small

utils

The Small Utilities

Version 2.0
For Windows 95

User's Manual

The Small Utilities 2.0 for Windows 95 have gone from being nice to have in their previous version to being essential system tools no computer should be without in this new version. This manual is your key to supercharging your computer with the productivity and power of the Small Utilities 2.0!

Named **Superior Shareware** by *Windows Magazine* in May 1996…

… and they'd only seen version 1.0 !

What are they saying about the Small Utilities?

- Windows Magazine named the Small Utilities "**Superior Shareware**" in May, 1996.

    - George Steffanos says of the Small Utilities package: "...has **constantly made my life easier and enhanced my computing experience**. The instructions for each utility are clear and concise, allowing even a novice like myself to quickly gain a working grasp of its function. For more advanced users, the author has added tips towards integrating the utilities in batch files, etc. ... This is one shareware package which I would recommend to anyone."

Contents

Notes

This documentation was created in Microsoft Word 95, and is in .DOC format.

If you do not have Word, I recommend you use the free MS Word viewer application (available on Microsoft's Internet World Wide Web page) to view the file. This is actually a stripped down version of Word with all editing functions removed, so it will print the file exactly as I formatted it. If you do not have this program (it came preinstalled on my PC), it is well worth getting since Word file format is very common these days -- its ubiquitous nature means a lot of documentation is going to be distributed in it -- and Word Pad does a lousy job with documents created in the full Word 95 product.

As a last resort, you can use WordPad or the Windows 95 Quick View program to view and print the manual, but I strongly discourage this since Word Pad will not preserve the page breaks, tables, and a lot of the formatting of the manual. Get the free Word Viewer!

If your non-Word word processor can swallow DOC files and come up with something that looks decent, you can use your word processor to view and print the manual, but you're on your own there.

Notes on typesetting:

- Some pictures in this manual may not exactly match what you see on your screen, since I have had to convert some to black and white so they would print out on black and white printers.
- My goal is to make this manual look good on the HP 600 series ink jet printers, since that's in line with the capabilities most printers have these days. If you have a better laser printer, the document will look sharper, and if you have a less capable printer the document will look okay but not wonderful. On most black and white printers, the manual will look decent.
- For the purposes of distribution to the widest possible audience, this manual uses only fonts which come with Windows 95.
    - This manual is laid out to look good printed single-sided or double-sided.

Shareware, Legal, and Registration Information

About The Small Utilities And Shareware

The Small Utilities 2.0 is being distributed as shareware. If you do not know what this means, it is a method of software distribution where end users can download (or acquire through other means) the entire software package, evaluate it to see if the software package meets the user's needs, and then pays for it if it does. The marketing overhead for the software is nonexistent, which means you as the end user are not paying for overhead. Your $100 for an over-the-counter software package at a software store funds full-page "feel good about the company" non-specific ads in trade magazines, unread direct mail which goes directly into the trash, and all those postcards giving you special deals on upgrades (which tend to come a month after you have already ordered the product from a catalogue for $20 less than the special offer's price). Shareware also allows for specialized software to be developed; with consolidation in the retail software industry, the only software which gets on shelves is that which can broadly appeal to everyone. This leaves out a lot of specialized but worthwhile software which could sell enough to support itself but still not be attractive to mass market outlets. Shareware allows a distribution outlet for software that might not appeal to the masses, but which would appeal to enough people to make it worth writing.

Shareware only works if users hold up their end of the bargain and pay for the software if they use it. By supporting shareware, you're enabling me to write more of these programs and enhance the Small Utilities. By not supporting it, you're telling me I'm wasting my time and that I need to move on.

The Small Utilities are priced as an attractive impulse buy. They're only $10, which pocket change compared to most software, and they are designed to perform specific tasks no other software does. If you're a Windows 95 user, there's no reason *not* to have the Small Utilities available on your machine.

Personal note: I do not believe in crippleware, software that is intentionally released with features disabled in order to give it a deceptively attractive price, and which is full of nag screens reminding you that the version you have is crippleware and that you should pay more money to upgrade. I've had enough of these inflicted on me to know how annoying they are. If a company is going to sell a lower priced version of a package, the version should stand alone, and not just be an advertisement for the bigger more expensive version. The Small Utilities 2.0 is complete. If you use it, you should pay for it; if not, delete it. *It's all up to you!*

The legal information follows:

License Agreement

This agreement explains when and how you may use the shareware version of the Small Utilities 2.0. This is a legal agreement which allows you, the end user, to use the Small Utilities 2.0 under certain terms and conditions. If you cannot agree to abide by what this agreement says you should not evaluate the Small Utilities 2.0. The Small Utilities 2.0 is fully protected by copyright under U.S. law and international treaty provisions.

You may use a shareware copy of the Small Utilities 2.0 for an evaluation period of up to 30 days, in order to determine whether the program meets your needs before purchasing it. Once the evaluation period ends, you agree to either purchase the Small Utilities 2.0, or to stop using it.

You may make copies of your shareware copy of the Small Utilities 2.0 to give to others, as long as you include all of the files that you originally received with your shareware copy. When you give a shareware copy of the Small Utilities 2.0 to another person, you agree to inform them that their copy is to be used for a time limited evaluation period, and that they must purchase the Small Utilities 2.0 if they continue to use it once the evaluation period has ended.

The Small Utilities 2.0 offers powerful capabilities. If you do not follow our instructions, or if you use the Small Utilities 2.0 improperly, you can destroy files or cause other damage to your software and data. You assume full responsibility for the selection and use of the Small Utilities 2.0 to achieve your intended results.

DISCLAIMER: YOU ARE NOT PAYING FOR THE TRIAL USE LICENSE FOR YOUR SHAREWARE COPY OF THE PROGRAM. ACCORDINGLY THE SHAREWARE COPY IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND.

LIMITATION OF LIABILITY: IN NO EVENT SHALL WE BE LIABLE FOR ANY INCIDENTAL, CONSEQUENTIAL, OR PUNITIVE DAMAGES WHATSOEVER ARISING OUT OF USE OF THE PROGRAM OR YOUR RELATIONSHIP WITH ME. SOME STATES DO NOT ALLOW EXCLUSION OR LIMITATION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, THEREFORE THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Registration Form

Please enclose check or money order for $10 in US currency drawn on a US bank.

Send to:

Scott McMahan
30 Clairmont Ave A-8
Asheville, NC 28804

Name ____

Company ____

Address ____

____

E-Mail ____

Phone _____________________________ Fax _____________________________

Register products to: ___ Company ___ Individual

If you want to use the Small Utilities at your entire installation, want to preinstall it on PCs, or have other volume/bulk needs: contact me for site licenses or OEM pricing. Special arrangements can be made to give you the best deal possible.

Send information on: ___ Multi-System Licenses ___ Reseller/OEM Pricing

Installing and Uninstalling The Small Utilities

Information on what the Small Utilities are, what they do, and how to use them follows in the manual, but I have included installation information first since you are likely to want to install and run the Small Utilities as quickly as possible.

Installing

If you received the Small Utilities as a ZIP file, you must unpack the ZIP file. (Presumably, you already did that to get to this manual!) I can't give you any specific instructions on how to do this, since there are so many unzip programs available now. Consult your unzip program's documentation if you need help. Or, if you received the Small Utilities 2.0 on diskette, place the diskette in the drive.

Once you have the distribution unpacked or in the disk drive, go to the Start menu and choose:

Settings / Control Panel / Add/Remove Programs…

If the Small Utilities came on diskette, the Add/Remove Programs… wizard will find it automatically. If it is not on diskette, you will have to type the full pathname to the directory where you unpacked it (or alternatively use the Browse… button and root around until you find it). The program which installs the Small Utilities is called setup.exe.

The setup program asks you in which directory you want to install the Small Utilities. Recommended is the COMMAND directory. In most cases it will be an acceptable default. If you chose a different directory, you must add it to your PATH statement if it is not already there.

The Small Utilities need to be in a directory that is on your PATH in order to function properly.

When you're happy with the settings, click the big Perform Installation button. Once the install starts, the buttons are replaced by a play-by-play of what the install program is doing. Also, during installation, the readme.txt file is displayed using notepad so you can peruse it as the install grinds away.

Once the install procedure has completed, you are given the option to go ahead and run the toolbar.

A "program group" window should also appear with icons for all of the Small Utilities.

DO NOT BE ALARMED: If you are using a higher resolution display on your system than 640x480, you may see scrollbars around some of the Small Utilities. Just resize the program's window to comfortably display all of the window's contents (the scroll bars will disappear when the window is big enough to show all of the contents), and when you exit the program the window size will be saved for the future. You'll only see the scrollbars once.

Uninstalling

To blend in with the Windows 95 landscape, and to be friendly to your system, the Small Utilities 2.0 can uninstall itself. It adheres to the normal Windows 95 uninstallation procedures. To uninstall the Small Utilities 2.0, go to the Start menu, and chose:

Settings / Control Panel / Add/Remove Programs...

and look through the (alphabetized) list of applications. Select the Small Utilities, and click the Add/Remove... button. A DOS-like window will appear, and several messages will come up.

Some important notes about uninstalling:

- Even though it says Add/Remove... on the Windows 95 dialog box's button, you may only remove the Small Utilities 2.0. This is noted by a "(Remove only)" on the Small Utilities line.
- The uninstall script is created at install time, based on the parameters you give the install program. If any or all of the Small Utilities files are moved for any reason after they are installed, the install will not work.
    - The toolbar program, tools.exe, must be on your path to complete the uninstall. It has the routines which delete the Small Utilities information from the registry.

Welcome To The Small Utilities

Introduction

I don't think I'm far off the mark to say The Small Utilities 2.0 for Windows 95 has gone from being nice to have in the previous version to being an *essential* system tool no computer running Windows 95 should be without in this new version. This introduction tells you what these programs are and why you should have them, and the rest of the manual is your key to unlocking the power of the Small Utilities.

The Small Utilities 2.0 package addresses the need for small, quick tools targeted toward specific areas in the Windows 95 interface which need improvement. All of these utilities are written because they address a specific need in Windows no one has addressed – until now! You need the Small Utilities because they perform important functions which no other programs do.

These programs are designed with the philosophy that they do one well defined task well, instead of doing a whole lot of ill-conceived tasks slowly (which has been the overall trend in Windows software). Furthermore, they are designed to run on any system, not a just high-end machine.

Uses to which these programs can be put are *unlimited*. They're designed to be part of your toolkit of solutions, something at your fingertips to reach for when you need help handling a specific situation. They're intended to help you reach solutions quickly: one of the biggest problems with Windows (and DOS) is that you can know *what you want to do*, but not *how to do it*. With the Small Utilities, the idea is that you will not find as many "if only I could…" situations. The Windows 95 which comes out of the box, even with the popular add-ons, often leads you down these dead ends. You can see your eventual destination, where you want to go, but not how to get there from where you are.[1]

My approach to these utilities is empowerment: my programs are the building blocks which allow you to build something all your own. No two people approach tasks the same way, or take the same route to the eventual solution. These utilities give you the materials you need to take charge and do tasks on your computer more productively and simpler. They're proven to be useful and specific. I personally use these utilities day in and day out, and can't imagine what I did before I wrote them. I also can't imagine all the things you could do with these: I use Xalc all the time in my work. Home computer users will benefit from being able to create shortcuts to Shutdown and from using the GUI user-friendly LZX program instead of the command line. Network or large installation maintainers can mix and match the Small Utilities with macro languages and batch files to make writing solutions easier. Computer non-gurus can read this help file and try to understand some of Windows' idiosyncrasies a little better (or at least realize they are not crazy, that things really are illogical or confusing); even experienced users will enjoy having quicker ways to do things and more power at their disposal.

[1] Which is particularly ironic when you consider Microsoft's latest corporate slogan.

Tested and approved: Every last line of code and every detail of the Small Utilities is aimed at meeting some need that is not met by Windows 95 alone. Most of the things which made their way into the Small Utilities come from my own experiences with the Windows 95 interface: I use it myself every day and watch what other people using it have to say about its drawbacks. The Small Utilities work to meet an important need, and I personally use them daily. They're practical because they're created by someone who uses them to solve his own problems, have value because they make your work easier and faster by providing utilities no one else does, and there's no reason why every system running Windows 95 shouldn't have them installed.

What's *New* In Version 2.0!

Version 2.0 is a significant rewrite of the product. I've gone through every aspect of the Small Utilities in light of all the feedback from version 1.0 with an eye for making them better and more useful. If you've seen 1.0, you'll be blown away by 2.0. If you're new to the Small Utilities, you'll find the missing link in the Windows 95 computing environment that you may not have even known you were missing!

Note: This version of the Small Utilities comes with three command line programs. These are full 32-bit programs which require Windows 95 to run. You can not run them from the DOS command prompt when you Shutdown To MS-DOS or boot directly to real mode MS-DOS without Windows 95. I'm including this warning here in case you find these utilities so useful you decide to put them on a DOS boot disk for emergencies. You will not be able to use them until you can bring Windows 95 up.

Among other improvements:

- All programs are now fully integrated with Windows 95 and support the new common dialogs, long filenames, new interface conventions, and so on.
- All programs are true 32-bit native Windows 95 programs.
- All programs now remember their window position and size and automatically restore it on subsequent runs! No more scroll bars in high resolution.
- Note now has a toolbar.
- Shutdown supports all possible ways to shut the system down both in the dialog and on the command line.
    - Run can run folders.

Changes From 1.0 to 2.0

The biggest change is that I am no longer going to support Windows 3.x with the Small Utilities. Why is this the case?

- First of all, Windows 3.1 provides no way to run Windows programs from the DOS prompt, or an executing DOS application. (I admit that a lot of 3rd party utilities allow it, but these are all different and incompatible). This seriously limits what you can do with the Small Utilities.
- Second, I have not encountered but one person who was actually using version 1.0 on the Windows 3.x platform. There's no enough demand for it to be worth my while! If it's the choice between adding cool new features everyone will like and grinding out code for a platform no one wants to use, I'll go for the former.
    - And, I firmly believe that Windows 95 is the wave of the future, and you should be using it. I don't want to support yesterday's fading technology.

Also, I've given up on the Windows help file and done this manual instead. There's little difference between having this manual online in electronic format and having a help file. You can easily read and search both online. And authoring a serious help file is a learning curve chasm I don't want to jump into. I can make this Word file look a lot nicer with bullets, tables, graphics, and formatting using a familiar word processor interface instead of creating help files using the bizarre help syntax, with which anyone who saw 1.0's help file will know I wasn't too successful. I couldn't even get bullets to work in a help file, let alone graphics and tables. Therefore, I've chosen to ditch the help file, and make the documentation a Word file, which is just as accessible as a help file with the Word Viewer program.

Future Additions

The Small Utilities 2.0 are the tip of the iceberg. The next version will be as big a leap forward from 2.0 as 2.0 was from 1.0. As you can see from some of the additions to 2.0, such as the toolbar's taskbar tray icon, I've only scratched the surface of what is possible. I've already got a long list of features which I want to add. Time is the only enemy -- I'd rather have a 2.0 release out that people can be benefiting from now and work on the next release than make people wait another 6 months.

Version 2.5 can only come about with your registration support!

Philosophy

The Small Utilities address an important need for small, modular, easy-to-access components in Windows. The Small Utilities can be:

- run from icons
- run from shortcuts
- run from hotkeys
- started by the Windows 95 Plus Pack's System Agent
- run from the macro languages of Windows 3.x and Windows 95 programs
- started from the command line
- called from batch files
- started from DOS programs which can start other programs
    - called from within the macro languages of DOS programs which can start other programs

The trend over the past few years has been towards bigger and more, which has the inevitable corollary of slower. The Windows shrink-wrapped application marketplace is dominated by huge applications which do any conceivable thing. They consume vast amounts of disk space. They grind your system to a halt. This means mixing and matching parts of the programs is not simple.

In this shortcut happy age of links, macro languages, batch files that can run Windows programs, etc., Windows 95 is well designed for small, functional, minimal modules. No one seems to be writing them, though! Where are all the little utilities? Applications are too big and too slow -- WordPad takes about 10 seconds to load on my development machine at work, a powerhouse with 32MB of RAM and a Pentium processor. (I enjoy racing WordPad and Paint to see which is slower.) On a 486SX, it is so slow I can leave and come back while it is loading. Something is desperately wrong. When people are writing applications that barely run on high end machines, it's time for a reality check.

Microsoft has envisioned small, interchangeable components, each of which is completely compatible with anything else on the system. Spell checkers, drawing programs, utilities, file managers, file compression software, etc. Unfortunately, among other things, Microsoft's foundation for these building blocks is OLE, which is just too slow to be of practical use on most systems today. Even though Microsoft espouses modularity, it curiously has made very little efforts towards designing small, modular components itself. Even the Windows 95 shell is monolithic, although it incorporates a lot of OLE technology. Third party utilities like Norton Navigator follow in Microsoft's footsteps creating huge, slow applications.

There's a lot of small shareware utilities, and even some freeware ones, which do some of the things the Small Utilities do, but the Small Utilities are differentiated from the rest because: *They work with all aspects of your system.* In general, commercial, shareware, and freeware utilities don't make a lot of effort to be compatible with older programs. They also offer limited functionality. A good example of this is shutdown programs: while shutdown programs are a dime a dozen, I haven't seen any which run from an icon, which run from the command prompt, which give you every possible way to shut down the system, and which come with detailed instructions on how to create a shutdown group.

Don't get me wrong: the Small Utilities do not replace certain big, monolithic applications. I personally use Norton Navigator and the Norton Utilities for Windows 95 all the time. Some software by its nature has to be big and expensive, other software doesn't. All different sizes and shapes of software have their place in the grander scheme of things.

Documentation Philosophy

The recent trend toward *no* documentation is alarming. Microsoft is probably the most visible perpetrator, since they were under a microscope with Windows 95's rollout, but they are not the only ones. You open the box and a CD falls out and there’s no manual, just a slim feel-good book (if you're lucky). Wizards, experts, coaches, and whatever are great as long as you want to do something they can do, but as soon as you exceed their narrow boundaries, you're on your own. The online help is often good, but unconnected reference information rarely gives you an insight into *how* to do something, only that it could be done.

With the success computer book publishers are enjoying not letting up in the least over the past few years (they're even writing books about products *before the products ship* now!), I have trouble believing that people do not want in-depth documentation. It’s not just page after page of dry parameters, usage syntax, and other stuff, but all of the background information and insight into how to use the product you hope the person or people who wrote the thing in the first place would have.

What's the old expression? Figures lie, and liars figure. Surveys, statistics, and numbers can be made to support any position. The problem is, companies are listening to too many of these people who think the documentation is "too hard" -- these are the same people who, if you sat them down and showed them how to do something on the computer and then immediately asked them to do it on their own, they couldn't. Of course they think the documentation is too hard, since they'd think *anything* is too hard. Companies are making huge, blanket assumptions about all users. But when the people who have to come in cold and solve problems (read: *clean up messes*) other people have with their computers, and I bet you fall into this category if you're reading this, concrete and comprehensive documentation is a requirement.

This is a general symptom of computing for the masses: software products need two levels of documentation. As long as users with no technical background will be using the product, it is important to provide "hand holding" documentation and programs whose functions are very obvious and discoverable, but on the other hand if the product is going to be used by technically sophisticated users, a deeper level of documentation will have to be provided for them to fully use the software.

Not only do I want to document my programs, I want to give you that little bit extra that makes the documentation worth reading. Ideas for how to use the programs effectively and in ways that may have never occurred to you, historical and background information about the reasons why things are the way they are, and even design tradeoffs I had to face. Documentation should be educational and interesting, not just a dry listing of a program's components and parameters.

Technical Support

Registered users get unlimited free e-mail tech support. You can send mail to softbase@mercury.interpath.com or scott@morphnet.com. If you have a reasonable[2] problem using the Small Utilities or integrating them with other programs, I'll be more than happy to help you out.

User feedback is also greatly appreciated. Many of the new features in 2.0 were suggested by users.

I'm also very interested in any bug reports or compatibility problems. I can only test the Small Utilities up to a certain point with the hardware and software I have available to me, and there's billions and billions of programs and configurations my software can be thrown into. If you notice a conflict or bizarre behavior, please let me know.

[2] This weasel word is provided in case anyone tries to take advantage of this policy. I'll be glad to help you up to a point, within reason. I'm not going to design your online transaction database, for example, so you can shut it down with the Shutdown program.

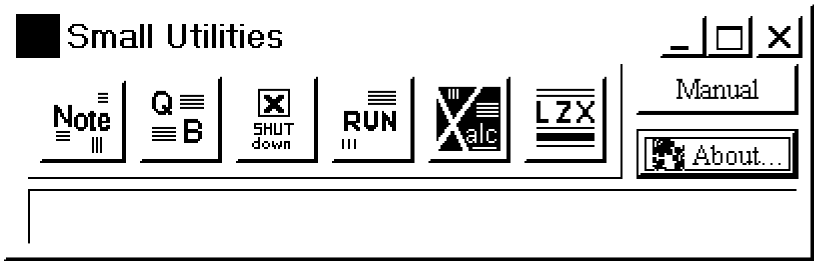
Troubleshooting and Technical Notes

The Small Utilities are small and easy to use, and rarely present problems. Yet something as complex as Windows 95 with all of the different third party programs available which can mix and mingle on the system naturally provides an opportunity for difficulties. This section lists all the oddities I (and the beta testers) have discovered.

- If you have another program with the same name as any of the Small Utilities, you could have problems if you run one of them first and then try to run a Small Utilities program. This is a Windows 95 defect and is not specific to the Small Utilities themselves. This situation occurred when a Small Utilities user found a program called "run" (which had nothing to do with the run included with the Small Utilities) on a Windows 95 book's companion CD-ROM. This run was a toolbar program launcher. If he started it, left it running, and then tried to use "run.exe" from the Small Utilities, the other run would just pop up and become the active window. As it turns out, it is impossible to simultaneously start two programs with the same EXE name under Windows 95[3]. It doesn't even matter if they're in different directories. The solution is to just rename one or the other's EXE to some other name.
- The Small Utilities make extensive use of the Registry to store various values. If anything happens to your Registry and the Small Utilities' entries get erased for some reason, the next time you start any of the Small Utilities they will complain "RegQueryValue FAILED" at least four times. This is a harmless error, since the programs will recreate the missing values when you close them again.
- The Norton Utilities have a bug in them that can come into play when they are started using the Small Utilities' Run module. Unless the Norton programs are started with the current directory set to their directory, they can't seem to find their DLLs. I consider this a bug in the other programs, not in my Run program, considering how integrated with the Windows registry the Norton programs are. They could easily check to see where the DLLs are installed.
  - Several other programs seem to suffer from this defect: the pinball program which comes with the Plus Pack is also one. I am looking into this issue and hope to have it resolved in version 2.5 with a compatibility kludge to work around the problem.

[3] To make this more complex, the bug is applicable only to certain cases: you can't start two 16-bit programs by the same name, but you can start one 32-bit and one 16-bit program with the same name. I'm exploring this issue further.

The Small Utilities' Control Panel



The Small Utilities' toolbar is a control panel that allows you to easily access all of the Small Utilities (with the exception of the bonus command line utilities). It is designed to let you have a one-stop location for all the different programs included in the Small Utilities set, so you can explore what is available in the package. As you will see all through this documentation, the toolbar is only one of many ways to run programs. The toolbar is like an interactive guide to the programs. Once you know what the Small Utilities are and what they do, you can come up with your own creative ways to use them.

The toolbar is designed to look like the kind of toolbars you see in Windows 95 and other applications. It, in fact, demonstrates just about everything a toolbar can do: tool tips, a status bar, a right-click menu (try it! right click somewhere that isn't a button on the raised area), buttons, etc.

The Manual button will open the manual (i.e. this document) with whatever program is set up to open .DOC files on your system.

The Manual button on the Small Utilities' toolbar illustrates a clever usage of the start command to achieve program and system independence. No matter what program is associated with .DOC on any machine (Word, Word Viewer, Word Pad, etc.), the start command -- when given the filename instead of a program name -- will automatically open the correct program associated with the file extension. The program which started the document did not have to know what program a particular system had on it for viewing/editing that file type. This technique is very powerful and has a place in any Windows 95 user's bag of tricks.

The Toolbar's Tray Icon

Along with the features of the toolbar, it also gives you a tray icon on the Windows 95 taskbar. Clicking this icon, when the toolbar is minimized, will bring the toolbar back to the foreground.

To effectively use the tray icon, I suggest starting the toolbar from your Startup group and running it as a minimized application (which you can do by going to your startup group in Explorer and pulling up the properties for the shortcut to the toolbar which you placed there). Then when you need it, you can click the tray icon, and when you are done, you can re-minimize the application.

In version 2.5, the tray icon will be much more powerful and give you more control over the utilities. It will also appear in each individual utility, with the option of using it or turning it off.

Yet another reason to register!

Note

The Note program is designed to give you a way to write notes and jot things down quickly. It also provides a quick place to put small amounts of text from the clipboard. Windows does not really have a good notepad utility.

I based Note on a real notepad, the kind I keep on my desk for writing notes. My notepad is just a blank page. It doesn't have a File menu or a Save As icon. When I write something on it, what I write stays there until I erase it by tearing off a page and throwing it away.

What Note is not:

- A text editor -- the poorly named "notepad" that comes with Windows 95 tries to be (and fails miserably) a full text editor, which is not what a notepad is at all.
- An MDI/multiple file document program -- Note is just a place to make notes. It is the equivalent of a scratch pad, not a notebook with dividers or a hanging file.
    - A word processor -- when you add support for multiple fonts and things, you sacrifice speed. Note uses one font.

What Note is:

- Designed to be used from a shortcut icon or hotkey
- Fully persistent so you can close it and reopen it to the same place you left off -- you don't need to minimize it. Just get rid of it until you need it again. This reduces clutter.
    - Quick -- you don't explicitly have to save anything, it is saved automatically on exit and loaded automatically on run. This speeds things up.

Using Note And The Clipboard

Note is designed to facilitate cutting and pasting fragments of text to and from applications quickly. Often I encounter situations where I need to cut and paste multiple things from, for example, a help file into some other program. Or from one program to another, or one document in a non-MDI program to another document in the same program. The problem is it's not possible (usually) to copy discontiguous items in a document to the clipboard, so a temporary holding area is needed to copy and paste one thing, then copy and paste something else. Note can be that temporary scratchpad.

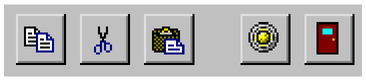
Remember you must highlight something in Note in order to copy or cut it.

More Notes About Note

Whatever you type will be saved in a file called note0000.dat in your Windows directory. This file is plain text, so it is possible to manipulate and process it using some other program. (Obviously, I can't recommend doing that, but I can't stop you either!)

Note's features can be accessed through the toolbar or the right-click menu.

The Note Toolbar



From right to left on the toolbar, the icons are Copy, Cut, Paste, About..., and Exit. The first three are the official Microsoft icons. The others are made up icons. All of them have tooltips in case you forget what they do.

Run

Unfortunately, the Windows 95 shell has the Run... command and its dialog box permanently built into it. It is not a separate program, which means you can't:

- create a shortcut to it on the desktop or in any folder
- attach a hotkey to it
- use it in batch files
    - call it from macros

And it also forces you to use Explorer as your shell to access it.

The Small Utilities' Run command is equivalent to the Start/Run... command, but it is a standalone program which may be started in any situation where you can start a regular program, including all the bulleted items above. See the section on Using the Small Utilities In Other Programs later on in this manual for ideas.

Using Run

Run is very streamlined and easy to use. Simply type a command and click the Run button (or just hit enter). If you would like to look for a particular command, or prefer to point and click, use the Browse... button to bring up a standard open common dialog box. (Since it is a common dialog box, you get full access to any third party utilities like Norton FileAssist.)

If you drag and drop a file from Explorer to the Run window, the filename will appear in the command line. This mimics the behavior of Start/Run....

Running A Folder

Run will run a folder if you type the folder name. You have to type the full pathname, since the dialog box in Browse… can't show folders as well as narrow the search down to executables at the same time.

The window which appears will be the same as what you would get if you opened Explorer, went to the folder, right clicked, and chose Explore From Here. This differs slightly from the behavior of the Windows 95 Start/Run… dialog, which opens a My Computer view of the folder you type. (I must thank George Steffanos for bringing this whole issue to my attention. I had never run a folder from the Start/Run… dialog, and did not know this feature even existed. The deviation from the Start/Run… behavior is purely my own programmer's license. I never use the My Computer view, seldom remember it exists, and see no practical application for it. However, I could make it an option in 2.5. If you'd like to see a toggle between Explore From Here and My Computer in the Run A Folder feature, slip in a note with your registration!)

Run's Command Line and Explorer

If Run is started with a command line parameter, that parameter is put in the command line of Run's main dialog. The main use for this is to facilitate the command line argument feature I present next, although I am certain that other uses for it will come up. (To be utterly ridiculous, open a DOS prompt and give the command "start run explorer" and see what I mean.)

The problem (or one of them, anyway) with Explorer is that it gives you absolutely no way to run a program with command line arguments. The old File Manager had a poorly documented feature which allowed you to select an item, do File/Run..., and have the name of the item you selected appear in the dialog box. You can then hit the End key, and type whatever command line arguments you like. (In fact, this feature may be completely undocumented anywhere. I discovered it almost by accident, and searched long and hard through official and third party Windows books and magazines for some hint that it existed -- with no luck -- before submitting it as a tip to Microsoft Magazine. My tip appeared in the 50 Fabulous Tips issue, so it must have been new to them as well!) In Explorer, you don't even have that much. With no File/Run... command, your only alternative is to use Start/Run... and then drag and drop the item in Explorer into the dialog box. This is a convoluted process that seems more trouble than it's worth.

The Small Utilities' installation program sets up a right-click menu item for some things (see the caveat in the next paragraph) in Explorer so that when you right click them, a Run... menu item is added to the right-click menu. If you click on Run..., you will see the familiar Run... dialog box with the thing you highlighted in the command line. This has three benefits which will endear it to you:

- First, you can give command line arguments to a program. If you right-click on LZX in Explorer, choose Run...; then when the Run dialog comes up you can hit End and the spacebar and type an argument. Notice what appears in the command line edit control is highlighted by default, so you have to hit End or some other cursor movement key to clear the highlighting. If you just start typing, it will overwrite what's there. This technique of supplying command line arguments to programs is very useful when dealing with batch files, DOS programs, and other things you would ordinarily have to run from a DOS box.
- Second, you can give a program to a command line argument. That is, if you have some unassociated file extension like .ME (as in READ.ME), you can open the file with Run..., hit the Home key, and type some program name like notepad.exe and hit the spacebar. The file name you selected becomes the argument to the application.
- Third, you can override the default program associated with an item. If a text file is associated with notepad.exe, you can Run... a .TXT file, and give it some other program name like your hex editor. This is particularly useful with files which naturally have more than one program associated with them, such as batch files or perl programs which you want to edit as often as you want to run. This way, you do not have to choose which association you want to go with the file, you can have it both ways.

A neat side effect of installing Run as a right-click menu item in Explorer is that it becomes the default menu choice for unassociated items. You can use the second option in the list above to supply a program to the unassociated file in order to run it as an argument to some application.

In this version, only legitimate objects have the Run... right-click menu option. In future versions of the Small Utilities, I will broaden this feature (and give you a way to turn it on and off). This sin of omission is not intended as a registration incentive -- that's just a happy side effect -- but a triage between throwing every neat feature I can dream up into the 2.0 release versus releasing what I have and letting you use a useful product in the meantime until I can add more features. Obviously, future versions depend on support.

Compatibility note: The Run... menu item does not show up in the Norton File Manager. This problem is not peculiar to Run, because many other things do not show up either (such as the Install option for INF files). I don't think this is a bug so much as an oversight, and hopefully it will be corrected in a future release of the Norton File Manager.

Run History and the Default Items

When you install the Small Utilities, a default Run history file is created with the three most useful programs which are not installed by default on the Start/Programs menu by the Windows install procedure. These are the three things I use Run to start the most often.

- Regedit -- a program which allows you to view or modify the registry. Modifying the registry is a very dangerous thing if you do not know what you are doing, so don't use this program without a good reason.
- Charmap -- an extremely useful program which allows you to copy any character in any font, whether it appears on the keyboard or not, onto the clipboard so you can insert them into a program.
- Sysedit -- allows you to view and edit all of the main system configuration files at once. Like Regedit, this can be dangerous if you don't know what you are doing.

If you didn't know these existed, don't feel bad. It isn't like they're documented anywhere.

It is entirely possible that one or more of these programs was not installed on your system when Windows 95 was installed. The defaults for Windows 95 installations can be very strange. If the program can't be found, try to use the Add/Remove Programs dialog box's Windows page to install the missing programs.

Shutdown

Something as important as shutting down the machine is rather halfheartedly implemented in Windows 95. That's unfortunate, since an orderly shutdown in an advanced operating system like Windows 95 is critical for maintaining the integrity of the file system.

Deficiencies in the shutdown facility provided include:

- Shutdown... is hard-wired into the Start menu. It is not a separate program. This has the same problems the Start/Run... command did in the last section: It can't be called from batch files or macro languages, and no shortcuts or hotkeys can be assigned to it.
- Not all shutdown options are available. Some options are available but bizarrely implemented and undocumented. (There's some kind of shift-click business you can do in the Start/Shutdown… dialog, but I'm not sure how it works. It isn't "discoverable".)
- It is impossible to shut down the computer from a batch file.
- It is impossible to create an automated shutdown procedure.
    - Start/Shutdown... is illogical. To go to "Start" in order to stop the machine doesn't make sense. (One could be reminded of Robert Fripp’s “When I say stop, continue”.) Although it makes sense in the warped way that all system functions are accessible from the Start menu, it doesn't make sense in normal English syntax and logic.

The Small Utilities' Shutdown program is designed to remedy all of the problems in the shutdown facilities of Windows 95.

Basic usage for Shutdown is simple: start the program and pick what you want to do from the dialog box that comes up. Note that there are many more options than in Start/Shutdown….

Warning: Any option followed by '(!)' will shut down or reboot the system unconditionally. Any running programs will be stopped regardless of the state of any documents or files you have open. If data is not saved, it will be lost forever. Use these options with extreme care.

Command Line Parameters

All of shutdown's choices are available from the command line as well as the main dialog. If no parameters are given on the command line, shutdown displays its dialog normally.

Usage:

shutdown [*how*]

where *how* is either a number or a word from this table:

Number	Word	Function selected
1	exit	Exit Windows
2	restart	Restart Windows
3	msdos	Exit to MS-DOS
4	reboot	Reboot the computer
5	logoff	Log off and back on as a different user
6	now	Unconditional shutdown*
7	uboot	Unconditional reboot*

* ***Warning***: These parameters cause an unconditional shutdown regardless of the state of the applications running. If any unsaved data exists in any running applications, it will be destroyed as the system shuts down or reboots. Use these parameters only if you're really sure you know what you're doing.

Note: Command line parameters are not just useful in batch files! When you create a shortcut to the Shutdown program, you may edit the shortcut's properties to give a command line parameter. This is useful if you want shortcuts to shut down or restart Windows from a double-click or a hotkey. You don't have to go through a dialog box.

Warning: Windows 95 is downright flaky when it comes to shutting down or rebooting. About 30% of the time, for no apparent reason, the shutdown or reboot will fail. *This is not peculiar to the Small Utilities' Shutdown program. Instead it is a compatibility or programming problem with the Windows 95 operating system itself.*

You cannot reliably shut down or reboot the machine from software only.

The only 100% reliable way to do it is to have an operator physically present at the machine in case something goes wrong. Usually, what happens is the "Shutting down Windows 95" screen comes up, and then a period of disk activity follows. Afterwards, the machine just hangs and never gives you the "It is now safe to turn off your computer" screen, or doesn't reboot. It is generally safe at this point to either power off or hit the reset button: it appears to do no harm to abort the orderly shutdown after it hangs like this. I've seen this behavior happen on every machine on which I've run Windows 95. I believe it is some type of problem in the interaction between the Windows 95 operating system, the BIOS and hardware of the machine, and the graphics card. I know of no solution to the problem other than operator intervention during the shutdown process. Again, this is not a bug in the Small Utilities itself, but a feature of the Windows 95 operating system.

The Shutdown Procedure

Although Windows 3.0 pioneered the feature which allowed you to run programs when Windows started, Windows has never had in its ten year history a way to automatically run programs when Windows shut down. Until now!

Most other operating systems provide a way to run programs at shutdown time, and with a multi-tasking operating system it is often desirable to do so. There are many things you can do in a shutdown procedure to make running your computer easier, such as automatically backing up files and automatically cleaning out old files.

With the Small Utilities it is simple to create a shutdown procedure: Write a batch file which contains all of the commands you want to run at system shutdown time. Then add the Small Utilities' Shutdown command to the end of the batch file. You may use any of the parameters to shutdown, such as "exit" to halt Windows normally, or "now" to do an unconditional shutdown.

You can, of course, create a shortcut to this batch file and use the shortcut to shut down the system.

Here's an example batch file:

```
@echo off
xcopy c:\work\*.* c:\backups
del c:\tmp\*.tmp
start shutdown now
```

Note: There's really no good way to have a Shutdown *group* on the Start/Programs menu the way the Startup group appears. Since group items are in no particular order, it would be almost impossible to make sure that the shutdown command was the last one run, and also tasks generally have to be done in a specific order. A batch file works much better in practice.

Quick Browse

The Quick Browse utility is a directory tree exploration tool that allows you to wander around the directory tree and open files with certain programs. It is designed to be a no-frills, small, fast program you can stick on your emergency disk or something.

The main screen has two panes. The left is a directory/drive pane like in the old Windows 3.1 common Open and SaveAs dialogs. The right is a list of files in the directory you've opened. (Folders do not show up on the right. You must open folders on the left to see sub-folders.) Once you have highlighted a file, you can push the Open Selected button.

The dialog shows you the file you selected with its full path (this is an editable text box if you want to change it), and a radio button with two choices: either use the default application registered to this file extension, or supply your own program in the edit box. (A Browse... dialog is provided for your convenience. You can browse your Windows Send To directory if you like.)

A really bizarre misfeature in this quick browser is when you have not selected a file and a directory is selected. If you do Open Selected and then Browse..., it will show you a list of all the programs in the current directory, and you can run one of them. The folder will be a command line argument, but most apps will ignore it.

Xalc

Like Homer Simpson designed the perfect car based on the perceived flaws of all the other automobiles he had driven, I designed a hex calculator that did everything the other calculators didn't do. Xalc is the calculator that corrects all these deficiencies I have found.

Windows 3.x's calculator is known for giving wrong answers. It doesn't cut it with me, since my main use for a calculator is to debug programs. A wrong answer or even the threat of one could set me back hours or days. Supposedly, the Windows 95 calculator is an all-new 32-bit program with the bugs fixed, but I have already seen mentions of bugs in the Windows 95 calculator in Microsoft's Knowledge Base. Xalc seems like it will stay a while. None of the other calculators for Windows that I've used will actually do arithmetic in another base. Norton Desktop's, NST Calc, Judy's 10 Key Pro, etc. do not do the kind of hexadecimal calculating capability computer users need.

Most Windows calculators make themselves look like handheld ones, which is pretty absurd since the design goal of a handheld calculator is to do as much as possible in as few chips and parts as possible. Surely a program running under a GUI with virtual memory to play with can do better. Windows calculators generally, though, have a single line for the display, which is bad. You can't tell if you've screwed up operand #1 if you've already typed in the operator and begun the second operand. I want visual feedback. Even my hand-held TI calculator will show me the entire expression I've typed up until I press enter. If a hand-held will, why can't a Windows-based one? Another really insane design choice in Windows is a 2nd Fn key! Why is this added bit of confusion and nonsense necessary when you can extend the window out a little and have one function per button? The idea is to minimize the chance of the user screwing up.

Xalc is meant as a programmer's calculator, and does not do fractions. It only does whole numbers, the kind you generally use calculating hex. Currently, things like transcendental functions are not supported. Log and x^y aren't supported in this version, but exponents may be supported later. Octal (base 8) is not supported.

Pronunciation: As to how "Xalc" is pronounced, I say "eks-alk", but it could be pronounced "zee-alk" as well. The name comes from "hexadecimal calculator", which someone I work with shortened to "hexalator" and I shortened all the way to "xalc".

Xalc features an Easter egg screen. Finding it is up to you!

Using Xalc

The Xalc window is divided into three regions.

1. The topmost pad: the top area has four display fields (operand 1, operator, operand 2), the quit and help program control buttons, and the mode indicator.
2. The keypad: Designed for hex or decimal (abbreviated as dec) input, the hex keys A-F are unclickable while in dec mode.
3. A control pad: This pad has the arithmetic operators, the mode switch, and an enter button.

Basic usage: Xalc calculates arithmetic expressions in the format

answer = operand1 operator operand2

where operand1 and operand2 are valid dec or hex numbers depending on the mode Xalc is in ("valid" is defined as anything Delphi's built-in integer type can swallow), and op is one of the supported operators (addition, subtraction, multiplication, integer division, and modulus).

At the top you see four display boxes. These are in order

```
operand1
operator
operand2
  answer
```

You input the operands and operators using the buttons provided, or the keyboard. (Xalc is not case sensitive.) 0-9, A-Z, and even a-z are recognized, all five operators (+,-,*,/, and %), and the Enter key does the same as the Enter button. Change mode is X (as in heX), and clear is Z because C is a hex number and can't be used. The Enter button or Enter on the keyboard will complete the calculation once you have entered everything.

Once Xalc has computed and displayed the answer for you, both operands and the operator remain visible on the screen until you type another number or an operator. This is so that you can check to make sure everything was entered properly. Once you type something, everything on the screen goes away and you can begin a new calculation.

Hints/tips: yellow popup tips appear to be the sin qua non for acceptable Windows programs. If you need tips, check the check box labeled Show Tips. Then, if you hesitate over anything that has a tip associated with it, the yellow tip/hint will pop up. This is off by default since tips are annoying if you know what you are doing.

Clipboard: The edit fields are highlightable and you can copy to the clipboard. You must use the Ctrl-Insert key combination, though, until I write an Edit menu.

Repeat calculating: If you want to use the answer as the first operand to a new expression, simply type the operator (omit typing the first operand) and the answer will become the first operand of the new expression. Otherwise, if you type a number, it becomes the first operand of the expression.

Modes: The mode switch controls whether or not you are calculating in hex or in decimal. If you happen to have entered any numbers when the mode switch is flipped, they immediately change to the new base. During dec mode, you can't enter hex digits. Xalc's behavior is optimized for quick conversions: just type a number in and switch modes. Besides clicking on the mode button, you may also double-click the mode indicator in the top panel to toggle the mode.

The Clear button erases everything and lets you start over. This is currently the only way to recover if you mistype or misclick, no editing is possible.

Menus: A File/Exit is provided for people who have wired Alt-F, X into their brains and can't handle a program without it. The Help menu pulls up the same dialogs the Help button does, only a little more differently.

Errors and weird situations: It is possible for you to get Xalc confused if you give it strange input. You can usually say OK to whatever dialog that comes up, hit Clear, and be okay. If it is really hosed, it may crash, forcing you to actually restart the program.

LZX

LZX fills a hole in the Windows applets collection: there has never been any way for Windows users to uncompress a file using a graphical user interface. Even though Windows has been using the same standard file compression scheme since 3.1! If you wanted to uncompress a file, you had to utter a command line incantation at the DOS prompt instead of choosing a file from a common dialog. This is still the case in the completely revamped Windows 95 user interface. That's progress.

When you use LZX, it is easy to uncompress single files from a compressed distribution. Often, you have to do this if a file becomes damaged or corrupt.

Compressed files which were compressed with the standard Windows compression normally have the last character in the extension replaced by an underscore. (E.g.: WRITE.EXE becomes WRITE.EX_) Not always, but this has been the convention used by Microsoft through the years and adopted by most other software which uses the Windows built in compression scheme.

Usage

Normally, you run the LZX program and fill in the dialog box it provides. You can graphically uncompress files by pointing and clicking from the standard File Open dialog. There are two edit controls on the dialog: the top one is the source, i.e. the compressed file; the bottom one is the new file you wish to create, i.e. the new uncompressed file.

Tip: To fill in the second edit field, which will generally look a lot like the first, use cut & paste: highlight what is in the top dialog box, copy it (with Control-Insert), and then tab to the second dialog box and paste it (with Shift-Insert). Then edit the new copy. Mastery of cut & paste like this will save you many hours of typing in LZX and all dialog boxes.

If no command line parameters are given to LZX, then LZX starts up like a normal Windows program and gives you full graphical control over everything.

If one command line parameter is present, LZX opens its dialog and puts the filename in the Compressed file edit box.

If two command line parameters are present, LZX operates in "batch" mode. The first argument is the compressed file to use as a source, the second argument is the new uncompressed file to create. This mode could be useful in batch files, although it overlaps with expand.exe's functionality.

LZX And The Send To Menu

LZX is particularly useful when combined with the Explorer right-click menu's Send To option. The reason for this is simple: a file compressed with the built in Windows compression scheme has no standard file extension. The extension is formed by taking the regular file extension and replacing the last character with an underscore. You can't associate a Windows compressed file with LZX the way you can associate a .WK1 file with Lotus 1-2-3. (Well, you could, but it would require associating all file extensions AA_ through ZZ_, AA_ through ZZ_, and all other non-alphabetic characters and numbers with LZX....! Your registry would probably explode.)

Any file can be sent to any of the Send To menu's menu items. When a file is sent to LZX, the normal dialog appears, with the selected filename in the top edit control[4]. All you have to do is type the new filename in the other edit control.

By default, LZX is added to the Send To menu automatically during the Small Utilities install procedure. If you do not want it there for some reason, use Explorer and go to the Send To directory in your Windows directory. Then delete the LZX shortcut you find there.

[4] I've noticed that short filenames get sent. This is a feature of the operating system for which I have no explanation.

Using the Small Utilities In Other Programs

The Small Utilities allow you to customize your computer to work the way you want it to, and to make things as smoothly and automatically as possible. The mix-and-match nature of the programs means that they can be plugged in where ever they can be useful, and some of those places are surprising. This section is going to walk you through using the Small Utilities with several different types of Windows programs. Our example is going to be Shutdown, but you could use any of the other Small Utilities similarly.

Small Utilities and System Agent

The Microsoft Windows 95 Plus Pack contains a scheduler called the System Agent. The Small Utilities, like any program, can be run from System Agent, both standalone and as part of complex batch files.

For example, it is possible to set up the Shutdown program to automatically and unconditionally shut down your system at a certain time by using the "now" parameter to Shutdown and calling it from System Agent.[5]

Small Utilities and VBA

Microsoft applications have Visual Basic for Applications (VBA) as their macro language. At least most of them do, or ought to eventually if you're lucky. If your Microsoft application does not yet have VBA, you will have to dig around for equivalents to the macro presented here. Pre-VBA versions of the various BASIC-like macro languages in Microsoft applications have differences (for example, they don't have the "Shell" function discussed below!).

Create a macro similar to this:

```
'
' Shutdown Macro
' Macro recorded 1/17/96 by Scott McMahan
'
'
Sub Shutdown()
   Shell "shutdown.exe", 1
End Sub
```

You can then create a menu selection and have it run this macro, or assign it to an icon in the toolbar. Different applications have different ways of accomplishing these tasks.

The Small Utilities and DOS

The conventional wisdom that we are bombarded with is that you can only use the new operating system fully if you upgrade to new applications written specifically for it. The logic here is driven by editorial content in magazines which derive their continued existence from not only the advertising revenue of the people writing the new applications, but also from the fact that giving

[5] Too bad you can't automatically start the machine, but the machine would be off and you couldn't run any programs.

rubber-stamp glowing reviews to new applications, and trumpeting new "features" (which may or may not be useful) is an effective way to generate editorial content. This whole school of thought ignores the fact that Windows 95 is particularly optimized to run old applications better than Windows or DOS! If older applications are adequate, no convincing reason exists to change them.[6]

On the other hand, there are convincing reasons to upgrade the operating system. One reason you may be using Windows 95 is for its advanced memory management for EMS and DPMI applications. Windows 95 is a tremendous improvement over DOS for DOS programs which require DOS extenders and EMS, and is worth considering for that reason even if you do not have the need for Windows applications. It is not unreasonable for a user committed to DOS to run programs which use DOS extenders, since even DOS applications have outgrown the old DOS limits, and it would not be out of the realm of possibility for such a user to have a system which meets the requirements for Windows 95. The switch to Windows 95 allows greatly advanced virtual memory and caching and other benefits.

As an example of a surprising way to make the Small Utilities useful, it is possible to run shutdown.exe from within 1-2-3 for DOS. It is not possible to shut down Windows from a DOS application otherwise, since the DOS application cannot call the Windows API. DOS users unfamiliar with Windows 95 may want an easy way to shut down Windows, such as Alt-F3 and choosing "shutdown" from a list. (Remember that Windows 95 can't simply be turned off without an orderly shutdown!)

A 1-2-3 for DOS macro might look like:

```
{system "c:\windows\command /c start shutdown.exe"}
```

You can then assign this to a macro shortcut key or to a named macro.

The macro command launches the new Windows 95 command interpreter (with a “/c” which means to just run the command on the command line and not act like an interactive shell) to start a Windows program, in this case shutdown.exe.

Some caveats: 1-2-3 must be shut down orderly. This macro instruction could be one step in a more complex macro that saves all open files and closes 1-2-3 itself. You may also have to adjust 1-2-3's properties so that Windows can close it automatically without having to prompt the user to close it. If your shutdown routine is smart enough to ensure all work is saved, then it could use the "now" parameter to shutdown.exe and not even prompt the user.

Although 1-2-3 is used as the example, any DOS application which allows you to run DOS programs from inside the program itself could use this technique.

Small Utilities and Lotus Applications

Lotus applications have SmartIcons and customizable menus. They are very flexible for the most part in allowing you to define your own automated procedures. The Small Utilities fit right into this extensibility. You can run the Small Utilities from either SmartIcons or menus, or as part of complex macro programs. The macro language for Lotus applications varies depending on the

[6] This being the opinion of someone with a disk crammed full of old DOS programs. Your mileage may vary.

program and the release. It is impossible to show a single macro for all applications. Consult the online help files for the application you are using, or the manuals, for details on specific macro language usage.

Example: Here is an Ami Pro macro to launch shutdown.exe:

```
FUNCTION shutdown()
        Exec("Shutdown.exe", "", 1);
END FUNCTION
```

Save this as a .SMM file.

You can attach this to a SmartIcon, or create a new menu selection for it. How to accomplish these two tasks depends largely on the application involved. The next revision of this documentation should have an Ami Pro macro that attaches a new menu item to a menu which will allow you to run a program. Until then, the macro documentation has a good sample of cut-and-paste code you can steal and then modify. It is not very difficult.

Lotus 1-2-3 for Windows is curious: I have release 4, which in my research does not have any way to start a Windows program from within the macro language. (Except for using the DOS-based {SYSTEM} command as above and the START command in the Windows 95 shell.) I looked at a book for R5 and it has a “launch” command, which is not present in R4. The “launch” command seems to be similar to the “shell” command in excel or the “exec” command in Ami Pro. Note that all of these are very thin wrappers to the WinExec() Windows API function. I do not understand why they all have to have different names.

Other Programs

If the programs you use are not discussed, the basic idea is almost certainly going to be the same. Most Windows programs complex enough to have a macro programming language support some way to start a Windows program. The trick is digging around and finding it, but if you do a lot of macro programming in a particular application you should be familiar enough with it that it won’t be a problem. WordStar 2 for Windows, as an example, lets you make the WinExec() API call directly from its StarBASIC language.

Obviously, you can call any of the Small Utilities from C/C++, Delphi, Visual Basic, etc. the same way you would any other program, by making a WinExec() call. Read the Windows API reference that comes with these development tools for more information on WinExec().

Bonus Utility: Hexdump

As an added bonus, the program hexdump.exe is included with the Small Utilities. Unfortunately, the base Windows 95 product does not come with a hexdump command the way UNIX versions do. Even if you are not a programmer, a hexdump program can be very useful because it will show you the printable contents of a normally unviewable file.

Caveat: This command line program is a full 32-bit utility and will only run under Windows 95, not in any of the DOS compatibility modes.

Usage And Sample Output

Usage:

```
hexdump <filename>
```

Information is sent to the screen by default. You need to use I/O redirection or the more command to capture and/or pause output.

Sample Output:

```
   (A)      (B)                                                    (C)
00000000  02 10 01 08 05 12 40 00  00 00 00 00 00 00 00 00  ......@.........
00000010  00 00 00 00 00 00 00 00  00 00 17 33 00 00 00 80  ...........3....
00000020  00 00 01 3C 00 00 7E 00  40 00 12 E8 00 00 01 C0  ...<..~.@.......
00000030  00 00 00 03 00 00 02 2C  00 00 00 19 00 00 06 3C  .......,.......<
00000040  00 00 00 00 00 00 06 3C  00 00 02 34 00 00 06 14  .......<...4....
00000050  00 00 00 02 00 00 20 78  00 00 00 06 00 00 08 74  ...... x.......t
```

(A) The Offset

The number on the left is the offset into the file. The offset is in hex, since each line is 16 bytes long.

(B) The Hex

On each line, 16 bytes are printed.

The 16 two-character hex values are divided down the middle with two spaces between the 8th and 9th bytes to make finding your place easier.

(C) The ASCII

In addition to the raw hex values, the ASCII value of the byte is printed in case you want to look for printable strings. You can find all sorts of interesting strings in binary files of all types. (Try running hexdump on things in your Windows directory, for example.)

Possible Uses For Hexdump

Even if you are not an experienced programmer, a hex dump utility can be a useful addition to your bag of tricks.

Here are some situations where hexdump could come in handy:

- Finding out where an executable comes from. Ever find a program like "mxlid.com" on your hard drive, and you can't remember what it is, what it does, or where it came from? Maybe the author put a copyright message in it, or there's some readable text from the help option (which you've forgotten how to call up on the command line).
- Examining a binary file for corruption. Ever open a Word document you created in Word yesterday and have Word convert it like it was a WordPerfect document and you get a bunch of little boxes on the screen? What went wrong? You can tell if your file is corrupt if you run it through hex dump: Files with all 00s and/or FFs are usually suspicious and could be corrupt. Unless it is a compressed file of some sort, you should be able to see some of your data or a header telling what kind of file it is.
- Determining the file format. Ever get a file that has an 1X5 extension, and you have no clue as to what format it is in? Most files have a fairly guessable header in them, and by running hexdump.exe on them, you can see enough of the header to get the idea. In the first page or two of the hexdump output, you should see something that will identify the file. Word Perfect files start with WP, for example.
- Finding printable text in file formats you can't otherwise read. Ever get a MIME attachment in your e-mail with a file in Moogleplex format you didn't have a clue what to do with? Maybe you can get the gist of it by hex dumping it. (This will not, obviously, work on compressed files which do not store plain text.)

Bonus Utility: fncase

As an added bonus, I've included my fncase.exe utility. This program is indispensable when you run into problems with filename case in Windows 95. If you've never run into this type of problem, the description of what this utility does may be a little hard to follow. This is one of those utility programs that until you need it, you might think "so what?", but when you do run across a situation where you need it, it is indispensable.

Basically, fncase changes the case of the filenames as they are stored on disk to either all upper or all lower case.

Windows 95 has a strange quirk in it: file names are not case sensitive, although they are stored on the disk in mixed case. Normally, since the file names are not case sensitive, the mixed case is not a problem. Where I run into problems with it is when I have to zip a file and then unzip that file on a system *with* case sensitive file names.

This happened to me when I was developing a World Wide Web page on a Windows 95 PC and uploading it to a UNIX web server. All of the references to filenames in my HTML documents was in uppercase, so if I said `<a href="IMAGE.GIF"> image </a>` it would be okay, on Windows 95, if I had image.gif, Image.gif, or IMAGE.GIF on disk. Under UNIX, though, this was a big problem since the UNIX file system is extremely case sensitive. I got tremendous errors as none of my files could be found! (What's really weird is this problem went undetected for a long time. I had been using an old DOS zip command which did not handle long file names, and using 8.3 filenames. The zip command under DOS converted all files to uppercase, so when they unzipped, my WWW page worked great because all of the file names had been converted behind my back to uppercase. But 8.3 got too constraining, so I switched to a new long file name compatible zip, which preserved the exact case of the file names and caused the havoc!)

Running this utility will convert all the files specified to have consistent case, no matter how they appear on the disk.

Caveat: this command line program is a full 32-bit utility and will only run under Windows 95, not in any of the DOS compatibility modes. You should always run the program with the -p option before using it for real and making sure it will do what you want it to do. There is no way to go back to the mixed case filenames when the program has been run.

Usage

fncase [options] filename1 [filename2...]

fncase is a 32-bit command line utility program.

Options:

- **-p** print out the conversion which would take place on the screen, but don't actually do anything. This option is recommended every time you plan to use fncase: run it with -p and see what will happen before you allow it to rename files on your disk!
- **-l** convert filenames to lowercase (this is the default)
- **-u** convert filenames to uppercase
  - **-? or -h** show a usage message

Notes:
- Wildcard filenames are permitted.
- Options may appear anywhere on the command line.
- Options can either be started by the characters - or /, both are accepted.
    - Options can't be combined together (i.e. -l -p, not -lp).

Extra bonus: shortcut

With the Windows 95 base product, it is impossible to non-interactively create a shortcut, and impossible to interactively create one without using Explorer (or some other file manager). The fact that a shortcut creation command does not exist astounded me. Its necessity is so obvious that I overlooked it for a long time. The shortcut command was the latest addition to the Small Utilities 2.0, and was so important -- a show stopper -- I delayed the product's schedule about a week.

Shortcut is a good reason why supporting the Small Utilities with your registration is so important. I use the Small Utilities myself day in and day out, and I develop software that solves a real need in the Windows world, and is invaluable. For more information about this utility and how/why I developed it, see the Background section of this chapter.

Caveat: this command line program is a full 32-bit utility and will only run under Windows 95, not in any of the DOS compatibility modes.

Usage

Shortcut takes two command line parameters.

shortcut Newfile Oldfile

- Newfile is the new shortcut you want to create. Note that the .LNK extension is automatically created for you, you should not specify it. (If you do put it, you'll get a file that ends in .LNK.LNK!)
- Oldfile is the existing file to which you want to create a shortcut.

Once you give the shortcut command, you will have Newfile.LNK which points to Oldfile.

Background On Shortcut

The shortcut utility was a late but important addition to the Small Utilities 2.0, and one whose development deserves discussion. I wrote the shortcut command when I was trying to create the part of the installation procedure which made the shortcut from LZX to the SendTo menu. I tried to do it in Delphi, but there was no way. I started scrambling around trying to figure out how to do something that took only a few seconds to do in Explorer. My reasoning went: If they can do it, I ought to be able to do it.

After utterly failing to find any way in Delphi to do it, I started looking at Borland C++, and using the IShellLink component of OLE from a C++ program. It took reading the *Programmer's Guide to Microsoft Windows 95* and eventually rooting around in the Windows include files to find out where IShellLink appeared (oddly, the *Guide* book tells everything about IShellLink other than which header file it is in!). I finally got a shortcut working in C++. It hit me like a ton of bricks: here I am doing what the Small Utilities is supposed to stop me from doing, rooting around trying to create an ad-hoc way of adding a feature to one program which was a feature every user and programmer needed and would benefit from having a command to carry out. Time is valuable, and most time spent on the computer is spent reinventing the wheel, rewriting the same code over and over. Come on: is it easier to pay me $10 for a shortcut command, or is it easier to reinvent the wheel every time you need a shortcut? As soon as I had identified the shortcut command as an extreme deficiency in the Windows interface, I knew I had to write it and include it in this 2.0 release. To not do that would be to ignore everything the Small Utilities stood for! The more I looked at the shortcut command, the more I realized it was important. Windows provides no way to create a shortcut from a batch file, for example, or easily from a macro language or install program (without using OLE itself from your macro program -- they haven't made enough nerve tonic for me to even try OLE in VBA!).

But again, the central theme of the Small Utilities comes through: to have a single one-stop solution and then quit reinventing the wheel. Once the shortcut command is written and works, then anyone can use it and quit worrying about OLE IShellLinks. They just call the shortcut command. End users, developers, and anyone else can use the command.

The Care And Feeding Of Command Line Parameters

During the late editing stages of this manual, I noticed that I talk a lot about command line parameters, but assume you, the reader, know what they are and how they work. After contemplating about that, I realized that it's probably a bad assumption. Windows 95, for the first time ever, allows you to (easily) give command line parameters to Windows programs, but the feature is so new and so undocumented that a discussion of it is warranted.

Command Line Parameters In DOS And Windows 3.x

Back when interactive computing was new, and the keyboard/screen combo was a "glass teletype", computers generally gave you a command prompt like '$' and allowed you to enter commands.[7] This command line model became the standard for UNIX, and came to DOS through a bizarre meld of UNIX and CP/M. Commands were like verbs, and parameters to commands were like objects and adverbs. Here's an example: "xcopy /e/s subdir1 d:\foodir". Xcopy, of course, is the command. It has three command line parameters, "/e/s", "subdir1", and "d:\foodir". Things that begin with a "-" or a "/" are generally options, which affect how the program is to do its work. ("/e/s" is the magic command line parameter to get xcopy to copy directories, for example. The slash is a CP/M option, and the minus is a UNIX option. Both can be found in different DOS and Windows 95 programs.) Other parameters indicate what for the command to work on. For DOS and UNIX command line parameters, position matters a great deal. Move, copy, xcopy, etc all use the "command source destination" syntax, where the position of the parameters has meaning. "xcopy file1.txt newfile.txt" means something very different from "xcopy newfile.txt file1.txt". (This is not true in all systems, however. The same way Spanish word ordering is much more flexible than English, some operating systems have position-independent parameters. Don't worry about them, though!)

Windows programs, like DOS programs, have the ability to accept command line parameters too. Programs like Write and Winword have oodles of undocumented and unused parameters that no one knows about, because there's no really good way to start these programs with parameters.

Command Line Parameters In Windows 95

Windows 95 finally allows you to give command line parameters to both console mode and graphical programs, thanks to many new features. You can mix and match batch files and interactive graphical applications. Some parameters even exist in strange places where you aren't aware you're using them, like the SendTo menu.

You can present command line parameters to Windows 95 in any of the following ways:

- From the properties page of a shortcut
- From a Run dialog box like the one in the Small Utilities or Start/Run…
- In batch files using the start command
    - From certain functions of the shell, like the SendTo menu.

[7] At the time, remember, interactive commands were actually a tremendous breakthrough. The alternatives around then were to create JCL files to submit jobs which would invoke utilities like IEBGENER with SYSUT1 and SYSUT2 DD statements to do things for you, as long as you had coded correct DCB information. If you think the DOS/UNIX command line is a bad user interface, it's just because you've never had to use anything worse.

Types Of Small Utilities Parameters

Just about all of the Small Utilities take parameters of some sort, and their use is well documented elsewhere in this manual. The parameters tend to fall into well defined functional groups, though.

The Small Utilities parameters allow you to:

- *Run the program non-interactively*: A lot of the command line parameters allow you to run the program without user input at all, which makes the programs useful to be called from batch files.
- *Help the program get started*: If the command line parameters do not completely run the program without interaction, they generally help you along by doing some of the interaction for you. These types of parameters are also very powerful when integrated with Windows 95's advanced shell features.
  - *Perform some undocumented function*: Occasionally, I have included a command line parameter which is totally undocumented, to make my life easier. The best example is "tools.exe", the toolbar program, which contains the code to uninstall the Small Utilities from the system registry. This code is not something you would ever want to access outside of the provided shutdown procedure, so I have not documented it whatsoever. If you used these parameters, you could damage the Small Utilities or your system.

In addition, of course, the command line bonus programs provide normal command line parameters.

Background Information

All of this chapter is background information on things that have immediate impact on the Small Utilities and computing in general. It isn't required reading, but may be of interest.

Microsoftisms

I've coined the phrase "*company*ism" to refer to a defining trait that's seen in all products, services, or what have you from a particular company or organization. When you are using that company's output, you notice *company*isms as you're using it. They can be good or bad, the only real requirement is that they're always to be found. One example would be a Nortonism: anything from Peter Norton has his picture on it, from books to software. That's a given when you purchase any of his books or utilities. Microsoftisms gave rise to the Small Utilities, so I want to discuss them here.

No matter how bad you think you have it with Windows, it could be ten billion times worse with IBM software. I originally coined the phrase "IBMism" to refer to the bizarre, counterintuitive, inconsistent, poorly documented, and oftentimes baffling "features" of IBM software I began noticing when OS/2 was inflicted on me. I later generalized the term to *company*ism.

I collect IBMisms now, and here are some of the best:

- One of the weirdest things about Windows 95 and OS/2 is installation. I've never had any trouble getting Windows 95 installed, and have had nothing but trouble with OS/2. What's weird about this is that given 100 people, probably about 50% have had a flawless, smooth installation and the rest have had problems. After getting my copy of Warp, I tried and failed for days to install it. I honestly didn't think I'd ever be able to get OS/2 installed on my machine. Finally, a co-worker installed it by *turning off the turbo switch*. The only problem with that solution is I had a Pentium 60, and the turbo switch doesn't actually do anything. Nevertheless, when the turbo was turned off, it installed flawlessly. That was one of the weirdest experiences of my life, because it was the first IBMism I'd ever encountered.
- You can't put semi-colon comments in your CONFIG.SYS file. What happens is OS/2 crashes completely during boot. It doesn't say syntax error or bad line or anything, the entire OS crashes. Since ; is a very, very common comment character, you'd think they'd handle finding one a little better than they do -- they could at least print an error message.
- I took an immediate dislike to the Launch Pad program, which seemed to add clutter to the screen and do nothing in return. I deleted it from the desktop. I deleted it from CONFIG.SYS's auto-run line. I deleted it from everything, and it still came back! It is apparently hard-wired into the OS's kernel to launch the Launch Pad every time you misclick or make a mistake. You can't get rid of it. Bizarre.
- After giving up on removing Launch Pad, I decided to move all the garbage all over the desktop into the Launch Pad launching area, to consolidate. I drug the object-oriented things on my desktop onto the launch pad, made sure they were there, and deleted them from the desktop. Then, I went back to Launch Pad -- and they were *gone*! I had to rebuild the entire system from scratch and reinstall everything. So much for an object-oriented desktop!
- I had the misfortune of porting a program to DB2/2 -- which required I install it. I was running OS/2 Warp, which was version 3 of OS/2. DB2/2 had a hard-coded version check for version 2, which meant it would not install on later releases even though the later releases would run it. (As usual, the install program did not pop up a dialog box saying "This software will only run on version 2 of OS/2", it just crashed.) I had at that

time just been converted to the benefits of software on CD-ROM, so I had ordered DB2/2 on CD. It took days and days to find the answer to the problem and a fix. Then, since CD-*ROM* is read only, I had to recreate the install diskettes to apply the patch. By this time it was hard to remember what I was going to do in the first place.

- Just to prove that DB2/2 wasn't a fluke, I got a Windows DB2 software development kit from IBM on CD-ROM. I just could not install this no matter what I did. I finally copied the distribution over to a hard drive, and tried installing it from there, and it worked flawlessly. Huh?

That leads us to the Microsoftism. Microsoft's trait is arbitrarily deciding what users will be allowed to have and what will be withheld from them, usually with no good reason for the decision. To paraphrase Ron Burk of *Windows Developer's Journal*, Microsoft's motto is "if we don't think you need it, why does anyone else?" Some necessary components are not included with Microsoft products, and some unnecessary ones are. Usually the omissions are characterized by the fact that the thing that's omitted is already written and tested and in use. Inclusion of these things would cost the company nothing in terms of money, manpower, or time. It's particularly frustrating to someone like me who likes to have resources at his disposal to fix problems to be limited for reasons I can't understand. It also makes you think Microsoft is judging either what the end user is capable of handling, or what they feel the end user wants.

Some examples of Microsoftisms:

- Icon editor: Since the beginning of Windows, end users have never been allowed to possess the official Windows icon editor. Microsoft has had an icon editor since the word go and it is included in the Windows SDK. As Microsoft Bob and the Windows 95 Plus Pack proved, people like personalized computers, and an icon editor which was already developed and working would cost them nothing to include and gain them a lot of warm and fuzzies. (To make matters more bizarre, IBM developed an icon editor "for internal use only" and then released it into the public domain to be downloaded for free. It's not a bad program[8].) Even with Windows 95, there is still no icon editor shipping with the operating system, although just about every other GUI from X Windows to OS/2 has one.
- compress.exe: Although end users were allowed to have expand.exe (this isn't strictly true, since the DOS install program was allowed to have it and magnanimously, Microsoft did not delete expand.exe after the install program was done), they've never been allowed compress.exe, which is all of 15K in size. Only software developers who get the Windows SDK are allowed to actually compress files. Even though it would cost Microsoft nothing, and would give the people using their products the warm and fuzzies, they chose not to allow end users to have compress.exe. (Note this saga continues to this very day, since the CAB file creator program has never been released to software developers or end users.)
- DOS 6 programs: The users actually won this round. Microsoft was not, in spite of distributing Windows 95 on a 600MB capacity CD-ROM, going to include Qbasic and the other MS-DOS 6 utilities which no longer shipped with Windows 95. After a huge uproar, Microsoft finally put the utilities on the final Windows 95 CD-ROM. But they wouldn't have if there hadn't been such a backlash!

[8] For *IBM*, of all companies, to have one-upped Microsoft in anything related to user interfaces or giving something extra to users, makes you question the underlying principles of the universe as we know them….

- cardfile: The useful, if stripped down, cardfile program was omitted from Windows 95 for no particular reason I can see. A lot of people used and liked this program. And it was deleted.

(Bizarre inconsistencies in software that simply cannot be explained in any rational way, but which are glaring and noticeable almost immediately, are another Microsoftism. Notice how in the printers dialog box, to add a new printer you double click an icon in the dialog box itself. But to add a new font, you have to go to the File menu and select a menu item. Microsoft's mission for Windows 95 was to make things "discoverable", which explains why to change the font in Explorer you have to go to the display properties, pick the appearance tab, and change the *icon* label size! I love to collect these Microsoftisms. I may publish a book some day.)

A related but separate Microsoftism is code reuse through cut and paste. Instead of designing modular systems, Microsoft creates example programs which encourages developers to include features by cutting and pasting the example programs into their own code. They do not create a utility that becomes part of the operating system which everyone, users and developers alike, can use and count on to be there and work.

The standard procedure seems to be:
1. Invent a system call of some type to perform a function, which requires building a parameter block in memory and passing it off to the system call.
2. Write a C++ class wrapper to turn a simple system call into a production.
3. Shovel all this on a CD.
4. Programmers then cut and paste the example (either in C or C++, or they translate it into another language like Delphi by hand) into their own programs[9].

In this case, end users lose big time because no general purpose utilities are created. The classic example is Run in the Small Utilities: for ten years, instead of having a standard system-wide Run program, every last Windows program had to create its own dialog box and call to the underlying system call WinExec(). File Manager and Program Manager in the old Windows 3 were victims of this: both implemented a File/Run... dialog, and both did it differently.

Shutdown and Run

Let's face it: Windows and DOS have always been very random. Things have happened to the OSes for which there is often just simply no explanation other than "that's the way it is". Different programmers on different teams through the years have been adding things that don't really fit in with the overall picture, because there *is* no overall picture. DOS and Windows are like an oyster building a pearl layer by layer, adding different system calls, commands, and utility programs to DOS and Windows. If you just get Windows 95, without an appreciation of all the history, it can be baffling why some commands behave the way they do. Why are commands internal, external, or completely inaccessible? Why can't you just type a command and have it run?

[9] My theory is there's only been one Windows program ever written. Everyone took this example program and modified it by cutting and pasting examples from the SDK documentation, until several examples were available. Then they added frameworks which automatically generated applications, which were modified by cutting and pasting examples from the framework documentation. Then there was a critical mass of code to swipe from!

To go back in time, the original DOS version *1.0* had all of what we think of as "commands" (DEL, DIR, etc.) included as part of COMMAND.COM, the command interpreter. (These are called *internal commands* because they're part of COMMAND.COM and can't run by themselves. They depend on COMMAND.COM.) It had no other standalone programs to help it other than EDLIN and DEBUG (I believe I'm correct in asserting these are the only two programs that came with DOS 1, but there may have been others -- I think LINK was one).

The next version of DOS had more internal commands, and some external ones -- meaning standalone programs. As DOS grew, some commands were added to COMMAND.COM, and some weren't. XCOPY is a particularly extreme example -- although it is a superset of COPY, they had to preserve the backwards-compatibility of making COPY an internal command, and XCOPY was too big to add to COMMAND.COM which has to be as small as possible, so DOS got two COPY commands which did essentially the same thing...!

When you add DOS to Windows, you create a situation where you can run a command like DIR or COPY and have nothing happen because you did not start the command interpreter and tell it to execute the command. This behavior has spilled over into Windows 95. Windows 95 adds the ability to run Win32 programs which can be started from the command line and run as Windows programs, as well as the ability to run Windows programs using the DOS START command. Plus, through the years, some DOS and Windows programs and commands are added and deleted, meaning that if you sit down at an average machine that's had DOS, Windows, and Windows 95 installed on it, you can be fairly certain that you'll have *no* idea what commands are available. (And I haven't even gotten into adding new third-party commands like UNIX utilities! Unless you installed the utility yourself and know how it operates, you can't be sure how good of a clone of the original the ersatz utility is. And it doesn't consider all the different utilities that do essentially the same function, but do it differently.)

You wind up with a lot of different kinds of programs:

- DOS internal commands (which require COMMAND.COM to run)
- external DOS commands (like XCOPY or EDIT)
- DOS programs (like 1-2-3)
- Win16 programs
- Win32 programs
- Win32 console mode programs (which look like DOS commands, but are full 32-bit Win32 programs you run from the command line, like hexdump)
    - Weird programs like DOS extenders which use Windows' memory management but exist in DOS.

Windows, at least pre-Win95 Windows, is a bizarre hodgepodge of every design philosophy you can think of, and something that *evolved* during a 10 year period. The early part of this period was spent passing the buck from one programming team to another. Windows was very much the punishment project people didn't want to get assigned. This kinda, sorta explains some of the bizarre things that have happened. I won't even mention the Windows API and internals. You'll need to develop a cast-iron stomach to look at them. But externally, strange things happen as well.

This hodgepodge resulted in strange code which has duplications and overlaps. Program Manager icons in Program Manager groups are completely separate and different from program icons on the desktop, and you can delete a Program Manager icon for an application and not delete if from the disk -- but when File Manager deletes an application, it is gone forever. Program Manager is actually an improvement over Windows 3.0's MS-DOS Executive, a “shell” that managed to single-handedly defeat the entire graphical user interface paradigm by looking like text! Program Manager is too new to be explained away as a relic of the ancient past. By the time progman was introduced, Windows was far enough along to have been influenced by good design practice.

Windows 3 was utterly schizophrenic in that the following two conditions were true at once:

1. You could not start programs from the command prompt
2. Programs accept command line parameters.

Think about it: a program accepts command line parameters, but you can't start programs from the command line[10]. You can, in Program Manager, actually supply a command line (using the Alt-Enter screen for an icon). But this is not documented anywhere, and most Windows programs support command line parameters that are also undocumented. Now that Windows 95 finally supports starting programs from the command line, it's too late to really use the feature because most programs have ignored it or not documented its use.

The Program/File/Task manager triangle is a good example of weirdness in Windows. All three of these do an ill-defined task, and overlap in strange ways, and suffer from inconsistencies. One strange example of duplicated functionality comes from the File/Run... command. Both File Manager and Program Manager have File/Run... commands, but there's no "run.exe" application that allows you to run a program, so both of these are part of the File/Program manager application itself, essentially duplicated functionality. Neither behaves consistently with the other -- File Manager will "help" you by filling in the highlighted file name on the command line, but Program Manager doesn't. Strange. The strangest part is Task Manager has no Run... command whatsoever, and it is the one which most critically needs one since you generally resort to it when neither File or Program Manager are working and you need to run a program to recover the system. Windows NT's Task Manager, obviously, provides a way to run programs.

What this is leading to is there is no standalone programs to provide interfaces to certain system functions, but there are many overlapping, inconsistent interfaces. Each application that wants you to be able to run a program will write its own interface to the fundamental, underlying WinExec() system call. That's why the Start/Run... menu item can't be put on the desktop. It isn't starting a separate program, it is creating its own dialog box and passing off what you type to the WinExec() call. It is hard-wired into the Start menu of the shell. Other programs can provide the same Start/Run... functionality if they want to. Some macro languages in applications allow you to call WinExec(), or provide their own equivalents which do nothing but call WinExec(). The problem is they're all different.

Start/Shutdown... is the same thing, almost. It is basically just a call to the underlying system call that shuts down the computer (in Windows 95, in Windows 3.1 it returns you to DOS), restart Windows, or restarts the computer. The Start/Shutdown... dialog is hard-wired into the shell, like Start/Run.... There's no generic "shutdown" program in Windows, which means each program which wants to shut down Windows (most install programs need to do this!) has to devise its own way to call ExitWindows(). Some give you a choice, some just do it without telling you anything, some tell you to do it yourself.

[10] It reminds you of a paradox like: "Everything I say is a lie. I'm telling the truth." There is no command line. You can give command line parameters. Are we getting into metaphysics here?

Windows, and DOS, do not follow a well-designed plan like UNIX. In UNIX, each major system call has a command that you can run in order to access the system call. For example, mkdir is a command that calls the mkdir() system call with a directory you specify. There are no internal or external commands, since all the command interpreter does is to interpret commands, not actually perform them the way DOS does with its internal commands (with a couple of rare exceptions we'll leave for the advanced class to discuss![11]). You can mix and match these commands any way you want to, in programs, "batch files", or anything else. There is a "shutdown" command that does nothing but shut down the system. You can run it at the command prompt, from within a program, or anywhere else.

What's the point? I'm trying to tell you by examples that the Windows interface is not particularly well thought out or modular. Everything that wants to access functionality usually does so by implementing the functionality as a part of the program. There are no general purpose interfaces to things like shutting down the system at a layer above the actual Windows API. If every application writes its own graphical interface to the underlying API calls, then the result is a lot of applications which are just different enough to be confusing and no general solution for when the need for the functionality comes up again. Although strides have been made to get people to stop duplicating functionality, we're not there yet.

It may be too late to save DOS and Windows, but I'm trying to complement the wide variety of command line utilities with a few GUI ones.

History of File Compression

Windows 3.1, and DOS 6 (I'm not positive about DOS 5), come with their own built-in file compression scheme. Sort of. It is not well documented, and I suspect that most users do not know it even exists.

Typically, files compressed with this scheme end in an underscore. This is a convention only, and there's no rule that says files *must* end in an underscore.

compress.exe: To create a compressed file, the Microsoft utility program compress.exe is used. This is a typical DOS command: it accepts the file to be compressed and the file to be created as arguments. It is well-suited to be called from batch files. Before you get too excited about having a file compression scheme at your fingertips which is built into Windows and DOS so you can use it to make compressed files anyone can uncompress, the catch is that you don't get the compress.exe program in the retail versions of DOS, Windows 3.x, or Windows 95. (Considering compress.exe is only 15K, there is no particular reason why it could *not* have fit onto the Windows 95 CD-ROM, in the extras directory if nowhere else. Since the compress.exe program has been around a long time, it would cost Microsoft nothing to include it and allow everyone access to file compression easily. They don't even have to document it, since they've given up on documenting their products!)

expand.exe: The uncompress program. It reverses the process started by compress.exe. Other than uncompressing instead of compressing, the two are almost identical. This utility is also a

[11] These are usually commands which affect the shell itself, like 'cd' or 'setenv' -- the shell itself must execute them, because it keeps track of what directory you are in or what environment variables you have available to programs.

DOS command line program. It is not particularly effective in batch files, since it gets deployed for ad hoc situations more than anything else. This utility is rarely used in Windows outside of "toolkit" situations where individual files from distribution diskettes need to be uncompressed in an emergency, since Windows install programs use the uncompression built into Windows itself. DOS installation programs (and I've seen very few over the years which use this compression scheme at all other than MS-DOS itself) could use it. The good news is that you have a copy of this if you have DOS or Windows.

lzexpand.dll: A library of uncompression routines. Any Windows program which wants to uncompress files can call on these routines, since they are built into Windows and can be expected to be available on any machine running Windows. Typically the only programs which use this library are setup programs. There is no equivalent "lzcompress.dll" available, so the compress.exe program Microsoft provides is the *only* means for creating compressed files.

CAB files: Microsoft has switched to a new compression scheme called CAB (for "cabinet") files. This new compression scheme is much more like PKZIP, since it allows multiple files to be compressed in a single CAB file. (Contrasted with compress.exe which compresses only one file at a time.)

One difficulty in using Microsoft's built-in compression is that it is totally undocumented. The compression scheme used by compress.exe and expand.exe are undocumented, and no third party replacements for them exist. CAB files are undocumented as well.

About The Author

The author of the Small Utilities, that's me, Scott McMahan, is by day a regular programmer who works as a software developer specializing in databases and C programming. About the only type of computer I've never seen or touched is an AS/400. I've used and programmed MVS mainframes (using SAS C), UNIX workstations (DEC Alphas and DEC Stations, Sun back when they had SunOS, Hewlett-Packard HP-UX, IBM's RS/6000), VMS (yuck!), and personal computer operating systems (Windows 95, NT, OS/2). Generally, every machine, OS, and software package I've ever worked on I've had to set up out of the box myself and administrate as well as use, so I've probably installed every kind of software in the world. I know C, C++, and perl well, and have gotten a Pascal education using Delphi. I know just a little about a few other languages like BASIC, REXX, and awk. I've done a lot of work with SQL using various databases. My big research area in college was the Internet, back in the good old days when no one knew it existed except the people on it. I wrote a mailing list administrator package in perl as my senior project in college, and have done a web page or two.

My machine at home is a Packard Bell something or other close-out model (I finally had to get a machine with a CD-ROM since all software just about is now distributed on CD-ROM), which is more powerful than the CAD graphics workstations I used in college, and whose onboard chip cache has more memory than my first computer, a TRS-80 CoCo II. I run Windows 95, and am a big believer that it is a great OS and has a lot of potential. That's why I want to develop Windows 95 software. The more I use the OS, the better I like it. It supports powerful new applications, all the old applications I have, the best games in the world[12], and throw in enough UNIX utilities and a vi editor clone it has a decent command line environment. The Win32 model is so much like UNIX that you can write C programs that look, feel, act, and smell like UNIX programs. I've been a Borland development tool user all my life when it comes to Windows. I wrote all of the Small Utilities which have graphical components in Borland Delphi. I think Delphi is the best thing to happen to programming since the invention of a compiler. The command line utilities were written in Borland C++, still the best C development environment going for the PC. I did this documentation file in MS Word 7 for Windows 95.

You may have seen my tips and information in:

- *Microsoft Magazine*'s 50 Fabulous Tips (Spring 1995)
- *PC Magazine*'s User-To-User (November 7, 1995)
    - *Maximize Windows* (December 1995)

These are a reflection of how I want to take my experiences of making things flow more smoothly and work smarter and more productively and share them. The problem is, it's hard to share things with a lot of people easily. My File/Run... tip in Microsoft Magazine was wonderful because a lot of people got to use a tip I'd only been able to tell a couple of people before that point. The Small Utilities are a way for me to take my abilities to simplify and streamline and share them with a wide audience.

I guess if I had to describe my role as a programmer and computer guru, I'd say I'm an efficiency and productivity expert. I always want faster, easier, better ways to perform tasks, whether it is

[12] Take this about games with a grain of salt -- I think Jill Of The Jungle, Wing Commander II, Dungeon Hack, and Epic Pinball are the best games in the world and sniff at "multimedia" titles which have little playing value and lots of fancy graphics. Your mileage may vary.

writing a library to make programmers' work easier, or the Small Utilities to make everyone's daily computing experience easier. The computer is designed to be tinkered with and improved over time to make performing tasks easier. It is not static: if you do not like something, or can do something better, all you have to do is create a way to make it happen.

From day one, my first real job, I've been specializing in tools to make myself more productive. I wrote a "visual" form designer (I use the term loosely since we're talking about a character mode terminal interface!) for a database product from the stone ages which required you to put the row and column numbers for everything you put on a form (*everything* -- every last word in headers, footers, the body, etc. -- whatever you're imagining, it was 100 times worse!). My tool took my visual file I made in a text editor and created the drudge work file with all the numbers the stone age database processed. I'm the kind of person who will spend an hour writing a tool to speed up a process rather than slug through the same laborious, time consuming steps. That led directly to the Small Utilities. If I can make myself more productive, why can't everyone benefit?

I like feedback, even negative feedback from people who don't want the Small Utilities. Many of the enhancements in this version are from that exact type of feedback! If I know why people don't like it, I can make it better. Don't hesitate to make any suggestions for the improvement of this product! I'm 100% customer driven. The software dreamup factory in my brain has two more shareware projects in the works for the next year or two, so I'll stay busy.

If you have a program you need written, let me know! I'm always available for ad-hoc contract work. I'm also available for producing manuals and other designs. If you think this manual looks sharp, imagine what I could do to yours!

Besides being a programmer, I have a lot of outside interests, but little time to actually pursue them. Besides reading 1000+ page programming books to learn stuff like OLE and Delphi so I can bring you the Small Utilities, I used to read regular books. (I, in fact, read James Michner's The Source at one time, if you can believe anyone would have that much free time!) I've been known to root for the Carolina Panthers (in fact, most of the Small Utilities 1.0 was written during their inaugural season while watching them play!) and the Atlanta Braves. Favorite players include John Smoltz and Greg McMichaels. I also have my own demo tape of original ambient/progressive music available (ask me about it!). I'm one of the biggest fans of the music group Genesis you'll find, and maintain *The Genesis Discography* (visit it at the URL http://www.morphnet.com/~scott, also listed with Yahoo), which is the most complete and largest source of info on the band ever assembled. I'm also trying to start a Sara Mornell fan club, except I appear to be the actress' only fan and no one else even knows who she is.