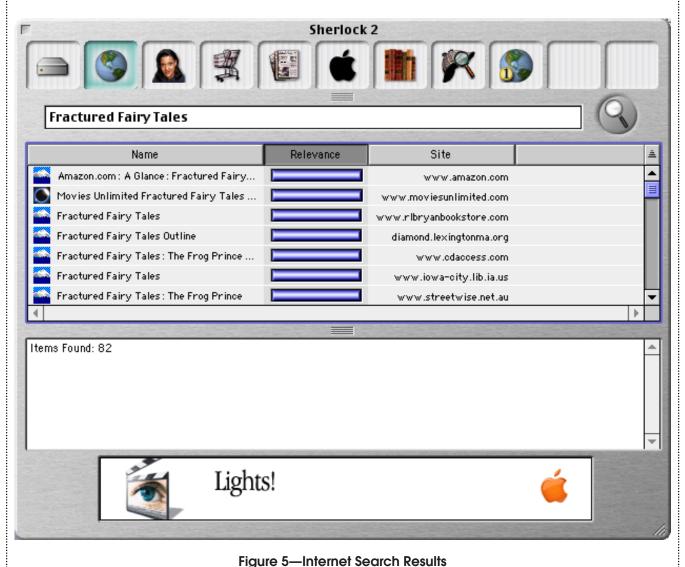
(continued from page 15)

effects, but are lacking in standard useful elements. A real title bar and a collapse box are both missing, so you can't collapse the window. The absent zoom box means resizing the window requires growing it manually via the grow box in the lower-right corner. All "brushed metal" areas are part of the window's drag region, so it moves under your mouse when you don't expect it.

Our nomination for "boneheaded interface move of the year" is part of the "brushed metal" interface. Sherlock 2 (and Sherlock) disable most functions *during* a search, forcing you to stop the search before changing anything or choosing menu commands. Fair enough—the options for stopping are standard and easy to find. However, the text field with search terms is one of those areas that becomes disabled while a search is in progress. That's also fair. What's dumb is that in the "brushed metal" interface, the disabled text looks *no different* than regular text, and the disabled field invisibly becomes part of the window's drag region.

Stopping is not instantaneous—it make take several seconds to abort an Internet search. During that delay, everything is still disabled. If you don't notice this, you might try to change your search parameters by clicking and dragging in the search terms field. Oops—it's still disabled, though it looks no different. And like other non-content areas, it's part of the window's drag region, so you've just picked up your window and moved it halfway across the screen.

This is unfortunately the major feature of the QuickTime 4 interface—trendy appearance triumphing over the familiar ways of working with a Macintosh we've all learned over the years. Nothing prohibits this program from having a standard human interface—or at the *very* least use an Appearance Manager theme that's visually exciting while maintaining the behaviors



all Macintosh owners expect. Clicking and dragging in disabled text fields should *not* move the window. That Apple would permit such shallowness in Mac OS 9's number-one feature is truly depressing.

Sherlock 2 takes another interesting departure from Sherlock-results of all searches display in the original search window. Sherlock displayed only one "Find" window at a time but permitted as many results windows as RAM allowed. Sherlock 2 can display as many windows as you like (up to RAM limits), but search information and results stay in the same window. The consequences are subtle but important. In Sherlock, starting a new search lost the original search terms, but they were summarized for you under the results list in the results window (as well as in the results window title). With Sherlock 2, the results replace the search criteria. When finding files, the results replace both the list of searched items and the "sentence-like" description of the search terms. For Internet searches, the terms remain visible in the text field but the results replace the list of sites. Figure 5 shows the results of an Internet search.

In either case, the only way to recover your original search terms is to double-click on the current channel. That replaces the results with your searching parameters—but then you have to perform the search again if you want to see the results. With Sherlock, you *could* see both the search inputs and outputs at once if you wanted. It's not possible in Sherlock 2.

This change is not related to the QuickTime 4 silliness. The single-window metaphor would work exactly the same with standard windows. We see the conceptual advantage in keeping searches and results in a single window, but it ironically makes less informa-

tion visible than the older method. Trading that for a slight conceptual advantage doesn't seem worth it. The QuickTime 4 crap *certainly* isn't worth it. It should either be retired or made optional as soon as possible.

Sherlock 2 Thrives

The most annoying aspect of the new interface's shortcomings is that Apple Computer may not get the message. Sherlock 2 is a great improvement on a great feature. People will use it, people will love it—and Apple may therefore incorrectly think the interface isn't an issue. We think Sherlock 2 will succeed not because of its interface, but in spite of it.

Sherlock 2 is everything we'd hope for from an evolutionary upgrade to Sherlock. Its strengths are so robust that the few holes it still has look even more glaring, like small splotches of red paint on an otherwise pristine canvas. Breaking down the wall between "Find File" and "Find By Content" makes both features immeasurably more useful. Logically distinct, selfrepairing and updating channels are inspired, and will be even more so when developers can provide their own with full support. People, shopping, and news plug-ins logically extend a great idea and make it even better.

Sherlock 2 will be part of your daily information arsenal regardless of the QuickTime 4 appearance. Lots of people still use MoviePlayer 3 instead of QuickTime 4 Player to avoid the insipid human factors choices, but we don't see anyone sticking with Sherlock instead of Sherlock 2. The best testimonial the program could hope for is true: it overcomes its own defective interface.

Mac OS 9

Sharing Your Personal Space Making One Macintosh Personal and Secure for Several

Apple has a large customer base in education, particularly K-12 schools, and it's rare to find a school at which every child has his or her own computer everywhere one is needed. The same is now often true of home computers as well—as kids and adults become more computer-literate, the home computer is not the sole domain of one family member who performs tasks on the machine for the rest of his unanointed household. Offices are a different story, but often many people share school and home computers, two of Apple's key current markets. This has always been something of a problem for the customizable Macintosh. You find the perfect desktop picture, the perfect system font, the perfect set of obnoxious sounds from *Monty Python's* latest. You position all your windows exactly perfectly, with the tabs on the pop-up windows aligned *just* so and the window positions of your five busiest folders neatly tiling when they're all open. You go to bed quite satisfied with your impending increases in personal pro-

Sharing Your Personal Space

(continued from page 17)

ductivity—and get up the next morning just to discover that your eight-year-old daughter got up first, replaced the desktop picture with Pokemon, moved all your windows, and reset the resolution to play Nanosaur. Before you can even say a word, she says, "Eww, daddy, you should have seen the ugly things someone did to your computer, but I fixed it for you!" So much for *that* head of steam.

Three of Apple's nine "Internet Power Tools"— Multiple Users, Voice Verification, and Apple File Security—although having little to do with the Internet itself, can help you share your computing experience in a much narrower sense by making it easier and more secure for other persons to use your Macintosh. A fourth new but unhyped feature—file signing—combines with Apple File Security to provide Mac OS 9's beefed-up security capabilities. Let's look at how Apple has made it easier for even the stingiest and most secretive among us to share and share alike.

A Shared History

Apple has been trying to address this problem for some time in schools through optional add-on software like At Ease and Macintosh Manager. A workgroup leader (called a "teacher" by people who aren't writing computer documentation) sets up individual users (students) and workgroups (classes or study groups) on a centrally-administrated server. When the kids sit down at a Macintosh, they boot into a login program where they identify themselves. The server then provides their applications, files, and some personal preferences to whichever computer they happened to pick. The students don't have to save their files on floppy disks or sit down at the same computer every time to get to their work on the hard drive. In fact, teachers can prohibit students from writing to or reading from the hard drive. Teachers can also force new documents into a specific folder on the server, make "drop boxes" for submitting homework (you can put a file in but can't take it back out), limit which applications students can run, and even control whether students can see removable media like floppy disks or CD-ROMs, just to keep them from sticking in a disk from home with a malicious application on it and wreaking havoc.

At Ease was originally consumer-level software, included as a Finder replacement on Performa systems to make the system easier for beginning Macintosh users. The interface presented two big, friendly panels—one for applications, and one for documents, hiding the complexity of the big ugly Finder from novices. Of course, you can't *do* much with At Ease, but schools loved it—a way to keep kids from doing much other than what they were supposed to be doing was most welcome. Starting with version 3.0, At Ease pretty much morphed into "At Ease for Workgroups," where it continued to grow and improve, eventually adding such features as Web access and sophisticated user tracking—not only the file management described earlier, but also tools like printer quotas (so kids don't print 6,000 copies of "You Suck" in 255-point Charcoal), and applicationspecific preferences for each user.

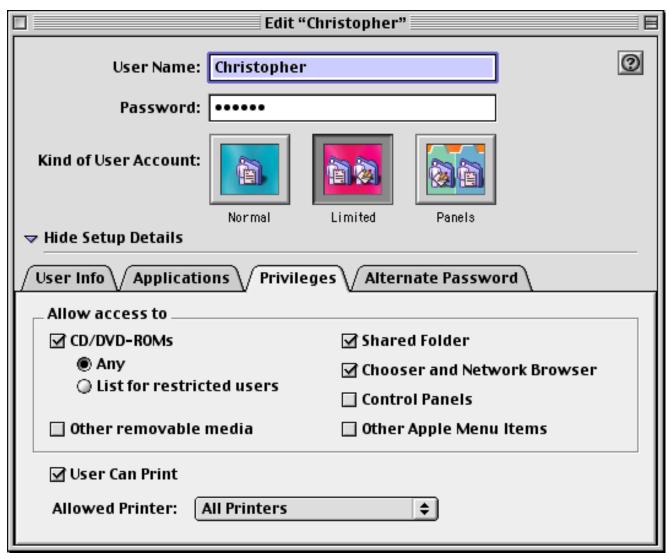
With Mac OS X Server, however, came a new challenge—NetBooting. At Ease wasn't up to this, so Apple replaced it with the Macintosh Manager. The newer program includes most, though not all, of At Ease 5.0's features, and works with both NetBoot and locally-booted computers. (Unfortunately, it removes At Ease's capability to prevent kids from *booting* a system from a local floppy or CD-ROM disk, a topic we'll come back to later.)

While the multiple-user capability of networked Macs has continued to evolve, the home version evaporated. At Ease's transition to a schools-only product left home-oriented Mac OS machines without much to soften the Finder learning curve. The "Launcher" control panel helped somewhat, as did the Mac OS 8 Finder additions of "Simple Finder" and button views, but that's been about it. At the same time, Apple has clearly been moving towards including functionality in the Mac OS that was previously provided by add-on software, from language kits to menu bar clocks.

Making It Personal

Enter the *Multiple Users Control Panel*, one of Apple's "Internet Power Tools" and the successor to both At Ease and the Macintosh Manager. Multiple Users stores preferences and restrictions for each of up to forty users on a single computer. When Multiple User Accounts are turned on in the Multiple Users Control Panel, the system changes in subtle but important and highly useful ways.

First and most noticeable is that system startup stops before you reach the Finder with a large "Welcome to Mac OS" dialog box. Depending on the global Multiple Users options, you may see a list of approved users with names and representative icons (the Control Panel comes with several you can use, or you can paste in your own) and three options: shutdown, change password, or log in. If the list isn't present, you see a simple text field where you must type a user name. Once you type or choose the right user name, you're asked for a password. Typing it is always an option, but another choice is one of Apple's other "Nine Internet Power Tools" that we'll get to shortly. If you mistype your password, the dialog box shakes





from left to right as if it's shaking its head "no," a cute interface gimmick lifted from NextStep.

If you log in as the owner, the Macintosh is the same as it always was, with one exception-the bottom of the "Special" menu now contains a "Logout" item with key equivalent Command-Q. If you log out, you must first confirm your intent (including acknowledging that logging out always empties your trash, a warning you see only if there's anything in the trash). Multiple Users then sends 'quit' Apple events to every visible application on the system, repeated every few seconds until they actually quit. (There's no way to stop this; once you say you want to log out, you're going to log out.) Faceless background applications continue to work. When all visible applications have quit, the Finder goes away as well, and you're returned to the "Welcome to Mac OS" dialog box. You can log in again, or pick a different user, or shut down the computer (including a dialog box that gives you fifteen seconds to cancel or shut down before it shuts down for you). Pressing the Power key or button gives you the option to restart, cancel, or shut down.

Three Degrees Of Freedom

Things aren't much different than before for the owner. Other users, however, are now grouped into three new categories, with the system now working a little differently for each. The owner specifies what privileges other users have on the computer, mostly by assigning them to one of three *user environments*:

Normal Users

A *normal* user sees the system much as the owner does, with a few notable differences. The desktop con-

Sharing Your Personal Space

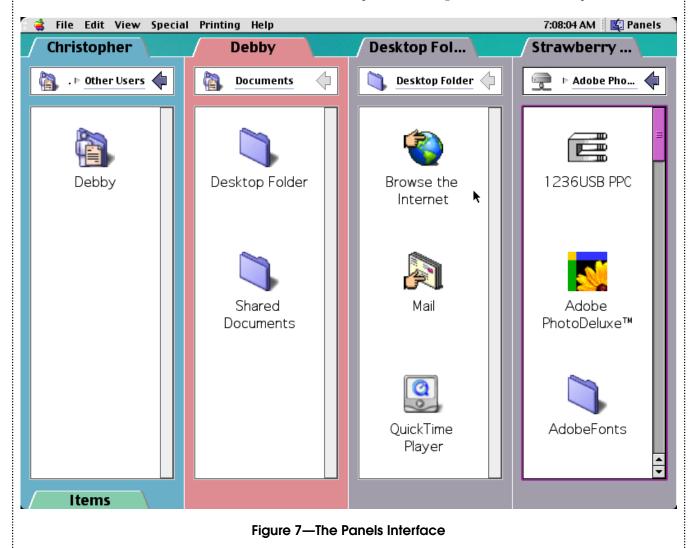
(continued from page 19)

tains an alias to a folder titled after that user name, and by default all of the normal user's documents are saved in that folder; it's the starting place for both Standard File and Navigation Services. There's a similar folder for every user who is not the owner. (We'll explore another folder, entitled "Items for" that user name, presently.)

By default, normal users cannot access any documents belonging to other users—though the owner can explicitly permit normal users to read, write, or read and write documents in other user folders. You need to read the checkbox allowing this carefully, though—when you check it for a given user, it permits *other* users access to *that* user's information, not the other way around. You can't make a normal user into a "super-normal user" who can see into anyone else's documents; you can only grant permissions for others to see into any given user's folder. They're called "normal" users because most things seem normal, but subtle restrictions are in place. Only the owner can access certain control panels that affect some system-wide settings. The list isn't documented, but it includes all Open Transport Control Panels as well as the Memory Control Panel. Those that affect how *you* use the computer are accessible—including Web Sharing, Date & Time, Keyboard, Mouse, and even Startup Disk. The owner can even let a normal user manage the Multiple Users Control Panel, allowing that person to change anything about any user (including decreasing his or her own privileges) except the owner. Only the owner may change the owner's user information, a logical precaution.

Limited Users

A *limited* user also logs into a familiar Finder environment but is not allowed access to many parts of the system. Figure 6 shows the setup panel for a limited user with user name "Christopher." The options are quite restricting: limited users can be prohibited from



accessing CD-ROM or DVD-ROM discs (or *all* removable media). They can be kept out of the Chooser and the Network Browser (improved in Mac OS 9), out of all Control Panels, out of all the other Apple Menu Items, and even out of the "Shared Folder," a special folder all users can read and write to so they can share documents without running into access problems. Limited users can be prevented from printing, also, or at least restricted to specific desktop printers (the restriction doesn't apply to printer drivers that don't create desktop printers, giving printer driver authors yet another nudge to update to the specification Apple released in February 1997 and implemented in Mac OS 8.5, *MWJ* 1998.08.17).

The "Applications" tab holds a scrolling list of items (each with checkboxes) that, by default, contains every application available on the system. A limited user can only open checked items-any application not checked in the list is visible, but if you're a limited user and you try to open it, the Finder tells you it "could not be opened, because you do not have enough access privileges." It doesn't suggest where you could get more. An "Add Other ... " button allows adding non-application items to the list, like Control Panels or documents. The Multiple Users Control Panel has built-in capability to screen out AppleScript applications so you don't have to walk the list and uncheck all of them. Aliases to all of the allowed items are in the "Items for " user name folder, an alias to which is maintained on the desktop for each normal or limited user.

Unfortunately, we found potential holes in the application screening feature. If a limited user tries to open a document belonging to a disallowed application, he's usually prohibited by the Finder. However, even though we tried disallowing SimpleText for a limited user, the program still launched for that user normally. Other disallowed programs behaved as they should have; perhaps Mac OS 9 exempts SimpleText from the restriction. Use caution, as we couldn't test all programs, and since we don't know why the screening fails on SimpleText, it may for other programs too.

The other serious restriction for limited users is that they can't save files outside of their own folders. They can read files anywhere, but they can only write files within the folder bearing their user name (or, if allowed, within the Shared Folder that all users may access). This prevents common Mac vandalism—trashing folders, making dozens of copies of large folders, throwing away your sister's science project, and so forth. Mounted server volumes are unaffected by this restriction, but AppleShare volumes have their own comprehensive set of access privileges; Multiple Users does not interfere with that, except that limited users may not open applications on servers unless they're explicitly permitted.

Panels Users

A *panels* user is the most restricted of all. The interface, as shown in Figure 7, is a combination of the former At Ease display and the NeXT File Browser—a group of very large panes with very large icons and text in them, thankfully configurable to be somewhat smaller in case you're not disablingly nearsighted. If you open a folder in one of the panes, a new pane opens up alongside the others. There is no overlapping; our iMac screen held four panes before extra ones started collapsing to tabs at the bottom of the screen. The top of each pane holds a left-pointing arrow that's darkened (enabled) if you can click on it to go "back" to the previous view. Like the Finder, each pane has separate view options, explaining why the "Items" pane uses smaller icons with Geneva 12 text while the other panes use very large icons with very large text. (The preferences include an option to "smooth" very large icons, since the icons themselves aren't that large and would otherwise look chunky at such magnification.)

This works in classic "At Ease" fashion by replacing the Finder. As you see in the top right of Figure 7, the application name is "Panels." Since there's no Finder behind the scenes, a panels user can't poke around and see inside areas where he's not permitted. If a folder or an alias to it is not within a panels user's personal folder, it's off-limits. Servers aren't that way, but owners can effectively eliminate this threat by removing access to the Chooser (and Network Browser) and to Apple Menu Items. That rules out logging on to servers or selecting "Recent Servers" alias items. To get to a server, an alias to the server would have to be in the panels user's personal folder.

Keeping Them All In Line

A machine's owner can also set global restrictions with the "Options" button in the Multiple Users Control Panel. There's an optional login message, and instead of choosing your user name from a list you may speak it if speech recognition is installed and turned on. There's a pop-up menu allowing alternate password selection; it currently contains only one item, as we'll see shortly. The owner can also automatically log out any user who is idle for a specified number of minutes, or just lock the screen to keep prying eyes away. CD-ROM and DVD-ROM content can be monitored on a disc-by-disc basis, with some discs completely disallowed, and others allowed but restricted to pre-determined content (as with limited and panels users, the owner specifies which content on the disk is allowed; the rest is disallowed). The owner may allow a

Sharing Your Personal Space

(continued from page 21)

guest account with no password (as a normal, limited, or panels user). The owner also determines whether users type their names or pick from a list, with the former being less convenient but more secure. Mac OS 9 can look for user information either from Multiple Users on the local machine, or from Macintosh Manager on a Mac OS X Server system on the local network. And finally, but not insignificantly, the owner can choose to be notified every time a user installs a new application so he may set access privileges appropriately.

Behind The Scenes

This probably seems like a giant kludge—folders vanishing and reappearing, applications not launching, and customized preferences being switched in and out. The latter, in particular, sounds like a ton of swapping files around—not an encouraging thought, because a crash during the middle of a swap could leave the system pretty thrashed. But in fact, there's not much of that going on—even less than with Macintosh Manager on NetBoot clients. Instead, Multiple Users relies on a System 7-level capability of the Mac OS, grabbing control of it behind the scenes in a way that makes the feature both less kludgy and extremely powerful.

Folding the Disk Space Continuum

The *Folder Manager* was originally a simple piece of Macintosh software that mapped values representing the locations of "special" folders (the System Folder, the Extensions folder, the Desktop folder, and so on) to their actual locations on disk. Introduced in System 7, the Folder Manager started out very simply—it used a resource of type 'fld#' to map the values to the names of folders within the System folder. That was enhanced in Mac OS 8 to allow tracked folders within other tracked folders (so, for example, the Folder Manager can track "Location Manager Modules" within the "Extensions" folder).

The Folder Manager originally served to track these folders across languages (the "Extensions" folder isn't spelled that way in Icelandic). Mac OS 8 also added a way for applications or extensions to add their own "special" folders to the list (*MDJ* 1997.07.24). Mac OS 8.5 expanded the list of pre-defined "special" folders by about 20, and added a mechanism for software to tell the Folder Manager (and the Finder) that specific files should be "auto-routed" to special folders. Before Mac OS 8.5, only the Finder decided whether or not to divert files dropped on the System Folder icon to subfolders within the System Folder, much less which files went to which folders. With Mac OS 8.5 and later, other software can add to or modify this routing list (*MWJ* 1998.03.23, 1998.10.19).

The clear trend is towards making the Folder Manager more comprehensive and more useful. In the past two years it's expanded far beyond simple language translations of common folder names into becoming a system-wide repository for folders that multiple programs might like to find and for more complete organization of files within the System Folder. With Multiple Users, however, the Folder Manager takes bold steps in a direction it had previously only inched—*redirecting* those folders on a currently running system.



Figure 8—The User Folder As Owner Sees It



Figure 9—The User Folder As That User Sees It

The original point of the Folder Manager was to prevent applications from hard-coding the locations of important folders. Now that eight years have passed and applications have taken this to heart, Multiple Users can take advantage of it. An Installer application might believe it's just asking for the location of the Apple Menu Items folder so it doesn't have to worry about foreign languages. Multiple Users goes one step further, making the Folder Manager return a *different* folder dependent on which user is logged in.

Private Folders, Private Files

Nothing really changes for the owner-all of the special folders are where you'd expect to find them. But for all other classes of users, Multiple Users redirects the Folder Manager to point at similarly named folders inside the user folder. Figure 8 shows the owner's view of a normal user's folder, and Figure 9 shows the same folder as seen by that logged-in normal user. Note the contrasts: Figure 9 shows several standard folders with their "special folder" icons; Figure 8 shows the same folders, but without the icons. The system is actually using these folders in the user folder as the special folders when that user is logged in. The Finder uses the Folder Manager to determine which folders are "special," and it only draws the icons for the tracked folders. The system isn't copying files back and forth, nor is it somehow hacking the System Folder to make it look right. Multiple Users has simply told the Folder Manager that while this user is logged in, these folders *are* the special folders.

It works in every significant way-if you drag a file to the System Folder that should go into the Apple Menu Items folder, it gets routed to the active Apple Menu Items folder in the user's folder, not to the "real" one in the System Folder. When logged in as a normal user, many of the folders in the System Folder no longer have the special icons because they're not the active special folders. Note, however, that it's not true for every special folder-those that affect the entire system remain mapped to the real System Folder and therefore inaccessible to limited and panels users. Unmapped special folders include Control Panels, Extensions (including the "disabled" variants), Appearance, ColorSync Profiles, Contextual Menu Items, Control Strip Modules, Fonts, Language & Region Support, Scripting Additions, Scripts, and Text Encodings.

More important are all the folders that *are* mapped—Apple Menu Items, PrintMonitor Documents, Desktop Pictures (not the entire Appearance folder, but the visually important part), Documents, Favorites, Internet Search Sites, Launcher Items, Preferences, Startup Items, and Shutdown Items. You see one item in Figure 9 that's not in

Figure 8—the Desktop Folder. Since the Finder asks the Folder Manager where the Desktop Folder is, Multiple Users can redirect it as well, giving every user his or her own customized desktop. The normal user's desktop folder is itself invisible to that user when logged in, just like your desktop folder is invisible to you now.

Making It Work

Multiple Users changes some other portions of the Mac OS aside from the Folder Manager to make things work as expected. It patches into the File Manager, the part of the Mac OS responsible for accessing files on disks, so limited users and panels users can't write to any folders except their user folders. This works by copying the AppleShare security model. Multiple Users returns errors when programs try to write in folders where they don't have access. It doesn't return the classic "file is locked" or "disk is locked" result, however, because too many applications over the years simply throw up modal dialog boxes and tell you to unlock the file or disk-something you can't do. Instead, Multiple Users returns the more complex AppleShare access results, as most programs over the years have been revised to understand that there are read-only servers, that some folders may have different privileges than others, and that the user typically can't "fix" this.

The only strangeness is a paradox involving the "Application Support" folder, one added in Mac OS 8 for applications that need space on the startup disk (and intended to eliminate System Folder residents like "Eudora Folder", the "Claris" folder, the "MS Internet" and "MS Preference Panels" folders, and so on). Mac OS 8.5 made it more useful by making the Mac OS automatically search for missing shared libraries in the Application Support folder, allowing developers to move them out of the Extensions folder (MWJ 1998.10.19). AppleWorks, for one, now uses this folder as it should. The Application Support folder is supposed to hold files that support the application across all users (dictionaries, plug-ins, and so forth), so it's not redirected for each user. But applications obviously need write permission to this folder, and limited and panels users don't get write permission to any folders that aren't redirected to their user folder.

Apple's solution is to cheat—Multiple Users makes the File Manager pretend that limited and panels users can't write to the Application Support folder, and all code that checks for access will believe it (the Finder, Standard File, Navigation Services, Panels, and other relevant code all lists the folder as read-only) but if a program actually *tries* to write to the folder, it (continued from page 23)

works. This is the only exception, and is noted in Apple's Multiple Users developer documentation. Software can also fairly easily determine if Multiple Users is running and which user is logged in (including whether or not the current user is the owner), allowing applications to tailor error messages appropriately. For example, we can imagine a not-too-distant future in which installer programs accidentally enabled for limited or panels users present clear error messages: "Someone with more access privileges, like this computer's owner, needs to install this software. You do not have permission."

Login: Setup Tasks

Although most things just work with Multiple Users, there are some minor revisions to the Finder and other system components to present a more seamless access picture. Most of Multiple Users's setup is handled by "Login." Like the Finder or the new "Panels" application, Login is a system-level application—the one the Mac OS launches first (instead of the Finder) when Multiple Users features are active.

Login is responsible for making sure each user's folder matches the privileges granted in the Multiple Users Control Panel. For example, owners may restrict panels users and limited users from accessing "Other Apple Menu Items," meaning the owner's Apple Menu Items. Normal users and others without this restriction each have their own Apple Menu Items folders, but Login makes sure those user-specific folders also contain aliases to all of the items from the Apple Menu Items folder in the System Folder. If the owner adds the restriction later, Login makes sure all aliases to disallowed items are removed from the user-specific folders before those users get to start using the computer. (Individual items in each user's Apple Menu Items folder aren't affected by the restriction.)

As noted earlier, the owner controls whether any user's folder is visible to other users, and with what permissions; the default is to disallow such access. Each user's folder has the user name as the folder's name and is located in the "Users" folder at the root of the startup disk.

Suppose you have three users: "Christopher," "Debby," and "John," and suppose that Debby's folder allows the other users to see her documents, while both Christopher's and John's folders prohibit such viewing. When either Christopher or John log in, they'll see their own folders plus "Debby" at the root of the "Users" folder, as well as the "Shared Documents" folders all users have access to (unless restricted by the owner). What's more, their own "Documents" folders (in their namesake folders) will contain a new folder, titled "Other Users," with aliases to Debby's user folder. However, Christopher will *not* see the "John" folder, and vice versa. In fact, inaccessible user folders are not only read-protected, but also made invisible so other users aren't tempted to look or tinker.

The concept of Multiple Users requires *some* file shuffling, and Login handles those tasks—but mostly by managing aliases to existing documents and hiding the folders of other users. Even if the system crashes at an inopportune time, Login can reset everything based on the proper settings the next time each affected user logs in. Other applications can learn which user is logged in (or even if Multiple Users is active at all) and act accordingly, but for the most part even the Finder seems remarkably happy with each user's settings as Login creates them.

Not For James Bond

All in all, Multiple Users offers a reasonably solid implementation to keep users out of each others' documents. Since the Login program is the "shell" (in traditional Mac OS parlance), it's the program you return to when all applications quit, so even force-quitting applications wouldn't provide a malicious user unauthorized access. Multiple Users itself is not an application, so its protections can't be eliminated through such Process Manager tricks.

However, it is *not* foolproof. Multiple Users encrypts each user's password so it can't be retrieved except by Login or the Multiple Users Control Panel, but the strength of the password encryption algorithm is unknown. Also note there are no Open Firmware updates with Mac OS 9—Multiple Users does not and cannot prevent anyone from starting the machine from a CD-ROM or other disk. Neither does it encrypt the disk itself, so once the system is started without Multiple Users's protection, all files and folders are just as accessible as if the owner were logged on (or Multiple Users were turned off).

Once you're up and running, the access restrictions work as you'd expect. The weakest link here is the restriction on launching applications. We've already noted that we couldn't keep any users from launching SimpleText, and we don't know if other applications are similarly exempt. The problem is that the restrictions are not in the Mac OS's Process Manager, but in the Finder and Panels. Each limited or panels users has an "Items for [user name]" folder containing the aliases of every application that should be allowed; applications not aliased into that folder should be disallowed. (Limited users can open these applications by double-clicking documents, but panels users see only these items in their "Items" pane.) The

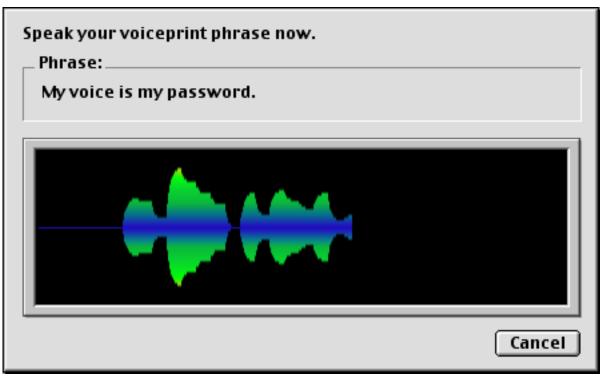


Figure 10—Voice Verification

folder is read-only so users can't add or remove items without permission.

Unfortunately, applications beyond the Finder and Panels can still launch programs (a facility necessary for some programs—for example, Microsoft Office has to be able to launch Microsoft Office First Run if it detects missing components), and Multiple Users has no interference there. If you give a restricted user access to a program that directly launches other applications, he'll be able to run whatever he wants. (DragThing and others of its ilk won't work they rely on the Finder to open applications, and the Finder respects Multiple Users's settings.)

Apple Help notes one of the biggest implications of this hole: if any restricted users can launch Script Editor, they can launch any program they like: "tell application 'blah' to activate". In turn, that opens a hole if restricted users can run any AppleScript that might stop with an error, because the standard AppleScript error-reporting dialog box allows editing the script. This is probably why Multiple Users has a built-in setting to allow access to all applications *except* AppleScripts (but, again, including Script Editor).

Alternate Passwords: Speak Up

Multiple Users maintains text-based passwords for all users, but it also allows *alternate authentication methods* (or "AAMs") if the built-in capabilities aren't enough. Another of Apple's "Internet Power Tools" is simply an AAM plug-in named "Voice Verification" that identifies users through their unique voiceprints (it lives in the Extensions folder). For now, AAMs' sole purpose is assisting with the Multiple User login process.

If the owner globally enables the "alternate password" feature, he can select from a pop-up menu to choose any installed AAMs (right now there's only one: Voice Verification). AAMs get control during the Login program's execution once it has determined which user is attempting to log in. Although Apple has not yet published specifications for AAMs, a wide variety of them may become available in the future, including ones that connect to corporate or university servers (imagine a super version of Macintosh Manager where you could have your customized preferences on any Macintosh across an entire university campus or company).

Voice Verification, however, is nifty enough to satisfy most people for a while. You choose a phrase for verification; the default is "My voice is my password." (For those who've seen the movie *Sneakers* about six billion times, it's hard not to say "My voice is my passport.") The phrase itself is immaterial—Voice Verification analyzes the sound of your speech and compares it to later repetitions of the same phrase, no matter what the phrase is. Just avoid picking a phrase you wouldn't want young users to repeat. You can hide

Sharing Your Personal Space

(continued from page 25)

the phrase in Login if you like, making it harder for someone to guess what he *should* be saying.

When you create your voice password (for consistency, referred to in all but a few dialog boxes as the "alternate password" since it could be using any AAM if more than one existed), the module asks you to record the same phrase four times. Each time you press a "record" button, speak the phrase normally, pause for a moment, and then click "stop." After it has all four instances, Voice Verification analyzes them to create a voiceprint that can survive minor inflections in tone but will not match anyone else's voice. It then asks you to try your new voiceprint, and you speak the phrase again. If it matches, your new voiceprint password is set. If not, you can try to match it again or rerecord it.

During login, selecting any user whose alternate password is enabled immediately tries the alternate password method. For Voice Verification, you instantly get a similar multicolored sound recording dialog box (shown in Figure 10) and are asked to repeat the phrase. If it matches, you're in. If you fail to match it three times, the AAM reports failure and falls back to the built-in typed password authentication. You can also skip the AAM by clicking the "Cancel" button in the voice matching dialog box. If the owner allows users to change their passwords, they can also reset their alternate passwords. That invokes the AAM again, and you can re-record your voiceprint password—once you've authenticated by typing your text password.

How Does It Work?

Our tests on a 333MHz iMac system were frustrated somewhat by problems with sound input—we could not get Mac OS 9 to record any sound from the builtin microphone using any program, including SimpleSound or the Sound Control Panel. We had the same problem booting into Mac OS 8.6, though, and after that Voice Verification started reporting that no microphone was connected—we suspect something has come loose inside and the internal microphone is not being recognized. We continued testing with a PlainTalk microphone plugged into the audio input port. That may make a difference—Mac OS Rumors, for example, recently reported that Voice Verification was easier to fool with an iMac internal microphone than with an attached external microphone.

Once set up, we found that positioning and voice inflection were key to making Voice Verification work. We recorded a voiceprint password while holding the microphone; it was easily recognized when we continued to hold the microphone, but not once we placed the microphone on the iMac. We re-recorded the voiceprint with the microphone in the new position, and then Voice Verification recognized it while the microphone stayed there. But even turning slightly in the chair and facing another direction made recognition fail.

We also found it necessary to record the voiceprint with the appropriate level of ambient noise. We recorded early voiceprint passwords in a quiet room, and then found Voice Verification would not recognize later attempts made in a noisier environment. However, once we recorded the voiceprint with plenty of ambient noise, Voice Verification had no difficulty recognizing it with the same level or noise or in the quiet room.

How you read the phrase also matters. When asked to read a prepared line, most people vary the pitch of their voice somewhat during the pronunciation, lifting a little on early words and declining towards the end (or rising towards the end if reading it like a question). At first, we thought that varying the inflection during verification made it fail, but later tests refuted that. Instead, the most important aspect seems to be *pacing*—making sure you read the phrase with the same delays and speed you used to create the voiceprint. If you hit all the words in about the same places, Voice Verification seems to accept various inflections. If you read it too fast or too slow, or pause between the wrong words, it will not accept the attempt.

Most of our testing focused on this aspect—getting Voice Verification to reliably recognize the right user. The other half is testing multiple voices to see if any of them can trick the AAM into allowing inappropriate access, and we haven't had the chance to conduct those tests yet. When we do, we'll update you on the results.

Encryption: Protecting Private Files

Yet another of Apple's "Internet Power Tools" is the new *Apple File Security* application for secure file encryption. The application supports the new "Encrypt" item in the Finder's "File" menu. The program uses a Fast Elliptical Encryption algorithm invented by Dr. Richard Crandall (formerly of Next Software, now of Apple Computer) to secure files with what Apple calls "industrial-strength encryption." In fact, it's the encryption technology in Mac OS 9 that restricts how Apple can export the operating system. Although Apple File Security is fully compliant with current (wimpy) US export restrictions, some countries have *import* restrictions on software that could hide information from governments. People in those countries (including China, Hong Kong, and Singapore) may not import Mac OS 9 without obtaining a license. Perhaps this is one reason Apple Computer closed its Singapore manufacturing facility in the past few years. Even with 56-bit or weaker encryption, export is forbidden to any country with which the US has a trade embargo: Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria, or any others.

There's a reason this sounds vague—Apple is not yet quite forthcoming about the nature of Apple File Encryption. We'll tell you how it works for users and developers (along with some other aspects of the security additions that aren't immediately obvious), but we can't tell you much about how it works inside. Apple is not publicly admitting to using its own Fast Elliptical Encryption, nor has the company specified how strong the encryption is. Even at last May's Worldwide Developer's Conference, Apple's "Security" presentations skipped over Apple File Security. The Mac OS 9 FAQ says that decryption other than on Mac OS 9 might be possible in the future with "third-party utilities," but won't go farther. Not to raise the pun, but on this topic, Apple Computer is quite cryptic.

What We Know

Certain parts of Apple File Security are pretty clear. You select items in the Finder and pick "Encrypt" from the "File" menu, or you drag the items onto Apple File Security. (The program only accepts files; you cannot encrypt folders or disks.) Generally the program both encrypts and compresses to save disk space, but it refuses to compress items in the System Folder. The application asks you to enter and confirm a new passphrase for every file. Each passphrase must be at least five characters long. A checkbox that's checked by default asks for permission to add the passphrase to your keychain (see "Living in a Wired World," this issue), and will prompt you to unlock the keychain if it's locked. Once that's all taken care of, Apple File Security encrypts both data and resource forks into a single encrypted data stream, stored in the data fork of a new file with file type 'enc2' and creator type 'crp2'.

Even though all encrypted files share the same file type and creator type (the generic icon is a regular document icon in gray holding the ghosted image of a smaller document icon with text, marked with a key in the lower-left corner), all encrypted documents in the Finder keep the icons they had before, adding only a key *badge* in the lower-left corner. Just as aliases in Mac OS 8.5 and later have a small overlay icon (the badge) with an arrow indicating an alias, and locked files have a padlock badge, encrypted files have a key badge over their original icon. Finder 9.0 manages this through a new resource added by Apple File Security to encrypted files. The 'badg' resource contains information about the encrypted file, but it also contains the file's original file type and creator type. Finder 9 uses these to grab the appropriate icon and draw it with the key badge.

Opening any encrypted file launches Apple File Security (all encrypted files are "Apple File Security documents"). If the appropriate keychain is already unlocked and the file's key is on the keychain, Apple File Security automatically decrypts the file and asks the Finder to open it. If the keychain is locked and you don't grant access, Apple File Security prompts you for the file's passphrase.

At first glance, Apple File Encryption appears to be a nice program. It's fast and efficient and, at a superficial level, works well.

What We Don't Know

It's what we don't know that raises questions. We don't know the precise nature of the encryption algorithm, nor how many bits each key is. We presume it must be using 56-bit encryption or weaker to pass US export controls, but Apple has not yet documented the details.

We don't know if Apple will release enough details to allow other people to write encryption and decryption programs like Apple File Security. The risk is that experts could find holes in either the algorithm (though so far "FEE" has withstood scrutiny) or its implementation that might make your files vulnerable. The bigger risk is that such holes exist and only the bad guys know about them.

Unless and until Apple releases full implementation information, only those using Mac OS 9 can use Apple File Security as an "Internet" feature. If you Email an encrypted file to anyone using any other OS including Mac OS 8.6 and earlier—they can't access it. There's no chance for existing encryption technologies like PGP to work with Apple File Security because it's all secret. It would be nice if PGP, Highware, and other security developers could include Apple File Security capabilities in their own applications, giving you more options for encryption, decryption, signing, and E-mailing the files. Apple needs to take the lead here, and there's no sign the company is stepping up to the plate.

With these questions unanswered, Apple File Security could at best be a *local* security solution. Without cross-platform capabilities or even the light of day on the program's innards, it's only suitable for file exchange with other people guaranteed to be running Mac OS 9. If you need secure file exchange, you undoubtedly need to work with more operating sys-

Sharing Your Personal Space

(continued from page 27)

tems and will have to investigate a more versatile thirdparty solution.

Where Apple File Security could work well today is with Multiple Users. You might have a Macintosh set up for several "normal" users with few restrictionsand then run across a document you want to keep private, like a picture of your child's birthday present. Suddenly restricting an older child's computer access by dropping him to a "limited" or "panels" user level would not be prudent, nor would changing the system to keep him out of your user folder if he's had access to it before as a matter of trust. Enter Apple File Security—encrypt the file, put the key on your keychain, and put the file in an out-of-the-way place. If your kid does find it, he'll only know that it's encrypted and what program it belongs to, not what's inside it. He won't even know who encrypted it if there are more than two users—keychains are part of each user profile, and your keychain isn't accessible to him.

Potentially, then, Apple File Security could be a fine complement to Multiple Users. But that potential hasn't been reached in Mac OS 9.

A Fatal Flaw

By now you may be asking yourself an important question: what happens to the original files once you encrypt them? Yes, the program deletes them, but does it do so securely? Normally, deleting a file does nothing to erase the information it contained; it just marks the disk space occupied by that file as "available" so other files may eventually overwrite it. Persons with file recovery software installed, like TechTool Pro 2.5's "Trash Cache" or Norton Utilities's "FileSaver" and "UnErase," can often recover normally-deleted files with minimal information loss. To securely delete a file requires *wiping* it—overwriting the file's contents with patterns or garbage information, sometimes several times, making any recovered file content-free (and damaging anyone's ability to recover magnetic traces from the disk platter itself).

Unfortunately, Apple File Security falls short. It *does not* wipe files that you encrypt, leaving the original much more easy for you (or, more to the point, other people) to resurrect. With file recovery software installed, recovery may even be trivial. Failing to wipe files before deleting them, a truly basic security technique, makes Apple File Security far less useful than it otherwise would be, and until Apple remedies this core deficiency Apple File Security is simply too out-of-the-loop to be useful in the wide world of the Internet or in the confines of a well-equipped Macintosh. It's not a real security solution, and it falls far short of

Apple's marketing promise as an "Internet Power Tool."

A Hidden Goody: File Signing

Thankfully, there's more to security in Mac OS 9 than file encryption. Tucked away in the same folder as the Apple File Security application (you might never look for it since the Finder launches it automatically) is another program named Apple Verifier. Open it and you're asked to select a file, but doing so is difficult. It's like what you see if you're trying to open a JPEG file through Navigation Services when you don't have many JPEG images lying around—lots of folders show up, but few actual files to select. If you find an acceptable file in Apple Verifier and select it, the program "verifies" the file and reports that it hasn't been changed since it was signed.

That's right—digital signatures are built into Mac OS 9. You'd hardly know it from Apple's public statements, but Mac OS 9 contains a rather complete set of system-level routines for signing files, complete with digital signatures in industry-standard *Cryptographic Message Syntax* (CMS) format (the same one used in the S/MIME3 specification and that house nearly all digital certificates). The Mac OS has needed basic cryptographic services for years; now, with digital signatures (and the Keychain, explored elsewhere in this issue), it finally gets it (*MDJ* 1997.04.03).

Apple isn't exactly hiding the feature, but it's not intended to replace PGP. The company's main goal for *Macintosh File Signing* is for developers to sign code. Mac OS 8.6 added URL Access, a powerful feature that gives developers extremely easy access to file transfers using FTP, the Web, or even local disks (*MWJ* 1999.05.12). Apple wants to have a very Net-savvy operating system, and they hope developers will use features like URL Access to make auto-updating applications—programs that automatically check the Net for updaters or new plug-ins or extra documents, download them and install them (all with your permission, of course).

This could potentially be horribly insecure—an attacker could spoof a given IP address and replace legitimate files with vicious ones that could damage or destroy your data. Before you go runnings strange code on your system, you need some way to be sure that code really is what it's supposed to be.

Enter Macintosh File Signing. With a set of system routines just about as simple for developers as URL Access itself, Macintosh File Signing implements a very Mac-oriented signature process. Once a file is signed, even a one-bit change in any of the "signed" portions make the signature invalid. Apple Verifier tells you that for files, and developers can build custom verification into their own programs. However, most industry digital signature standards work on a single stream of data. That's not very good for Macintosh files that have both data and resource forks. It's actually much worse—some of the resources in Macintosh files are *supposed* to change with normal use. If you give an application a larger memory partition in the Finder, you've added or modified a 'SIZE' resource in that application's resource fork. Any digital signature that covers the entire resource fork will thereafter be invalid.

Macintosh File Signing works around this with a new resource type: 'sig#'. The resource, specified for developers in the Macintosh Security Software Development Kit (still in pre-release), tells Macintosh File Signing which parts of the data fork to include, and which resources should not be included. A default file 'sig#' resource includes the entire data fork but excludes all icon resources, the 'badg' resource that Apple File Security might add, and the 'vers' resource with ID 2. A default application 'sig#' resource also excludes 'SIZE' resources. A 'sig#' resource for a text editor might include the data fork but exclude any resources devoted to non-content information (window positions, scroll bar settings, and so forth). When an application calls on Macintosh File Signing to create a digital signature (based on the 'sig#' resource settings), the result is stored in a 'sign' resource. That explains which files show up in Apple Verifier's "Verify File" Navigation Services dialog box: the program only looks at files containing a 'sign' resource, because they're the only signed files.

The flexibility of the 'sig#' resource gives great Macintosh advantages. Without an implementation that recognizes resources, file signing would require turning every forked file into a single data stream, as with StuffIt or MacBinary, both to sign *and* to verify signatures. That's impractical. By excluding volatile resource types, the signature remains valid long after you've started using the file—an extra bonus in the view of some people, but a definite part of the Macintosh Way (why should a program appear "invalid" just because you've used it?).

At present, however, Macintosh File Signing is limited to files (no folders or volumes) and aimed at developers. The "Apple Signer" application is only in the Security SDK, and quite frankly, it's remarkably unstable on our test systems (it won't sign anything and it seems to damage the Desktop Database; it's clearly marked "pre-release"). Signing requires more sophistication than verifying—anyone wanting to sign files must first get a valid signing certificate from a company like VeriSign or Thawte Consulting onto the signing system's keychain. Once your keychain has both a public and private signing certificate, you can export the public part so others can use it on their keychains. (We'll have more explanation in our Keychain coverage later this week.)

Why Bother?

Macintosh File Signing has a twofold aim. First, once the tools get in gear, Apple hopes to offer simple file signing to all developers. When you grab a piece of software from the Internet, you'll at least be able to verify that you got the version signed by the developer, and you should be able to continue verifying it for as long as you want. If you suspect a virus has infected an application, you can just re-verify the digital signature.

This level of security is nice, but it's not a real solution to Internet problems. Consider Microsoft's "ActiveX" technology in the Windows versions of Internet Explorer. Web site developers embed ActiveX components on a Web page, and the browser actually downloads the components and executes them as part of displaying the page. To protect against problems, Microsoft lets users require that the components have a valid digital signature—but if site after site after site requires new ActiveX components, how many times do you really want to see a dialog box prompting you to check a signature against a developer to approve the component? Most people eventually water down or weaken the security out of annoyance. It's a handy option, but it's too broad to answer general security issues.

Combine this with URL Access, though, and you've got something. Imagine a new version of Adobe Photoshop that comes with an embedded Adobe public signing certificate. Through URL Access, Photoshop could regularly connect to Adobe's Web site, find new components (or plug-ins), and automatically download and install them-with a twist. Photoshop wouldn't simply verify that the signatures were valid-it would also insure that the files were signed by Adobe's own digital certificate, the public half of which would be built into Photoshop itself. The combination makes any deception much more difficult. An attacker would not only have to spoof Adobe's Web site to get Photoshop's traffic, he would also have to make Photoshop think his malicious files were actually signed by Adobe's secure private certificate. The chances of pulling this off are almost nil, giving automatic updates with great security. Apple has apparently implemented something similar in the "Software Update" Control Panel (see "Living in a Wired World," this issue).

Macintosh File Signing has routines built in to sign files, opening the door for developers to add signing capabilities to their own programs. Want to sign a file before sending it in E-mail? Developers can allow

Sharing Your Personal Space

(continued from page 29)

it—and since they know their own file formats, they can make sure the signature only covers the essential parts of the document. Putting file signing in everyone's hands will require a much clearer picture of certificate creation than Mac OS 9 provides by default, but that can come with time.

File Signing is built on top of a set of CMS routines that provide fully extensible, cross-platform, industry-standard cryptographic messaging services. Unfortunately, the support is only half-baked in Mac OS 9; only those parts necessary for File Signing to work are public at this time. Apple has discussed CMS at several technical conferences, however, and you can expect more robust support in future versions of the Mac OS. Consider what you see in Mac OS 9 the baseline support for digital signatures. It's only getting better from here.

Gotchas and Kudos

We think the three "Internet Power Tools" we've examined here-Multiple Users, Voice Verification, and Apple File Security-are similar in that they're not really Internet-related at all. Multiple Users is a good way to share a local machine within a household or classroom (when obtaining user information from a server instead of locally), but aside from keeping separate sets of Internet preferences it's not an online feature. Voice Verification is even less related to the Net. Apple File Security is billed as a way to securely transfer files, safe from Internet eavesdroppers, but that's only true if everyone you need to communicate with is using Mac OS 9. As we noted, it flat-out fails to provide local file security. File signing, the fourth new feature we've examined here, is truly a stepping stone to more Net-based software distribution and better verification, but the signing part is not yet in users' hands.

Don't let this fool you. The marketing may be misdirected but most of the features are solid. We were fully prepared to hate Multiple Users, to find it an unsupportable hack involving tremendous amounts of disk thrashing and a minimal illusion of customized environments. Instead, thanks to the Folder Manager, we find it to be powerful and about as well-implemented as you could expect in today's Mac OS. Voice Verification is a nice trick in its own limited ways, but the alternate authentication method it demonstrates may add tremendous value to Multiple Users in the future. Apple File Security's blatant shortcomings are somewhat of a mystery, but we're hopeful that Apple will fix it soon.

Compatibility problems will arise as these features come into wider use. Programs that don't run well today on NetBoot clients (or from CD-ROM) will dislike Multiple Users just as much. Some programs don't take full advantage of the Folder Manager and will therefore have problems (for example, some applications "know" that the Preferences folder is on the startup disk—not true under Multiple Users if obtaining data from a Macintosh Manager server). Faceless background applications may need substantial revision because the "special" folders change locations while they're running, every time a user logs in and logs out. (Multiple Users provides ways for all faceless background applications to keep abreast of such changes so they can respond accordingly, but some of the FBAs will need programmer attention to use those methods.)

Apple specifically warns against extensions that piggy-back off the Finder. Some software waits until the Finder launches and then, assuming it will never go away, takes up some of Finder's memory or uses some of its application environment to make an extension behave more like an application. These are the kinds of programs that crash if you ever quit the Finder, because the parts of Finder space they stole go away when Finder goes away. This has never been what you'd call a bright idea. With Multiple Users, the developers won't be able to brush it off with "Oh, just don't quit the Finder." With Multiple Users active, the first program to launch is Login, not Finder-and as soon as you do log in, Login goes away. Similarly, every time you log out, Finder (or Panels) goes away and Login returns. Any extension piggybacking off the first program launched-whether Finder or Login-will find itself bringing down every multiple-user system.

Multiple Users and its associated friends are a solid first implementation for an operating system designed around the "one person, one computer" vision. It does not make Mac OS into a true multi-user operating system, like UNIX and Mac OS X Server are and like Mac OS X will be. Even though HFS Plus supports it, Multiple Users does not allow you to set owners and access privileges for individual local filesusers are either completely unrestricted or are restricted in one or more broad ways based on their user folders, and that's it. A remote login capability is absent. Files are not assigned owners on disk as they're created. As the Mac OS moves inexorably towards Mac OS X, we wouldn't be surprised to see the interface and methods of Multiple Users adapted for that strong multi-user operating system, but we're not there today.

In the end, that probably doesn't matter. If you have four people and one computer, Multiple Users, Voice Verification, and Apple File Security make it easier for you all to get along with your single piece of technology. It's far more elegant than any other Mac OS solution we've seen, and worth your examination if you have to share a computer. Especially if you get to be the owner.

Tage 51

Mac OS 9

Living in a Wired World Merging the Internet with Your Desktop

"The Best Internet OS Ever" would certainly need tools for managing the myriad and byzantine ways computer owners use-or are forced to use-the Internet. These days that means, in large part, the Web. Apple doesn't develop its own Web browser anymore (R.I.P., Cyberdog), but bundles both Microsoft Internet Explorer 4.5 and Netscape Communicator 4.61 with the Mac OS 9 CD. Both of those programs manage bookmarks for Web sites, handle FTP file transfers, and generally try to be one-stop shopping centers for your Internet needs. Microsoft Internet Explorer must be the default browser because Apple and Microsoft agreed in August 1997 that it would be (MWJ 1997.08.31), but from the Mac OS's point of view it's a better choice anyway-it uses the settings in the Internet Control Panel, based on the public domain Internet Config system. Netscape Communicator *can* use Internet Config, but it really doesn't like to and doesn't hesitate to tell you so. Netscape Communicator includes a built-in E-mail client; Microsoft relies on the separate Outlook Express 4.5 (version 5.0, released this week, was too new to qualify for the Mac OS 9 distribution discs).

But the real question for an "Internet operating system" is how the OS itself can make things both better and easier for Macintosh owners and Macintosh developers. For example, the hotly-disputed "integration" of Microsoft Internet Explorer into Windows has had at least one undeniable effect: every Windows program now has full access to the same browser-quality HTML display code as does the browser itself. Programmers who want HTML support need not reinvent the wheel, and users get a consistent display and experience across all HTML-savvy programs. Whether or not you believe Microsoft did this to eliminate a competitor, the capability is one many applications are happy to exploit. As we've noted before, Eudora Pro for Windows relies on Microsoft's HTML code to display HTML-formatted E-mail. Eudora Pro for Macintosh does not, and its HTML capabilities are far weaker.

We'll shortly look at new features in Mac OS 9 that bring familiar Macintosh services to the Internet: they take parts of the Mac OS that have traditionally been AppleTalk-only and make them work over the Net-centric TCP/IP protocol. Before we blur the lines between AppleTalk-based LANs and the Internet though, let's look at two of the key new features Apple hopes will help you organize a secure shared world. And for the first one, we mean "key" literally.

Keychain Access: Unlocking Your Net Life

Apple Computer's story for the first half of this decade was inventing great new technologies, overengineering them, releasing them a few years before the hardware could handle their needs, leaving them to twist in the Macintosh wind with little assistance from the company, and finally cutting off their oxygen (as well as the oxygen of developers who foolishly trusted Apple's promises). Every now and then, though, an idea or two from one of those abandoned enterprises resurfaces because it was just too good to kill. QuickDraw GX Typography is gone, but it lives on in spirit and truth in Apple Type Services for Unicode Imaging, containing many of the GX features but working with regular QuickDraw. PowerTalk's "Catalog" of people and services is the faint echo of Mac OS 9's Network Browser (coming up later in this article). And Mac OS 9's Keychain Access is the lineal descendant of PowerTalk's "Keychain," minus the fancy Finder manipulations that made it a desktop icon you couldn't move or duplicate.

Keychains survived because they're indescribably useful, even moreso in the Internet age. Just about every shared resource identifies you by "user name" and "password." Trying to keep track of more than a handful of such data pairs is an onerous task, especially since ideal security requires a variety of passwords that are frequently changed. Consider these options that Timbuktu Pro 5 can force on users at the machine owner's behest: passwords may not match any of the three previous passwords, may not be any "common" passwords (including first name, last name, common sequences of digits or letters such as "QWER-TY", a day of the week, a month, season, first name, or sporting term), and passwords must contain a specified minimum number of characters. Oh, and they can expire after a set number of days (even one day), requiring you to change them frequently.

Page 31

Living in a Wired World

(continued from page 31)

Sure, it's *secure*, but who can possibly keep up with all that? Multiply this by two or three file servers and add half a dozen Web sites or FTP servers and you're hosed. A few logical management choices exist, and none are good. Using one password on all services is risky because anyone who discovers it can then hijack your identity all over the Net. If you rotate a few passwords through various services, an unscrupulous administrator who can see your password on his site could keep a list of all the passwords and try them on other sites. Or you could write them down, but that gives away the store to anyone who can find the paper. You wouldn't want to keep that in your office unless you could lock it up, so you might take it with youand lose access to all your servers if you lose the paper (or your wallet, or your purse-and let's just hope those passwords didn't include anything related to credit cards or online banking...).

Programs like Alco Blom's US\$25 Web Confidential help a great deal—they maintain all these various files in a single place and protect it with very strong encryption. But it's not a systemic solution—programs don't have a way to access the information in Web Confidential's database, so the best you can do is find them and cut and paste them. Even if programs could talk to Web Confidential, it opens the possibility of an attack, where a rogue program might try to find all your passwords. Even guarding against that possibility, every program would have to add compatibility with Web Confidential—or with other programs offering similar features but slightly different APIs. In the end, the best way to provide comprehensive and secure access is to build it into the Mac OS. That way, all programs can use it without requiring a third-party program or juggling multiple utilities.

Using Your Keychains

Figure 11 shows a typical Mac OS 9 keychain. Unlike PowerTalk, where the keychain was a special entity with a desktop icon, Mac OS 9 keychains are normal documents, typically stored in the "Keychains" folder of your "Preferences" folder (including the mapped version for Multiple Users, described in "Sharing Your Personal Space," this issue), but accessible anywhere. Feel free to keep yours on a floppy disk and shuffle it between computers (or, with current models, carry a disk *and* a USB floppy drive). Each keychain can contain a variety of identifying items; Figure 11 shows passwords from Apple File Security, AppleShare, and Anarchie 3.6.1 (listed as an "Internet password"), along with both the public and private





certificates for signing files with Macintosh File Signing.

The keychain is not Web Confidential—you have only a few ways to manually add keys to any keychain (the main way is to drag a server alias or Internet Location file into the Keychain's main window). The idea is different—the keychain is a convenience for you *and* for the applications that continually prompt you for passwords. If a program needs authentication information, it should first check any available keychains to see if the information it needs is securely stored in one of them. If so, the program gets the data from the keychain and uses it without bothering you. If not, the program has to ask you for authentication.

Another difference is that keychain-savvy applications add a checkbox to the login dialog box that, if checked, adds the data to the keychain as you authenticate. But it's not quite that simple, because every program's needs are different. AppleShare logins require a user name, password, and optionally a volume on the server; FTP servers typically want a user name, password, and an optional starting directory; certificates might include an expiration date, a purpose, and other attributes. Since only the application itself knows what information it really needs to log you in, only the program is in a position to tell the keychain what data needs to be saved. The bad news is that managing all this diversity requires keychain-savvy applications. The good news is that several of these are already available, including Web Confidential itself plus the BlackWatch 1.5 password-protected screen saver, URL Manager Pro 2.6, the Transmit 1.5.1 and Anarchie 3.6.1 FTP clients, Netopia Timbuktu Pro 5.21, and probably several others. The better news is

that keychain support is undoubtedly slated for your favorite Internet programs now that Mac OS 9 is out. The best news is that with built-in program support, you don't have to do anything other than check the "Add to Keychain" checkbox to make it work—no copying, pasting, retyping, or other such inconvenience. Once an item is on the keychain, your work is mostly done.

Mac OS 9 allows multiple keychain files, but each person will typically have only one. If you have multiple keychains, one of them is the *default keychain* to which new items are added. Mac OS 9 prompts you to create a keychain file if you don't already have one the first time any program tries to use a keychain. You do have to remember the keychain's password (and no, you can't add it to another keychain), but that's about it.

Your keychain is *locked* when you start the Macintosh, meaning no programs (or other users) can access the sensitive data on it until you provide the password. If any program wants access to that data, you're prompted to unlock the keychain first. An unlocked keychain is a security risk—anyone wandering by your machine has access to any resource (server, FTP site, or even digital certificate) protected by the keychain. Therefore, Keychain Access allows individual security settings for each keychain file, accessed through the "Edit" menu. It's well-protected—the menu item is only available if the keychain is unlocked, and you must then enter your keychain password *again* to change the settings. Available changes include password, a checkbox to allow access

Living in a Wired World

(continued from page 33)

to the keychain without warning, plus options to automatically lock the keychain after a set number of idle minutes, and to automatically lock the keychain when the system goes to sleep.

You can also get information on any of the keychain items while it's unlocked; server items also include a "Go There" button that tries to take you to the server in question (for Internet passwords, this happens by default through the Network Browser). All password items include a "View Password" button so you can actually see the passwords protected by the keychain—but, as with changing settings, you must again enter your keychain password even though the keychain must already be unlocked just to get that far. These extra parameters ensure that someone who happens to come across your machine with an unlocked keychain won't be able to see your passwords or change your keychain protection.

For both protection and convenience, the keychain asks for your approval any time a program wants into it. Figure 12 shows the dialog box you'll see the first time any program wants access to an item on the keychain. You can grant permission on a case-by-case basis (seeing this dialog each time), on a program-byprogram basis (so, for example, you could allow Apple File Security into the keychain without further prompting), or on a keychain-by-keychain basis (allowing all programs unfettered access to that particular keychain). If case-by-case approval is too much of a headache for you, feel free to grant permissions to specific applications, but try to avoid opening the whole keychain to every program. The dialog box is there to warn you about a rogue program trying to examine your keychain items, so don't disable it lightly. Even Keychain Access, the program that manages the keychains, needs your approval to automatically change a keychain.

An unlocked keychain with approved access clears the way for the program to use your keychain items: the program fetches what it wants and moves along, and you'll encounter no more obstacles within that program getting to anything protected by passwords stored on the keychain. If you deny access using any of the cautionary dialogs, the application should prompt you for authentication information as it would if Keychain Access were unavailable.

Apple File Security tries to use the Keychain quite transparently—not only does it add its keys to a keychain if you allow it, but it also removes them when you decrypt the file. Once the file is unscrambled, you're prompted for a new passphrase if you scramble it again, so why keep the old key around? Don't be spooked if you think file passwords are vanishing they are, as you no longer need them.

Keychains work well with Multiple Users, with one caveat: Multiple Users automatically creates a keychain for each user, named for the user name. If you've already used Mac OS 9 without Multiple Users, you'll probably already have a keychain file active, but Multiple Users will create a new one named after the name of the "owner" user. Unless you choose "Keychain" as your owner name, you'll wind up with two keychain files. You can keep both open at once, or choose to use the new owner one as your primary file, but it could surprise you if you're not expecting the new keychain.

Behind The Keys

Underneath the interface that you see, the internals of the Keychain Manager have a difficult task. It must balance easy access for programs (including the Keychain Access Control Panel) with the security needed to protect your confidential data. The sensitive parts of the keychain are protected with US export-approved 128-bit RC2 encryption, so anyone who happens across your keychain file won't be able to use it against you. The difficult part is keeping that data encrypted while easily using any part of it.

Every item on the key chain is split into *attributes* and *data*. Attributes describe an item; the data is the sensitive part that only sees the light of day if you grant permission. All items share common attributes, such as the *class* of the item (AppleShare password, Internet password, digital certificate, or generic password), a creation date, a modified date, a label, description, and comment. Other attributes are specific to each class of item—for example, items of class "Internet password" include a protocol (HTTP, FTP, or others), a directory, a port number, and a security domain. Not every attribute is filled in for every item of a given class, but the spots are there.

The Keychain Manager finds items based on their attributes. Apple insists that attributes are encrypted within the keychain files, so if someone gets your keychain file they can't use, say, a disk editor to find out what it contains. "Encryption" is a strange word, though—the Keychain Manager allows access to all of the attributes at any time, even if the keychain is locked. For example, the "Keychain Scripting" application (located in the "Scripting Additions" folder), makes it easy to find the name of every item: 'tell application "Keychain Scripting" to get name of every key of current keychain'. It chugs for a while, but it returns the entire contents of the "Name" column in the Keychain Access window for that keychain. Unfettered access to "encrypted" attributes means the

Page 35

Keychain Manager has everything it needs to decrypt them built-in—including the algorithm *and* the key.

That sounds scary, but it's no cause for panic. Someone, someday, somewhere will disassemble enough of the Keychain Manager to figure out how to decrypt those attributes, and there will be yet another giant uninformed "scandal" about how these attributes aren't secure. The scandalmongers will be right, but it won't be important. The encrypted attributes only specify the *kinds* of items on your keychain. The supersecret part—the information you need to actually login to a server or site—is stored in a keychain item's *data*, not in its attributes. It's simply no big deal if another person sees your keychain item attributes.

Item data, in contrast, is secure because it's encrypted, and the decryption key is made using a password that only you know. Item data can't be searched, nor can any user or program retrieve an item based on the data (even if you knew a password stored in a keychain, there's no way to trace it back to a key item and figure out what the password goes to). The Keychain Manager APIs do not allow retrieving item data-even inside a program-without your explicit permission (see the dialog box in Figure 12). Finally, for items that are themselves keys (like a public and private PGP key pair, something Network Associates could choose to put on the keychain), the item data for private or symmetric keys is simply not available. We're not sure why you'd want to keep a private key on a keychain if no program can retrieve it, but Apple's 1999 WWDC presentation maintains that you want it that way.

Item data is encrypted using a symmetric encryption algorithm derived from the keychain's password itself-without the keychain password, you cannot retrieve item data. Since the decryption key is computed from the password, no keychain ever stores its own password. Moreover, the Keychain Manager doesn't store passwords anywhere else on disk. The only way into a keychain is for you to unlock it. That normally involves you typing a password, but programs can send a password to the Keychain Manager and ask to silently unlock a keychain. Yet even this is safe from attackthe invisible unlocking method only works until the first failure. If any program (including an AppleScript) ever sends the wrong password to a keychain, all future attempts to silently unlock any keychain will fail. The only way around it is to restart. Forcing a shutdown and restart cycle between hacking attempts effectively eliminates brute-force attacks.

Hand-carved brute-force attacks are also difficult. If you enter the wrong password in a keychain authentication dialog box, it's no problem. For the first few attempts. As the failed attempts accumulate, the Keychain Manager fights back, adding dead time after you ask it to authenticate and before it returns a result. The "chasing arrows" in the dialog box spin, and the system goes on, but your only option is to wait or cancel. The dead time remains even if you cancel and start over. What's more, the dead time increases exponentially at an impressive rate. In our testing, the first four mistakes all took less than one second to reject. The fifth attempt took three seconds; the sixth, twelve seconds; the seventh, eighty-five seconds. The eighth attempt took over ten minutes. With exponential growth, the ninth attempt would probably take well over an hour (we didn't try). The only way around this is a reboot, which itself thwarts automated methods of attack.

Programmers will find the routines for working with keychains and their items pretty easy to understand and implement. The Keychain Manager includes high-level routines for working with AppleShare, Internet, and Certificate items, plus a slightly more complicated one for generic password items. There have been minor changes in Mac OS 9's implementation compared to the earlier Keychain releases available only to developers, so there's an outside chance some "keychain-savvy" programs will need minor revision for Mac OS 9. Most of those programs should already have those changes in place by now. And thanks to "Keychain Scripting," AppleScript authors can perform almost every task available to the low-level programmers.

Your New Keychain?

A new system-level service like the keychain, especially one that requires application support, can take a while to catch on. But unlike some other technologies, the Keychain has a jump on widespread adoption. Apple has talked about the Keychain for a long time, and thousands of users are familiar with it from PowerTalk. As you've seen, several applications already support the keychain, and so do the relevant parts of Mac OS 9 itself—AppleShare client, Apple File Security, and Network Browser all support the Keychain out of the box. The Keychain Access Control Panel, as well as Keychain Scripting and a Keychain Control Strip Module for quick access to any available keychain, add further robustness. One potential trouble spot: we've seen reports that in AppleShare 6.2 and earlier, servers can't be added to keychains if the System Administrator has set AppleShare to deny local storage of passwords. This may be fixed in version 6.3, but we haven't checked.

The next big step is getting Web browsers to add keychain support, details that aren't public yet. Once the two main Macintosh browsers do so, the Mac OS keychain will be as indispensable as the keychain in

Living in a Wired World

(continued from page 35)

your pocket. And it may happen soon—keychains are file-based in Mac OS 9, but they work from read-only media like CD-ROMs and some AppleShare servers. Apple says that future Keychain Manager releases will support both removable tokens and smart cards, and the keychain file format is already based on the industry-standard CDSA data storage model. With a future Mac OS release, you could store all your private passwords on a single encrypted smart card (with a keychain file to back it up), pop it out of a FireWire card reader, and take it with you to work or even to an ATM machine.

The Net has been throwing keys at all of us for years. It's about time we got a real tool to corral them.

Software Update: Piecemeal Installation

Many Macintosh owners use the Internet to stay abreast of new software releases—if you just bought that great new utility at your local Mac software store, you might need the latest iMac Firmware Update or Power Macintosh G4 ROM Update or Incredibly Obscure Firmware Reburning Utility for peak—nay, *optimal* performance. One way to keep up with the changes in the Mac OS is to check the "Recent Changes" page on Apple's "Software Updates" Web site. Automatically updated several times per day by a massive WebObjects database, you can easily find the latest releases, download them, mount the omnipresent disk images, install them, reboot, and feel suitably connected to Apple's engineers.

Not many people take this approach. This irks Apple, for the company would like to release bug-fix updates and feel reasonably comfortable that people are installing them and using them. The smaller the number of different existing configurations, the easier Apple's engineering becomes. That saves both time and money, commodities Apple loves to conserve. If Mac OS 9 is the "Best Internet OS Ever," it should include some kind of tool to facilitate these necessary updates, shouldn't it?

Indeed it should, and in fact it does. The **Software Update** feature in Mac OS 9, yet another of the "Internet Power Tools", handles this mildly onerous task for you. When launching the Software Update Control Panel (really an application), you get a nice friendly, colorful "Update Now" button. Clicking the button connects to the Internet, where Software Update checks Apple's hidden magic server, the same one that holds the Sherlock plug-ins for automatic updates, using the same URL Access feature introduced in Mac OS 8.6 that had such great potential (see?). The utility then sees what's been released since the last time you checked (or any recent changes— Apple isn't documenting details, but we did some spying to watch Software Update in action). After connecting to the database, Software Update looks to see if you already have the current updates. If you do, great, you're done (right now, all Mac OS 9 purchasers have all updates, so don't expect to see this in action instantly). You can also schedule updates on a regular basis, as with Sherlock 2 indexing, so you don't even have to remember to check for updates.

The updates all come from Apple, not a thirdparty Web site over which the company has no control, so all files are digitally signed. URL Access verifies their integrity automatically on download, and you can repeat this yourself at any time with Apple Verifier (see "Sharing Your Personal Space," this issue). Software Update refuses to install any file whose digital signature fails verification.

Driving You Sane

Programs like Insider Software's UpdateAgent have done this downloading trick for years. But UpdateAgent, long a favorite of ours (*MWJ* 1999.05.29), can't do one of Software Updates' documented tricks: automatically installing new files in the proper System Folder locations. This is hard to test at the moment because, having installed Mac OS 9, our test system is up-to-date (you'd hope so, wouldn't you?). We can and did test it with a separate part of Software Update: device driver acquisition.

USB and FireWire both feature hot-swappable devices, drivers that load on-the-fly, and no restart requirement after adding a driver but before using a new device. Software Update takes this one step further. In Mac OS 8.6 and earlier, plugging in an unrecognized USB device gave you a generic modal dialog box warning that the driver for the named peripheral (often "Unnamed Device") was not present, so you couldn't use the thing. In Mac OS 9 with the Software Update Engine in your Extensions folder, the same situation launches Software Update. The program tells you, through an equally modal dialog box, "Drivers needed for the USB device 'Unnamed Device' are not available. Would you like to look for these drivers over the Internet?" You may assent or cancel. Continuing gives the same indefinite progress bar as Software Update's main function, but with text that says "Looking for Drivers..." instead of "Looking for Updates...". It checks a different Apple server, passing it the identifying information for your renegade USB device: vendor ID, product ID, USB class, USB subclass, protocol, version, and USB interface reference. If Apple's server has an update for your file, Software

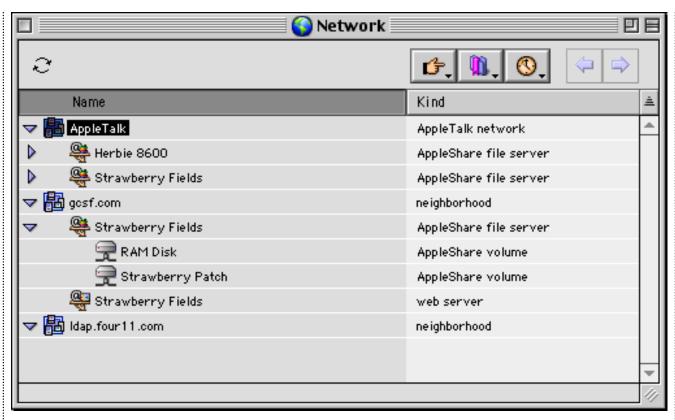


Figure 13—Network Browser 2.0

Update downloads and installs it. Otherwise you get a dialog box expressing regret and urging you to contact the manufacturer.

We had a crapshoot trying this one out. Our first attempt to invoke automatic updating was a USB floppy disk drive from TEAC, whose driver we had not installed on the Mac OS 9 test machine. Nice try-Mac OS 9 includes built-in support for most USB floppy drives as part of the USB Mass Storage device class. Next we tried an Entrega USB-to-Serial converter, a device class we know has no built-in support. Sure enough, we got the offer to check the Internet for drivers-but Software Update came up empty-handed. The device has been out for several months, but Apple's server knew nothing about it. We understand from reports and a demonstration by Phil Schiller that the feature works quite well for input devices, especially those controlled by Apple Game Sprockets. And although Apple has neither demonstrated nor specifically discussed the issue-nor were we able to test itour examination of Software Updates makes us believe it also downloads and installs FireWire device drivers if you wish.

Self- updating system software is a grand idea, nicely secured with digital signatures and automatic installation. The device driver feature is even nicer, but it could become either a hot feature or a curiosity based on how Apple handles its driver servers. If the company excludes drivers other than its own or those of the largest developers—an entirely possible choice given Apple's tendency not to distribute code it hasn't tested—people with dial-up connections may stop looking (why occupy the phone line and wait a few minutes to see another "Not found" dialog box?). But if Software Update becomes the gateway to all the USB or FireWire drivers you could want, it will redefine "plug-and-play." Software Update is no replacement for UpdateAgent, for it doesn't examine items that aren't part of the Mac OS itself. Yet the items Software Update does examine are usually the trickiest to manage, and the program does it for you. We'd call that a power tool.

Network Browser 2.0: Netly Neighborhoods

One year ago we told you about Mac OS 8.5's **Network Services Location Manager** (the NSL Manager), a new system-level service that, through a plug-in architecture, uses protocols like DNS and SLP (service location protocol) to discover TCP/IP services the same way the Chooser "discovers" AppleShare file servers. Instead of having to know an

Living in a Wired World

(continued from page 37)

IP address before connecting to a server, NSL could find and display a list of all IP-based file servers in your domain. NSL wasn't fully baked in Mac OS 8.5: it only included plug-ins for SLP and DNS (and those DNS servers needed slight modification to return useful results), and it had no way of providing a standard human interface for choosing "discovered" services. Network Browser 1.0 gave some inkling of a promised TCP/IP browser, but it only worked for AppleShare file servers.

Figure 13 shows the new and improved **Network Browser 2.0**, another of the "Internet Power Tools." Apple promised a standard human interface for NSL, and the company delivered in the form of Mac OS 9's NSL 1.1 library. Apple also promised new NSL plug-ins for LDAP (Lightweight Directory Access Protocol, an Internet standard for servers supplying directory information about people) and AppleTalk's Name Binding Protocol; both are also included. That gives NSL all the support it needs to completely replace its former proprietary code with the NSL Manager, leading to the greater functionality in Network Browser 2.0.

AppleTalk is but one neighborhood (an NSL term for a hopefully-local network) in the new Network Browser. Apple's documentation is unclear on exactly how neighborhoods are defined-for AppleTalk it's a zone, for SLP it's an IP subnet. We're not sure what it is for DNS; Network Browser 2.0 calls our zone "gcsf.com" even though that domain resolves to an IP address halfway across the country, but the Network Browser finds services on our local area network for "gcsf.com". If you don't have a default domain configured in the TCP/IP Control Panel, the SLP services often show up in a neighborhood called "Local Services" (the name is different in non-English languages). You can add new neighborhoods with a command from the "File" menu; you remove them by pressing the "Delete" key with the neighborhood selected, but the deletion doesn't show up until you quit and relaunch Network Browser.

Figure 14 shows the "Network" selection in Mac OS 9's Navigation Services dialog box (this one for saving Sherlock 2 criteria). It looks awfully familiar. NSL 1.1 includes the code to let any application display and use network browsing functionality—the NSL Browser. Network Browser 2.0 relies on this, as does Navigation Services 2.0, presenting a consistent

Open: Sherlock				
Network 🗢 🗘		6 . 🕅. ()_	
Name		Date Modified	±.	
AppleTalk		-		
🗢 🛅 gosf.com		-		
🤝 🚇 Strawberry Fields		-		
RAM Disk		-		
Strawberry Patch		-		
Idap.four11.com		-		
Local Services		-		
			Ŧ	
Open saved search criteria				
0	Cancel) Open		

Select a Service			
@We	b servers 🗢 🗗 🗗 🚱 🗘 🤤	>	
	Name	±.	
- ₽	AppleTalk		
マ 🛱	gcsf.com		
	🚑 Strawberry Fields		
▽ 闘	Idap.four11.com		
▽ 🛱	Local Services		
		Ŧ	
	nttp://www.testserver.com/		
0 Q	5 items Add Neighborhood Cancel Choose	כ	

Figure 15—NSL Browser's "Select Service" Dialog Box

human interface throughout the system. It's even more flexible than regular Navigation Services: the NSL Browser can be part of a regular window, as in Network Browser 2.0, and doesn't have to be part of a separate dialog box.

But dialog boxes are part of the game, too. There's a new interface for browsing network services and obtaining a URL for the one you've selected, shown in Figure 15. Again, the human interface is consistent with both Navigation Services and Network Browser, with one exception. The pop-up menu in the upper-left corner lets you choose the kinds of services that appear in the list: Web servers (with http and https protocols), FTP servers (ftp), Telnet hosts (telnet), AppleShare file servers (file, afp, nfs), news servers (nntp), directory servers (ldap), media servers (rtsp), and remote applications (eppc, "program linking," though this didn't really work for us, as we'll explain shortly).

You can see these protocol types yourself by holding the "Control" key while the dialog box opens. The NSL Browser shows you more information, including the plug-in used to create each neighborhood, the number of current searches underway (while the "chasing arrows" are moving), and the names of each plug-in by their neighborhoods in the "Shortcuts" popup menu. An easy way to play with the dialog is to launch Script Editor and run a script with just the command "Choose URL". Each application defines what protocols should show up in the dialog box; the ones we've listed are the "Choose URL" defaults.

The browsers, whether in Network Browser or in other dialog boxes, are fully Macintosh-savy. Dragging any icon out of the browser pane creates either an alias (to a file server) or an Internet Location file (a URL clipping file). The standard key equivalents all work (with the exception of Command-T for "top" in the non-Navigation Services dialog boxes), and the windows are resizable and remember their positions. And, in a nice surprise, Network Browser 2.0 is a minimal FTP client, with full drag-and-drop file transfer support. Note that this is part of the application—FTP is not supported in the NSL Browser functionality in Navigation Services. Trying to use an FTP URL in an "Open" or "Save..." dialog box gives you an "unsupported protocol" error dialog box.

Living in a Wired World

(continued from page 39)

Network Browser 2.0 is quite an evolution from the AppleTalk-only version in Mac OS 8.5. Apple's press release says it "makes finding file servers, FTP servers, and Web servers as easy as selecting a local printer, in small workgroups or the largest organizations." Technically, Apple could have appended "or the Internet," because Network Browser 2.0 could eventually make at least parts of the Internet as easy to use as AppleTalk-based LANs are now. But Apple was wise not to overhype this capability at this time, because most of the Internet still isn't as user-friendly as AppleTalk. The potential is certainly there—but the delivery is still in the future.

What's Wrong With NSL 1.1?

Nothing, really—this is a cool piece of code. The problem is that it relies rather heavily on **Service Location Protocol** (SLP). As we noted a year ago (*MWJ* 1998.10.19), AppleTalk has always required devices on the network to register their names and broadcast their presence to the larger network. TCP/IP never has and probably never will, and that's why you must know either a domain name (or a URL based on it) or an IP address to connect to an Internet service.

Both Apple and Open Door Networks have bought heavily into SLP, and with little wonder: SLP is what lets TCP/IP servers broadcast their name and capabilities to at least the local network, giving it the advantage AppleTalk has always held (Open Door Networks founder Alan Oppenheimer, not coincidentally, is an SLP promoter and one of the inventors of AppleTalk).

The problem is that not many servers aside from Apple's and Open Door's products support SLP at all. The NSL Manager only has four ways to discover services: SLP, add-ons to DNS servers, AppleTalk's Name Binding Protocol (NBP), and LDAP. Most networks aren't using the extensions to DNS that NSL Manager needs; AppleTalk and LDAP can't help find TCP/IP services. That leaves SLP, and too few servers understand it today. In our testing, the brand-new WebSTAR Server Suite 4.0 didn't show up in any NSL Browser tests for any of the myriad services it supplies. In fact, the only servers we could get NSL Manager to find were from Apple and Open Door: File Sharing, File Sharing over IP (from Open Door's ShareWay IP), and Mac OS 9's Web Sharing. Apple claims to have a Network Services Location Manager Network Administrator's *Guide* to describe how to make your servers show up in the list, but the manual is not on Apple's Web site where Mac OS 9 Help says it is.

We also had bad luck trying to get the NSL Browser to find "remote applications" over IP, even though that feature is added in Mac OS 9 (see "Program Linking: the PPC Toolbox," this article). Pressing Control while opening the NSL Browser shows that it's trying to find such applications through (you guessed it) SLP, not by finding remote machines and looking for applications. Mac OS 9 allows programs to talk to each other over IP, but that support does not include SLP discovery for running applications. Right now applications may have to add that SLP support themselves if they want to show up in the NSL Browsers. We doubt many programmers are itching to adopt that burden.

Some will be disappointed that Network Browser doesn't include printers like PowerTalk's original catalog services browser did. You shouldn't be surprised here-it's hard to improve a printing architecture when one doesn't even exist. What would happen when you try to "open" a printer icon in Network Browser? Should it select the printer? Give you printer information? What about dragging a printer to the desktop-does it make a desktop printer even though the majority of non-Apple printer drivers don't support desktop printing? How does it all work for printer drivers that still execute crucial routines only when you pick them in the Chooser? Everyone wants to dump the aging Chooser, including Steve Jobs, but a new browser program can't do it alone. In our opinion, Network Browser is wise not to try.

Apple's NSL Manager is caught in a chicken-andegg problem: servers don't support SLP because few programs use it, and few programs use SLP because most servers don't implement it. But Apple Computer can easily be a 500-pound gorilla in the marketplace it was Apple's choice to use USB in the iMac that finally let that serial bus take off with peripheral manufacturers. NSL 1.0 in Mac OS 8.5 gave a decent reason to investigate SLP, but not one serious enough for developers to start jumping since there was no built-in human interface to see the results. Now there is-and thanks to modular code, it's in the high-profile Network Browser, the growingly-pervasive Navigation Services, and in a new dialog box all its own for "choosing" a URL. It might be enough to get server developers' attention-or at least get Microsoft's attention, which would have the same effect. In the meantime, it works well on local networks running Mac OS 9 and even serves as an FTP client in a pinch. Not earth-shattering innovation, but a good start.

Sharing IP Freely

The last two of Apple's Nine "Power Tools" are rightly grouped in one topic—sharing using TCP/IP—but examined separately: File Sharing, enabling networked computers to share documents, programs, and other files; and AppleScript, enabling programs running on networked computers to share data.

The Macintosh has always been a superbly networkable computer, with every model since January 24, 1984 featuring built-in local area networking. Unfortunately, that advantage hasn't survived in the Internet age. AppleTalk is a "chatty" protocol, constantly sending small packets of information over the wire to other machines in the same zone (the *neighborhood* in SLP or Windows parlance). The frenzy of packets is desirable in some ways; it's what makes things like self-discovery possible. Every time you turn on an AppleTalk printer, it broadcasts its existence and address to all devices on the local network; that's how the Chooser keeps its lists updated.

But it's a system designed for slower wires and smaller packets. Most AppleTalk data packets hold less than 255 bytes. Each packet also has latency associated with it—a fixed delay for sending and receiving a packet regardless of the actual transmission speed. As the standard disk drive on a Macintosh has grown to 10GB (from 400K), files have become larger. Speeding up network transmission isn't just a matter of faster wires; it's also about using larger packets with less traffic to reduce latency. AppleTalk was designed for a day when losing lots of packets made keeping them small a good thing-it made retries faster. Today's networks have lower packet loss rates and faster wires; they want a network protocol that takes advantage of these features. The answer, of course, is TCP/IP, the Internet's native language.

Apple's support for TCP/IP has grown with the Internet. The company was early out of the gate with a TCP implementation in MacTCP, but it didn't get the attention it deserved until the Net came along. Open Transport shook things up and eventually made the Mac's TCP/IP implementation quite strong. Products and peripherals from printers to servers started adding TCP/IP support. The only thing remaining for Apple was doing the same in its own software—migrating existing services to a dual TCP/IP and AppleTalk networld.

File Sharing Over IP

One of the two remaining "Internet Power Tools" brings File Sharing, the personal AppleShare server technology added in System 7, to the TCP/IP world. AppleShare file services are significantly faster over TCP/IP than over AppleTalk, so this is a win for everyone involved. It's just not very exciting. For starters, Apple didn't add this to the Mac OS as much as purchase it from Open Door Networks; Mac OS 9 contains a limited version of that company's ShareWay IP Personal Edition, a product available for US\$79 for the past few years. The company also makes more industrial-strength versions, moving up to the "Professional" model with SLP proxies, logging, centralized administration, and the ability to place multiple servers on an IP network from a single Macintosh.

The technology on the client side isn't news, either—Apple first changed AppleShare to work over IP with the release of the professional AppleShare IP 5.0, announced with the "Rhapsody" OS strategy in January 1997 (*MDJ* 1997.01.07) and shipping three months later (*MDJ* 1997.04.22). That's when Apple revised the AppleShare client software to work with either AppleTalk or IP servers, and it hasn't changed much since then. The most interesting change of late is that Apple now fully supports Open Door Networks's "AFP URL" format, where a URL with the protocol "afp://" describes an AppleShare server running over IP. (The AFP "helper application" is Network Browser.) Of course, it was this way in Mac OS 8.5 as well, so it's not *really* news, but it's close.

Apple and Open Door have integrated their technologies well. The "File Sharing" Control Panel is your one-stop shop for all dual-protocol sharing needs. Both File Sharing and Program Linking have added a simple checkbox beneath the status area, allowing you to enable clients to connect over TCP/IP. If you check it, the IP service starts, and the Control Panel displays a URL by which others may access your zippier serving (protocol "afp" for File Sharing, protocol "eppc" for Program Linking, as we'll discuss below). File Sharing now also displays your IP address, making it easy to share with someone who will just type it after clicking the Chooser's "Server IP Address" button. You can still see some of the piggy-backing; for example, ShareWay IP depends on existing File Sharing strongly enough that you can't enable IP-based File Sharing without also enabling the AppleTalk version. A recent Tech Info Library article adds to this logical conclusion, pointing out that you must have AppleTalk active to share files via IP. These are generally exceptions rather than typical problems, though.

We are not minimizing this. It's truly a power feature, and many folks will find even more utility from it once Open Door makes available their upcoming Mac OS 9 upgrade, bringing security and logging features to the IP File Sharing service. It's very cool, and we're already using it. It's just not news. Great features don't have to be exciting.

Program Linking: The PPC Toolbox

The great advantage of a modular operating system is that adding new features to low-level compo-

Living in a Wired World

(continued from page 41)

nents enhances the value of all the others built upon them. You may not recognize some of the low-level component names, but they're still inside your Macintosh, plugging for you day after day after day. One of those components gets a significant IP boost in Mac OS 9, and Apple is avoiding mentioning it because you probably wouldn't know what they were talking about. They're talking lots about the *benefits* of this improvement, but not its source. For you to understand it, we need to review the way Macintosh programs talk to each other.

One of System 7's most touted features was interapplication communication. On a single system or across networks, the Mac OS added comprehensive capabilities for programs to share data with each other. You may think of this as happening through AppleScript, or even though Apple events. Beneath both of those, however, is a technology called the *Program-to-Program Communications Toolbox*, or "PPC Toolbox." This all happened years before PowerPC microprocessors changed the way most Macintosh folk think of the abbreviation "PPC," but that hasn't changed the Mac OS; the term "PPC Toolbox" refers to interapplication communication.

The PPC Toolbox handles all standard communication among applications on the Macintosh, even between two applications on the same computer. PPC Toolbox uses network-like terminology of *ports* and *sessions* to represent the idea of opening a connection (perhaps an authenticated one) between two programs for communication. When the two programs are over a network, PPC Toolbox uses the AppleTalk Data Streaming Protocol (ADSP), as well as the Name Binding Protocol (NBP) to turn machine names into AppleTalk node numbers. PPC Toolbox doesn't care what goes across its connection; it just manages the flow and ensures that data gets from application 'A' to application 'B'. In the modular Mac OS design, the next highest level is the *high-level event*, a part of the Event Manager that uses PPC Toolbox to send discrete chunks of information from program to program, treating them when they arrive just like any other kind of event (a keypress, a mouse click, a disk insertion, or others). On very rare occasions, you can see PPC Toolbox remnants peeking through the rest of the system via data using its four-character identifying type: 'eppc' (extended program to program communication).

High-level events also don't have many structure rules; they just break a PPC Toolbox session into more manageable chunks. Programs communicating with each other would need to agree on all the formats in their high-level events so they'd know how to interpret the data they receive. Comprehensively defining such interpretation rules is massively complex, so Apple did it for programmers, publishing a protocol for highlevel events called the "Apple Event Interapplication Messaging Protocol." High-level events conforming to this specification are (you guessed it) *Apple events*.

You're probably already familiar with Apple events as they're thrown around from program to program, directing your computer on its tasks. When you "open" a URL, the Internet Control Panel launches the appropriate helper application and sends it an Apple event with the URL to fetch. The Finder sends and receives Apple events constantly—every time the Finder does something in response to another program's request (open a window, quit, reveal an icon), it's all due to Apple events. In fact, Apple events are so useful that an entire system of controlling them through text-based programs is included in the Mac OS. You type English-like words, and the system turns your instructions into Apple events and sends them. This grand experiment is called *AppleScript*.

Why go through the grand event tour? You need to understand this hierarchy to realize what's happened in Mac OS 9. AppleScript is built on Apple

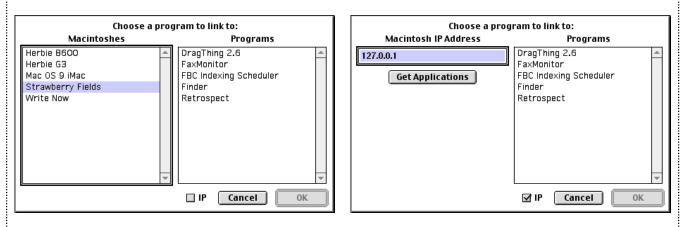


Figure 16—Classic PPC Browser with IP Checkbox

Figure 17—PPC Browser Using IP

events; Apple events are built on high-level events; high-level events are built on the PPC Toolbox. In Mac OS 9, Apple has changed the PPC Toolbox to allow communications over IP as well as through AppleTalk. Because of the modular structure, *everything* above it suddenly has IP capabilities as well, including Apple events and AppleScript.

The PPC Toolbox includes a standard human interface to choose an application to talk to, on your own machine or anywhere on the network. It's called the PPC Browser, and it's shown in Figure 16. (You can play with one on your own system by executing the AppleScript command "choose application".) If your AppleTalk network has zones, they appear in a new list in the lower left corner of the dialog box, just as they do in the Chooser. New in Mac OS 9 is that little checkbox marked "IP." Figure 17 shows what happens when you check it: the list of AppleTalk machines goes away, replaced by a field where you type an Internet address and click the "Get Applications" button. Once you have the address right and tell the PPC Browser so, it connects to the remote computer via IP and lists the applications capable of communication. (This all requires Program Linking to be enabled in the File Sharing Control Panel; linking over IP requires enabling Program Linking over IP.)

The difference is simple: instead of opening a PPC connection to a remote machine by AppleTalk specification, you do it by IP address. If PPC Toolbox can handle IP, so can high-level events, and so can Apple events, and so can AppleScript. That's the real story behind Apple's ninth "Internet Power Tool."

AppleScript Via IP

PPC Toolbox support for IP probably won't affect your daily Macintosh use much. If you regularly send Apple events or scripts to remote computers (and are therefore familiar with the PPC Browser shown in Figure 16), you'll get to start using the IP-enabled version in Figure 17. Everything else happens transparently—just faster, and over distances. Another advantage of IP is that, being the *lingua franca* of the Internet, you can send IP packets all over the world. You can now send an Apple event from your Internetconnected machine to any other Internet-connected Macintosh running Mac OS 9 or later (with Program Linking enabled, natch). You'll pick the same programs in the same dialog boxes; the system handles the gory details.

Except in AppleScript. Since the scripting language turns your English-like words into Apple events, it requires a *terminology* telling it how to do that. The terminology for naming a remote application involves AppleTalk terms: 'tell application "Finder" of machine "Strawberry Fields" of zone "The Big Zone."' In Mac OS 9, you want the option to specify a remote machine using an IP address. AppleScript uses the URL in the Program Linking section of the File Sharing Control Panel as its identifier; it's formed with the PPC Toolbox's internal identifier, 'eppc'. If your machine is on the Internet at address 38.195.223.64 with the name "fredbox.mycompany.org," AppleScript can address it as either <eppc://38.195.223.64/> or <eppc://fredbox.mycompany.org/>.

Unfortunately, AppleScript 1.4 in Mac OS 9 is, at press time, the worst-documented set of changes to AppleScript in years. We know of at least seven noteworthy changes in AppleScript, but they are barely mentioned in AppleScript Help with Mac OS 9, meaning scripters will have trouble using them. Neither are they mentioned at all on the AppleScript Web site. The Mac OS 9 Technical Note mentions the changes but doesn't explain how to use them in a script. And none of the CD samples that we found use any of the new features. In fact, the only sample we've *seen* from Apple demonstrating any of the new features is an online picture of a script in Apple's marketing material for AppleScript over TCP/IP. Study it carefully; it may be all the documentation you can get.

Here's what we know:

• You can use an 'eppc' URL for the "machine" parameter of a remote computer instead of a string presumed to be the AppleTalk computer name

• You can allegedly include a user name and password for a remote machine in your scripts, instead of forcing the authentication dialog always onto the screen, but we can't find out how (another choice would be putting the password on the keychain and scripting keychain access)

• A new "using terms from" block tells AppleScript to use event terminology from one program while sending Apple events to another, eliminating the need for the "double-tell" block described in the Mac OS 8.6 Technical Note

• Script Editor is now a Carbon application (ready for Mac OS X) and can save compiled script applications in Carbon-ready form also, if you wish (referred to as "classic applet" and "Mac OS X applet" in the "Save As..." dialog box)

• The "Info For" scripting addition now gives more information about a file—it tells you if the file is busy or not, so a script can check a file repeatedly until it's no longer in use

Living in a Wired World

(continued from page 43)

• AppleScript's long-standing error-catching mechanism is a series of statements in a "try" block followed by error-handling code in an "on error" block; the "on error" block is now optional, so you don't have to include an empty one just to get the script to compile

• The new "Keychain Access" scriptable application gives script writers almost as much access to keychain items as machine-level programs get

• The new "Speech Listener" scripting addition accepts a list of words and, using Speech Recognition, returns which word in the list it heard (if any) in a given time period

• Due to architectural changes, AppleScript no longer loads with extensions; it loads with "components" like QuickTime digitizers and file translators; if your system has a strange dependency that requires loading AppleScript after some other extension, prepare for trouble (components always load first and can't be made to load after extensions)

That's it. A nice set of features, especially given how little change was necessary to get AppleScript working over TCP/IP. Note, however, that any script using new terminology features in AppleScript itself (like URLs for remote machine names or the "using terms from" block will neither compile nor run with earlier versions of AppleScript or of Mac OS. All you need to take advantage of this is the documentation Apple has yet to supply. A Thursday phone call to AppleScript product manager Sal Soghoian seeking clarification was not returned by press time.

The Nine Wonders Of The Release

Unlike Mac OS 8.6, Apple's choices for marquee features in Mac OS 9 are surprisingly meaty. Sherlock 2, Multiple Users, the Keychain Manager, Network Browser 2.0 (and NSL Manager 1.1 on which it is based), and interapplication communication over TCP/IP are all worthy additions in the best Macintosh tradition—taking what we already know, making it stronger, and using the Mac OS's own design and capabilities in the most flexible ways.

Mac OS 9

Watch This Space Hidden Technologies Promising Future Benefits

Each release of the Macintosh operating system contains behind-the-scenes changes. Invisible to most people sitting at a computer, they have no apparent immediate effects. They do, however, lay the foundation for the features of applications yet to come. Sometimes it takes a while: aside from Network Browser, for example, few programs use the Network Services Location Manager introduced in Mac OS 8.5 (MWJ 1998.10.19, plus "Living In A Wired World," this issue), and the Apple Type Services for Unicode Imaging (ATSUI)—the intellectual progeny of QuickDraw GX Typography—is still badly needed yet unused by most applications (MWJ 1998.10.19). On the other hand, URL Access, introduced in Mac OS 8.6, is showing up in custom AppleScripts all around the Macintosh community, and is now the basis for Apple's Software Update Control Panel ("Living In A Wired World," this issue).

Predicting how the new features in Mac OS 9 will change the software landscape is no small chore, as it depends on whether and how quickly developers update their software. We prefer to let you draw your own conclusions, so we'll explain the major invisible and structural changes in Mac OS 9 to provide a good first step for your assessment. First are two of the biggest "hidden" changes in Mac OS 9—access to HFS Plus features, and the Mac OS debut of *real* font management support, built right in. Following that we examine four other new "invisible" technologies: the HTML Rendering Library, a new Multilingual Text Editor, plus the Carbon and Core Foundation technologies that pave the way for Mac OS X.

HFS Plus: File Manager Expansions

It's hard to believe, but Apple first released the HFS Plus file system with Mac OS 8.1 nearly two years ago (*MWJ* 1998.01.12). The revision to HFS to account for modern file system features and much larger volumes and files has been enormously successful—

reports of trouble and disk damage have been few and far between. The major problem has been that some of HFS Plus's modern features are inaccessible to applications. Files on HFS ("Mac OS Standard") disks cannot grow larger than 2GB each. HFS Plus ("Mac OS Extended") files aren't so limited, but all the Mac OS routines that programmers use to access files are. For example, programmers must be able to access every byte of a file, so the Mac OS allows programs to start reading or writing at any point in a file. However, the value they pass to the OS for this purpose is restricted to 31 bits, so there can't be more than 2^31 bytes (two terabytes) of data in any fork (data or resource) of any file. HFS itself has similar limitations. HFS Plus does not, but since the only way to access the files are through the HFS routines introduced in 1986 (and updated in 1991's System 7), the limitation is still effectively in place.

In Mac OS 9, Apple's file system engineers started adding new application program interfaces (APIs just like humans talk to the computer through the "human interface," programs talk to the OS through the APIs) for extended HFS Plus features. In addition to removing the limit of 348 open file forks, boosting it to 8169 ("Mac OS 9 Basics," this issue), Mac OS 9 adds support for reading and writing file forks larger than 2GB, setting and preserving HFS Plus's 255-character Unicode file names, reading directory information as easily as reading a file, and support for more than two forks in a single file.

Fork This

Different parts of Apple Computer seem to be moving in different directions concerning forked files. On the one hand, the Mac OS X people are cleaving to the OpenStep application architecture, which eschews resource forks. After all, OpenStep (and NextStep) didn't require file systems that offered resource forks, so applications written for OpenStep (and now for Cocoa, the OpenStep descendant programming environment native to Mac OS X) take what a Mac OS application would put in a resource and keep it in a separate, small file. Special folders located with the application hold all these files for each language version of the application. To keep you from diving into these folders and messing up the moral equivalent of a resource fork, the OpenStep Workspace Manager normally prevents you from seeing them-instead, the entire application folder looks like a single application file.

These are called *packages*, and they're supported in Mac OS 9, not just for applications derived from NeXT interfaces but for Mac OS programs as well. In Mac OS land, a package might include plug-in folders, scripts, configuration files—anything a developer wants tied to his application and doesn't want you messing with unless you know what you're doing. In the Technical Note describing packages, Apple points out that some of the files developers typically drop in the System Folder (help files, dictionaries, and so on) could just as easily live in packages as in the System Folder. It's not a bad recommendation, save that it's something like the fifth about-face Apple has pulled on the "where to store application-specific files" issue since 1991, so don't expect developers to react as if this is Received Wisdom.

The benefit of packages is that for most purposes they work like folders (the Finder and Navigation Services are two software components that treat packages like single units, not folders). They can be archived, distributed, backed up, and otherwise manipulated with standard file utilities. And they don't require resource forks, so they work even on file systems like UFS (the UNIX File System used in Mac OS X) that don't offer multi-forked files.

At the same time, however, the Mac OS File Manager team is moving in quite a different direction. Far from giving up support for the resource fork, the File Manager in Mac OS 9 at least theoretically expands support to many forks in a file—maybe even dozens. This is not as insane as it sounds. The Finder and other applications need to track *metadata* about a file—information about the file itself. File type, creator type, modification dates and times, creation dates and times, access privileges (new in HFS Plus), and even the backup state are all examples of metadata.

In the world as we know it, metadata is stored in a file's directory entry. That makes adding new kinds of metadata rather difficult, since directory entries are fixed-length and can't be revised without breaking every program that accesses disks directly (including most disk utilities). At some point, Apple might want to let applications like the Finder track such information with the file but not within a directory entry.

That's where the idea of extra forks becomes useful. The forks in the Mac OS 9 File Manager are named, not numbered, so no one has to arbiter the use of fork #2 or #15 or #99. The data fork has no name (the empty string), and the resource fork is named "RESOURCE_FORK". There might be an "EXTRA_DATA" fork, or a "FINDER_PRIVATE" fork, or who knows what else? Each fork would have its own contents independent of any other forks. When you copy or move the file, all the forks go with it, so any metadata or other information in the other forks stays with the file.

The downside is the same set of problems brought by resource forks, multiplied by however many forks a file has. The data in extra forks would regularly be lost

Watch This Space

(continued from page 45)

unless all programs that copy files see revisions to copy all forks, not just the data and resource forks. Compression and encoding programs and standards (StuffIt, Compact Pro, MacBinary, AppleSingle) would need more revisions to account for multiple forks. AppleDouble (an Apple standard that keeps all "Macintosh" information in a second file and leaves the data fork as one file, making both parts easy to transfer and reassemble on a target system) might be revised to keep all non-data forks in the second file, or it might become obsolete. And don't forget third-party developers-no one knows what they would do with all those forks. Some might try replacing every separate file in an application with a new fork in the application file, leading to huge single-file programs (imagine Microsoft Word 2000 in one file), an idea almost diametrically opposed to the one espoused in "packages."

At the moment, this is an academic debate—the APIs are present for multiple forked files, but no file system implements them, not even HFS Plus (it has theoretical support, but in Mac OS 9 you can't create files with more than two forks). HFS will never support more than two forks per file, and most non-Apple file systems will never support more than one. But on the other hand, multiple forks are a direct analog to extra file-specific information that already exists in modern new file systems like NTFS (the Windows NT File System) and UDF (the Universal Disk Format found on DVD discs). Multiple named forks might be necessary just to preserve those contents, regardless of whether Macintosh information winds up in other forks.

Extra forks could work well if they're used only to track extra information not intended to survive E-mail or some kinds of disk copying,. But without significant systemic changes, applications won't know about extra forks and will leave them behind when copying or sending files. If the eventual destiny of extra forks is to hold disposable information, that will be fine. But if interoperability with NTFS and UDF is a priority, it will mean more changes in how Macintosh applications handle files.

Evolutionary Changes

Multiple forks are the biggest change in Mac OS 9's File Manager, and as noted, they're still largely in the theoretical stage. The other major changes largely serve to bring the File Manager's APIs up to speed with the HFS Plus file system.

Since HFS Plus's release, the File Manager had been "dumbing down" HFS Plus's capabilities to fit the

older HFS model on which the File Manager's routines have been based since 1986. Although an HFS Plus disk can have files with names both longer than 31 characters and using Unicode characters that don't appear in your system's default text encoding (for example, a file name containing a Chinese character on a US English Macintosh), the File Manager in Mac OS 8.1 through Mac OS 8.6 "translated" these names down to an acceptable 31-character single-encoding name, embedding the file's catalog node number in the name if the file's Unicode name was otherwise unpresentable. It all worked fine-you could open, read, write, and otherwise manipulate the file-but you couldn't figure out what the real Unicode name was, nor could you change it. Similarly, HFS Plus disks could have files larger than 2GB, but no programs using the Mac OS File Manager could satisfactorily manipulate them.

Mac OS 9 includes a wide range of new routines designed for native access to all the features of HFS Plus, eliminating these headaches and inconsistencies. The new routines are designed to avoid future problems: like much of the Mac OS, the new capabilities are based on opaque data structures, meaning Apple doesn't tell developers how to see inside them. Instead of saying "A reference to a file contains a disk number, a directory number, and a file name," the new routines hide the information. They say, "Here is a reference to a file. We're not telling you what all it contains, but we'll tell you that it contains at least as much information as the old version. If you want any of the information out of it, pass it to these new routines and we'll give you back what you want. Don't try looking directly inside, because it may change."

Now, instead of "dumbing down" HFS Plus for old routines, the new File Manager APIs get to "smarten up" other file systems so programmers can use one new set of routines and forget about the other ones. The changes are designed so programmers of external file systems can implement them if they wish, but the File Manager will take up the slack if they do not. One new routine, by way of example, can provide information about all the files in a directory and in any subdirectories it contains in one convenient operation. This may be rather painless for HFS Plus because it keeps such information handy, but it would take quite some time to compute on other file systems (think "Get Info" on a directory in the Finder, and the amount of time the window says "Calculating size..."). The new File Manager takes the fast route for file systems that support it, but it falls back to the same manual way of examining each file directly for file systems that don't (like HFS and, probably, most external file systems). If you're lucky you get results much faster, but in the worst case you don't get them any slower. That's the rule of thumb for the File Manager additions in Mac

Mac OS 9 Hidden Features

OS 9. The new capabilities aren't earth-shattering, but they're necessary tasks in the world of huge disk drives and multi-lingual files.

What Mac OS 9 Doesn't Fix

When people hear about large file and long Unicode file name support in Mac OS 9, they're almost immediately discouraged because there's no easy way for them to use it. The Finder uses the new calls internally to properly copy large files, to gain a little speed, and to preserve Unicode file names wherever possible. Finder, however, didn't get a *major* overhaul in Mac OS 9—its internals are still largely limited to 27-character disk names and 31-character file names. Finder doesn't currently accept Unicode input for file names.

There are far more issues involved in full human interface support for HFS Plus than you may be considering, many revolving around the long Unicode file names. Apple chose Unicode for HFS Plus file names because those file names then wouldn't be limited to a single text encoding—you could put one or two Katakana characters in an English file name without difficulty. Displaying such a file name, however, is hard. Your system might lack Japanese fonts, and then Finder has to figure out how to display the name—and let you edit it—without screwing up the original file name too much. The Text Encoding Converter can help, but it's not magic.

And not to belabor the obvious, but 255 character file names are *long*. Finder has never had to deal with a situation where a file name on a single line might not even fit on the screen in your chosen "views" font. The trickiest part, however, may be deciding what happens when you copy a long file name to an HFS disk (like a floppy disk, since HFS Plus doesn't support such small disks). Does it get truncated to 31 characters? What happens if a file whose name has the same first 31 characters is already on the target disk? Should the Finder (or the File Manager) maintain a "parallel directory structure" with the HFS Plus information that doesn't fit on an HFS disk so it's there when you copy the file back? Adding support for these kinds of features isn't as easy as changing the number "31" to "255" in some file and recompiling the Finder.

While new programming methods eliminate some of the most complex methods programmers were forced to use under the old File Manager APIs, the system isn't exactly modern: it's still single-threaded, meaning it can only handle one file request at a time and can't be interrupted. Most of the File Manager is still in 68K assembly language because it's complex and it works correctly; rewriting it in PowerPC code could create lots of bugs for little benefit. HFS Plus and the new File Manager APIs are written in a highlevel language, so they're already portable to PowerPC, but they're compiled as 68K code to avoid the performance penalty involved in switching between the existing 68K File Manager code and any new PowerPC code.

Apple's engineers say that in normal cases, the path through the File Manager to the device drivers that do the actual work is pretty quick, and the device drivers are PowerPC native. It's the single-threading that makes Mac OS file I/O so slow. You'll notice that when Apple shows off its super-fast computers at demonstrations, it never runs disk-intensive tests demonstrating databases or Web servers or the like. That's because Windows, UNIX, and Linux can all process multiple file requests at the same time while the Mac OS cannot.

The File Manager additions in Mac OS 9 do not correct this, and quite frankly, we think it's unlikely to ever change in classic Mac OS because it's too big of a compatibility risk. Look at all the gnashing of teeth taking place over direct access to the FCB table and imagine what would happen if Apple changed some more substantial implementation detail to deliver multithreaded file access. Everyone professes that they understand Apple must break some very old programming techniques to deliver long overdue new features—until it actually inconveniences them personally.

The changes are about the future—they lead to a more robust transition to Carbon applications, provide support for HFS Plus features to developers (if not directly to end users), and eliminate some of the ambiguity that led to problems like error 119. You won't see much benefit to these changes when you install Mac OS 9, but as programs that implement the new features become available, you might even start using Unicode file names—outside the Finder, at least. It's a building block technology. The technically curious can explore more details about the new HFS Plus File Manager at Apple's developer Web site.

A Real Font Manager

If you do any serious work with text on the Macintosh, you can probably name at least four font management utilities off the top of your head (we came up with Adobe Type Manager Deluxe, Font Reserve, Suitcase 8, and Master Juggler Pro). Programs to activate and deactivate fonts, rearrange font menus, validate fonts, repair fonts, and otherwise boss them around are ubiquitous because the Mac OS itself has always lacked real font management capabilities. Third-party developers have just tried to take up the slack.

Watch This Space

(continued from page 47)

A History of Font Non-Management

The first Macintosh dumped all of its fonts in the System file as resources. They weren't large and had to be available at all times; the Resource Manager made that possible. (The same was true for the code that draws windows, menus, and controls, as well as the code for desk accessories, drivers, and even applications themselves—all stored in resources. It was the thing to do, and on a 128K system with a 400K floppy disk, it worked well.

For fonts, however, this design was fatally flawed. None of the original engineers foresaw what desktop publishing or even WYSIWYG word processing would do to their tightly-packed system. Apple's original specification allowed for only 256 different fonts, a number eclipsed within months. Apple had no plans for stylized fonts, and outline fonts were far beyond the computer's capabilities. Meanwhile, developers and typographers understood that fonts and resources were identical—each font worked exactly like every other resource, with no special management or other code necessary to make extra font resources work properly.

Apple started adding font capabilities piece by piece, yet everything stayed based in resources for compatibility. When the font family description resource and "new font" bitmap font resource were added in 1986 with the Macintosh Plus, the system overcame the limitation of 256 fonts per language, but at the cost of multiple resources per "font family" instead of a single 'FONT' resource. TrueType, in 1991, added 'sfnt' resources and outline fonts. Some of these were so large, especially for double-byte systems, that Apple had to add "partial resource" calls to the Mac OS—the traditional "read this resource into a chunk of RAM" paradigm fell apart on 4MB computers dealing with 2MB Kanji font resources.

Even at this stage, fonts and resources were still synonymous. This may be hard to grasp fully without a counterexample, so contrast it with QuickDraw pictures and the 'PICT' file and data type. Support for 'PICT' data is built into QuickDraw, the Mac OS component that draws everything you see on the screen. Like fonts, the 'PICT' data type is completely documented—developers can get inside it and build their own. But a QuickDraw picture could live in many places—a resource of type 'PICT', a file of type 'PICT', or just in a chunk of memory prepared by an application. QuickDraw deals with the idea of a picture—it will create one for you or draw one that you give it, but it does not care how you store them.

Fonts had no such luxury. For the Mac OS to find and use a font, it had to be stored in the right resource type in the System file. Developers couldn't say "this data is a font" or "this file is a font," or more usefully, "Here is a font-go store it somewhere and do what you want with it, but make it available to the system." Those who wanted control over fonts had to deal with resources, period. The overlap led to some unfortunate programming practices, as developers learned that they could generally use resource-oriented tricks with fonts. They could put resources in application files instead of in the System file and QuickDraw would find them-until you switched applications. They could give resources funny names starting with "%" and "." to keep the Macintosh from displaying those resources in a "Font" menu. They could override the Mac OS fonts with their own versions in separate files, even though QuickDraw was never prepared for the idea that the same font would really be different for different applications.

Adobe Type Manager, for its part, came in around 1987 and sat on top of all of this, patching into the deepest levels of font and resource interaction, waiting to see if its outline font rendering code would be needed for any given text. Apple had never intended any of those areas to be open to third-party developers, but Apple documented enough about *how* it worked for Adobe's keen engineers to get in the thick of the process and grab control where necessary. But as both QuickDraw and PostScript fonts have grown more complex, the Mac OS's text-handling code has grown to accommodate it. ATM has been stuck not only keeping up with these changes but trying to implement its own new features, both in font management and for PostScript font features. It's a nightmare.

The "Fonts" folder, added in 1993's System 7.1, engendered unwarranted optimism. The idea of not keeping fonts in the System file any more was certainly appealing, but behind the scenes, the Mac OS stretched almost to the breaking point to make it work. If fonts are just resources, every font has to be in an open resource file for QuickDraw to find it. Applications couldn't be revised to open all the font files every time they launched, and it wouldn't have worked anyway since many programs expected the fonts to live in the System file and the System file alone. To make the boondoggle work, Apple revised the Resource Manager to pretend that every font file was really part of the System file, through a horrendous hack that more or less merges each font file's listing of resources (the *map*) with the System file's resource map. The same code also has to resolve conflicts between fonts that have the same resource IDs (a part that was broken between Mac OS 8.1 and 8.6, according to Master Juggler Pro developer Alsoft). It was a classic Rube Goldberg system, and still failed to solve any of the core problems. What's worse, in normal situations, Macintosh users could drop so many font files in their "Fonts" folder that they took up more than a third of all available slots for open files (128 out of 348).

QuickDraw GX's release in 1994 finally brought a true font architecture to the Macintosh. GX unified all the various font formats into the TrueType font storage format (even PostScript fonts) and introduced the Open Font Architecture (OFA). For the first time, Mac OS text capabilities were built on a modular architecture designed for expansion in every way. Developers could not only add new fonts and completely manage the ones already in the system, but also add entirely new font types. TrueType or PostScript not enough? Write your own code to render an outline font and plug it into the Open Font Architecture; store the fonts in the TrueType format, and everything magically works. It was the solution that desktop publishers needed like no other, except maybe for a robust printing architecture that QuickDraw GX also provided.

Of course, GX failed, for a litany of reasons we won't repeat here. Printing is still as big a mess as it ever was, but the typographical model lives on in ATSUI—including the OFA. However, one of ATSUI's main goals was to make QuickDraw GX's advanced typographical features available in the world of regular QuickDraw. That meant working with existing fonts, not just with brand new ATSUI-capable fonts; one of the most frequent complaints about QuickDraw GX was that it required "converting" PostScript fonts into a new format that non-GX systems couldn't use. ATSUI eliminated this problem, but ditched the GX style of font management with it.

Font Management in Mac OS 9

The absence of real, built-in font management has been an intolerable situation for at least eight years, especially with content creation the focus of so many Macintosh customers. Mac OS 9 finally adds it: for the first time, the Mac OS treats fonts as unique objects. They may come from font resources, TrueType "data fork fonts" (the ".TTF" and ".TTC" files seen on Windows systems), Adobe Type 1 files, or even "OpenType" fonts that put Type 1 fonts in a TrueType wrapper as QuickDraw GX did. The Font Manager knows about these fonts and tracks them, making details about them available to any program that wants a font inventory had to do this work itself, usually by directly accessing the font resources.

The Font Manager now handles the basic font management tasks that applications have craved for years. In Mac OS 9, the Font Manager can count fonts, enumerate them one by one (with complete information), and provide the TrueType-style font tables for each font—even those not stored in TrueType format. Since the Font Manager now abstracts various kinds of fonts into a standard "font object," it can rely on its modules and the OFA scalers to convert native data into a standard format all programs can use. We can't overstate how important this is—reasonable data on all fonts can now be available to programmers who haven't spent months or years studying all available font formats and how to find them on the Macintosh. No more "do this for bitmap fonts, this other thing for TrueType, and this third thing for ATM depending on the version number."

Using opaque data types (as the HFS Plus File Manager does), the new Font Manager code still tracks font families by a reference number-but this time, it's an opaque number, not transparently the ID of a 'FOND' resource for that font family. Programs use the new font family reference just as they used the old "font number" (really a font family resource ID) before; QuickDraw now knows how to deal with all of it. New Font Manager routines provide all data programmers formerly had to extract by knowing how the font subsystem worked. The font families are sets of opaque references to actual fonts-a family might be "Times," while actual fonts could be "Times Roman", "Times Bold", and so on. The font family references work with QuickDraw, and the font references work with ATSUI, unifying the two text drawing systems for the first time.

Even though Mac OS 9 allows more than twentythree times as many open files as Mac OS 8.6 and earlier, Apple expects that professional customers will still want to activate and deactivate sets of fonts for particular uses and jobs. The new Font Manager has that built-in as well, allowing for the arbitrary activation of fonts directly from their files, with deactivation later whenever you want. The old system can't do that deactivating a font in the middle of an application might disturb its resource chain and cause a crash. Programs that use the new font management APIs, however, get activation and deactivation support as part of the conversion.

The Font Manager even steals a trick from Suitcase circa 1990: every time Suitcase changed the active fonts, it incremented a certain location in low memory. Suitcase-savvy applications checked this value every time you brought them back to the foreground and, if it had changed, rebuilt their font menus. (Most of the major desktop publishing applications do that to this day.) Apple can't exactly mimic the trick because they're discouraging the use of fixed-address values, so they built it into the API. It's called a *font generation*. A program can ask for the font generation

Watch This Space

(continued from page 49)

whenever it wants; it works like the old Suitcase number. When the number increments, the font collection has changed and it's time to rebuild font menus.

Speaking of rebuilding font menus, the new Font Manager does that, too. It includes functions to both create and update a "standard" font menu—a *hierarchical* font menu, ordered by font family, built into the Mac OS. Applications that switch to the new font management routines get this much-admired feature with almost zero additional programming—a developer's dream. The Mac OS 9 version of the Font Manager doesn't include a way to extend the "standard" font menu, but it's marked as important to consider for future releases.

The new Font Management system has the potential to eliminate strange resource dependencies in text drawing. Remember how Adobe Type Manager requires you to install at least one bitmap version of any PostScript font you want it to use? That's because the text drawing system is based on those pesky resources—if no QuickDraw font is installed, no program will see it and ATM can't patch in to take over drawing for it. The new Font Manager should treat each PostScript font as its own unique entity, eliminating the "at least one bitmap" requirement.

Backwards and Future Compatibility

For Mac OS 9, the new font features are a *parallel* API to the older, resource-based font system. The hundreds of programs that walk through resources and examine fonts directly still work—in fact, if we understand the implementation correctly, it seems that if a new program activates a PostScript-only font globally (for all applications), the Font Manager will synthesize a 'FOND' resource so older applications still work properly. The API also includes ways for third-party font management software to hook in and do its magic; the version in Mac OS 9 incorporates development-level comments from DiamondSoft, makers of Font Reserve, and other font management vendors.

Apple is unfortunately ambiguous about the future—developer documentation strongly implies that the older system may go away in the next major Mac OS release or in Carbon, but if that's the case, the company needs to be stating this *much* more strongly than it is. (Failure to make perfectly clear that a formerly-good programming technique is really and truly going away leads to problems like Mac OS 9's new error 119.)

Developers need a push—the new ways are far easier and more robust than the old ways, but the new ways only work on Mac OS 9 and later, ruling them out for programs that should work with Mac OS 8.6 and earlier. The most clear language we could find said that the older, resource-based system of text drawing would be "deprecated" in Mac OS X—implying that it will still work but that programmers should change away from it as soon as possible. Other passages imply that the older compatibility routines are to be present only in Mac OS 9. Apple has released the technical developer documentation on the Mac OS 9 Font Manager, but it also fails to make the future of the older system perfectly clear and unambiguous. Apple should do so as soon as it can.

A Big Font Deal

In Mac OS 9, for the first time, Apple has provided a robust, supportable, backwards-compatible set of font management routines that developers can fully exploit without having to rewrite every part of their text drawing code. That's great news. It's even better that Mac OS 9 users see benefits like hierarchical font menus, better integration of various font types, a connection between QuickDraw applications and ATSUI applications, and built-in *real* support for activating and deactivating fonts.

The new font management system is monumentally important, especially if Apple successfully migrates developers to using it. Fonts have always been the unwanted stepchildren of Mac OS graphics; previous attempts at more robust font handling have always been coupled with more industrial-strength typography systems that developers never adopted *en masse*. Apple realizes the new Font Manager won't provide every feature every professional customer needs, but it also realizes that the Font Manager must be the foundation on which all future font-handling tools are built.

Early implementations may contain bugs—since the resource-based font process is one of the most abused parts of the Macintosh, there's every reason to expect problems in the most font-intensive applications. They'll be fixed, though, and the Macintosh world will be a better place. These features are *long* overdue. When they take full effect within Macintosh applications, you'll see why we're so excited.

HTML Rendering Library

Every now and then you hear about a new feature in software and you think, "Wow, that's really going to help," only to find out later that it's not nearly as powerful as you thought it was. Such is the case with Mac OS 9's built-in **HTML Rendering Library**. You're forgiven if you thought this was something like the integration of Microsoft Internet Explorer into Windows—suddenly all applications had the same

Page 51

HTML capabilities as the Web browser itself, turning HTML into as core a data type as text or images or movies.

It would be nice, but it's unfair to be disappointed in the HTML Rendering Library because its goals were never so lofty. Mac OS 8.5 introduced "Help Viewer," a new application for displaying HTML help files. The HTML Rendering Library is the engine driving Apple Help; Mac OS 9 just makes it available to other programs.

The HTML Rendering Library is designed to take over a window and render HTML in that window, be it from a file, a file URL, or data already in memory. It can draw in a smaller area than a window, but that area will contain scroll bars if necessary. It contains several routines programmers can use to manipulate URLs (finding the root URL for anchors, extracting page titles, changing file URLs into file specifications, and so forth). The library itself has no link-tracking facilities—it only renders HTML, it's not the full Help Viewer application. However, the HTML Rendering Library does allow programs to hook in to track URLs, keep history buffers, and redirect URLs appropriately.

The new library does a decent job; you can try it out by opening HTML files in the Help Viewer application. It's a lightweight rendering engine, handling HTML 3.2 or earlier only. Because it's simple, it's fast—significantly faster than either major Web browser for the same HTML files. And it's unencumbered by support for plug-ins, networking code, or the other tasks a Web browser has to manage. Most of the HTML files we tried rendered quickly and well. It couldn't quite handle some of the pages on our Web site, probably because of the intricate tables used to place objects on the page without forcing a given page width.

Without built-in code like the HTML Rendering Library, HTML on the Mac OS has been restricted to the Web browsers and to the big applications that can justify implementing part of an engine (like Eudora Pro). The new library is neither an integrated Web browser nor a state-of-the-art HTML powerhouse, but it should make HTML-based data a little more prevalent in Macintosh applications; it might even let you occasionally see HTML data in documents, or lead to simple HTML-based utilities (HTML to 'PICT' converters or similar tools). The HTML Rendering Library is perfect for programs like Help Viewer where authors want the convenience of HTML and programmers don't want the hassle of drawing it. We like it.

The Texty Generation

Programming a Macintosh text editor is hard. System 7.1-level text editing features are difficult

enough: multiple fonts, styles, and sizes (meaning variable line heights and variable character widths on every line), aligning text on the left or right or both (or centering it), mixing multiple languages in a single document or even a single line, dealing with both left-to-right and right-to-left text in a single line, properly accounting for font metrics including italics that slant outside of each character's rectangle (MDJ 1996.12.05), proper clipboard use of both text and styles, and even full color support. We discussed the difficulties in implementing these features when MacCyclopedia dissected **TextEdit**, the built-in text editing facility of the Mac OS (MDJ 1997.04.30). Programmers still use it, despite its annoying 32K limit on text and lack of support for tabs or pagination, because it makes the necessary chores easy and manageable.

Routine Obsolescence

That was the state of the art eight years ago. Since then, text has taken off in new directions, starting with Unicode. The Unicode standard (which Apple helped develop) attempts to make it easier to get more of the world's languages onto computers by encoding every major character from every major written language (including the thousands of Japanese and Chinese characters), using two bytes for each character instead of the one byte used by most existing text encodings. TextEdit has absolutely no concept of Unicode, and its limit of 32K bytes per record would limit it to 16K Unicode characters even if it did.

QuickDraw GX Typography, released with QuickDraw GX in 1994 (*MDJ* 1997.01.01), added major typographic capabilities to the Mac OS-the kind of stuff that some programs like QuarkXPress and Adobe InDesign still don't have. Examples include automatic ligature formation, context-based glyph substitution (such as substituting a real fraction for two numbers separated by a fraction bar, or using an em dash instead of two hyphens), access to hundreds or thousands of glyphs in a given font, optical alignment, hanging punctuation, powerful kerning and tracking, multiple character ligatures, and dozens more. QuickDraw GX is now truly dead; even the smaller "GXGraphics" extension is unsupported and unusable in Mac OS 9. ATSUI implements most of QuickDraw GX's typographical concepts, and we'll probably review those in the months ahead when MacCyclopedia examines the current state of advanced Mac typography.

Here, the point is that GX and ATSUI add many powerful features regarding placement and choice of glyphs that are totally beyond the scope of a simple

Watch This Space

(continued from page 51)

editor like TextEdit. These aren't just typographical affectations, either: languages like Arabic and Devanagari aren't as "simple" as English in that the actual shapes of glyphs change depending on context (i.e., which glyphs they're next to on a line). Systems like GX and ATSUI can correctly render text in these languages without hand-tuned adjustments; TextEdit and others based on simple character models cannot and never will.

While both QuickDraw GX and ATSUI include support for these features, they're both line-based systems that have no built-in support for text that spans more than one line. They'll help applications figure out where to break lines (since such decisions aren't always obvious in non-English languages or with complex white space characters), and ATSUI even keeps line breaks in its own structures for help in drawing multiple lines of text, but that's as far as it goes. Simple-seeming tasks like selecting multiple lines of text get complicated in these models. Managing the human interface for the GX and ATSUI font model is also difficult, as neither architecture is much help with fonts (thankfully, the new Mac OS 9 Font Manager is).

The evolution of Mac OS text handling, then, has created a gap. On one hand, the venerable TextEdit is still around, with full support for 1991-era features, but programmers have to glue in anything above and beyond them—even drag-and-drop support. And TextEdit is still limited to 32K of text, even on machines with 1GB of RAM (for perspective, 32K is the square root of 1GB, so a huge machine could hold well over 30,000 TextEdit records, none of which could overlap). On the other hand, ATSUI offers the most advanced typography of any personal computer operating system available today, but the Mac OS offers no simple text editing library to go along with it. Translation: developers must either find an alternative to TextEdit, invent their own, or do without ATSUI.

Developers, still big on not reinventing the wheel, have generally stuck with TextEdit. Those demanding more than 32K of text have typically turned to WASTE, Marco Piovanelli's "WorldScript-Aware Styled Text Engine." WASTE 1.3 includes all of TextEdit's features, without the 32K limit. It also adds built-in undo, drag-and-drop editing, justified text (both left and right sides), the ability to embed objects like pictures in the text, and customizable low-level ways to modify how WASTE draws and measures text. Moreover, both WASTE and its source code are free—the only "fee" required is sending Piovanelli a full copy of any program that uses it.

Piovanelli is working on WASTE 2.0 (the current version is 1.3), a revision now in alpha testing to

include multiple undo and redo support, separate rulers for each paragraph, Carbon support, built-in routines for printing and for matching text, and Unicode translation (though not native Unicode editing). WASTE 2.0 will still be free, except commercial programs needing source code will have to pay a onetime US\$100 licensing fee. No version of WASTE, however, supports ATSUI or includes anything like its layout features.

So the Mac OS's text-handling dilemma has remained: an incredibly advanced typography still sits inside, where it's been since Mac OS 8.5, and very few applications are using it. Apple needed to get developers on board.

The MLTE Answer

Mac OS 9 attempts to do so by including Multilingual Text Editor 1.1, a new set of system-level services designed to parallel TextEdit but with modern features and programming models. Although debuting with Mac OS 9, the editor (abbreviated *MLTE*) works on systems as early as System 7.1, picking up where the last major revision to TextEdit left off (although MLTE only runs on PowerPC processors). On such systems, MLTE uses QuickDraw and the Script Manager to measure and draw text, just as TextEdit and WASTE do. However, MLTE automatically finds and uses ATSUI under Mac OS 8.6 and later. Programmers don't have to worry about the differences unless they want to use ATSUI features-the same code works with either architecture. It just provides better results with ATSUI's superior typographical engine.

MLTE eliminates TextEdit's 32K barrier. It also supports tabs, justification, and margins, called a *ruler* in classic MacWrite terms, but like TextEdit, MLTE 1.1 supports only one ruler per text record. MLTE supports 32 levels of undo and redo, including a way to return text to a program to display in a menu to describe what action will be undone or redone (such as "Undo Typing," "Undo Format Change," and so forth). TextEdit has no file-handling code, but MLTE includes routines to read and write several kinds of files: plain text files, text files with common style resources, Unicode text, and a new MLTE-inspired format that supports text or Unicode with pictures, movies, and sounds.

MLTE implements the full Macintosh Human Interface Guidelines for text editing, complete with all arrow keys and modifier equivalents, as well as the drag-and-drop text guidelines (including copying the selection instead of moving it if you hold down the Option key). It provides applications with all the support they need for dragging and dropping text into MLTE records from other applications, too. MLTE Mac OS 9 Hidden Features

provides full "Edit" and "Font" menu support (not using the new Font Manager routines, unfortunately, because MLTE works on earlier systems where they're not present), and builds separate font menus for QuickDraw and ATSUI clients reflecting differences in how the two architectures treat font families. (MLTE isn't hung up on its "Font" menu—programmers can use the new Mac OS 9 Font Manager API instead if they wish.) It even sports a "password" mode where all characters are drawn with a single glyph, like a bullet or a diamond.

MLTE has been in progress at Apple for a long time—it was scheduled for Mac OS 8.6 but pulled from the release shortly before the beta cycle began.

Does MLTE Solve Anything?

We think so, but developer adoption won't be a cakewalk. MLTE's programmer interface is somewhat more complicated than TextEdit's. Like the new File Manager and Font Manager, MLTE uses opaque data structures. In contrast, TextEdit's innards are completely transparent, giving developers lots of flexibility but simultaneously tying Apple's hands—developers using those innards means Apple loses backwards compatibility if it changes them. Programmers accustomed to total control over the editor will probably lean towards WASTE, where source code is still available and the API is more TextEdit-like.

Apple is understating MLTE's real benefit: it works equally well with QuickDraw or ATSUI, meaning developers can write to MLTE's features and know they'll get the best possible representation. Benefits in international markets will be even better. One reason developers use TextEdit and WASTE is because they know their programs will automatically work with various languages and writing systems-both TextEdit and WASTE feature built-in support for Japanese, Chinese, Korean, and other "difficult" script systems. However, as we've noted, that simple character model is not sufficient for correct visual representation of some languages. ATSUI handles this automatically, whereas both TextEdit and WASTE 2.0 can't-it's just too difficult to implement. ATSUI inherits from QuickDraw GX, which in turn inherits from the "Line Layout" project started at Apple in 1989. It includes at least 15 programmer-years of coding from some of the most linguistically talented programmers in the world. Marco Piovanelli is very good but he can't duplicate that gargantuan effort.

But for this feature to work at all, Apple must follow through, as it did with the Appearance Manager, and release a version of MLTE that works as far back as System 7.1. (Although the MLTE code itself works on older versions of Mac OS, that does little good if Apple doesn't release it as a separate extension qualified on those systems.) Until and unless they do so, there's a ton and a half of code in MLTE that serves no purpose whatsoever. If MLTE is only available for Mac OS 9 and later, all the work in it to work with QuickDraw and the Script Manager is wasted space— Mac OS 9 and later always contain ATSUI, so the older compatibility code is never needed. It's that same code on pre-Mac OS 8.6 system that make MLTE so valuable, but only if Apple releases and licenses it in a timely and inexpensive way.

Developers have already outgrown MLTE 1.1's limited support for some features. One example is tab support. MLTE's idea of a "tab" is a fixed-width interval that repeats across the available area. If your tab stops are "one inch" and your MLTE editing area is seven inches wide, you get seven tab stops (counting one at the right edge, or left edge if using a right-toleft language). A tab character makes the next character draw at the next tab stop. If you then type more characters before the tab, they fill in the white space gap until the characters reach the tab stop; the tab then pushes the next text out to the next tab stop. The data structures offer future support for various types of tabs (left, center, and right; decimal tabs aren't mentioned), but MLTE 1.1 supports only left tabs. There's no support for irregularly-positioned tabs like those allowed by every word processor since MacWrite 1.0. And while MLTE does add printing support (TextEdit is infamous for having none and not reacting well when used with printer drivers), it's so basic that there's no obvious way for developers to combine printing an MLTE text record with anything else on a page (like a header or footer, for example). That won't fly very far.

Another problem isn't really MLTE's fault, but it prevents the new technology from advancing the cause of advanced typography. ATSUI, like its ancestor QuickDraw GX, offers dozens of advanced typographical features based on intelligence built into fonts by their designers. Since the advanced capabilities depend on what the fonts offer, and since there are dozens of options, building a standard human interface to let you choose your favorite features is a mammoth task. Unfortunately, Apple has responded to this so far by punting. QuickDraw GX included no recommendations for how to present these complicated options to Mac OS users, and neither does ATSUI. MLTE could have taken up the gauntlet, but it didn't. Developers don't want to take on that much work for something Apple might replace with a standard in the future. Thus, the most powerful feature of ATSUI typography is still remarkably invisible because no one has bitten the bullet and designed a consistent way to present it to people who might want to use it.

Watch This Space

(continued from page 53)

Apple *really* needs to resolve this, because ATSUI font capabilities far exceed those of the similar "OpenType" standard. Microsoft and Adobe, OpenType's creators, say OpenType's model is superior because programs aren't "locked in" to the algorithms built into the system software. ATSUI and GX supporters say that argument is a crock—developers can override or turn off every font feature, and leaving the algorithms out of the OS just ensures that the advanced features show up only in programs from companies large enough to spend thousands of programmer hours recreating them. Which companies might that be? Why, Microsoft and Adobe, of course (see *The Weekly Attitudinal*, "Welcome to ClosedType!", *MDJ* 1997.04.25).

Proof In The Pudding

Apple can fix most of these problems—it can release MLTE for System 7.1, fill in the holes, and add the features that didn't make it into the first release. But that won't be enough. As with Navigation Services, Apple must start using MLTE in its own programs in place of TextEdit. If Mac OS 8.5 had included Navigation Services, but all of Apple's programs still used the older Standard File operations, how many people would have seen the advantages of the newer "Open" and "Save..." dialog boxes or pressed developers to adopt them? So will it be with MLTE. The Finder, Note Pad, Scrapbook, SimpleText, Script Editor, and other programs included with Mac OS have to commit to MLTE before developers will bother doing so.

We hope Apple makes this commitment, because MLTE's concepts are important. Adoption of ATSUI and Unicode will advance the Mac OS, both typographically and in worldwide markets. Apple has long held a technological lead in these kinds of features but has been less effective at translating that lead into "must have" products. MLTE is another chance for the new, improved Apple to carry through on its own promise.

The X Files

Finally, we'll look briefly at Mac OS 9 technologies aimed squarely at Mac OS X. At the 1998 Worldwide Developers' Conference, Apple announced the Mac OS X strategy. The new OS is to be based on a UNIXlike core, with Next's foundation and application environment (now called "Cocoa") on top of a Mach kernel. The existing Mac OS, now at version 9, maintains its traditional architecture with incremental improvements like those we see in this release. For compatibility, Mac OS X will run existing Mac OS applications, unmodified (provided they don't directly touch hardware), in a form of Mac OS "emulation" called the "blue box." The downside is that such applications don't get any of Mac OS X's marquee features—protected memory, preemptive multitasking, and *dynamic resource allocation* (where the operating system provides what is needed when it's needed, instead of requiring you to figure out things like application partition sizes in advance).

This was basically the "Rhapsody" model, and it has a serious flaw. Only new applications get the benefits of the new operating system; older applications gain no benefits unless they're virtually rewritten from scratch. Very few developers were interested in that proposition. Under Steve Jobs's leadership, Apple took a step it was never willing to take before—introducing a new, third application model; one heavily based on the existing Mac OS, but discarding all the parts of it that prevented Apple from adding the modern features to earlier efforts like Copland.

The interim layer is called **Carbon**. It works for two reasons: because it lets developers bring their existing Mac OS applications to Mac OS X with full features and relatively minor modifications (a few months instead of a few years of coding), and because Apple pledged to bring Carbon's code to the existing Mac OS as well. It's a sound strategy-developers who take the time to write their applications to the Carbon specification write one application that gets all the advantages offered by both the classic Mac OS and by Mac OS X. Carbon is almost entirely based on the classic Mac OS, but its lowest-level services are more based on the Next "Foundation" classes. The reworked version of that Next material supporting Carbon is called Core Foundation. Both it and Carbon see first public release in Mac OS 9. Let's look at Carbon first.

Carbon 1.0: Slowly Starting

Today's PowerPC applications connect to PowerPC-native Mac OS code through shared libraries. Some are stored as separate files; some are built into the System file. The major portion of the PowerPC-native Mac OS is in a library called "InterfaceLib," and all PowerPC-native Macintosh applications *link* to it. Carbon applications bypass that in favor of "CarbonLib," a library stored in a file of the same name in the Mac OS 9 extensions folder. CarbonLib contains the Carbon 1.0 release. It's a cautious step—it loads *only* on Mac OS 9. It will not load on either earlier or later versions of Mac OS, to prevent people from taking this early release back to Mac OS 8.6. Apple has promised to release Carbon for the older 8.x versions of Mac OS, but this version doesn't qualify.

Carbon aims to change the existing Mac OS so that the transparent data structures that have caused so many compatibility headaches go away. Since programmers already have to change their programs to gain Carbon compatibility, it's the perfect chance to give them new ways to perform tasks other than looking directly at internal Mac OS structures Apple documented fifteen years ago. For programmers, the changes are minor. Instead of peeking into a fixed location in memory to find the height of the menu bar, Carbon applications ask the system (either Mac OS or Mac OS X) to just tell them how tall the thing is. In just about every place where programs have previously peeked, they'll now switch to asking. Each such change is minor, taking mere seconds in a program's source code.

However, there may be thousands of them to make in a large application. From QuickDraw on up through the human interface tools built on it, fifteen years' worth of transparent data structures are being replaced with "black box" opaque data structures. Unfortunately, there is no real documentation on most of these changes as Mac OS 9 hits the shelves. Those in Apple's developer programs have more access to Carbon information than does the general public, but even that information is lacking.

For example, dozens of data items in QuickDraw drawing environments (called graphics ports) are hidden in Carbon, because the graphics port is an opaque data structure-programs can't peek into it. To compensate, Carbon adds dozens of small functions to get and set the information from the opaque structure. Instead of looking in a graphics port for the current font size, a programmer asks Carbon for the font size of the given port with the new call GetPortTextSize. Pretty simple—if there were any documentation of the new QuickDraw routines. But the interface files that tell compilers how to work are all that's available right now. Most of the new QuickDraw routines are pretty easy to understand, though, and the more complex changes to higher-level system software (the Window Manager, the Menu Manager, and so forth) are covered by preliminary documentation.

Carbon's Printing support sees some of the most major changes, largely because the existing Mac OS printing code is an abomination. The Macintosh lacks a real printing architecture—programs basically pass their printing requests almost directly to printer drivers, which do all the work. Even the simplest of tasks is convoluted in Macintosh printing because no printer driver is *required* to support more features than the original ImageWriter driver did in 1984. Have you ever used a program that told you to pick "Page Setup..." and set the page orientation to landscape (sideways)? It seems dumb, but printer drivers aren't required to support different page orientations, so there's no way for the program to *tell* the printer driver "use landscape mode." The driver may have no clue about it.

Carbon Printing tries to fix some of these problems—like adding a way to set the page orientation. Such changes repair some of the worst deficiencies in the current printing non-architecture, but they stop short of making it completely usable. Printer drivers are still free to support-or not support-features like arbitrary rotation, dashed lines, and polygon smoothing. Embedded ColorSync profiles are still handled through QuickDraw picture comments; a driver may or may not support them, and applications have no way to determine which is which. Carbon's printing fixes, welcome as they are, may cause problemseither the new functions won't work with all printer drivers, or drivers will need updating for Carbon. We suspect the former, since there's been no word of new printer driver specifications, and an explicit goal of Carbon printing is to use Mac OS printer drivers under Mac OS and a new Mac OS X printing system on that platform.

Apple has also fallen short on a few Carbon promises. The company said all new routines in Mac OS 8.5 and later would appear in Carbon as well, but Carbon 1.0 hasn't quite caught up to Mac OS 9 (specifically, the new HFS Plus File Manager APIs are not included in Carbon 1.0, though they're promised for a future release). There's no word on when Carbon will come to older versions of the Mac OS, and other details remain decidedly sketchy.

What's The Good News?

The good news is that it's here, and that it works. Apple's commitment to Carbon is clear when examining the programming material for Mac OS 9, some of which is now available. The "Universal Interfaces" that Apple releases for all developers to use in building their Macintosh applications are now thoroughly "carbonized," with every Mac OS routine not available in Carbon clearly identified. This allows programmers building Carbon applications to find out whether they're using non-Carbon code sooner rather than later. Previously, the application-building process would fail in the final step (linking); now, non-Carbon routines cause errors during compilation—the first step.

Developers can build Carbon applications *today*. Script Editor in Mac OS 9 is a Carbon application, and it can also save script applications ("applets") as Carbon applications (although for now they'll only run on Mac OS 9). Programmers can not only begin

Watch This Space

(continued from page 55)

the Carbonization process, but also test with shipping code on shipping systems. Almost all of the new Mac OS 9 features are included in CarbonLib, at least according to the Universal Interfaces, making Carbon a real option for programs that already require Mac OS 9.

Most, if not all, of the key Carbon questions seem decided. It's full steam ahead from here.

Core Foundation 1.2

OpenStep, the Next-created application environment, is universally hailed as one of the easiest and most powerful systems ever invented for programming personal computers. Every operating system needs a set of core services so every application doesn't have to duplicate them, and Next's services are some of the best out there. Kept in the "Foundation" classes of the object-oriented OpenStep system, they provide useful features in ways that make programmers smile. For example, existing libraries and systems ask programmers to allocate memory for some object, then fill it in, then manage the memory until done with whatever is occupying it. In the Next world, programmers just create the object they want; OpenStep takes care of the memory allocation, even allowing programs to share the object amongst themselves. When each program is done, they tell OpenStep they're done with the object; the system automatically releases it when no program is using it any longer.

These kinds of conveniences can, on a large project, shave weeks off the total programming time. The drawback is that, as object-oriented classes, they've only been available to object-oriented programs written in languages compatible with OpenStep. Carbon changes that, bringing some of the core OpenStep functionality to the Mac OS world through a series of services called "Core Foundation."

We're not going to examine Core Foundation in detail because it's the kind of tool only a programmer could love. We just note for the record that Mac OS X developers *will* love them. It'll suffice to list some of the capabilities available though Core Foundation tasks common in the modern world of computing yet not built into the Mac OS:

• Opaque types for all data, easing future compatibility and reducing dependence on implementation details, leaving Apple free to change Core Foundation's insides without breaking programs

- Sharing of data between programs through common types and the ability to discover information about an object
- Built-in support for Unicode strings and enough utilities to manipulate them more powerfully than a standard "C" compiler can manipulate regular ASCII text
- Optimized storage for Unicode text—not every character requires two bytes for storage—allowing compact representation of arbitrarily-long Unicode strings (they're not limited to a few hundred characters)
- Support for the same concepts as in Mac OS 9 packages, complete with a built-in easy-to-use plugin architecture any application can use
- Built-in routines for easy internationalization, allowing one application to contain several language versions with correct run-time selection of the appropriate one
- Standard support for the kinds of structures programmers use in their code every day—arrays, dictionaries, sets, "bags," and trees—for storing large amounts of data and referencing it quickly
- New routines that store and retrieve Core Foundation data in XML files, allowing programs to use the hot standard as a native storage format

Documentation on Core Foundation, like that of Carbon, is incomplete; the XML part apparently has yet to be documented at all. But there's enough present for programmers who want to use the services to get started. Core Foundation isn't likely to be too important for developers porting existing Mac OS applications to Carbon, but it's sure to be a big hit with those starting new applications.

The Carbon and Core Implications

The Carbon strategy has been the "future" for nearly a year and a half. In Mac OS 9, Apple has released the first deliverable versions of that. The documentation isn't complete, and some functionality is still missing, but it's mostly there. Most observers expect a new Mac OS X developer release before the end of the year. That should kick-start preparations for the new OS; most developers have been largely unconcerned about Mac OS X while so much has been going on in Mac OS 8.5 and Mac OS 9 (and with good reason). New work on Mac OS X will revive its importance for most developers, because it's clearly the next major revision of the Mac OS (after all, there's not going to be a Mac OS 10.0 before Mac OS X). Carbon and Core Foundation in Mac OS 9 make that work much less theoretical—if a company decides to totally go Carbonic and prepare for the new OS, they can still release their applications for Mac OS 9. It's no longer a bet on an unreleased operating system: Carbon is here now.

When Apple fulfills its promise to bring Carbon to older versions of the Mac OS, it will clearly be the preferred path for application development. Why use the fifteen-year old transparent data structures when the new Carbon methods are just as easy and work under Mac OS X? Why reinvent the wheel with common computing tasks when Core Foundation does it easily and with panache? Thanks to Carbon in Mac OS 9, the future is now.

The Invisible Benefits

These are not features you often see discussed in detail. Installing Mac OS 9 doesn't let your existing applications write files larger than 2GB, or use Unicode file names, or magically see ATSUI text editing, or overcome the limitations of the "Fonts" folder, or even start rendering HTML in existing windows. Everything we've discussed in this article requires changes from application developers—changes you're not likely to see in a week or two.

They are, however, the building blocks of future applications. There has been no ATSUI-capable text editor before Mac OS 9; now there is, and it may become available for older systems. Carbon was vapor-ware until Mac OS 9; it's now a viable alternative for future application development. HFS Plus features were accessible under Mac OS X Server but not under the classic Mac OS—until Mac OS 9. Developers now have the tools they need to bring great new features to your applications. Watch these areas for future developments, hopefully exciting ones.

.....

Mac OS 9

The Nitty-Gritty Everything Else We Could Find

In 1996 and 1997, Macintosh owners complained (with some justification) that Apple Computer was focusing all its resources on the "next great OS"—first Copland, then Rhapsody. Mac OS 8's release in July 1997 started to change that perception, but not completely, as it was widely known that the release was scheduled to be "Mac OS 7.7" until Apple changed the version number to gain leverage in Mac OS licensing contracts (*MDJ* 1997.03.06). Mac OS 8.1 fixed bugs and added HFS Plus support, but was still easy to completely grasp (*MWJ* 1998.02.02).

Mac OS 8.5 (*MWJ* 1998.10.19) substantially changed the playing field, packing in so many changes both visible and invisible that it was Apple's most significant Mac OS release since 1991's System 7. Mac OS 8.6 (*MWJ* 1999.05.12), though ostensibly a minor revision, had almost as many changes as Mac OS 8.5, seeing revisions and improvements in almost every major component. Each of these releases contains so many updates that they're almost as big as all changes from System 7.5.3 through Mac OS 8 combined.

Mac OS 9 continues this recent pattern. We've so far covered the "Nine Internet Power Tools" and basic

hardware requirements, plus six invisible technologies that open doors for better products in the future. But Mac OS 9 includes *many* more changes besides. In fact, we haven't even mentioned much of what you'll see once you start using your Mac OS 9-powered computer, because so many of the visible changes are little ones—hundreds of them, most of them requiring no more than a sentence to mention, but each potentially affecting how you use the computer. For example, you may notice some applications in Mac OS 9 and later using very large cursors where only software with "Kai" in its name did so earlier. That's because QuickDraw in Mac OS 9 allows cursors larger than 16 pixels square, but it doesn't take more than a sentence to let you know that.

We call these changes the "Nitty-Gritty" of a Mac OS release, and we've saved them for last in our Mac OS 9 coverage. First we'll examine Control Panels, then proceed to Apple Menu Items; other Applications, Utilities, and Extras; Extensions; and finally, the System File.

(continued from page 57)

Control Panels

One major change in Mac OS 9 affects *all* Control Panels: they run as separate applications. The change has its history back in the days before the Macintosh could run multiple applications at once. In those days, before System 5 and MultiFinder (remember then?), a running application owned the entire machine. The only way to run multiple programs at once was through *desk accessories*—small programs like the Calculator, Note Pad, and Scrapbook that, through special programming rules, were actually hosted by the running application.

MultiFinder changed that because not to do so would have created a strange human interface-desk accessories you opened in one application would require that application to be frontmost to use them. Open the Calculator in Microsoft Word (3.0, of course) and you'd have to switch back to Word to use the still-open Calculator. Instead, MultiFinder used a new application, "DA Handler," to host all desk accessories. It patched the Mac OS to take over the standard ways applications opened desk accessories to effect these changes-in fact, it's not until Carbon 1.0 that applications can finally get rid of those older mechanisms. For compatibility, you could open a desk accessory while holding down the Option key and MultiFinder would still allow the application to open the DA in the application's partition.

When MultiFinder became the "Process Manager" in System 7, desk accessories became stand-alone processes. System 7 and later create a small application to host each desk accessory individually, allowing each to appear in the Application menu under its own name. The compatibility work-around was eliminated—all desk accessories open in the new way, since DA developers had well over three years to revise their code to work with the new method. (In fact, Apple officially told developers to avoid writing desk accessories; they should instead write small applications to gain more capabilities and to exploit System 7's multitasking.)

Control Panels are a similar but subtly different story. Until System 7, the *Control Panel* was a single desk accessory. The left side of the window held a scrolling list of the different "panels," and the right side contained the controls for each panel. As a single DA, the Control Panel worked like any other desk accessory. System 7 changed the model. The panels (always stored as files of type 'cdev' in the System Folder) were now hosted by the Finder itself instead of by a stand-alone desk accessory. The Finder opened each panel in its own window, so you could use multiple "Control Panels" at once. The files also moved into their own "Control Panels" sub-folder of the System Folder. But since the Finder is the host, they don't show up separately in the Application menu, and they're still restricted to the old programming model of the Control Panel desk accessory, circa 1986.

Apple has been pushing developers away from true desk accessories and true control panels for many years. Mac OS 8 introduced two new file types for applications: 'APPD' for applications that should autoroute to the Apple Menu Items folder when you drop them on the System Folder, and 'APPC' for applications that need to work like control panels (not only auto-routing to the Control Panels folder, but also examining them during startup to see if they need to load code like traditional control panels or extensions, *MDJ* 1997.07.22).

Now, in Mac OS 9, Apple takes the evolution one step further. The Process Manager now opens traditional control panels (files of type 'cdev') as standalone applications, just as it has done for desk accessories since System 7. If a control panel file is really an application—that is, if it contains a 'CODE' resource with ID 0 like a 68K application, or a 'cfrg' resource identifying a PowerPC "application" type code fragment—the Process Manager launches it as an application. Otherwise, it creates a stand-alone partition of about 200K to host the code in the 'cdev' resource under the 1986 rules. If control panels need more memory, the Process Manager will respect the partition settings in a 'SIZE' resource with ID -16474.

Not every control panel will behave well in this environment—some of them have gotten used to being hosted by the Finder and may not handle smaller partition sizes very well, or they may have been piggy-backing off the Finder's ability to send Apple events, or otherwise exploiting the Finder's hosting skills. As with MultiFinder in 1987, there's a workaround: hold down the Command and Control keys while opening a control panel and the Process Manager will revert to the System 7 behavior of launching it inside the Finder's partition. There's already word out that Microsoft Office control panels require this workaround, but we note that they might just need more memory, so a 'SIZE' resource might fix the problems.

Apple has put developers on notice that they won't be as patient with control panels as they were with desk accessories. Specifically, this Finder-based workaround won't be around for three or four years. As Apple's developer documentation phrases it, "This technique only works for control panel files containing a 'cdev' resource. As this feature is only present for compatibility issues in Mac OS 9, it will likely be removed in the next system revision."

Appearance 1.1.4

The Mac OS 9 version of the Appearance Control Panel has no documented changes other than fixing a bug where previous versions "leaked a large amount of memory" in the system's partition every time the control panel was opened. However, while the control panel itself is mostly unchanged, the Mac OS 9 CD includes several new "themes" (collections of Appearance settings, not code-level "themes" like "Apple Platinum") built around custom desktop pictures. To be frank, these things are gorgeous. At least two options are built around each of the standard iMac colors; the choices include Blueberry Oxygen, Blueberry Union, Grape Gravity, Grape Mission, Lime Horizon, Lime Sharp, Quantum Foam (our favorite), Strawberry Baby, Strawberry Parabola (also cool), Tangerine Fusion, Tangerine Melt, and the flyingsaucer inspired tangerine-like theme Roswell. The "CD Extras" folder on the Mac OS 9 CD-ROM contains alternate versions of some of these desktop pictures (including a blueberry version of the "Roswell" picture), along with the same pictures for larger resolutions (though all are SVGA-type: there are 1280 X 1024 pictures, and 1024 X 768 pictures, and 800 X 600 pictures, but not 832 X 624 or 1152 X 870 versions, both traditional Mac OS resolutions). The CD also contains many other pictures, as well as large desktop patterns and "Additional Desktop Patterns," a Scrapbook file containing all the desktop patterns from past Mac OS releases (but the Scrapbook can't open files from locked media, so copy it to your hard disk before double-clicking it).

ColorSync 3.0

IMATEC's PR screeds and Apple's missing crossplatform versions notwithstanding, ColorSync 3.0 is long overdue. Originally promised for late 1998 or early 1999 (*MWJ* 1998.03.23), the new version goes even farther towards integrating color management with everyday computer tasks.

ColorSync is a hit with publishing professionals, but the concepts are still a bit tricky for those without lots of computer experience. They don't have to be. In essence, every color-capable device produces color slightly differently. For example, the light-producing colors on a monitor are brighter than the reflective inks on paper—a monitor can display brilliant yellows, but only special inks on glossy reflective paper can come close to matching it. ColorSync uses *profiles* that describe exactly how each device displays color. With an accurate profile for your display and another accurate profile for your color printer, ColorSync-savvy applications can adjust their output so what you see on the screen matches the printer's output as closely as possible—including flagging those colors that display perfectly on a monitor but are too bright to print on the printer. Such colors are called *out of gamut*.

ColorSync has always supported a *default profile*, the one ColorSync uses when no other profile is specified (*MDJ* 1997.03.20). With ColorSync 2.5, Apple expanded that to a default profile for RGB devices and a separate default profile for CMYK deviceseffectively giving you a display default and a printer default (MWJ 1998.03.23). Version 2.6 added a default "gray space" profile (but omitted any way for non-programmers to set it, MWJ 1999.03.06). ColorSync 3.0 takes this concept much farther, turning profiles from a system-specific setting into a part of regular document work. The revised ColorSync Control Panel has two tabs (the third "About..." tab from ColorSync 2.6 is now relegated to a proper "About..." box because the "control panel" is now a true application). A popup menu near the top of the "Profiles" panel chooses between "Profiles for Standard Devices" and "Default Profiles for Documents." The former option allows selecting a different default profile for your input device (like a scanner), your display, your output device (typically a printer), and a "proofer" (an intermediate output device, like an inkjet printer, for cases where final output is too expensive for multiple tries). The latter option allows choosing default profiles based on colorspaces for embedding in documents: an RGB profile, a CMYK profile, a grayscale profile, and a "Lab" profile (for the scientific Lab colorspace used in advanced color work, something only graphics professionals typically encounter).

ColorSync 3 allows applications to find all eight of these default profiles, giving substantially more flexibility than the former RGB and CMYK defaults. What's more, using an Open Transport-like interface you can save all these settings into stand-alone documents containing those profiles called workflows. You can take a workflow with you from one system to another, or switch between several if you regularly change devices or production systems and need different default behavior. The benefit to eight kinds of default profiles (instead of two or three) is that you can now set your preferred ColorSync behavior in the control panel. Once applications are revised to support ColorSync 3, they'll pick up your choices from there, instead of every program asking you in a different way to rechoose the same basic profiles.

That's the big news in ColorSync 3.0, though we're sure it has the usual minor enhancements and bug fixes. AppleScript support remains about where it was in version 2.6—the scripting dictionary is still constructed the same semi-awkward way, but it does have support for all the new default profile settings. Even

(continued from page 59)

though the ColorSync Control Panel is now an application, you must still target scripts at the "ColorSync Extension," which still employs its strange dual extension-background application nature for automated tasks. Developers can read more about ColorSync 3.0 in Technical Note #1185, including information on a new but malfunctioning routine to automatically open the ColorSync Control Panel (so programmers don't have to implement their own human interface for profile selection).

Date & Time 8.2

The two-digit year problem is eliminated from the Date & Time Control Panel-you enter the year as a four-digit value now. Curiously, a preliminary developer document says that Apple has changed how the hardware time is implemented. The "clock" in the Mac OS is nothing more than a simple chip that increments a value once every second. By definition, "zero" means midnight on 1904.01.01. The 32-bit counter fills up at 6:28:15 AM on 2040.02.06, so the next second it wraps around to zero. According to the preliminary hardware document, Apple has changed how the bottom half of the possible value range is interpreted, with zero now meaning 6:28:16 AM on 2040.02.06, incrementing again and filling up to just under the halfway point. The new interpretation stops at 9:43:23 AM on 2108.02.25. The new minimum time value in this interpretation is the halfway point of the old range-a 32-bit value of 0x80000000, or 3:14:08 AM on 1972.01.19. The theory is that, "in the future, a user's machine is more likely to run with a date set past 2040 than it is to run with a date set in the distant past." The new definition of this "epoch" means you can no longer set dates in the Date & Time Control Panel earlier than 1972.01.19.

Except that it doesn't seem to be true. The Date & Time Control Panel in Mac OS 9 has a 1904-2040 range, just as before. The Mac OS utilities for turning a count of seconds into a date and time still return the same values they always did. This is to be expected: the proposal calls only for changing the hardware date value so clocks now count up from 1972, not for changing any of the existing date formats. The plan is to overcome the 32-bit limitation in the hardware, not to change how all Mac OS dates are interpreted. But the necessary consequence—the inability to set the clock to a year before 1972—isn't in force in Mac OS 9.

The new OS does include some new date and time management routines to convert local times to UTC (universal time coordinates, formerly Greenwich Mean Time), including time measurement dividing a second into 65,536 parts with "no inherent precision." These are part of the System file and accessible to Mac OS 9 applications, but aren't in the Carbon 1.0 library.

File Exchange 3.0.2

In Mac OS 8.5, this control panel combined two others of old: PC Exchange (for accessing disks formatted with PC, Windows, or Apple II file systems), and Macintosh Easy Open (the interface to the Translation Manager allowing products like QuickTime and MacLink Plus to convert files from one application format to another). In Mac OS 9, the documented changes are all to the file system side. It now supports 120MB SuperDisk discs in PC formats better, no longer assigns incorrect volume sizes to discs formatted by the PC file system code, notifies the File Manager when a volume mounted by File Exchange has been unplugged (for example, if you unplug a USB floppy drive with a mounted PC disk) provided the disk's driver supports the notification, eliminates some problems with reformatting PC disks into Mac OS formats, no longer allows you to name a DOS disk " " (a blank space), and provides more information when you click the "Mount Now" for PC SCSI disks but they actually can't be mounted. File Exchange also supports new features of the HDI (high-density disk driver), but those features are not enumerated. There are also internal bug fixes, as usual.

File Sharing 9.0

We've already noted the addition of File Sharing and Program Linking via TCP/IP through Open Door Networks' ShareWay IP product bundled with Mac OS 9. In addition, the File Sharing Control Panel now contains the functionality of the former Users & Groups Control Panel—all access privileges for AppleTalk File Sharing and any utilities based on the same access privileges are now set in the new third tab of the File Sharing Control Panel's interface. Plus, your password is now masked with bullets as you type it instead of only after you finish typing it.

FontSync 1.0

FontSync is a new technology for Mac OS 9, one that Apple has half-embraced. It's included with the OS and is partly installed by default—it ships as both a control panel (application) and an extension; the extension is part of the standard installation but the application winds up in the "Font Extras" folder in the "Apple Extras" folder at the root of your startup disk.

Figure 18 shows the FontSync control panel. Appearances are deceiving—the technology is more





comprehensible than it looks. We've already discussed in past *MacCyclopedia* entries how outline fonts contain various kinds of data—the outlines for the glyphs themselves, the measurements for those glyphs, the mappings between glyphs and characters (not always one-to-one, since an ATSUI or OpenType font can have many different glyphs for the same character one QuickDraw GX font has about three dozen separate ampersand glyphs), kerning data, and so forth.

FontSync seeks to use these various points of information to eliminate a huge problem in publishing workflows—invisible font substitution. For years and years, font gurus have warned against having both TrueType and Type 1 versions of the same font installed because you're never sure which one you're going to get (the Mac OS will always pick the TrueType version, but some graphics programs that work directly with PostScript fonts might pick the Type 1 font without your knowledge). Many of the older imagesetters and other high-end printing products can't handle the more robust TrueType fonts, and those presses aren't items you replace every year. Or, just as bad, you might take a print job to a service bureau or send it to a high-volume printer but find out (belatedly) that the printer thinks he has the same fonts you do, but they're subtly different, resulting in poor layout and character spacing problems.

This story from Apple's FontSync developer documentation is worth repeating:

The worst example I have heard so far occurred in an issue of the magazine Woman's Day. One of the designers at the magazine got tired of manually tweaking kerning in QuarkXPress, and changed the

(continued from page 61)

kerning table in their fonts so that it happened automatically. Of course, when the magazine was sent to prepress, the prepress company had the unmodified version of the font. As a result, the text reflowed, and the last line of every article in the magazine was cut off. The error wasn't caught in proofs, and in fact wasn't caught until the magazine was on the press and many of the subscriptions had already been mailed. An employee at the printing plant pulled a copy to read, and noticed the last line of the article she was reading wasn't there. The mailed copies had to be recalled and the press run redone, and the prepress company had to eat the total cost, which was about US\$1 million.

FontSync is neither part of the Font Manager nor directly integrated into text drawing in any way. Instead, FontSync takes a font and makes a font reference from it, containing mathematical reductions of the various kinds of font information designed to compactly but uniquely identify the font contents. You can think of each part of a FontSync reference like a checksum, but more complex. On request, FontSync compares font references. If they match, the fonts are known to be identical. The control panel shown in Figure 1 lets you decide which portions of the font references must match before FontSync says the fonts are functionally equivalent. The "Don't match fonts if both references are missing data" checkbox tells FontSync not to return a match if both of the font references are missing one of the items you've said must match.

Font references are more complex than checksums because the issues are more complex. Suppose Adobe releases a new version of your favorite font, and this new font contains extra glyphs (such as the euro character or new small caps) in *addition* to all the existing glyphs. You can argue that if the outlines and metrics for all glyphs in the old version of the font are unchanged in the new version, then the two versions should match even though standard comparison methods would fail. However, the finer the comparisons become, the larger the font references must be to accommodate all the data. In FontSync 1.0, font references are about 2K or 3K per font.

FontSync leaves the door open for this kind of specialized matching in the future, but for now it's an all-or-nothing comparison. As Apple sees it, the more narrow comparisons could correctly compute that two different fonts are identical for all relevant purposes. However, if a narrow comparison says two fonts are different, a broad comparison must by definition report the same result. As Apple's FontSync developer documentation puts it, "For the customers we are initially targeting, however, the primary requirement is that fonts not be misidentified. Sophisticated font substitution is not a requirement at this stage. To put it another way, false negatives are OK, but false positives are anathema."

FontSync font references are abstract data types, but the technology also works with font profiles-collections of font references stored in a file. Font profiles are always in files; they don't exist as resource or RAMbased structures. Until applications add support for FontSync's matching capabilities, font profiles are how you'll interact with FontSync. The control panel itself provides no interface for creating or matching font references or profiles. Applications can do this, or the FontSync Control Panel will do it through AppleScript. The "Font Extras" folder contains two sample scripts-one to create a font profile, and one to match a font profile against the currently installed fonts. If you want to use FontSync 1.0, you must study these scripts: not only is AppleScript the only way to do it, but the scripts also use verbs like "match against" and "create font profile" that aren't in the FontSync Control Panel's scripting dictionary.

FontSync 1.0 is a limited first implementation. It can only create font references or profiles for installed fonts. Since all matching is done through AppleScript by a faceless background application, FontSync can only work with globally visible fonts—application-specific fonts can't be profiled or matched. FontSync seems to know about Mac OS 9's new font management capabilities (see "Watch This Space," this issue), so it can probably keep up with newly-activated fonts when they're activated with those features—but we doubt it can notice fonts activated through pre-Mac OS 9 font managers like Suitcase 8, ATM Deluxe, or Font Reserve. Thus, if you switch fonts around and intend to use FontSync to profile them, you may need to restart.

AppleScripts are thus limited, but other software is free to profile any fonts it wishes (as long as they're globally installed) without restarting. We expect publishing developers to release more robust tools soon, like regular applications that compare individual fonts or profiles on demand, or plug-ins to embed font references in QuarkXPress and Adobe InDesign documents. That's when this technology will start paying for Mac OS 9 upgrades many times over. After all, if you're *Woman's Day*, US\$1 million will upgrade a lot of systems.

General Controls 7.7.2

Although it's not officially documented, one of our favorite Mac OS 9 changes eliminates a pet peeve of ours by revising the wording of the "improper shut**MW** Mac OS 9 Special

down" dialog box (the one that displays while the built-in Disk First Aid examines your hard disk) to shorten it and eliminate all that nasty passive voice. The message now reads, "Your computer did not shut down properly. Disk First Aid is checking your hard disk and will repair any problems. If your computer is having problems, see Mac Help for troubleshooting information." *Much* better.

Memory 8.1.1

The Memory Control Panel is now PowerPC native and has AppleScript support for getting and setting virtual memory, RAM Disk, and disk cache settings from scripts. The control panel also now considers the memory used by ROM-in-RAM systems when computing maximum values, so it won't let you choose settings so large there's no room for the Mac OS ROM image. Mac OS 9 also fixes a systemic bug that prevented you from turning off a persistent RAM Disk; it's not part of the Memory Control Panel, but you might have cursed at it a fair amount if you experienced the problem. You can no longer set the RAM Disk on a PowerBook 1400 to more than 60% of available RAM because this makes the machine's ROM issue the "death chime" on startup, generally a bad thing.

Monitors 8.5.1

A remnant of the AppleVision days is finally dead and gone—the combined "Monitors & Sound" Control Panel has been retired, split once again into two distinct control panels (stop us if you've guessed this): "Monitors" and "Sound." In Mac OS 9, the Monitors Control Panel has all the functionality you'd expect from the older combined panel—color depth, resolution choices, plus contrast and brightness on models with appropriate display controls (like PowerBook and iMac machines, for example, and perhaps some AppleVision displays), and monitor geometry controls for those systems whose displays can be controlled from software (such as iMacs).

Mouse 8.0

The venerable Mouse Control Panel is now an application as well, because it too is scriptable—script writers can now get and set the tracking speed, the double-click speed, the "mouse tracks" setting (the "ghost" images that follow the cursor, useful for following its tracks on LCD displays). Scripters can also activate or deactivate the "thick I-beam" cursor that's easier to see on the same LCD screens. The only visible change, however, is updated graphics in the Mouse Control Panel's window—the pictures of the mouse are now of the round Apple USB Mouse, not of the older original Apple ADB mouse (circa 1988).

Sound 8.1.2

The other half of the former "Monitors & Sound" Control Panel is now also a separate application, but, like the new Monitors Control Panel, is not scriptable. A list on the left of the window allows you to choose from available services: alert sounds, input, output, and speaker setup. The last of those includes, on appropriate systems, separate volume controls for left and right speakers, and a test mode of white noise that allows you to equalize the sound to your satisfaction.

Most notable, however, is the first change to standard alert sounds since 1991. Gone are Droplet, Quack, and Wild Eep (the voice of a former wife of a current Apple executive, if you must know). We'll only provide brief descriptions of the sounds and their titles: ChuToy (a child's squeaky toy), Glass (a crystal ringing), Laugh (self-explanatory), Logjam (a revving chain saw), Pong2003 (a short electronic dual tone), Purr (an electronic repeating sound), Submarine (a sonar ping), Temple (a stereo brass chime), "Uh oh" (a woman saying "uh-oh"), and Voltage (a buzzing sound). "Simple Click" and "Whit," introduced with Mac OS 8.5 themes, remain.

Software Update 1.0

New to Mac OS 9, and already covered in "Living in a Wired World."

Undocumented But Changed

As usual, several other control panels have changed in Mac OS 9, but the company has released no information about what changed or why. Here are the new version numbers of those we know changed in a standard desktop installation:

- Apple Menu Options 1.1.6
- Energy Saver 2.5.2
- Keyboard 8.3.2
- Launcher 3.1.3
- Startup Disk 7.7.6
- TCP/IP 2.5 (part of Open Transport 2.5.2)
- Text 8.0

Apple Menu Items

At first it seems there are few changes to the Apple Menu Items. Apple's Mac OS 9 developer documentation lists changes to only two items: Network Browser 2.0 and Sherlock 2. But if you look at an

(see "The Nitty-Gritty," page 64)

(continued from page 63)

Installer log, more is going on here. Several of the items that formerly had standard Apple Menu Items placement are now located in the "Applications" folder at the root of the startup disk: AppleCD Audio Player (or Apple DVD Player if your system has an Apple-supplied DVD drive), Graphing Calculator, Note Pad, and SimpleSound. The "Automated Tasks" folder has moved to the "AppleScript" folder in the "Apple Extras" folder. The "Internet Access" folder inside Apple Menu Items is now gone; the "Browse the Internet" and "Mail" applications that redirect to the appropriate helper (through the Internet Control Panel) have aliases on the desktop and are located in the "Internet Utilities" folder of the "Internet" folder. "Connect To ... " is gone; you can now connect to a URL from the Network Browser or from other applications using NSL Browser functionality.

The "Remote Access Setup" application is part of Apple Remote Access, where changes are not adequately specified. We'll discuss the "Speakable Items" folder with PlainTalk 2.0 in the "Extensions" section of this article. Otherwise, the version numbers of all the aforementioned applications are the same as in Mac OS 8.6, with one exception.

AppleCD Audio Player 2.3.1

There are no release notes for this version. The only change we can discern is one tipped to us by a reader: on new machines with three-dimensional sound, "3D Stereo" is now an option in the "Sound" hierarchical menu, along with "Left", "Right", and "Stereo." ("3D Stereo" is the only of the four with no balloon help.)

Applications, Utilities, and Extras

We won't attempt to catalog all the extras on the Mac OS 9 CD-ROM, for they are numerous and mostly undescribed. The CD Extras folder alone contains modem scripts, AOL 4.0, lots of ColorSync extras, Eric's Solitaire Sampler, OSA Menu, iDo Script Scheduler (lite), HyperCard Player (and update), Palm Desktop 2.5, firmware updates for Power Macintosh G3 and iMac computers, and music videos from both Barenaked Ladies ("Call And Answer") and Static X ("Push It")—and lots more. Apple is not responsible for most of these items; those that it does create (like the "Network Extras" are mostly one-trick ponies that are already familiar to those who need them). We encourage you to prowl through the CD Extras folder and read a bunch of "Read Me" files if you want to know more about the bonus files.

As far as actually installed programs go, one is underspecified: Disk First Aid 8.5.5 is included with Mac OS 9 without release notes. Fortunately, most of the other standard applications are documented enough to give you an idea of what you're getting into.

Finder 9.0

Changes in the stalwart of the Macintosh desktop are minor and evolutionary, but in line with the other features of Mac OS 9. For example, while the Finder doesn't have the overhauling necessary to fully support HFS Plus file system features, it does preserve Unicode file names when copying files (though it won't if the name changes—if you "Duplicate" an HFS Plus file with a Unicode name, the duplicate has only an HFS-style altered name with "copy" appended), and it uses the new HFS Plus File Manager routines to accurately copy and display information about files larger than 2GB. The theoretical HFS Plus file limit is 16.8 million terabytes, but the existing Mac OS driver model limits that to a "mere" two terabytes, or 2048GB.

Finder 8's invisible files, like "DesktopPrinters DB" and "OpenFolderList DF," used for tracking items like what folders you have open and your desktop printer configuration, are now themselves stored in an invisible folder named "TheVolumeSettingsFolder". These folders are kept in user folders when Multiple Users is active, enabling Finder to track each user's desktop configuration separately. And the "Encrypt" item in the "File" menu invokes Apple File Security, as previously noted ("Sharing Your Personal Space," this issue).

Finder 9's support for packages is somewhat different than you might expect. The concept is inherited from Next technology, where the file systems have no resource forks and all separate chunks of data are stored as small individual files. Recall that a *package* is a folder with its "bundle" bit set and a special kind of alias at the top level of the folder pointing to the "main file" of the package (see "Watch This Space," this issue). Finder 9 treats packages just as if they were actually the package's main file, complete with dragand-drop support and "Get Info" mechanisms for changing memory partitions for packages built around applications (the normal kind). There is one major difference: when Finder tries to open any file that's *inside* a package, it sends the package's application a list of the files through the standard "open document" Apple event, but places the list in a new optional parameter called the "Finder package document list." This happens even for aliases to items inside the package that the application couldn't open (for example, an alias to a package's built-in shared library file). The differences mean that applications **MW** Mac OS 9 Special

generally need to know they're in packages; regular users shouldn't go creating packages out of existing application folders for convenience.

Package properties are available through AppleScript, as are new optional properties for the LaserWriter 8.7 scriptable printing interface (see "Extensions," this article).

Apple Guide 2.5

Apple Guide is now a supplemental part of the overall "Apple Help" system, given an overall version number of 1.5 in Mac OS 9. There's only one documented change: the internal Apple Guide routine that returns the Apple Guide version now correctly returns version 2.5. Version 2.3.1 and 2.4.1 in previous OS releases both returned version 2.3.1 for the traditional "oops, it's a bug" reason.

Apple System Profiler 2.4.2

Apple's standard system examination application lacks release notes this time; it seems identical to version 2.2 from Mac OS 8.6 upon inspection. However, the scripting dictionary has substantial improvements, allowing scripts to fetch almost everything the application can display in its main window, complete with dictionary explanations of what most parameters are. The scripting dictionary only applies to the main functions—it won't return listings of extensions and control panels, for example, since you can do that yourself by scripting the Finder.

Help Viewer 1.5

The Apple Help application no longer has HTML rendering code built in; instead, the code that was in Help Viewer is now in the new HTML Rendering Library (see "Watch This Space," this issue). Apple says that Help Viewer also now allows searching locally for Internet-based help and viewing it, but Apple provides no explanation of how this might work.

Package First Aid 1.0

The Mac OS 9 CD-ROM contains a new utility, Package First Aid, that's not installed on your hard drive. It serves only one purpose—if you have a folder that has been incorrectly designated as a package, drag-and-drop it onto Package First Aid to fix it. It does not construct packages; it only deconstructs them. And, just like a dialog buried deep in Sherlock 2, no one looked at it very carefully after Apple decided this would be Mac OS 9: if you double-click the application to open it, you're told "Package First Aid is used to repair folders that incorrectly appear as packages under Mac OS 8.7 and later." Oops.

Drive Setup 1.8.1

Apple's utility for managing hard disks that come with Apple systems has been expanded to allow formatting partitions with UNIX File System (UFS) format, in both Mac OS X and Mac OS X Server flavors, as well as supporting "Linux Home" and "Linux Opt" partitions. Other reports say Drive Setup now allows formatting and installing drivers on various brands of third-party hard disks, but Apple has made no moves towards officially supporting any drives not shipped by Apple. Use Drive Setup on third-party hard drives at your own risk.

Apple Network Assistant 4.0

Mac OS 9 includes version 4.0 of the Apple Network Assistant client software. ANA is Apple's remote administration program for networks of Mac OS computers, including screen monitoring, screen sharing, chat, intercom, locking the screens of remote users (to get their attention), and more. Think "Timbuktu Pro" but less mature and not cross-platform, with some classroom-specific features. Network Assistant 4.0 is apparently needed for Mac OS 9 compatibility, so Apple includes it on the Mac OS 9 CD-ROM. You must perform a custom installation to get it, however, and it requires the not-yet-available Apple Networking Assistant 4.0 administrator to configure. If you use ANA, you'll want to know about the update, but everyone else can ignore it.

Extensions

We'll try to group extensions into logical groups, such as treating Open Transport and Apple Game Sprockets as functional units despite large numbers of files. As with all other aspects of Mac OS 9, expect to find bugs fixed in places where they're not documented.

Apple Game Sprockets 1.7

Two and half years after Apple, as a cost-cutting move, declared Game Sprockets development dead, a resurgent Apple with a sharper focus on games and entertainment has reversed that decision. Mac OS 9 provides the proof. The Game Sprockets are small code modules designed to take common game programming tasks out of the hands of developers and place the burden on Apple, speeding development

Page 66

The Nitty-Gritty

(continued from page 65)

and allowing for standard interfaces. A famous example is InputSprocket, a set of human interface input routines that reads from keyboards, mice, joysticks, gamepads, and other controllers. When game developers write using the InputSprocket routines, their games automatically work with new game devices provided the devices have simple InputSprocket drivers. So goes it with all of the sprockets.

• **DrawSprocket 1.7** makes it easier for programmers to "take over the screen," as they do on PC games, without running into the finicky Macintosh display system. DrawSprocket can even allow for pageflipping (rapidly switching between alternate screen buffers) on systems that support the feature, one otherwise unavailable to Mac OS programs. The new version improves performance copying pixels to the screen, fixes bugs, and adds program-level support for better management of "contexts," the term DrawSprocket uses for a drawing environment.

InputSprocket 1.7, as mentioned, provides a standard API for programmers to interact with all kinds of input devices. The new version adds a universal USB "HID" (Human Interface Device) sprocket driver, so any USB HID-compliant device works with InputSprockets off the shelf, no custom driver required. There is apparently a bug in this implementation: Alessandro Levi Montalcini has already noted that as shipping, InputSprocket 1.7 uses this new generic driver for all devices, ignoring custom drivers even if you have them. Aside from this problem, InputSprocket 1.7 consolidates all of the Apple-supplied drivers into a single file, adds support for devices with more than two axes (three-dimensional input) or more than three buttons, and includes a calibration dialog box for all supported input devices.

• NetSprocket 1.7 includes code that makes it easy for game developers to write networked games, painlessly sharing real-time data and activities across AppleTalk or TCP/IP networks. (NetSprocket, unfortunately, is not cross-platform, so applications using it can't interoperate with PC games.) The update in Mac OS 9 fixes bugs and adds features that let application developers manage players and teams more easily. A host game can now unilaterally drop players from a game instead of waiting for them to quit, for example. Developers can now set their own time-out values to identify non-responsive clients. Further, the NetSprocket code now resides in a single file instead of in two files. • **SoundSprocket 1.7**, the last of the sprockets, adds pinpoint 3D sound facilities for game developers so sounds can be rendered as if they're coming from a specific point in space. In this revision, the 3D sound is "superior" to previous versions (probably by leveraging the built-in 3D sound support in most current Macintosh models, but it's likely there are improvements regardless of hardware). It also adds routines for developers to make it easier for OpenGL applications to use SoundSprocket.

AppleScript 1.4

We covered the changes in AppleScript 1.4 when discussing the new capability to script remote machines via TCP/IP ("Living in a Wired World," this issue). We've also covered changes to scriptability in other components as we've found them with those components. If you'd like more information on all AppleScript-*related* changes in one convenient place, try the AppleScript Sourcebook page for AppleScript 1.4, a handy and easy-to-follow repository for all scriptrelated information.

AppleShare Client 3.8.5

This version of the AppleShare client software (for logging onto, mounting, and using AppleShare file servers, including File Sharing volumes) is required for Mac OS 9; earlier versions will not work. Additions include keychain support (an "Add to Keychain" checkbox, with Command-K key equivalent, allows inserting the server information and your password on a keychain), DHX user authentication for encrypted logon with Mac OS X Server machines, and support for machines capable of waking up from "dozing" (a light form of sleep) over network events ("Wake-on-LAN").

You might notice two changes in unusual ways. Programs can tell the AppleShare client to log onto servers and how securely to do so (sending passwords with one-way encryption, two-way encryption, in clear text, or whatever other methods are supported). Before now, the client would fall back to less secure methods than requested—if a program wanted a twoway secure log in but only clear text authentication was available, the client would log onto the server with clear text authentication. It no longer does this, so if a program asks for a secure method that's not available, it will not be able to log onto the server unless it retries and asks for less secure authentication. You can work around any such problems you see by logging onto the server yourself.

The AppleShare client still lets you check volumes on a server to be mounted at startup, but the method Mac OS 9 Hidden Features

has changed. Instead of storing server information and passwords in a relatively insecure "AppleShare Prep" file, the security options are now to save your name only, or save your name and password in the keychain. The AppleShare client places an alias to the server in a new "Servers" folder in the System Folder, and once you unlock the keychain on startup, you're automatically logged into the servers. (Note, however, that if a password for the server is already on the keychain, you'll get a cryptic "Could not add this to the keychain" alert.) The change allows you to stop automatically logging onto servers just by removing them from your "Servers" folder. Ironically, it does not work with Multiple Users-there are no duplicate "Servers" folders mapped to each user's folder, just a global one in the System Folder. Users who cannot write to the System Folder won't be able to set automatic log-on servers from the client. They'll have to place server aliases in their own "Startup Items" folders with the passwords either on the keychain or entered at startup.

This new method requires the Keychain to work. The same feature is present in AppleShare Client 3.8.4, shipping with Power Macintosh G4 (AGP Graphics) systems running Mac OS 8.6. Those systems have no keychain, so the automatic log on feature is disabled. The workaround, according to a Tech Info Library article, is to downgrade to AppleShare Client 3.8.3 or upgrade to Mac OS 9.

ColorSync 3.0

The ColorSync 3.0 extension, part of ColorSync 3.0, contains the background-only application that implements the scripting interface; it also includes the actual ColorSync code so applications can use it (much like the QuickTime extension contains the QuickTime code). ColorSync 3.0 was discussed earlier in the "Control Panels" section of this article.

Find By Content 3.0.1

The Find By Content extension provides the indexing and searching services used mostly by Sherlock but also available to other applications (*MWJ* 1998.10.19). In this release, developers have access to a new routine that was previously only part of Sherlock's domain—indexing a specific folder or file on demand. Apple hopes that application developers will adopt this as part of the "Save..." routine for documents, keeping your Sherlock indexes up-to-date without requiring too many regular indexing operations. We've seen no real sign that developers are intrigued by this, but it's an interesting idea.

In addition to Western European languages plus Japanese and Korean, Find by Content can now index text in any of the following languages: Afrikaans, Arabic, Farsi, Urdu, Catalan, Croatian, Cyrillic languages, Czech, Danish, Dutch, English, Estonian, French, German, Greek, Hebrew, Hungarian, Icelandic, Italian, Japanese, Korean, Latvian, Lithuanian, Norwegian, Other (using the standard Roman alphabet), Polish, Portuguese, Romanian, Slovak, Slovene, Spanish, Swedish, Turkish, and Yiddish. You control how many of these languages are in your normal indexes through the "Languages..." button in Sherlock 2's preferences. Once you've added a language to your index, you can only remove by recreating the index.

LaserWriter 8 version 8.7

In Mac OS 9, Apple continues to try to make up for the lack of a real printing architecture (since the death of QuickDraw GX and the lack of significant improvement in Carbon) by turning the LaserWriter 8 driver into an architecture of its own. Over the past several releases, the LaserWriter 8 driver (used for almost all PostScript output on the Macintosh; the rest comes from Adobe's AdobePS driver, built from (more or less) the same source code) has added "hoses" so developers can redirect the driver's output (using Desktop Printing) over any kind of hardware connection, a Printing Plug-Ins Manager for overseeing these hoses and other additions to the driver, a Download Manager for applications to use in sending data directly to the printer (instead of faking print jobs) complete with converter plug-ins for massaging the data, a detailed job log for expert analysis of the PostScript processes used in printing, and much more. As a sign of how much work is happening in this area, note that about 28% of the developer-level Technical Notes released by Apple in the past year are about the LaserWriter driver.

LaserWriter 8, version 8.7 continues this questionable trend, but in ways that are hard to dispute because they provide features professional users have needed for more than a decade. PostScript output filters provide high-end developers with their long-time dream-a way to view the stream of PostScript output by the printer driver and add, modify, or remove sections of it before it's sent to the output device or file. It eliminates the need for a generation of hacky workarounds, like resources full of PostScript that get sent to the printer before starting or bizarre font workarounds to force specific driver behavior. Output filters are real PowerPC-native code, living in files in the Printing Plug-Ins folder, and they should fulfill every PostScript modification need developers have ever expressed.

(continued from page 67)

Also new in version 8.7 is scriptable printing. The libraries associated with LaserWriter 8 version 8.7 define a new "print options" record in AppleScript that allows setting a ton of properties: number of copies, starting page, ending page, N-up layout, job log settings, and more. These settings are private in each driver, and applications do not know how to tell any driver (including LaserWriter 8) how to modify them-the printing non-architecture has no APIs for fine-level control, and the settings are normally completely under your control in the "Print..." and "Page Setup..." dialog boxes. Programs have no way to access or interpret your choices, much less change them. Version 8.7 works around the problem by including Apple event code that converts the new print settings record into a standard "print record" used by applications to control all printing processes. The driver handles the conversion, so applications don't have to cheat by trying to control private settings. The idea is right, but making it work requires revised applications-each program with scriptable printing has to add the scripting terminology for the new record and call the appropriate routines to convert it into a usable form. Whether this happens or not is anyone's guess-AdobePS doesn't have this feature yet so InDesign may pass, and QuarkXPress tends to add such features slowly (Quark, you may recall, told users for years not to even use LaserWriter 8 instead of fixing its own printing code). Plug-ins for both applications may take up the slack.

On a more visible level, LaserWriter 8 version 8.7 has a revised "Color Matching" panel. If you choose ColorSync Color Matching, you now have two radio buttons for the printer profile. One lets you choose from a menu of all available profiles, as before, but the other one uses the "output profile" as configured in the ColorSync Control Panel. Pick this choice and choose "Save Settings," and you'll be free to manage your profiles from the control panel instead of in every printing application.

Minor new features include a new PostScript query that allows PostScript spoolers to tell the printer driver a job should print in the foreground and not the background; spoolers often like to communicate with the driver and ask it questions, but this is only possible during foreground printing while the driver has the "live" control it needs. The driver has always synthesized QuickDraw styles for TrueType fonts when those styles aren't available—for example, slanting a plain font to create an "italic" font if no drawn italic TrueType font is installed. Until version 8.7, it didn't do this for double-byte fonts because they were too large; now it does, based on font subsetting technology introduced in version 8.6. And last, but not least, the extensive job log now includes FontSync font IDs for your help in diagnosing problems.

Language Kits

The Mac OS 9 CD-ROM includes, free of charge, updated versions of all products formerly sold as "Apple Language Kits." These additional packages (available through custom installation) include fonts, keyboard scripts, and WorldScript extensions that provide the Mac OS with the code and data necessary to allow writing, typing, and displaying non-Roman languages. The "Roman" script, built into all versions of the Mac OS, is fine for all languages that use the "Roman" alphabet or minor variations (for example, Spanish adds the ñ character, French adds the ç). Languages with substantially different textual requirements need different OS routines.

Mac OS 9 includes kits for the following languages or writing systems: Arabic, Central European languages, Cyrillic languages, Devanagari, Gujarati, Punjabi, Hebrew, Japanese, Korean, Simplified Chinese, and Traditional Chinese. These replace the Chinese Language Kit 1.1.1, Japanese Language Kit 1.2, Cyrillic Language Kit 1.0, Arabic Language Kit 1.0.1, Hebrew Language Kit 1.0.1, Korean Language Kit 1.0, and Indian Language Kit 1.0. (Some Language Kits supported more than one language or writing system.)

The "CD Extras" folder on the Mac OS 9 CD-ROM includes disk images with language kit extras for each of these languages, offering extra fonts, and some alternate keyboard layouts. The extras also includes more explicit support for a Unicode script, including a programmer-level keyboard script that allows entering Unicode characters directly by their character numbers as four hexadecimal digits.

Mac OS Runtime for Java 2.1.4

This latest version of Apple's Java virtual machine was released in August (*MWJ* 1999.08.14) and is unchanged since then.

Multiple Users 1.1

We explored the Multiple Users system in-depth in "Sharing Your Personal Space."

Network Services Locator 1.1

We explored the NSL Manager, residing in the Extensions folder, in-depth in "Living in a Wired World."

OpenGL 1.1.1

Apple's developer documentation says Mac OS 9 comes with OpenGL 1.1, implementing the full OpenGL 1.1 API as defined on multiple platforms. This isn't quite right—in fact, Mac OS 9 comes with OpenGL 1.1.1, apparently a minor bug-fix release. No further information is available.

Open Transport 2.5.2

The networking code in Mac OS 9 is, as of press time, one of the more frustrating areas of change: the substantial and significant revisions are sparsely documented. The only available information is in Apple's Mac OS 9 developer documentation, and that source is long on Mac OS APIs and short on information about what's changed within the TCP/IP and STREAMS implementations. The developer-level Open Transport software development kits and Web pages haven't changed in many months; Open Transport "Read Me" files are neither installed nor on the Mac OS 9 CD-ROM, and the only Tech Info Library article on the subject reprints the sparse information available in the Technical Note.

We've seen first-hand that the new Mentat TCP 3.5 stack included in Open Transport 2.5.2 has a better reverse DNS resolver; it recognizes delegated DNS records where earlier versions did not. The Mac OS implementation is improved by faster creation and deletion of *endpoints* (the software end of a network connection), eliminating "copy stall" problems when moving large blocks of data over TCP/IP, allowing protocol stacks to stay open across sleep and wake-up cycles, and reducing the number of Open Transport library files from six to two.

The developer documentation says Mac OS 9 includes both a client and a server version of Apple Remote Access, but you'll miss it if you're not careful. The Remote Access Control Panel now includes an "Answering" menu item in the still-misspelled "RemoteAccess" menu. Though the associated dialog box, you can tell Remote Access to answer the phone and connect to either that single computer or an entire local area network via AppleTalk. You can also allow TCP/IP clients to connect remotely using PPP, and set an optional maximum connection time. If Remote Access is installed, the Users & Groups portion of the File Sharing Control Panel expands: each user's window gets a new "Remote Access" panel that lets you allow or deny dialing in on a user-by-user basis, and even allows a call-back at a predetermined number for extra security. This new feature replaces the separate Apple Remote Access server, priced at around US\$100. But that's not all-dial-on-demand performance is now improved, Apple says, and the

new release fixes some problems with third-party wireless modems (those are the only kind, of course— Apple doesn't make wireless modems).

There are still reports of some problems, but overall DHCP behavior should be improved in Open Transport 2.5. DHCP is a protocol that lets your computer get all its TCP/IP information from a central server through a "lease" that expires after a set period unless your client renews it. Internal timeout values have been adjusted so you shouldn't notice "pauses" when your computer tries to negotiate a new or renewed DHCP lease. The packets Open Transport sends still conform to the DHCP standard, but they're now "more compatible" with Windows NT (and similar) servers that stretch the standard a little bit.

Open Transport isn't your typical TCP client unlike those on UNIX and other systems, Open Transport doesn't have to load the TCP stack during startup, and can load and unload it at will. Open Transport 2.5 adds a way for applications or other code to know when the stack's status is changing (or about to change), including ways to know if the TCP link is running over Ethernet, 802.3, or PPP.

We'll keep an eye out for more complete details of changes inside Open Transport 2.5 and publish any such information when we find it.

PlainTalk 2.0

After a few years of inactivity, Apple has revised its speech recognition software with a flourish. In addition to Speech Recognition 2.0, Mac OS 9 includes English Text-to-Speech 2.0.2 and Mexican Spanish Text-to-Speech 2.0.2. There isn't much news about what's revised in those components for turning text into the spoken word, but the reverse implementation is much expanded.

For starters, there is a new "Application Speakable Items" hierarchy of folders within the "Speakable Items" folder. Each folder is named for an application it controls, but precise naming isn't necessary-the folders must contain an invisible alias to the application they reference. Speakable Items uses these aliases to recognize applications, so you can name the folders something more convenient (like "Netscape" instead of "Netscape Communicator 4.6"). If the frontmost application matches any of the invisible aliases in this folder hierarchy, all the items in that folder become speakable items for that application *only*. Apple ships four sets of application-specific speakable items: for the Finder, Netscape Communicator, Microsoft Internet Explorer, and Microsoft Outlook Express. Sample Web browser speakable items include "go back", "go forward", "page down", "page to bottom",

(continued from page 69)

"make this page speakable", and "reload this page". Items in the top-level Speakable Items folder remain recognizable at all times.

Apple's speech recognition is not continuous-it doesn't turn everything you say into text. Instead, it listens to your phonemes and matches them against a list of possible matches. For Speakable Items, the list is the names of all applications in the global and application-specific Speakable Items folders. AppleScript authors can now take advantage of this by asking the "Speech Listener" application (in the "Scripting Additions" folder) to "listen for" text, providing a list of possible responses with an optional spoken prompt and timeout value. Script writers can now add this rudimentary speech recognition to scripts without having to resort to a global speakable item, a definite plus. The "Speakable Items" application also has a few scripting commands, including enabling and disabling active listening, and showing recent applications.

Of course, the supplied speakable items commands have been updated for Mac OS 9 features and compatibility. The "show me what to say" command lists all available speakable items in a floating window, both built-in and application-specific (also available from the "Speakable Items" scripting interface). Examples of new commands include "reference channel" (to activate Sherlock 2; other channel commands are also included), "insert my E-mail address", "shop for books", "take a screen picture", and many more. Speakable Items itself includes global commands not stored as items in the folder, such as "listen continuously", "turn on push to talk", "show application commands", and a great new feature that lets you choose almost any simple button, radio button, or checkbox just by speaking its name.

Speech recognition in Mac OS 9 is a lot beefier than it was before. It's a custom installation option, but if you found previous versions of PlainTalk at all interesting, you should check out this version.

The System File

And now, ladies and gentlemen, we come to the nittiest and grittiest of it all—the contents of the System file (or, in some cases, the updated "Mac OS ROM" file, or both). Many of the longest-lived Mac OS technologies long ago migrated from extensions into the System file itself, and that's where they see changes in new OS releases. This is some of the most technical material in our Mac OS 9 coverage, so don't fault yourself if it takes multiple readings to understand. We do our best to translate from geekspeak, but you can still detect our geek accent. Also note that some com-

ponents in the System file do not have recognizable or meaningful version numbers.

ATSUI 2.0

You may have heard about a change in Apple Type Services for Unicode Imaging even though ATSUI probably has absolutely zero effect on your Mac OS use (since we can't think of a single ATSUI application for general audiences). This inheritor of QuickDraw GX Typography is more robust in almost every way: it uses less memory, breaks lines according to Unicode's idea of a line break for a given language and does so faster, now handles tab characters (something QuickDraw GX never did), and adds a bunch of new bug fixes and features for developers to exploit.

The noteworthy change is that ATSUI now includes Apple's own PostScript Type 1 font renderer-not a version of ATM, but built-in code written and owned by Apple. The scaler, part of the Open Font Architecture, supports Type 1 fonts in TrueType wrappers (either QuickDraw GX or OpenType fonts) in CID, regular, or multiple master formats; it also supports the old-style "5-3-3" font files (but not multiple master versions), and Adobe's original composite font and "naked CID" formats. Speculation about this Type 1 scaler ripped through some publishing circles, but don't get too excited too fast-as noted, it's only for Open Font Architecture clients like ATSUI. QuickDraw, used by all non-ATSUI programs, is not an OFA client. The scaler does not make Type 1 fonts available to non-ATSUI applications.

CFM-68K 4.0

Although Mac OS 8.5 and later are exclusively PowerPC releases, the system still included CFM-68K fragments, so some parts of it were accessible to 68K applications. That was a mistake—only programs on 68K computers can call CFM-68K routines. (PowerPC machines don't allow CFM-68K access; developers were required to build "fat" applications and access the PowerPC-native code fragments for such technologies on PowerPC machines.) Accordingly, Apple has removed several CFM-68K code fragments from the System file since no application can access them.

Communications Toolbox

This code, originally introduced in System 6.0.4 with support for choosing connections, modems, and file transfer protocols, was not always correctly returning the maximum allowable serial port speed. This has been fixed.

Device Manager

The low-level software that talks directly to device drivers has implemented changes to allow multiprocessing tasks (Apple's new, recommended way for dividing a program into separate, simultaneous tasks) to call device driver routines synchronously (waiting for a return event) if an asynchronous variant of the call is available. Synchronous calls from low-level tasks are tricky, because if the system and application don't carefully manage the sequence of events, the machine can lock up while the tasks waits for the call to complete and the call needs more information from another part of the system that can't get control. The solution in Mac OS 9 is not perfect, but it improves the status quo.

File Manager

The routines in the Mac OS that mediate access to files on any disk have been significantly updated to include HFS Plus file system features (see "Watch This Space," this issue). And, as noted throughout the FCB table brouhaha, the maximum number of open files is now 8169, way up from 348 in Mac OS 8.6 and earlier (MWJ 1999.10.09). The File Manager follows the Device Manager and allows multiprocessing tasks to call it synchronously if asynchronous variants of the same call are available; some routines have no asynchronous versions and they're still inaccessible from MP tasks. Lastly, a new File Manager error (-503) is the "Hardware Gone" error. Disk drivers should return this error if they're asked to read or write to a disk device that has been unplugged and can't be plugged back in. You may see it a lot if you hot-swap USB disks. (FireWire disks have a separate FireWire capability, called "Wait for device replug," that forces you to reconnect a disk the system wasn't done with.) One compatibility offshoot of the new HFS Plus routines: applications that patch the entire File Manager so they can watch every bit of disk activity (like antiviral products or on-the-fly disk recovery tools) need revision so they patch the new HFS Plus routines as well.

File System Manager

The names are similar but the purposes different: while the File Manager arbitrates access to all files on all disks, the File *System* Manager is specifically in charge of the "plug-ins" that interpret non-Macintosh file systems. The File Manager calls the File System Manager when any program makes a file request to a volume that's not HFS or HFS Plus format (including DOS, ProDOS, ISO 9660, UDF, and all those others implemented through extensions). The new edition of the File System Manager includes many new routines making it easier for FSM plug-ins to get to file control blocks—since they can no longer touch the FCB table, the File System Manager gives them fast and robust routines to get everything they need. Plug-ins can also now check and validate file permissions for file forks. The new HFS Plus routines add lots of capabilities, like access to multiple forks or files larger than 2GB. The File Manager includes a compatibility layer to turn these new calls into old-style calls, so FSM plugins need not implement these for Mac OS 9 compatibility (however, they will need revision to deal with the changes to the FCB table). Any FSM plug-in authors who *want* to handle the new calls directly, however, may do so by including the code and setting a specific flag bit in their volume control blocks.

Folder Manager

We discussed the extensive changes to the Folder Manager for Multiple Users with that technology (see "Sharing Your Personal Space," this issue). Other changes to the routines designed to help applications find special folders include more control over the finding operations, faster routines to identify folders, a way to pick a special folder off a specific disk when it might be available on multiple volumes, and the new capability to resolve special folder aliases. Note that this requires the code using the special folder to do the same. For example, you can't replace your active Extensions folder with an alias because the Mac OS boot code will not follow the alias during startup and load extensions from a different location. You could, however, replace the Preferences folder with an alias because all code looking for that folder should be using the Folder Manager. Be prepared for compatibility problems, however, if you try to use aliases to move special folders off the startup disk. Apple has warned developers not to assume special folders are on the same disk as the System Folder, but not all programmers have yet heeded the warning in revised code.

Font Manager

We discussed the extensive new Font Management routines as one of Mac OS 9's noteworthy invisible features (see "Watch This Space," this issue). Aside from that, the major change in Mac OS 9's Font Manager is fixing the bug that created incorrect 'FOND' resources and wrote them to files. This is the same bug fixed by Apple's Font Manager Update 1.0 (and in a different way by Alsoft's Corrupt FOND Fixer and DiamondSoft's FONT Fixer utilities). Those are

(continued from page 71)

unnecessary with Mac OS 9, as the problem is repaired at the source.

Mac OS USB 1.3

Development versions of Mac OS USB 1.3 have been available for months (*MWJ* 1999.06.26). There are many changes in the new version, shipping in final form in Mac OS 9 (but not separately available at press time), but to be frank, most of them are only interesting to USB device driver authors. You can read about the changes online, but you probably only care about a few of them.

USB device drivers written for Mac OS 9 (or USB 1.3 and later) can suspend and resume operations when the system goes to sleep. The new version displays device names when the "not enough power" dialog box warns you of bus problems. Performance is better for time-sensitive bus transactions when virtual memory is enabled. USB mice work better on USB CardBus-equipped PowerBooks. USB 1.3 includes support for USB Audio class devices—microphones and speakers, with mute, volume, bass, and treble adjustments. There's now built-in support for USB Communication Class devices that support the USB Communication Class abstract control model. All in all, a worthy release, especially when new drivers take advantage of the new capabilities and bug fixes.

Memory Manager

Mac OS 8.1 included new virtual memory functions that mark areas of memory as unimportant. If a program is done using a chunk of memory, it can tell VM about it. VM will then not bother writing that chunk of memory to the "VM Storage" file on disk before reusing the RAM, saving disk access and valuable time. In Mac OS 9, the Memory Manager itself uses these new functions. When a program is done with any chunk of RAM and releases it back to the system, the Memory Manager marks it as "unimportant." If any programmers are releasing memory and then referring to it anyway, it's a serious programming error, and is now more likely to cause problems than before. The Mac OS 9 Memory Manager also fixes a bug so that it no longer crashes when programs, for who knows what brain-dead reason, try to allocate a chunk of RAM sized within 32 bytes of 4GB.

Mixed Mode Manager

The part of the Mac OS that seamlessly transitions between PowerPC code and emulated 68K code sees a

revision in Mac OS 9 so that "accelerated code resources" (a special format of 'CODE' resource containing PowerPC native code) don't crash if the Resource Manager moves them in memory after they're prepared for execution.

Multiprocessing Library 2.1

Apple's now-preferred solution for dividing tasks into simultaneous chunks (for preemptive multitasking within a single program,or in preparation for offloading tasks to separate chips in multiprocessor systems) now allows most File Manager and Device Manager calls from preemptive tasks, as noted with those technologies. The library also now supports systems with more than 1GB of RAM.

Navigation Services 2.0

Introduced with Mac OS 8.5, the new non-modal "Open" and "Save As..." dialog boxes have caught on with developers and most users alike, though some remain more comfortable with the older modal "Standard File" dialog boxes. We suspect this is because after 15 years, people are comfortable with Standard File and have enough patches (Default Folder, ACTION Files) to make it workable. Nonetheless, Navigation Services is the future-it's supported in Carbon, and Standard File is not. Navigation Services 2.0 recognizes packages by default, presenting them as an individual item instead of as folders (though developers can override this). Standard File does not respect packages, always showing them as folders, and Apple advises that it will not be revised to fix this.

Other changes in Navigation Services 2.0 include the use of the new NSL Browser functionality for more comprehensive Internet connectivity (see "Living in a Wired World," this issue). The rest of the changes are for developers—easier ways to create previews and attach them to documents, a way to turn off the beveled frame around the custom control area, and new methods to both automate navigation actions and keep you from performing inappropriate ones. Developers can now relax file filtering to display more files in the list, and can prevent adding the files you choose to the "Recent Items" menu. For the most part, however, Navigation Services works like it did before, with minor enhancements and bug fixes.

Notification Manager

Added in System 6, the Notification Manager is the recommended way for programs to get your attention if they're not currently the frontmost application. Posting modal dialog boxes in their own application Mac OS 9 Hidden Features

layers isn't productive; you may never see them if they're hidden behind other windows. The Notification Manager allows three levels of attentiongetting: flashing an icon in the Application (or Apple) menu title, placing a diamond beside the application's name in the Application menu, and posting an alert. These are the wide rectangular alerts that totally block all other activity, like the ones that say "Finder requires your attention. Please chooser Finder from the Application menu."

In Mac OS 9, those blocking alerts are gone. The Notification Manager now posts such messages as floating windows with the appearance of a Post-It® note; you can continue to use other programs or go about your business with the message still on-screen. This may cause compatibility problems for a handful of programs that assumed (after ten years of no changes) that as long as the message was up, the system would not proceed. That's no longer true. Programs with the bad assumption may try to reuse the memory occupied by the message before you close the floating window. MacsBug hilarity ensues. We doubt you'll experience such problems, but it is remotely possible.

Process Manager

The Mac OS code that manages and shares processor time among multiple applications now launches control panels as stand-alone applications (as discussed earlier in "Control Panels," this article). Mac OS 9 also fixes a bug that could, in unspecified circumstances, cause slower Apple event performance.

PPC Toolbox

As noted in "Living In A Wired World," the Program-to-Program Communications Toolbox now includes an "IP" checkbox, allowing your machine to link to programs on other Macintosh systems via TCP/IP (see "Living in a Wired World," this issue). Developers also gain access to a new call that can stop a listing of ports in mid-operation (never before possible). More significantly, PPC Toolbox works with the keychain to allow invisible program linking. There has never been a way to open a program linking connection to another machine without PPC Toolbox asking you for your user name and password-for security reasons, the OS does not allow silent connection to other systems with supplied authentication material. That prevented background-only applications from linking to programs on other machines-since PPC Toolbox would insist on an authentication dialog box if guest access was disabled, and since backgroundonly applications will crash if they try to draw anything like that, no linking was possible. In Mac OS 9, PPC Toolbox can use a password on an unlocked keychain to silently link to remote systems. If the right password isn't on the keychain, or if the keychain is locked, any such attempts silently fail because a background-only application can't prompt you to unlock the keychain. It's still overly-secure, but it's more flexible than it was.

QuickDraw

Vital to the Mac OS since 1984, the system's core drawing architecture gets some much-needed improvements in Mac OS 9. For the first time ever, QuickDraw allows arbitrary transfer modes-rules for how the source and destination pixels of a graphics operation interact. If you copy a red square onto a blue background, you could get a totally red square, or one that's blended with the background to show purple, or an entirely blue result that ignores the source, and so forth. QuickDraw has always maintained several useful transfer modes, but Mac OS 9 allows programmers to supply custom code that interacts with source and destination pixels. Two new transfer modes demonstrate the capabilities-a color exclusive or mode, and an "add noise" mode that operates a bit like a Photoshop filter.

Mac OS 9 allows cursors larger than 16 pixels square, including hardware support for animated color cursors—developers pass all the cursors in a series to QuickDraw and it takes care of changing the image on a regular basis. New routines for off-screen drawing environments improve the performance of some pixel-level access functions, as well as allowing programmers to create such "graphics worlds" directly in AGP memory on AGP systems for maximum performance. QuickDraw in Mac OS 9 also fixes a bug that could leave out one column of pixels when scaling graphics for landscape printing on non-PostScript printers.

Sound Manager

Every few years Apple rewrites the Sound Manager to clean out the cruft and make it faster. Mac OS 9 includes such a rewrite, "featuring improved performance and robustness." The new implementation only lets you have one sound input source active at a time (to reduce confusion and complexity), and fixes a problem that made clicking noises during sound playback on some PowerBook systems with virtual memory enabled. Apple doesn't see the reimplementation as a big deal; there are no notes to developers to check carefully for compatibility problems or use new routines instead of old ones, but sound changes have a way of showing up in unexpected places. Keep your ears open.

(continued from page 73)

StdCLib 3.5

The "Standard C Library" is to the PowerPC Mac OS as Core Foundation is to Mac OS 9 and beyond. This set of standard routines is commonly implemented on every platform where the "C" programming language is available. Apple provides a PowerPC-native implementation of these routines in the System file for maximum performance. The new version includes many changes, but they're only of interest to programmers, who can read about some of them online.

Thread Manager

Like the Multiprocessing Library, the Thread Manager is an Apple-sanctioned way for applications to split themselves into multiple parts that run simultaneously. It's not as blessed for the future as the MP Library is, but it still works well. In Mac OS 9, three routines that should have returned errors on preemptive threads in PowerPC applications now do (they didn't before). Programs can also make a stopped thread ready from a PowerPC application, but they never could before. Apple warns developers that this last fix has not been applied to the 68K version of the Thread Manager, so it behaves as before when called from older 68K applications.

Unicode Text Utilities 1.1

Mac OS 8.6 introduced a set of new Unicode Text Utilities, providing text comparison and sorting for the complex Unicode encoding system (*MWJ* 1999.05.12). These utilities evolve in Mac OS 9 with new routines programmers can use to discover properties of Unicode characters, find text boundaries, and find line breaking points appropriate for given language areas. The routine to translate keypresses to Unicode also has some bug fixes. Documentation for the new versions isn't public yet, but the pre-Mac OS 8.6 documentation may be found online.

UTC Utilities

Although Mac OS 9's new routines for converting back and forth to Universal Time Coordinates and obtaining time measurement finer than one-second resolution are in the System file, we discussed them with the Date & Time Control Panel earlier in this article.

Virtual Memory Manager

VM is about as robust as it's going to get with the existing Mac OS. Until Mac OS X provides an industrial strength kernel and file system, virtual memory can't grow beyond the limitations of the Mac OS's singlethreaded file manager and partitioned application memory model. Within those limits, however, it's doing pretty well. Mac OS 9 adds a single new call that returns information about the disks VM uses for backing storage (the call is not available in Carbon and won't be because it inherently assumes the classic Mac OS VM model). Two bugs were fixed as well, both of them obscure: VM no longer marks a page of RAM as "recently used" (meaning it stays in RAM) when a non-DMA disk driver tries to write that page to disk and reuse the RAM, and it fixes a very rare bug when using fast striping RAID arrays for VM Storage and one 4K page falls across a stripe, locating parts of it on multiple drives. In that bizarre case, a read of the affected 4K chunk could cause data corruption.

Rest In Peace

Before ending, let us pause for a moment of silence for five Apple technologies abandoned once and for all in Mac OS 9. Their service is appreciated, and their demise mourned by those who depend on them.

• **Apple Telecom Software**, for the GeoPort modems, is officially dead.

• **Color StyleWriter 4000** drivers, for the StyleWriter printers made by Hewlett-Packard, are not supported in Mac OS 9 and will be removed by the Installer if detected.

• The **Energy Saver** Control Panel is still present and supported, but the API used by applications to control energy settings is absent and no longer supported.

• The **LaserWriter 8f** driver, a version of LaserWriter 8.2 with special code resources implementing a PostScript Fax option, crashes under Mac OS 9 and will not be updated.

◆ And we bow in respect to **QuickDraw GX**, the graphics architecture Apple released in 1994 that blew away anything else in the industry before or since. GX Printing went away in Mac OS 8, and per prior notices, the QuickDraw GX Graphics and Typography engines are no longer supported in Mac OS 9. Reports from MultiAd Services, a company using GX in its Creator2 product, indicate that the GXGraphics extension still works in Mac OS 9 but only if you remove ColorSync. GX only knows about ColorSync 1.0 and can't coexist with the new version. Its ideas live on, and we shall miss it. ◆

The Weekly Attitudinal

Do You Really Want It? After All This, The Breakdown On Mac OS 9

After two weeks knee-deep in lubricated Mac OS 9 innards, allow the *Attitudinal* to be succinct: Yes, you want Mac OS 9. You may not want it this week, and that's OK, but you will want it in the near future.

If you're either a multi-lingual person or someone who needs to connect remote computer networks together via modem, the new bundling alone is worth the price. The new OS includes seven former language kit products that would have previously set you back nearly US\$1000. The former Apple Remote Access Personal Server 3.0 would have lifted around US\$125 from your personal financial empire. If even one of these eight bundled items is useful to you, the US\$99 upgrade price (*before* the US\$20 rebate) is already a bargain.

The Attitudinal is, as usual, less impressed with Apple's marketing focus, but the Nine Num-Nums of Upgrading are fairly tasty this time out. Sherlock 2 has problems but great new features, and the Attitudinal completely disagrees with TidBITS that the new interface for searching on multiple file attributes is a {"giant step backwards" lhttp://db.tidbits.com/getbits.acgi?tbart=05625}. The Attitudinal invites anyone to construct a search with five criteria in Sherlock and then try to eliminate the third one. Your only choice is to wipe out the last three and start over, or to try to change the third item so that it's non-restrictive. These "More Choices" and "Fewer Choices" dialog boxes have long outlived their usefulness due to this awkward modification, and the Attitudinal says good riddance. Pfft.

The Attitudinal's main problem with the Nine Internet Power Outlets (or whatever) is that each one seems aimed at a different audience, and none alone is enough to please said group. Multiple Users is a fine add-on for folks who need to share one computer among multiple household members. The upgraded Network Browser fits in well for installations with many networked Macs all running Mac OS 9. Apple File Security keeps snoopy-noses out of sensitive files if Multiple Users isn't enough, but it's unsuitable for Internet work or heavy-duty personal security. Apple events over IP, ColorSync 3.0, LaserWriter 8.7, and internal font changes will tickle the publishing market right in their color separations. None of these target groups is likely to dance with glee at the features intended for the other.

Yet, as when the iMac was introduced, those criticizing Mac OS 9 for no single stellar feature miss the *gestalt* of the overall package. Combine all these features with the death of the archaic limit on open files, the evolution of Sherlock 2's Internet searching for items sorted via price or availability, the overdue expansion of PlainTalk 2.0, the nifty but superfluous Voice Verification, and tons of work on the details from the desktop themes to new alert sounds (a few "Mac OS 8.7" alerts notwithstanding), and you have a very nice evolution of your primary system.

Those who criticize the release for having too many features also miss the point—who's forcing you to use them? If it's your machine and yours alone, don't turn on Multiple Users, ya nimrod. Enjoy thousands of open files and use that extra RAM you bought the way God intended. Don't need LaserWriter 8 features? Try using ColorSync with your EPSON printer and get better output. Skip AppleScript over IP and enjoy more productivity with the dozens of tiny changes that enable new features. Quit looking for something to complain about.

If your favorite applications are on the "incompatible" list right now, then by all means wait a few weeks. Starting now, Apple's new machines all ship with Mac OS 9 installed and no guarantee earlier versions of the OS will work; developers will not leave these lucrative markets long untapped. You do not owe Apple an upgrade any more than they owe you features aimed precisely at your particular needs, but if the preponderance of the changes leads towards a better experience for you, go and get it. No pressure, no guilt.

It's not "a whole new Macintosh for US\$99." It's not going to take the Internet by storm. It's not even going to increase your absorption of essential vitamins and nutrients. It makes your Macintosh easier and more fun to use. Once compatibility problems are resolved by third-party developers, it will increase your productivity. That's all one could ask, and Apple has delivered.

It matters not if you are the first or last on your block to install it. Grab it when you and your software collection are ready. But yes, Virginia, you will want this OS upgrade. It is thoughtfully assembled and more encouraging than frustrating. The *Attitudinal* pronounces Mac OS 9 worthy.

As if you care.

About MWJ

MWJ, the Weekly Journal for Serious Macintosh Users, is published each Saturday morning with small schedule variants for holidays and other forces of nature. Each week's issue contains summaries of all the week's Macintosh-related news and product releases, complete with detailed examinations of those areas worthy of further study. Most issues contain one or two feature articles as well, a number that bloats significantly when major news warrants.

The material in this Mac OS 9 Special Edition was originally published in *MWJ* 1999.10.23 (the day Mac OS 9 was released) and *MWJ* 1999.10.30. The discussion of error 119 in "Mac OS 9 Basics" originally appeared in *MWJ* 1999.10.09. This special issue combines our original two-part coverage into a single document, and integrates follow-up material into the source articles (for example, the discussion of Sherlock 2 advertising appeared as a follow-up in *MWJ* 1999.10.30, but here it's integrated with the rest of the Sherlock 2 coverage).

MWJ is available in both PDF and plain text formats, both digitally signed each week using PGP technology. Subscriptions cost US\$10 per month. An absolutely free three-issue trial subscription is yours for the asking at our Web site (we send one follow-up letter after your third issue, but that's it—we do not share your E-mail address). Our readership includes major Macintosh developers, journalists, and executives. We would welcome your evaluation of our journal.

Here are some of our relevant Internet addresses:

Letters to the Editor

<letters@gcsf.com>

"Ask the Staff"/"MacCyclopedia" <ask@gcsf.com>

Subscription Changes <subscribe@gcsf.com>

Comments Not for Publication <mwj@gcsf.com>

Press Releases

<pr@gcsf.com>

Other Questions or Information <info@gcsf.com>

MWJ • Washla Lauraa

The Weekly Journal for Serious Macintosh™ Users

is published by



blisherMatt Deatherage	
itorMike Opitz	
oducts Editor Jackie Verdun	
0	
8 5 /	
tters <letters@gcsf.com></letters@gcsf.com>	
ss	
Magnetic nour information	
0	
ong opinion, parody, biting	
casm, and things you need to	
ow. Those easily offended	
ere.	
	itorMike Opitz oducts EditorJackie Verdun ntributing Editor Jerry Kindall tters <letters@gcsf.com> Contains news, information, ong opinion, parody, biting casm, and things you need to ow. Those easily offended ould seek information else-</letters@gcsf.com>

Regular issues of *MWJ* are digitally signed using PGP technology to verify the integrity of the transmission. Our DH/DSS corporate PGP key may be obtained online.

Copyright © 1999 GCSF, Incorporated. All Rights Reserved.

All trademarks are the property of their respective holders.

P.O. Box 1021 El Reno, OK 73036-1021 (405) 262-1399 (405) 262-1560 Fax <mwj@gcsf.com> <http://www.gcsf.com>