Chapter 20

Palm OS: A Class Case Study

In this chapter:

20.1 OVERVIEW

In Chapter 4 we discussed some elements of the Palm operating system. That discussion was mainly limited to issues that arose as we studied the more complex design goals of this OS compared to the ones studied before it. This chapter is nominally about the Palm OS versions prior to version 5. This OS represented a particular niche in the hierarchy of OSs that was described in Part 2 of this text. As such there is not a great deal more that can be said about this OS that was not covered in Chapter 4. So this chapter starts with a series of sections that parallel the other two case study chapters and provide some details that were not relevant to Chapter 4. But other material is added here that helps place this OS in the computer industry as it is evolving today. We discuss some aspects of programming such platforms and the trends of the applications that are developing in the industry.

We begin this chapter with a brief restatement of the type of environment that the Palm OS is designed for in Section 20.2. Sections 20.3 through 20.5 briefly summarize the related points of Chapter 4. Section 20.6 then discusses several developments that

have occurred in the area of the input/output subsystems in the Palm OS beyond the fundamental things we covered in Chapter 4. These features are typical of new handheld platforms. Sections 20.7 and 20.8 also summarize related sections of Chapter 4. Section 20.9 explains the nature of the cross-development systems needed to develop programs for such a limited environment. PDAs, cell phones, and multimedia players were originally different sorts of devices, but these platforms are currently undergoing a merger. So Section 20.10 discusses how software is evolving in these platforms. It also touches on some of the developments in later releases of the Palm OS. We conclude with a chapter summary in Section 20.11.

20.2 THE MULTI-PROCESS OS ENVIRONMENT

The Palm OS and its environment were discussed in Chapter 4, but they are reviewed here for convenience. The Palm OS is designed for a very specific type of environment. There are several characteristics of this environment that restricted the design of the OS. The environment characteristics are briefly outlined in Table 20.1.

The primary characteristic of the Palm OS is the small screen size. This means that the user is only interacting with one program at a time so that applications assume that their window fills the entire screen at all times except for small notice boxes that may pop up in front of the main window.

Although later models sometimes included disk drives, especially as an add-on feature, the initial machines did not include them and the OS design assumes that all programs reside in primary memory. The lack of a keyboard means that the OS must provide for handwriting recognition. This feature is a real-time application, so a real-time kernel underlies the Palm OS. User applications, however, are not real time and are single threaded. The resulting OS design choices are listed in Table 20.2.

TABLE 20.1 Unusual Characteristics of the Palm Platform

Small screen size
No secondary storage
No text keyboard—touch screen
Limited power for better battery life
Slow CPU to reduce power
Limited primary memory

TABLE 20.2 Unusual Characteristics of the Palm OS

Programs never stop
No demand paging (virtual memory) or disk caching
Single-window GUI
Multiple text input options
Real-time OS tasks but non-real-time applications
No application multithreading

20.3 PALM PROCESS SCHEDULING

20.3.1 Real-time tasks

The process scheduler in the Palm OS is a preemptive multitasking priority scheduler. It will dynamically determine which task that is ready has the highest priority and it will interrupt the running of a less important task to run a more important one that becomes ready. The underlying OS is a real-time kernel for support of handwriting recognition, but user application programs cannot access these functions. Handwriting recognition is divided into two parts: stylus tracking and character recognition. The stylus tracking task processes interrupts from the stylus using a standard interrupt mechanism. When the stylus tracking determines that the stylus has changed direction, stopped, or is no longer touching the screen, it will calculate a vector describing the movement and will pass this information on to another task that is running, the character (graffiti) recognizer. If this routine recognizes a character, then it will pass this information on to the OS so that it can decide what to do with the character. Usually the character will be passed to the application that has the focus to be placed on a control on the current form where the user is entering text. Of course, the character might be a control character instead of a text character and the application may then be given a message telling it about the event. If the screen touch is not in the graffiti area, then the OS must detect screen taps on form buttons or pass the information on to the application—perhaps it is a drawing tool, for example.

20.3.2 Other tasks

Only a single-user application has the focus, and that application is most likely waiting for user input as just described. But it is normal for Palm systems to have background communication functions running such as telephony, database synchronization, Bluetooth connection to local devices such as headphones, and Internet access for browsing and email. In addition, certain user features such as searching for a name will invoke a search function in applications that do not currently have the focus. Tasks that do not have the focus will be running in an event loop waiting for signals about events requesting them to do some work. The Palm scheduler module will see that each application gets some time to do its work.

20.4 PALM MEMORY MANAGEMENT

Processes in the Palm OS are always resident in primary memory. Once a process has begun running it never really stops. It may lose the focus, in which case it will not be running anymore, but it is still there waiting to be selected from the menu again and resume execution.

The memory manager in the Palm OS treats a large block of a primary memory as a heap. As memory is allocated and freed on the heap, the eventual result is external fragmentation. This requires occasional compaction to aggregate larger blocks of memory. To facilitate this, items in memory are addressed indirectly through a

memory pointer table (MPT). Thus, when the memory manager moves an item, it merely updates the MPT entry that points to the item. The details of this mechanism were covered extensively in Chapter 4.

20.5 FILE SUPPORT

The Palm OS programmer's documentation refers to "databases," but these are actually random access flat files. They are accessed by an "index" value that is a 16-bit integer. Records are of variable length and can be resized dynamically, and added and deleted. The file manager maintains an index for each database giving the current location in memory of each record in that database. A database must fit entirely within a single memory card.

20.6 INPUT/OUTPUT SUBSYSTEMS

Early Palm devices were merely used as PDAs. As was mentioned, these systems have evolved to cover many additional functions including games, cell phones, Web browsers, and media players. The initial Palm releases had limited functionality in the audio area in particular, and these have been enhanced significantly through the various releases. In addition, the platform evolution has also seen advances in communication and networking functions. This section discusses the audio functions. The networking functions are covered in a later section in order to maintain a parallel structure with the other case study chapters.

20.6.1 Audio I/O

The continuing development of technology has led to a rising interest in portable music devices. In addition, the advanced games that users want to have on these machines needed enhanced audio features. The initial sound support in the Palm OS was limited to short sounds for alerts and a few noises for games. Later versions added support for a low-level implementation of a **musical instrument device interface** (MIDI) so that more elaborate musical sounds could be created by an application. This led to a number of interesting applications for musicians to use a Palm OS device as a simple musical tool, such as a tone generator and a metronome. Later versions of the platform also added more advanced sound support, allowing these devices to be used as cell phones and to play music files. Later they also added the ability to perform voice recording and playback—audio notes to oneself—and the recording and sending of audio to other cell phones.

20.6.2 Stream I/O

For purposes of ease of programming, the Palm OS also includes a version of file streams similar to the **stdin/stdout** functions available in most C language libraries. These functions use the database structure discussed in the File Support section but allow easier porting of some applications to the Palm OS.

20.6.3 RAM disk driver

The underlying OS is designed to be used in embedded systems where there is commonly no secondary memory. But many applications are developed to run from a standard file system–style interface. Therefore, the AMX OS comes with a predefined RAM "disk" driver that uses a portion of RAM to emulate a disk drive. This allowed the Palm OS to readily define a RAM drive as a DOS-formatted floppy drive so that porting applications to the Palm were easier and programmers did not have to learn a separate interface to use the system.

20.6.4 Cameras

The development of low-cost, high-resolution CMOS image sensor technology has meant that many cell phones and PDAs now include cameras. Moreover, since larger memories are now available, the cameras can even record video files as well as static images. So now these devices are capable of transmitting image and video files in addition to audio files.

20.6.5 Communication circuits

Bluetooth and 802.11 Wi-fi communication circuits are now available and have been incorporated in the latest Palm devices. These allow other synchronization pathways but also a merging of the PDA and cell phone device classes and to Internet access devices such as the Blackberry™.

20.7 GUI PROGRAMMING

The GUI environment on the Palm platform was extensively covered in Chapter 4. The principle factors that distinguish this platform are the small screen size and limited memory. As a result of the small screen size, the Palm system does not support tiling the forms of applications. (The Palm OS uses the term "form" for a normal window.) Pop-up boxes from a single application are allowed but the pop-up boxes must be closed before the application can continue. (The Palm OS calls these boxes windows.) The limited Palm memory led to the development of specific windows that can be created by Palm applications merely by filling in specific data structures and calling OS routines. The OS will then take on the task of displaying the window and closing it when the user selects an option.

20.8 NETWORK PROGRAMMING

20.8.1 Personal data synchronization

It is natural that a portable device would need to have strong support for communication protocols. Since the Palm devices were initially envisioned as PDAs, the most important communication application was synchronizing with a PC so that the data

in the handheld unit could be backed up. Accordingly the initial interfaces provided with the Palm OS were low-level drivers for serial, infrared, and USB ports. At the same time there was a higher-level interface provided for writing applications for synchronization personal information such as contact lists and appointment calendars. A consortium of interested vendors was created known as the Versit Consortium. They have defined a set of standards concerning **personal data interchange (PDI)** that include standards for a **vCard,** an electronic business card, and **vCalendar,** an electronic calendar and scheduling exchange format. These standards are now maintained by the Internet Mail Consortium. The Palm OS includes a library that allows an application to open a PDI stream as either a reader or a writer to facilitate the development of synchronization applications.

20.8.2 Other data synchronization

Some users will be concerned with developing custom applications that go beyond traditional PDA applications. They may have specific data files that need to be synchronized between a Palm application and a similar application on another platform. So the Palm OS provides **exchange libraries,** which act as plug-ins to an OS module called the Exchange Manager. They allow Palm OS applications to import and export data records without being concerned with the transport mechanism. For example, one exchange library always available to Palm Powered™ handhelds implements the IrDA protocol, IrOBEX. This allows applications to beam objects by way of infrared from one Palm Powered handheld to another. Similar exchange libraries exist for other hardware ports and other protocols such as the **SMS (short message service)** library, email protocols, and the Bluetooth library.

20.8.3 Internet applications

During the last several years the Internet has risen in popularity to the point where it is almost mandatory that handheld units be able to access many of the popular features found there. In particular, these include accessing World Wide Web (WWW) sites as well as the email protocol already mentioned. Accordingly, more protocol stacks and APIs have been added to the Palm OS to support networking applications. The first addition was the widely used and well-known lower-level **Berkeley Sockets** API. This interface allows a programmer to connect to services on other systems using a variety of protocols without having to implement that protocol in the application. The interface included with the Palm OS allows either TCP (connection-oriented) or UDP (connectionless) communications.

The second level of protocol supported in the Palm OS includes support for Application layer protocols such hyperText Transport Protocol (HTTP), the protocol used for the WWW. This protocol is used by Web browser and Web service applications for Palm devices. There are some interesting problems to be solved when developing a browser for a Palm unit because initially few websites are developed with the very small screen space of a handheld unit in mind. However, current HTML attributes allow a Web server to determine that a browser is running on a mobile platform and to adjust its output to fit.

TABLE 20.3 Telephony API Service Sets

Service set	Functionality
Basic	Functions always available
Configuration	Configure phones including SMS
Data	Data call handling
Emergency calls	Emergency call handling
Information	Retrieve information about the current phone
Network	Network-oriented services, including authorized networks, current network, signal level, and search mode information
OEM	Allow manufacturers to add features to the Telephony Manager and provide a new set of functions for a device
Phone book	Access the Subscriber Identity Module (SIM) and address book
Power	Power supply–level functions
Security	Provide PIN code management and related services for phone and SIM security-related features
Short Message Service	Enable reading, sending, and deleting of short messages
Sound	Phone sound management, including the playing of key tones and muting
Speech calls	Handle the sending and receiving of speech calls; also includes Dual-tone multi-frequency (DTMF) signaling

20.8.4 Telephony applications

In Section 20.10 we discuss the current merging of PDA devices with cellular telephones. The Palm Telephony Manager provides a set of functions that allow an application to access a variety of telephony services. The telephony API organizes the functions in groups called service sets. Each service set contains a related set of functions that may or may not be available on a particular mobile device or network. One of the API functions allows the application to find out if a given service set is supported in the current environment. A list of some of the more common service sets is shown in Table 20.3.

20.9 PROGRAMMING ENVIRONMENTS

The resources available on a computer designed to run the Palm OS are usually not sufficient to develop software. The Palm OS programming website suggests that programs designed for a Palm-based system be designed to support only a minimum amount of data entry. This suggestion is made partly because of the difficulty of inputting data with the handwriting recognition, but also because of the very limited screen display. Instead, Palm suggests that the user should mainly input data on a desktop system and use the Palm system for referencing the data. Furthermore, once a program is running on the Palm OS there would be no simple way of getting any debugging

information displayed and the environment is obviously not well suited for the entry and editing of program source code. Moreover, there rarely are printers attached to Palm systems and as was noted before, the CPUs are very slow and there is generally limited RAM and rarely any secondary storage. Most program development is therefore done on another system, a concept known as cross-platform development.

There is a wide variety of languages and tools for development of software for the Palm OS. Some is available from Palm itself and others are available from third parties. These include commercial integrated development environments (IDEs) such as CodeWarrior™ from Metrowerks and free tools such as PRC-Tools, which is a gcc-based compiler tool chain for building Palm OS applications in C or C++. Tools that are supplied by Palm include a Software Development Kit (SDK) that includes the headers, libraries, and tools for Palm OS platform development on Windows, Linux, and the Mac OS. It also includes a version of the Palm OS running in native X-86 code on a Windows machine. This emulation offers an easy way to test applications destined for the Palm OS for compatibility. Compilers are also available for developing applications in other languages, including Visual Basic, Pascal, Forth, Smalltalk, and Java.

An essential feature is a package that Palm calls an Emulator. This is a software package that emulates the hardware of the various models of Palm OS platform devices on Windows, Linux, or Mac OS computers. Since various platforms have different features available in their ROM, ROM images for use with the Palm OS Emulator are available to emulate each desired model.

Once a program has been developed with the cross-platform tools, it can be installed on a Palm device using the synchronization tools included with Palm PDAs that are available for the various cross-development platforms.

20.10 SIMILAR SYSTEMS AND CURRENT DEVELOPMENTS

One of the difficulties facing authors who write about computer science is that the state of the industry changes so rapidly that a book is not reflective of the latest developments even on the day it is printed. Operating Systems are no exception. There have been rapid developments in the hardware systems used for the Palm OS and others of its ilk discussed in this chapter. In addition, a different functional view has captured the minds of the public and the vendors that have forced some changes in the OS. As a result, other OSs that were developed for this different view have some features that are more complex than the Palm OS described here. But then, so do later versions of the Palm OS.

In this section we describe some features found in other OSs for small systems. We mostly mention the Symbian OS. This OS is developed by Symbian Ltd. It is a descendant of Psion's EPOC OS and runs only on ARM processors. Symbian is a consortium of manufacturers of cell phones.

20.10.1 New functional models

In addition to the use of more advanced CPUs, the basic functions of small handheld systems have also evolved in the last few years. At the beginning of this century the products in this area were mostly envisioned as either PDAs or cell phones. In PDAs

the applications were things like phone and address books, appointment calendars, calculators, memo pads, to-do lists, specialized data bases, and an occasional specialized application. In cell phones the main application was the phone book or contact list. In either case, these were primarily standalone applications that required occasional connection to another computer for purposes of synchronization, backup, and loading new applications. In the cell phone the actual telephone application was a real-time task that was considered to be a fundamental function of the device rather than a separate application. These cell phones were closed systems in that installation of additional applications was not part of the design.

Lately, however, a new model has evolved for handheld devices. This evolution has come about partly because of the revolution in the availability of communications technology and ubiquitous connectivity. The devices are now positioned as mobile communications platforms—but as much more than just replacements for cell phones. The main feature that distinguishes a cell phone from a PDA is that PDAs are turned on, used for some single function and then turned off, while a cellular phone is normally left on most of the time and is continuously connected to the network and waiting for incoming calls. In addition to waiting for calls, a cellular phone does other work to manage the connection such as keeping the time synchronized and conversing with the cellular network so that if a cell phone is turned on the network knows where it is. In order for the PDA functions to be used while the cell phone is handling the network connection, the OSs for these mobile communicating devices have to incorporate multiple active tasks at the same time. In a pure PDA device such as the original Palm products, the OS provided only a few separate tasks so that handwriting recognition and synchronization could take place while a **user interface (UI)** application was also running. (You may recall the screens are so small that there is no room for more than one UI application to be executing at a time.) However, there was no provision for applications to provide any separate background function such as managing the connection to a cellular network and checking for incoming calls. As a result, these OSs all have added more features to support application multithreading. More importantly, applications can now start a thread as part of a background task in addition to the single foreground UI thread. An example of the utility of such a feature would be that a service can be built that will handle multiple TCP/IP connections at the same time, so that several TCP/IP-based applications can all be running at the same time using a single TCP/IP multithreaded service. The TCP/IP service will be running as a "user" application rather than as part of the OS kernel. Another important example is a Web browser. When a page is fetched from a server, the images and other included items are not automatically sent. The browser must parse the initial page and then individually request each referenced element. The browser must be able to continue to work on displaying the main page for the user while the other elements are being fetched from the server. This gives the user some immediate access to the contents and a smoother browsing experience.

As in the case of the cellular phone connectivity, other applications can benefit from these background tasks without having control of the UI. Some of the more obvious ones are playing an audio file, downloading new audio files to play, instant messaging, and having an email program connect to a server to check for new email. Other, less obvious background functions exist as well, such as synchronizing

changes to distributed databases and updating installed software. These new tasks help cell-phone manufacturers to differentiate their products from one another. In addition, users are asking for these features because they are beginning to value the instant access to information through messaging, email, and the World Wide Web. In order to provide these features, these devices have incorporated advanced hardware components to handle the multiple communication streams, multimedia streams, and so forth. The OSs have had to improve as well. Not only can applications start many threads and set different priorities for each thread, they have new mechanisms to synchronize between the threads and with other processes, to share memory segments with other processes, to communicate with other processes, and to protect databases.

20.10.2 Advanced communication models

Other advanced facilities being provided by the OSs now include encryption and other security mechanisms, new Data Link layer modules such as Bluetooth and 802.11x, and APIs so that new user applications can easily access these OS functions. As the cost of bandwidth from communication services continues to decrease, we are beginning to see more and more intense multimedia applications. The near term projections of the marketplace include more streaming multimedia applications. These applications have heavy hard and soft real-time requirements. These small OSs will continue to evolve with the increasing requirements.

20.10.3 Thread scheduling

As discussed earlier, cell phones are used somewhat differently than PDAs. PDAs are generally turned off when not being used, but cell phones stay on all the time so that they can wait for incoming calls and keep the network updated about the location of the phone. In addition, the demands of the cellular technology are such that real-time tasks are needed to service the network. Accordingly, the scheduler used in the Symbian OS is a priority-based multithreading scheduler. Any application can be a "server" and can create multiple threads of execution within its address space. The Palm OS included a real-time scheduler because of the needs of the graffiti handwriting recognition program, but user applications were not able to create real-time tasks or threads. We discussed such schedulers in Chapter 8.

20.10.4 User interface reference design

The user interface (UI) for most PC systems is very flexible. Windows can fill the screen, shrink to a smaller size, move around, cover one another, and so on, depending on the whim of the user. These smaller systems, however, have simpler interfaces than personal computers do. Often the application assumes that its window fills the entire screen. There are generally three different types of UI in such devices. They roughly represent the classes of a cell phone, a PDA, and a handheld computer. The user interface class is an abstraction of the features that the members of each family have in common. These classes are summarized in Table 20.4.

The challenge for the OS designers and for application programmers is to write OSs and applications that will run on any of these different platforms without a major

TABLE 20.4 Small Systems Device Families

Cell phone	Small vertical screen Keyboard with digits and a few buttons Almost no user input Application has full screen
PDA	Larger vertical screen Stylus input and a few buttons Limited user data input Application has full screen
Advanced	Larger still horizontal screen Full QWERTY (usually) keyboard More extensive user input Application windows can overlap, etc.

rewrite. The desire for such portability forces system implementation into strict object-oriented designs in order to isolate the UI functions from the rest of the application. Industry estimates are that about 80% of an application can be isolated from the UI.

The rise of the popularity of the Internet have lead to the incorporation of several standard applications in these new devices. In particular, users want to send instant messages, work with their email, view (or listen to) streaming multimedia transmissions, browse the World Wide Web, and upload multimedia files to their websites. These small systems have very limited screen size, and Web pages are typically set up for at least 800 × 600 pixels. As a result, a browser on a cell phone has to work really hard to make an intelligent display of a larger Web page. Initially a separate standard was created for building Web pages intended to be viewed on a handheld system—**wireless markup language (WML),** a part of a larger standard, **wireless access protocol (WAP).** Later developments seem to indicate that standard browsers can be modified to display standard Web pages on handheld systems. This is an area of active research.

An additional protocol has been developed for sending short text messages when a phone call is not appropriate. For example, it might be used when the recipient is in a very noisy environment, in a lecture or arts performance, or in a meeting. IT can also be used for short queries where a complex interaction is not needed. This protocol is the short message service, or SMS. It allows the sending of messages up to 160 bytes long. It can be used similarly to **instant messaging (IM)** services on normal PCs, but IM is typically interactive while SMS typically uses one-way messages.

20.10.5 Location-aware applications

Another obvious but still interesting facet of these systems is that they move around with the user. After some time it became clear that there were some interesting applications that could be created if the application knew where the phone was. The initial impetus was probably the emergency location system that has been mandated for cell phones. In an emergency there is obviously a great benefit available if the cell phone can tell the emergency call handlers where the cell phone is located within a few tens of meters.

There are many other location-based applications that can be created as well. Since the cellular carriers in the United States were mandated to have the network able to locate the phone, they have decided to make lemonade out of those lemons by devising services that they can offer to their users (for a modest fee, of course) based on the current location of the phone. Where is the nearest pizza restaurant? Dial *1411 (or some similar special number) and a friendly operator will get your location from the network, ask what you are looking for, search a location-indexed database for the nearest Chinese restaurant, and give you the information. Some of the other obvious applications are general driving directions, traffic reports, and weather reports. Of course, these systems can also be computerized, eliminating the human operator.

An interesting question is, how does the phone (or the network) find out where the phone is? One answer is the federal GPS system. There are a few dozen satellites that are in orbit purely for this function. Initially they were installed for the benefit of the military, but since using the satellites merely involves listening to their broadcasts, it was impossible to keep civilian uses out forever. By locating several satellites at one time any device can determine its present location, including altitude. This is an extremely accurate mechanism, to within a few feet in many cases. However, the hardware costs are still somewhat high compared to most of the rest of the phone. Fortunately, there are at least two other ways to find the location of a phone. The first is just triangulation of the phone by the network. All the cells that can hear the phone will report the timing of the signals from the phone and the network will be able to locate the phone within a hundred feet or so. This is not accurate enough to drive a car, but it is usually accurate enough to locate the nearest post office, for example. The other location mechanism is for the network merely to report which cell is currently servicing the mobile device and perhaps a distance from the tower based on signal propagation times. Although this method is even less accurate than the triangulation, it is still accurate enough for many purposes.

20.10.6 Later Palm OS releases

Beginning with release 5 the Palm OS supports an ARM processor instead of the Motorola CPU used in previous platforms. Beginning with the 5.4 release the PAI came to be known as Garnet. The PalmSource company that had been spun off of Palm Inc. was purchased by a Access Co. Ltd. They have created a release of Linux with the Garnet API for use on mobile platforms. The latest release of Palm OS was version 6. It is named Cobalt.

20.11 SUMMARY

In this chapter, we discussed further the features and concepts of a simple modern OS—the Palm Operating System™ developed by Palm, Inc. This OS was developed for small handheld devices. Although this is a single-user system, it can concurrently run some OS processes and a small number of applications.

We started this chapter with a recap of the process scheduling and memory management functions of the OS. We then followed this by discussing several additional I/O subsystems in the Palm OS, GUI and network programming, and by explaining the process of developing programs for these limited

environments using simulators and cross-compilers on larger systems. We continued with a discussion of some similar OSs for limited environments and how they differ from the Palm OS, including later versions of the Palm OS itself, actually a different OS for a different CPU. We further discussed some of the new types of applications emerging from the convergence of PDA and cell phone platforms.

BIBLIOGRAPHY

Exploring Palm OS: Palm OS File Formats, Document Number 3120-002. Sunnyvale, CA: PalmSource, Inc., 2004.

Exploring Palm OS: System Management, Document Number 3110-002. Sunnyvale, CA: PalmSource, Inc., 2004.

Palm OS Programmer's Companion, Volume 1, Document Number 3120-002. Sunnyvale, CA: Palm, Inc., 2001.

Palm OS Programmer's Companion, Volume 2, Communications, Document Number 3005-002. Sunnyvale, CA: Palm, Inc., 2001.

Palm OS® Programmer's API Reference, Document Number 3003-004. Sunnyvale, CA: Palm, Inc., 2001.

Rhodes, N., and McKeehan, J. *Palm Programming: The Developer's Guide.* Sebastopol, CA: O'Reilly & Associates, Inc., 2000.

SONY Clié, Personal Entertainment Organizer, Sony Corporation, 2001.

WEB RESOURCES

http://www.accessdevnet.com (ACCESS Linux Platform Development Suite)

http://www.freescale.com

http://www.freewarepalm.com (free Palm software)

http://www.freesoft.org/CIE/ (*Connected: An Internet Encyclopedia*)

http://www.imc.org/pdi/

http://oasis.palm.com/dev/palmos40-docs/memory%20architecture.html

http://www.palm.com (Palm home page)

http://www.palmsource.com/developers/

http://www.pocketgear.com/en_US/html/index.jsp (software for mobile devices)

http://prc-tools.sourceforge.net (programming tools supporting for Palm OS)

http://www.symbian.com (Symbian OS)

http://www.w3.org/Protocols/ (HTTP, primarily)

http://en.wikipedia.org/wiki/Graffiti_2 (article on Graffiti 2)

http://en.wikipedia.org/wiki/Palm_OS (history of the Palm OS versions)

REVIEW QUESTIONS

20.1 Since almost no websites are developed with the assumption that the screen size is 160 × 160 pixels, of what use is the HTTP protocol support?

20.2 What does a Palm device need a RAM disk driver for?

20.3 How does a programmer go about creating and testing programs for the kind of handheld platforms discussed in this chapter?

20.4 What are some of the new device types and features that have been added to the Palm platform since the earlier models and what sorts of applications do they facilitate?

20.5 What is a vCard?

20.6 What is a "location aware application?"

20.7 Describe the three different families of handheld systems.

20.8 One of the major differences between a cell phone and a PDA is that a PDA is turned off and on and a cell phone usually stays on most of the time. What major feature did this force to be included into OSs designed for cell phones?