CT-Shell for Windows

CHAPTER 1 - Installation

This chapter describes the process of installing CT-Shell onto your computer. It is a very simple process, and because it is much easier to understand how CT-Shell works if you try out features as you read about them, these directions are placed first in the manual. It is strongly suggested that you go ahead and install CT-Shell before going beyond this chapter.

Because it is expected that CT-Shell will be installed by novices and experts both, this first chapter is split into two parts: Expert Installation and Novice Installation. The Expert Installation part contains brief directions, without a detailed explanation of the steps that are involved. The Novice Installation part contains a full explanation of each step.

If you are very familiar with computers and with Windows, you will want to install CT-Shell according to the Expert Installation part. If you are new to computers or if you have just recently started using Windows, you will appreciate the Novice Installation part.

If you fall somewhere in between, you should start with the Expert Installation directions, and look to the Novice Installation part for more details if necessary.

...but first a word about shareware

This version of CT-Shell is not free software, but a kind of software that has become known as *shareware*. That means it is copyrighted material, with all rights reserved by Computer Training, but that you are granted certain limited rights with respect to the software.

Specifically, you are permitted to use CT-Shell v1.0 for a period of 30 days for the purpose of evaluating it, and you are permitted (even encouraged!) to distribute *unmodified copies* of the original distribution archive to others for their evaluation. (So long as the contents remain original, the archive may be

repackaged using any archiving program, such as LHArc, PKZip, Zoo, etc.)

At the end of the limited 30-day evaluation period, you must either discontinue using CT-Shell and remove it from your computer, or you must register your copy and pay the required fee (for this version, $35.00 US). Payment of the required fee will provide the following:

1. A clean, original copy of the newest version, including any incremental improvements since the version you received.

2. A professionally printed, tabulated, laser-typeset, graphically illustrated manual.

3. A special softkey (a small file) that will remove the (Unregistered) identification at the top of your CT-Shell window, and bypass the original sign-on reminder that you're using an unregistered evaluation copy of the software. Your key will unlock future incremental upgrades at no additional cost, allowing you to download new registered versions of the software when they become available.

4. Voice support by telephone, and a special enhanced level of access to Computer Training's bulletin board system.

5. The honorable feeling of using the product legally, and the satisfaction of knowing that you've made the author a much happier person.

The Computer Training BBS phone numbers (Programmer's Resource BBS) are:

(206) 823-2831 - 2400 bps
(206) 823-1917 - 9600 Courier HST

The Computer Training main office phone number is:

(206) 820-6859 - in the Pacific time zone

There is a file distributed with this shareware version called CTREGIST.DOC. That file is a registration/order form in ordinary 80-column ASCII format, that you can easily print (using CT-Shell, if you want, or just copy it to your printer). Please mail it with your payment to the address shown on the form.

The requisite disclaimer of liability and copyright notice are located at the end of this document. You may want to print a copy of this shareware version of the documentation and refer to it as you explore CT-Shell.

Note that there are several places in this file where reference is made to illustrations that are not included in the shareware documentation. They were not left out as incentive to register and obtain a complete manual - rather the graphic images did not survive the translation from Word for Windows format to Windows Write format. *Every word* of text from the original registered version of the manual is included in this shareware copy. The Write format is used for the shareware manual, as everyone who has Windows has Write.

Be advised that there is also a high level of help available from the Windows online help system, which you can access through the *Help* menu in CT-Shell. Enjoy!

Expert Installation

CT-Shell is distributed in a self-extracting archive file named CTINSTAL.EXE. You can install the program from the DOS command line or with a RUN command in the Windows Program Manager or the Windows File Manager. When that archive file is run, it will copy its contents to the current working directory.

First, change to your Windows "home" directory, to be sure that your copy of CT-Shell and its configuration file CTSHELL.INI will end up where they can be found again later! Also, many of the tutorial examples that are suggested in later sections of this manual refer to files and directories that everyone should find in the Windows directory, so it's a good place to be while you're learning to use CT-Shell.

Installing from DOS

To install the program from the DOS command line, change your current directory (if you haven't done so already) to the Windows directory. Then issue the command:

d:CTINSTAL

where d: is the drive where your copy of the program is stored. Most often it will be on a floppy disk that has been inserted into drive A:, so the command to install it would be:

A:CTINSTAL

If your CTINSTAL.EXE program is in another drive/directory, such as on a network path, use an appropriate path in your command, but be sure your *current directory* is where you want the program to end up. Having installed CT-Shell, you can start Windows and begin a CT-Shell session with a command like:

WIN CTSHELL

Installing from Windows

If you are currently running Windows, and are installing the program from either the Windows Program Manager or the Windows File Manager, select the FILE menu, then the entry named RUN... You'll be presented a dialog box that has a place to type the command as shown above in the section on Installing From DOS. When the program has been successfully installed, simply select the RUN... entry again and give the command:

to begin a CT-Shell session. Or, if you are working from the Windows File Manager, you could select the file named CTSHELL.EXE and doubleclick the mouse on it, or press <Enter>. In short, use any of the methods you usually use to run an .EXE file from within Windows.

When the installation is complete, you will have the following new files in your Windows directory:

CTABOUT.DOC- ASCII text file telling briefly about CT-Shell
CTREGIST.DOC - registration form
CTSHELL.EXE - the executable program
CTSHELL.INI - sample initialization file
CTSHELL.HLP - help file
CTSHELL.WRI - Windows Write format document file

and possibly

CTREADME - updates to printed manual, included only if needed

Since it is a Windows program, you probably don't need a whole lot of instruction before you use CT-Shell. In fact, if you know Windows, you already know most of CT-Shell. It's a good idea, however, to look over each chapter of this brief manual before you experiment with the corresponding features of CT-Shell.

Although this program has been designed to be as safe as possible, and to ask you for confirmation before it does anything dramatic, by its very nature *any DOS shell* has the potential for disaster if misused. Because you can delete an entire directory full of files with just a few clicks of a mouse, for example, you'll want to be sure you know what you're doing *before* you do it.

Novice Installation

This section isn't meant to be demeaning, just sympathetic! Not everyone has worked with Windows for a long time, and not everyone who uses a computer is a "Power User"... yet. CT-Shell offers so much to *all* Windows users that we want to be sure no one is left confused about installing it.

CT-Shell is distributed in a self-extracting archive file named CTINSTAL.EXE. That's a kind of file that is like a container for a collection of other files, stored in a compressed form. You can extract those files from the archive, and return them to their original size and condition very easily, just by "running" the archive file.

When you run CTINSTAL.EXE, it will put several files into the current working directory, including CTSHELL.EXE, CTSHELL.HLP, and CTSHELL.INI. Those are the three main parts of CT-Shell - the program itself, its associated help file, and its initialization file. You may also find that a file named CTREADME has been included as well. This file - if it is included - contains additional new information and perhaps corrections that were discovered after the manuals were printed.

Before you install CT-Shell, it is a good idea to change to your Windows "home" directory, to be sure that your copy of CT-Shell and its configuration file CTSHELL.INI will end up where they can be found again later! Also, certain optional tutorial exercises that are suggested later in this manual will assume that you are in that directory, and will refer to files and directories that everyone should have there. Therefore, the Windows directory is the best place to be while learning to use CT-Shell. If you're not already there, give DOS a command that looks like this:

```
CD  \dirname
```

where *dirname* is the name of your Windows directory. Chances are, you've named it something like \WINDOWS, or another popular choice is \WIN. If you have installed Windows in the directory on the current drive called \WIN, then the command to change to there would be:

```
CD \WIN
```

This is a brief explanation, and assumes that you understand what a directory is, and that you know in which one you installed Windows. You can find out more about DOS directories in your DOS owner's manual, if necessary.

If you did not install Windows yourself, and are not sure where its home directory is, you may need to do a bit of detective work. Starting at your root directory (you can get there with the command CD \ ) look for one of the directories mentioned above, either one named WIN or one called WINDOWS. Change to that directory, and see if you can locate the file WIN.INI in it. If so, you've found your Windows home directory, and that's where you'll probably want to install CT-Shell.

If you have not been able to find your Windows directory at all, you can still install CT-Shell, but you'll want to be sure you do it in a directory that lies along your executable DOS path. If you need more information about the DOS path, refer to your DOS manual, or perhaps ask for some help from a friend who has worked longer with PCs.

At this point, it is assumed that you are in your Windows directory, and ready to install CT-Shell there. It can be installed from a DOS command line or from within Windows itself, however installing from DOS is simpler, and the results are exactly the same, so that's the approach we'll take here.

To install the program from the DOS command line, issue the command:

```
d:CTINSTAL
```

where d: is the drive where your copy of the program is stored. Most often it will be on a floppy disk that has been inserted into drive A:, so the command to install it would be:

```
A:CTINSTAL
```

Remember that the extracted files from CTINSTAL.EXE will end up in your *current directory*, so don't change to the disk where the installation program is. Be sure that you run CTINSTAL.EXE from the place where you want the files to end up, by specifying the drive letter in front of the program name, as described above.

When the installation is complete, you will have the following new files in your Windows directory:

```
CTABOUT.DOC- ASCII text file telling briefly about CT-Shell
CTREGIST.DOC   - registration form
CTSHELL.EXE    - the executable program
CTSHELL.INI    - sample initialization file
CTSHELL.HLP    - help file
CTSHELL.WRI  - Windows Write format document file
```

and possibly

CTREADME - updates to printed manual, included only if needed

You can start Windows and begin a CT-Shell session with this command at a DOS prompt:

WIN CTSHELL

If you ordinarily start Windows with command-line switches of any kind, such as /s to start Windows in Standard Mode, you can include them between WIN and CTSHELL. The next section shows a way that you can modify your Windows SYSTEM.INI file so that CT-Shell is automatically started by Windows whenever it starts.

An important word of caution: since it is a Windows program, you probably don't need a whole lot of instruction before you use CT-Shell. In fact, if you know Windows, you already know most of CT-Shell. It's a good idea, however, to look over each chapter of this brief manual before you experiment with the corresponding features of CT-Shell.

Although this program has been designed to be as safe as possible, and to ask you for confirmation before it does anything dramatic, by its very nature *any DOS shell* has the potential for disaster if misused. Because you can delete an entire directory full of files with just a few clicks of a mouse, for example, you'll want to be sure you know what you're doing *before* you do it.

Installing As a Shell in SYSTEM.INI

This section will be of interest both to experts and to novices. CT-Shell makes a wonderful replacement for the Windows Program Manager, and it can be configured to start automatically, every time you start Windows. You'll need to make a simple change in one line near the beginning of your SYSTEM.INI file, which is located in your Windows directory.

If you have already installed CT-Shell, you'll find it quite simple to make this change, and the change you'll make is easily reversible, so everyone is encouraged to give this a try. Although you'll be trying out some features of CT-Shell that haven't been explained fully yet, very explicit directions are provided.

What you want to do is really quite simple. You want to change one line near the beginning of your SYSTEM.INI file so that instead of reading:

shell=progman.exe

it reads:

shell=ctshell.exe

Here are simple, step-by-step instructions that will show you the easy way to do that from within CT-Shell, with the assumption that everyone has the NOTEPAD.EXE editor available (it is included with Windows). A command to edit your SYSTEM.INI file using the NOTEPAD.EXE editor is already installed in your CT-Shell menu as a convenience, since this is a file that you may need to modify from time to time as you adjust and configure Windows.

1. From the menu at the top of the CT-Shell window, select the *Win* menu item. Click on it with the mouse to display the *Win* popup menu, or if you prefer to use the keyboard, press the key combination <Alt-W> to access that menu item, then press <Enter> to display the popup menu.

2. Using the mouse or the keyboard, select the entry named *SYSTEM.INI File*. This menu entry has been designed to start the NOTEPAD.EXE editor, and tells it to edit the SYSTEM.INI file in your Windows directory.

3. Use the editor to change the line shown above, so that it specifies CTSHELL.EXE as the program to run as the Windows shell, rather than whatever is there now (probably PROGMAN.EXE,

as shown). Remember, you can always reverse this procedure by changing that line back the way it was. An alternative method is to place a semicolon in front of the original line, changing it to a comment, then add a line to run CT-Shell as the shell, like this:

```
;shell = progman.exe
shell = ctshell.exe
```

That way, you could easily change things back later by changing the semicolon to the line that runs CT-Shell.

4. Save your file and exit from NOTEPAD, using the *Exit* command from the *File* menu. You don't need to save the file first, as NOTEPAD will ask you about that before it quits.

The change that you've made will take effect the next time you start Windows. Instead of starting a session with the Windows Program Manager, you'll begin your session with CT-Shell.

Note that PROGMAN.EXE is an ordinary executable Windows program, and you're quite able to run it from CT-Shell, if you ever want to. In fact, as you'll see in a later chapter, you could easily add the Windows Program Manager to your CT-Shell menu, allowing you to select it any time you want it. Most people who have tried CT-Shell, however, far prefer it to Program Manager.

If you have decided not to install CT-Shell as your Windows shell at this time, you may want to install it in one of your program groups within Program Manager, to make it easier to start when you want it. You'll be able to begin a CT-Shell session just by clicking on its icon in the group that you've assigned it to.

Multiple Configurations

Note that no matter how or where you start CT-Shell, you may provide an optional initialization filespec on the command line after the program name. Rather than looking for its initialization file, CTSHELL.INI, in any of the places where CT-Shell would normally expect to find it, CT-Shell will instead use the name you provide as an initialization filespec. Thus, you may create multiple configurations, each of which customizes your CT-Shell session to work a different way.

Once you understand how to add new custom features to your menus, for example, you might want to create a version of CTSHELL.INI that is specially designed for programming with a particular language. Or another version that is designed purely for word processing projects. Just remember to provide the

whole filespec, including its directory path, since CT-Shell doesn't provide any kind of file name default when you specify the initialization filespec yourself.

To start a session that is customized for programming, you might start CT-Shell with a command like this:

```
CTSHELL  c:\win\program.ini
```

or to start a word processing session with a customized menu, you might use:

```
CTSHELL  c:\win\wordproc.ini
```

Of course, either of these would require that you create those xxx.INI files. You might begin by copying CTSHELL.INI to the new names, then modifying it to suit your purposes.

This handy feature also makes it easy for multiple people to use a computer on which CT-Shell is installed. Each of them can create an initialization file that suits the way they individually work, and change to it when they're using the machine.

One last point should be made here. As a Windows application, multiple instances of CT-Shell may be run at the same time quite effectively. All the instances share the same program code, so it is just the data that differs among them. You will find that the first copy of CT-Shell requires very little memory to run in, but that a second and third copy require even less. Don't be afraid to start several at the same time, using different configurations, if that turns out to be useful to you.

CHAPTER 2 - Overview

This chapter provides a quick summary of CT-Shell's overall capabilities. Don't be concerned that some topics are introduced here and not explained fully right away. Later sections of this manual will provide all the details.

From this point on there will be various comments and explanations that go beyond what you must know in order to use CT-Shell effectively, and they are meant to be additional information for more advanced users. Those comments will be placed into footnotes, so that the flow of the material is not impeded.

Generally speaking, you can ignore all footnotes as you read the rest of this manual, except when you'd like more details about the particular topic of discussion.

CT-Shell is a DOS shell that was originally developed for use in advanced programming courses, as a replacement for the DOS 4.x shell. It allowed programming students to change easily and quickly to the directories where they were to work, and made it easy to do their work with routine commands from its menu.

CT-Shell for Windows is still a DOS shell, though it now takes advantage of the Windows 3.x environment. You can use it to launch DOS and Windows programs, to copy, move, list and delete files, and yes, programmers still use it to build programs.
CT-Shell is quite configurable; in fact, *every entry* in the menu is defined in the

CTSHELL.INI file, and you can change any of them or add more entries, all to suit your needs. It is very important for you to realize that your menu need *not* stay just the way it is, and that you can add entries to run all your favorite programs, change to the various directories where you do your work, and automate routine tasks such as disk backups, so you can accomplish them by clicking on a menu entry. However, the fastest way to find out what the program is all about is to start it using it with the supplied sample configuration file.

So that CT-Shell can find its initialization file, CTSHELL.INI, make sure that it is in the same directory as CT-Shell, in the Windows "home" directory, in the Windows "system" directory, in any directory along your executable DOS path, or in any directory

specified in a network installation. Those are the various places where CT-Shell will be able to find it without help. To simplify things, you might just want to keep it in the Windows directory, as suggested earlier in the chapter on installation.

Two Kinds of Menu Entries

CT-Shell popup menu entries can contain two different kinds of commands: commands that run programs that are not part of CT-Shell (such as the programs that you use every day), and commands that are internal to CT-Shell. The latter are implemented using a special set of *keywords* that CT-Shell recognizes. Most of the entries in the sample CTSHELL.INI file use the CT-Shell keywords, since they will work exactly the same on everyone's computer.

However, there are a couple of entries in the sample CTSHELL.INI file that run external programs, as examples of that second kind of entry. You may already have used one of them - the menu entry that used NOTEPAD to edit your SYSTEM.INI file - if you made the modification to install CT-Shell as your Windows shell. Using the same techniques, you'll be able to run any of the programs you use on a day-to-day basis from the menu in CT-Shell, with much more versatility than you might imagine!

As you can see if you take a look at it, the external programs in the sample CTSHELL.INI file have been chosen as ones that every Windows user is likely to have, so they should also work on nearly all computers. Don't get the impression that you can run only Windows programs from within CT-Shell. You'll be able to make any program that you can run from within Windows a part of your menu, including DOS applications.[1]

Keep in mind that we're speaking in broad generalities in this Overview, and that you'll learn much more in later sections about how to add new entries to your menu. At this point, it is important for you to understand all that you'll be able to do with CT-Shell.

Besides running programs, CT-Shell makes it easy for you to change to any directory on your disk drive. There's an example provided that changes to your Windows "home" directory, and can take you there from anywhere else on your system. When you've learned to customize your installation, you might want to install an entry to take you to your root directory, to a word processing work directory, to where you work on spreadsheets, and so on. There's no practical limit to the number of entries you can have in such a menu!

Your CTSHELL.INI file is an ordinary ASCII text file, and you can modify it with nearly any editor or word processor, not just

with Windows NOTEPAD. In fact, your own personal editor is probably one of the very first things you'll want to add to your CT-Shell menu, so you can use it whenever you need to modify a text file.

CHAPTER 3 -The CT-SHELL Window

This chapter provides a tour of the main CT-Shell window, and the features that you'll see as you explore it. It is highly recommended that you start the program using the supplied sample CTSHELL.INI file, as all these features can be experienced even before you customize your system.

(Graphic not in shareware Write version of documentation.)

Note that everything you see above should look the same on your system, *except* the contents of the files list, the contents of the path window just above it, and the numbers in the status window in the lower left corner.

Menu

When CT-Shell is run, at the top of its main window is a menu that shows several items, such as *File*, *Edit*, *Dirs*, and more. This menu is based on the entries in the CTSHELL.INI file, and you will later want to revise the sample file to include your own program choices. As you add entries to your CTSHELL.INI file, those new options will appear in the CT-Shell menu, and you'll be able to select them from within the program.

This manual contains much more information about menu entries in later sections, with instructions about how you can customize yours. For now, just realize that you'll have almost unlimited freedom to create such a menu with selections that are perfect for your system and what you do with it. Most routine operations can become entries in your menu, so that a keypress or a mouse click is all that's needed to accomplish them.

In the menu that the sample initialization file creates, you'll find a number of things in the *File* menu that you can do with the current file (the one file that's selected with a dotted outline in the files list window), and a directory that you can change to in the *Dirs* menu. The *Edit* menu contains a number of choices, one of which you have probably used already to edit your SYSTEM.INI file. You can also edit your CTSHELL.INI file itself with the first entry in that menu, after which you can reload the modified menu by pressing <F7>.

The *Shells* menu offers an ordinary DOS session, and an additional CT-Shell window if you ever need a second one (or a

third one, for that matter). The menu labeled *Tagged Files* contains many of the same options that are provided for the current file in another menu, but this one applies those options to a whole list of files that you have tagged, rather than just the one current file. (The next section will cover file tagging in much greater detail.)

Utils is where you'll see some utility options. That would be a good place to add the utility programs that you use daily. The *Help* menu provides access to the CT-Shell help system, and to some other options that provide you with information about CT-Shell and about your system.

Function Keys

There are pre-assigned meanings for most of the function keys, but <F9> and <F10> have been kept available for the user to define. In the sample configuration file, they are assigned to *Solitaire* and *Configure*, as you will see when you edit that file.

Solitaire, of course, is that wonderful Windows diversion that so many of us need from time to time. It is an example of how you might configure one of your function keys to run an external program.

Configure will allow you to set all of the options that have to do with printing file listings. You may prefer to assign those two keys to different tasks, such as starting up a database or a spreadsheet program. Entries in your CTSHELL.INI file make that possible, as you'll see.

Much of what CT-Shell does with files can be done either with a current file or with a set of tagged files. When a file is tagged, its entry in the files list at the right is highlighted, letting you know that it has been selected for an operation. The first five function keys are devoted to managing those file tagging operations.

As you read this explanation of the function keys, feel free to try them out by tagging and untagging the files in your current directory. You won't cause anything to happen to those files just by doing that, and it's easier to understand the process if you see it happen, rather than just reading about it.

<F1> Toggle Current

Toggles the tagged/untagged condition of the current file. If, for some reason, you want to be sure that *no* files are tagged, you can turn OFF (and back ON) the current file by pressing <F1>. You might want to do something with all the files in the directory *except* the current file, for example. You could

do that by selecting the one file you want excluded from the command, then press <F2> to tag all the files, and finally press <F1> to untag the current file.

<F2> Tag All

Tags all the files (but not directories or drives) that are in the files listing to the right. You can perform any number of operations on a set of tagged files, such as to copy them all somewhere, delete them all, etc., and this keypress tags them all.

<F3> Untag All

Untags all the files, regardless of how many were tagged, or how they got that way.

<F4> Invert Tags

Inverts all the tags. You might want to tag some of the files in the current directory, copy those tagged files to a floppy disk in drive A:, then copy the rest of the files in that directory to somewhere else. <F4> will tag all the previously untagged files, and untag the ones that were tagged.
If that doesn't sound clear to you, drag the mouse part way down the list of files (with the left button held down) to tag a few of the files, then press <F4> several times and see what happens. You can finish your experiment with <F3>.

<F5> Tag By Name

Tags by name. If you want to copy all the .EXE and .COM files from the current directory to a floppy disk, you could press <F5> once, specify *.EXE when CT-Shell asks you for a wildcard spec, then press <F5> again and specify *.COM.

After you press <F5> you will be presented a dialog box (a question-and-answer panel) that prompts you to enter a filespec for tagging. That filespec may include the ordinary DOS wildcard characters, such as you would use to delete or copy certain files at the DOS prompt. Both the * and the ? wildcard characters work here as you would expect them to.[2]

<F6> Original Path

Returns you to the original path where CT-Shell was first started. As you work with the program, you will have many reasons to change to other drives and/or directories. <F6> will always return you to your starting point. Thus, you will want to consider starting the program originally in a "main" directory, such as the one in which you're working on a

current project.

If your AUTOEXEC.BAT file automatically starts Windows for you, you should consider having it do a CD (change directory) command just prior to starting Windows. Then, whether CT-Shell is started automatically via your SYSTEM.INI file, or you start it yourself from Windows, you'll always be able to press <F6> to return to that starting directory.

There is a CT-Shell keyword called HOME that lets you change the CT-Shell "home" directory to the current directory. That way, when you press <F6> you'll return to here, rather than the directory where you started CT-Shell.

This is a great function key to experiment with! See if you can find a way to change to a different subdirectory than the one you started this session in, and press <F6> to transport you instantly back. If you can't figure out how to do it just yet, don't get discouraged. We're coming to that part pretty soon.

<F7> Reload Menu

Reloads the menu. You can easily customize your CTSHELL.INI file with an ordinary text editor, as mentioned already. In fact, one of the entries in the default EDIT menu item uses the Windows NOTEPAD editor to change CTSHELL.INI. After you make your modifications, you can simply press <F7> to load the new version, without needing to exit CT-Shell and restart it.

<F8> Print File(s)

Will print a formatted and line-numbered listing of the current or tagged files. Note that this description uses the term *listing* as it is commonly used in computer jargon, to refer to a printed (hard-copy) version of the file contents. It does not mean a list of the files in the directory. If you would like a printed copy of your Windows xxx.INI files, for example, you might tag them all using <F5> and specifying *.INI as the file spec, then press <F8> to print them all - assuming, of course, that you have a printer connected to your computer. If you press <F8> without first tagging a set of files, the one current file will be printed.

Listings are formatted with a left margin that can be hole-punched. Lines of text can be numbered, as can pages, and at the top of each page can be a header that identifies the file, its creation time and date, and the time and date when it was printed. Refer to two keywords, PRINTER and CONFIG, which are explained in the CTSHELL.INI Reference

Chapter, for more information about setting up your printer driver and changing the format of your file listings.

<F9> and <F10>

Are reserved for the user. In the sample CTSHELL.INI file they are assigned to *Solitaire* and *Configure*, as mentioned above, but you can easily reassign them to other tasks that you want to be able to invoke with a keypress. These options are explained more fully in later sections of this documentation. [3]

<Esc> Parent

The escape key is used to change to the parent directory. So that the same operation is easy to accomplish with the mouse, the <Esc> key is represented on the screen along with the function keys.

The display of these keys at the left of the CT-Shell window allows you to click on a button with the mouse, to accomplish the same thing as pressing the keys themselves. Thus, whether you prefer using the keyboard or prefer using the mouse, you can have it your way.

Status Display

Below the listing of the function keys is a small window that displays the current time, the amount of RAM that is available (including virtual memory if you're running Windows in Enhanced 386 mode) and how much room is left on the current disk drive. The latter two measurements are displayed in megabytes, to the nearest hundredth.

Current Path

Just under the menu bar, and above the files display window, is the current path. As you navigate around your disk drive, you can glance here to discover quickly where you are. Watch this as you press <Esc> to move up in your directory tree, and as you press <F6> to return to your starting point.

You can also click the mouse on any part of the path that's displayed, and you'll change immediately to that directory. Thus, you can move upwards in the directory tree by pressing <Esc> to move to the parent directory, or jump directly to a directory that is more than one level higher, by clicking on it in the path display.

Directory changes made this way are "permanent" in the sense that CT-Shell will stay, and continue to work, in the new directory that you've chosen. However, you are still able to

press <F6> at any time to go directly to the original drive and directory where CT-Shell was started.

Files Window

The display of files contains considerable information that is always conveniently visible. One of the biggest advantages of a visual shell over an ordinary command line is that so much more information can be made available at all times.

Rather than trying to remember which file you came here to copy, you can *see* which file it was. You can tell this without needing to issue a DIR command, and unlike a DIR command, the file names here won't scroll past faster than you can read them. You can move both upwards and downwards in this list of files using the keyboard cursor keys, or by clicking the mouse on the scroll bar to the right of the window.

This section will explain the information that is displayed here, and tell you how you can change nearly all of it from within CT-Shell:

(Graphic not in shareware Write version of documentation.)

The largest window contains a display of the files in the current directory. Information displayed for files includes name, extension, size in bytes, last modified date, last modified time, and attributes.

Deleting Files

If you press <Del> , the current file or an entire set of tagged files can be deleted. You are prompted for confirmation before that happens, of course! There are additional deletion options, including a CT-Shell keyword called DELDIR. (See Chapter 4 for a reference to CT-Shell keywords that you can use in your menu entries.) This keyword is implemented in the sample CTSHELL.INI file in the *File* menu item, and called *Kill Dir*.

If you are doing disk maintenance and would like to delete an entire directory full of files (and possibly other subdirectories within it as well), first make sure that the current file is the directory you want to delete, then select this menu entry. You'll be asked to verify deletion of the directory with a dialog box like this, which is worded to get your attention:

(Graphic not in shareware Write version of documentation.)

If you click on the [NO] button, nothing will happen to the directory. And because this operation is so potentially disastrous if it is misused, even if you click on the [Yes] button you will be asked to verify one more time:

(Graphic not in shareware Write version of documentation.)

The sections that follow each describe one of the components of a line in the files display window, and what you can do with that information. In most cases, you are able to change it (except for the file size), and you'll find directions here for doing that. With a couple obvious exceptions (such as deleting a file or directory), you will probably want to try out the various keywords, commands, and menu options that are described here, as you read about the files window.

Name

Remember, directory names are displayed in uppercase, to distinguish them from file names. The filename extension, if any, is included in this field. In addition, following the directory and file listings, the name field will display the various disk drives that are available on the system.

If you would like to change the name of a file or a directory, you can easily do it with the CT-Shell RENAME keyword. (See Chapter 4 for a reference to CT-Shell keywords that you can use in your menu entries.) This keyword is implemented in the sample CTSHELL.INI file in the *File* menu item, and called *Rename*.

If you invoke this menu item, you will be prompted for a new name for the file:[4]

(Graphic not in shareware Write version of documentation.)

Size

The size in bytes of the file is shown here. You are also able to find out how many total bytes are included in a set of files that have been tagged, by using one of the special CT-Shell keywords, TAGGED, which is explained in a later chapter. This keyword is implemented in the sample CTSHELL.INI file in the *Tagged Files* menu item, and called *Files Tagged*. Selecting that entry while you have a number of files tagged presents a display that looks like this:

(Graphic not in shareware Write version of documentation.)

You are also able to discover how many subdirectories, files, and bytes a directory contains, using the CT-Shell DIRSIZE keyword. This keyword is implemented in the sample CTSHELL.INI file in the *File* menu item, and called *Size of Directory*. Selecting that entry while a directory is the current file presents a display that looks like this:[5]

(Graphic not in shareware Write version of documentation.)

Date

The date when the file was last modified (created or updated) is shown using the conventional mm/dd/yy format.

Both the time and the date for a file or a group of tagged files can be changed using the CT-Shell keyword, SETDATE. This keyword is implemented in the sample CTSHELL.INI file in the *Tagged Files* menu item, and called *Set File Date/Time*.

To change the date/time for any of the files in the current directory, first tag the one or more that you want to change, then select the menu item *Tagged Files*. Click on the entry named *Set File Date/Time* to open a dialog box that looks like this:

(Graphic not in shareware Write version of documentation.)

Input is checked to be sure that you are providing valid date and time values, and you will not be able to leave the dialog box with bad values in it by clicking on the [OK] button. If you want to abandon the operation, however, you can always click on the [Cancel] button. If you do that, no changes will be made.

Time

CT-Shell displays the file's creation time in its full resolution, which is to within two seconds. DOS displays only hours and minutes when you use its DIR command, although the number of seconds (to the nearest even number) are stored by DOS in the disk directory.[6]

The time applied to a file is of particular importance to programmers who work with a program maintenance utility called MAKE, or a variation of it. MAKE tests file date/timestamps to determine whether one type of file is newer than the type of file that is created from it, and rebuilds the target file if necessary. Sometimes it is important to give a file a date/time that is newer than another file, and there exists on many systems a small utility program whose only purpose is to change a file's date/time to the current date/time.

CT-Shell has two keywords that provide this service, called TOUCH (which changes only the current

file) and TTOUCH (to update a list of tagged files). They are both assigned to menu entries named Touch, in the appropriate popup menus in the sample CTSHELL.INI file.

Attributes

The file attributes are displayed as a series of characters which may include any of the letters *RHSDA*, for *R*ead/only, *H*idden, *S*ystem, *D*irectory, and *A*rchive, respectively. These attributes indicate that a file has certain properties, which may affect how you and DOS can access and use it. Following this listing of the attributes are directions showing how you can use CT-Shell to change file attributes.

Read/only

A file with the read/only attribute cannot be modified, overwritten or deleted. DOS simply won't allow the operation to happen, unless the read/only attribute is first removed.[7]

Hidden

Hidden means that a file won't show up in an ordinary DIR command from the DOS command processor, and the DOS COPY command won't copy a hidden file.[8]

Note that you can even use CT-Shell to hide an entire directory, so that others who use the same computer won't realize it's even there (unless they also use CT-Shell or another utility that displays hidden files). You can still change to the hidden directory, execute programs from it, and edit files in it - by specifying the directory name in your commands - yet it remains invisible to DOS.

System

System means the file is a special type which is part of DOS itself. Examples of this type of file include the two parts of DOS that you'll find in your root directory, named differently depending on which version of DOS you're using. CT-Shell will display these two files with the attribute letters RHS.. (or maybe just .HS.., depending on your version of DOS) showing that they have two or three of the attributes explained so far.

As an exercise, you might change to your root directory as you read this, and identify those files on your system. This is an attribute that you're not likely to assign to a file, unless you're a systems programmer who is writing a replacement for part of the operating system.

Still, you should know what it means, and you should be careful not to delete or accidentally damage any file that has the system attribute.

Directory

Directory makes the file a subdirectory, rather than a data file or a program. In the DOS system, subdirectories are special files that contain information about the files that are stored under them.

As is the case with other attributes, this one implies what can and can't be done with a file so identified. For example, you can change to a directory, but you can't change to a file. You can TYPE or DEL a file, but you can't do either with a directory.[9]

Archive

The archive attribute means that a file has been changed since the last time it was backed-up. Most backup programs, such as the DOS BACKUP command and commercial programs like CPBACKUP from Central Point Software, Inc., use this attribute to determine which files need to be processed when a differential backup is done.[10]

When you glance at your CT-Shell files display, you can easily see which files have been modified since your last backup. When a great number of files have the archive attribute displayed, or whenever particularly important ones do, you should begin to feel uncomfortable enough to do another backup!

Changing Attributes

Besides displaying the attributes, CT-Shell makes it easy for you to change most of them. You can't turn a program into a directory, but you might want to make a file read/only, for example, to prevent its being accidentally deleted or overwritten.

You can alter the attributes for a single file or for a group of tagged files if your CTSHELL.INI file contains a menu entry that uses the keyword ATTRIB (see the later section on CT-Shell keywords for more information about ATTRIB and other CT-Shell keywords). The sample CTSHELL.INI file contains entries for both the *File* and *Tagged Files* menus that implement this keyword.

When you invoke that menu entry, CT-Shell will present a dialog box like the one shown above, that lets you determine which attributes are to be turned on and which ones are to be

turned off.

This would be an excellent time to experiment with this feature. If you're in your Windows directory still, select the file 3270.TXT. Use the *Attributes* entry in the *File* menu to change that file to read/only status. Now select 3270.TXT again and press the <Delete> key to delete it. Go ahead and confirm the deletion, and see what happens.

Disk Drive Display

At the end of the files listing you'll find entries for all the disk drives in your system. Each is identified as to type and each (except floppies) has its current remaining capacity.

The capacity of floppy disks is not displayed, as those disks are removable, and Windows would report constant disk errors if CT-Shell kept trying to access empty drives. If you want to know how much room is left on a floppy disk, simply change to that drive by doubleclicking on its entry. You can easily change back to the current drive the same way afterwards, or by pressing <F6> to return to your starting drive/directory. CT-Shell always displays the free space on the current drive, even if it is a floppy disk, so you'll want to make sure there is a disk in the floppy drive before making that your current drive.

When you have changed your default drive to a floppy disk, by the way, that drive's activity light will remain on all the time, as CT-Shell checks once each second to remain aware of how much disk space there is. It will appear that your disk is constantly busy, but that's not actually the case. It's just that a floppy disk drive doesn't shut off until about two seconds after its last access.

If you access it once a second, as CT-Shell does to keep track of remaining space, no matter for how brief a time, the light will remain on. No harm comes from this. If you open the drive to change disks, you'll get an error message from Windows. Just click [Retry] when you're finished changing disks, and everything will be okay.

Extended Selections

CT-Shell's file window is programmed to allow extended selections. Thus, you'll find that you can tag multiple files by holding down the <Shift> or <Ctrl> keys as you tag with the mouse. <Shift> will allow you to extend a selection to include contiguous files (a group all together) and <Ctrl> will let you select any files, even if they are separated by others that you don't want tagged.

You can also mark a series of files using the keyboard. Press the key combination <Ctrl-Shift> and move the bar up or down with the keyboard cursor keys. As the highlight bar moves up or down, the files that it passes over will become tagged, just as if you'd dragged the mouse over them.

Doubleclicking Entries

Things happen when you doubleclick the mouse on an entry in the files list! (You may also move the highlight bar to an entry and press <Enter> to do the same.)
When you do either of these things, what happens will depend on what kind of file is selected:

Directories

If it is a directory, then you will change to that directory. This is another of CTShell's "permanent" directory changes, and it will continue to work in the new directory until you change again. However, you are still able to return to your original startup directory by pressing <F6> at any time.

CT-Shell lists all the directories first in the file list, to make it easier to travel around your drive by clicking on directory entries. Be reminded that you can use the <Esc> key to change to the parent directory at any time, and that you can click the mouse in the path window (just above the files list) to change to any place in the path above this directory, up to the root directory.

Executable Files

If it is an executable file (to CT-Shell, that means .EXE, .COM, .PIF or .BAT) you will execute that file. This provides a convenient way to run programs that are in the current directory, and which require no command-line arguments. Those that reside elsewhere, and those that do require command-line arguments should be installed as menu entries instead. See the later sections for more information about setting up your menu to run programs.

DOS programs (DosApps) that have a .PIF file (Program Information File) available somewhere in the path will be executed according to that .PIF file, so if you need to customize the way your DosApps run, simply create a .PIF and keep it available in a directory that's part of your DOS executable path.[11]
You can use the Windows utility program named PIFEDIT.EXE to create and edit .PIF files, and your Windows manual contains more information about these files and

how to modify them.

Programs that were designed to be run under Windows (WinApps), are executed just as they would be from the Windows Program Manager or the Windows File Manager.

Known Extensions

CT-Shell checks the [Extensions] profiles in your WIN.INI file, and can "run" files that are not themselves executable, but for which you have provided an extension in your WIN.INI file. Thus, it is likely that if you doubleclick the mouse on a .WRI file, you'll start up Windows Write and can edit that file. If you doubleclick on a .CRD file, you'll start up the Cardfile database program, etc.[12]

Drive Specifications

If it's one of the entries at the end of the files list that describes a disk drive in your system, doubleclicking on it will change to that drive, which will then become the default, or current, drive. As in many earlier examples, CT-Shell will continue to work in the new drive/directory until you change away from it, but you can instantly return to your original startup directory by pressing <F6>.

Command Line

An extremely important CT-Shell feature is its command line, with the only visible manifestation of that feature in the main window a button named [Command Line] that appears just below the function keys to the left. To access the command line without using the mouse, simply press the key combination <Shift-Enter>. Here are some of the important features of the CT-Shell command line:

DOS Commands

Most common DOS commands like CD, RD, MD, COPY, and DEL are handled internally in CT-Shell, without using the DOS command processor at all. Most of the DOS commands that CT-Shell handles offer an enhancement over their DOS counterparts. For example, the CD command will allow you to specify a drive as well as a directory to change to. The COPY command uses a buffer up to 16 times the size of the one DOS uses, allowing many files to be copied with only one disk read and one disk write, for better efficiency.

If you enter a DOS command that CT-Shell cannot handle

itself, it will pass that command along to your DOS command processor for evaluation. Fortunately, all the most-often-used DOS commands can be dealt with smoothly within CT-Shell, without involving the DOS command processor at all.

Most users will probably prefer to change working drives by first scrolling to the listing of drives at the bottom of the files list and then selecting a drive using the mouse or the keyboard. However, you are also free to enter a drive letter, followed by a colon, as a command on the CT-Shell command line, just as you would at a DOS command line. Such a drive change is handled internally by CT-Shell.

CT-SHELL Commands

Some additional CT-Shell commands may be issued from this command line as well:

Find

It often happens that someone needs to edit one of the many text files that are part of a programming or word processing project, and can't remember for sure which file contains the text. The FIND keyword is designed to find a string (of characters) wherever it may occur within any of the files in the current directory.

Use quotation marks to enclose strings that include embedded spaces. This example shows how you might look through the current directory to find the places where you have used the function named "wsprintf":

(Graphic not in shareware Write version of documentation.)

Note that the command itself (FIND) is not case-sensitive, and may be entered in uppercase or lowercase. However, the string that is being sought *is* case sensitive. You'll want to be sure your <CapsLock> is not on when you look for a string that contains lowercase letters!

If that string is found in any of the files in the current directory, a list box like the one shown just below will be created that contains all the matches that were found. All leading spaces are removed from the lines before adding them to the list box, so that you can view more of the significant parts of that line. Up to the first 75 characters are included (starting with the first non-space character), which should be enough to help you verify whether that line is the one you're looking for. The entries are alphabetical according to file name, and in line-number

order within a file.

One of CT-Shell's most useful features is available to you at this point: if you select one of the entries (using either the mouse or keyboard methods) that file will be loaded into your editor automatically. Even better than that, if your editor is one that will accept a line number on the command line along with the file name, you can even load the file and jump directly to the line that contains the string you asked CT-Shell to find!

For you to edit one of the files that was found this way, your CTSHELL.INI file must have an entry in its [EDITOR] section called EDITORNAME that identifies your editor by name, so CT-Shell will know what program to run. The sample CTSHELL.INI file contains:

```
EDITORNAME=NOTEPAD.EXE
```

as a default, but most serious programmers will prefer to change that name to another editor.

For CT-Shell to be able to load the file *and* jump directly to the line where the string was found, your editor must be capable of such a feat in the first place, and you must also include an entry in the [EDITOR] section of CTSHELL.INI called EDITLINE that shows how to give such a command to your editor. The sample CTSHELL.INI contains:

```
EDITLINE=
```

which disables the feature, since the default NOTEPAD.EXE editor can't handle line numbers in this manner. There are more details and examples in Chapter 4 of this manual, which is the reference to the entries in CTSHELL.INI.

Move

If you want to move a file quickly from one place to another, rather than copying it, you can use the CT-Shell MOVE command. The syntax is just like the ordinary COPY command, but the move is much faster.

Where

If you want to know where a file is on the disk, you can use the WHERE command. Start the command line, then issue the command like this:

(Graphic not in shareware Write version of documentation.)

where you type in the name of the file you want to find and its extension, if any. The customary DOS wildcard characters are acceptable here, so you could search for files such as:

where *.dbf

or

where copy??.bak

If you want to locate all the files with a given file name and *any* extension, you can simply enter it as FILENAME. If there is no dot in the name that's entered (and thus no extension has been used) CT-Shell will automatically append the .* to the name. If you actually do want to locate a file named FILENAME that does not have an extension, you can provide just a dot for an extension. That tells CT-Shell not to add the .* to the end.

After you've provided the name, the mouse cursor will change to an hourglass temporarily, letting you know that CT-Shell is busy as it searches the current disk drive for that file. CT-Shell will display information in a list box about all the matching files that it finds:

(Graphic not in shareware Write version of documentation.)

You even have the option to select one of those entries and go straight to that directory. Just doubleclick on the entry, or select it with a single mouse click and afterwards click on the [OK] button. Alternatively, you can select it with the keyboard cursor keys and press <Enter> to complete the command. If you have selected an entry and decide afterwards *not* to change to that directory, simply click on the [Cancel] button.

There is some potential for confusion where the FIND and WHERE commands are concerned. After all, you might think to use FIND to find a file, and WHERE to locate where a string is. About the only way to keep them straight is to remember that the DOS command FIND looks for a string within a group of files, and CT-Shell's FIND command does the same.

Also, there have been a number of public domain utility

programs developed over the years that are named WHERE or WHEREIS, and are used to locate files on a drive, as does the CT-Shell WHERE command.

Command Recall

CT-Shell maintains an internal doubly-linked list of previous commands, and lets you scroll through them to select a command to issue again. Each command that you type at the command line is added to the list, and there are three options for deleting old commands that you no longer want to scroll through.

(Graphic not in shareware Write version of documentation.)

After one command has been given at the command line, you'll see a [Delete] button the next time you invoke the command line. That will allow you to delete the earlier command. Perhaps you misspelled a file name, and the command wasn't successful, and you don't want to accidentally issue that same command again.

After more than one command has been issued, you'll see options that let you delete from the current command upwards, from the current command downwards, or just the current command itself. The default is always to delete just the current command, so you don't accidentally remove several that you'd like to use again later.

Whenever you start a CT-Shell command line, you can use the <Up> and the <Down> keyboard cursor keys to scroll through the list of past commands. When you've found the one you want to use, you can press <Enter> to accept it, or click the [OK] button with your mouse. If you prefer to select from the list itself, just click the mouse on the small downwards-pointing arrow to the right of the command line itself. That will open the associated list box, showing you any existing commands that are available to be reused. If there are none, the box will be empty.

CHAPTER 4 - CTSHELL.INI Reference

Some of the power of CT-Shell isn't unleashed until you make a few simple modifications to your CTSHELL.INI file, to customize it for your system and for the way you work. This chapter describes the various areas in that file that can be modified by the user.

Options are recorded there for future sessions, and all the special menu entries that you create are stored there. As long as CTSHELL.INI is stored in the current directory, a directory along your DOS path, or in your Windows "home" directory, CT-Shell will be able to find it when it needs the file. It is suggested that you keep CTSHELL.INI in your Windows home directory with your other important xxx.INI files.

Far from the programming required to customize some similar products, CT-Shell lets you work with simple objects (characters like ! and #) that take on special values in your commands. Even those who have never written a program, a script, or a macro are encouraged to have a shot at this! It's really VERY easy to do. Here's what you need to know about the various sections in your CTSHELL.INI file:

Accessing CTSHELL.INI

Your CTSHELL.INI file is an ordinary ASCII text file that can be edited with nearly any editor or word processor. Although it isn't a powerful editor, the Windows NOTEPAD editor is fine for the light-duty work of customizing your initialization file.

If you followed earlier recommendations, your CTSHELL.INI file is probably stored in your Windows directory. If that's the case, you'll be able to edit it quite easily using the *CTSHELL.INI File* entry from your *Edit* menu. If you have stored it somewhere else, simply change to that directory, select the file, and select *Notepad Editor* from the same menu.

In short, do whatever's necessary to edit your file and save the new copy. You can put your menu entry changes into effect afterwards by pressing the <F7> function key, or by exiting and restarting CT-Shell.

The following subsections describe the five parts of your CTSHELL.INI file, and what changes you might want to make to each of them:

Editor

The first section is marked [EDITOR] and contains two settings that tell CT-Shell whether you have a text editor and whether it has a particular capability. These settings are used in conjunction with the FIND command, and will allow you to edit the file that contains a string of characters that you have asked CT-Shell to find for you. These settings look like this in the sample CTSHELL.INI file:

```
[EDITOR]
EDITORNAME=NOTEPAD.EXE
EDITLINE=
```

On the assumption that anyone who has Windows has the NOTEPAD editor, that's the default, even if this section is missing from the CTSHELL.INI file. Most programmers use more of a heavy-duty text editor however, and will want to change this entry to contain that editor name instead (including its path, if necessary).

The EDITLINE entry tells CT-Shell two things about your editor: whether it can start at a line number that is included as part of its command, and if so, what command-line switch is used to invoke that feature. If EDITLINE is left empty as in the sample file, CT-Shell will simply load the selected file into the editor, but not attempt to start at a particular line number. If EDITLINE contains any characters, it will be added to the edit command just before the line number.

Here's an example that would work for the popular QEdit programming editor (from SemWare, Inc.), which has an executable file named Q.EXE and which uses the switch -n to tell it what line number to start on:

```
[EDITOR]
EDITORNAME=Q.EXE
EDITLINE=-n
```

When CT-Shell puts together a command to execute your editor, the actual file name and line number are combined with the editor name and switch. Assuming a file called FILE.EXT and assuming that the desired string was found in line 123 of that file, the command that CT-Shell would create from all this would look like:

```
Q.EXE FILE.EXT -n123
```

If your editor does not offer a way to start on a specified line,

just leave EDITLINE blank in your CTSHELL.INI file. If you edit your CTSHELL.INI file to change these settings, you may use the <F7> function key to reload your menu, which will also cause these editor settings to be reloaded from the file.

Options

The [OPTIONS] section provides a place to tell CT-Shell which disk drives you want it to ignore as it creates the file listings. It can be convenient for all available drives to be shown in the listing, as their free capacity is displayed, and the user can doubleclick on their entries to change to those drives. However, each time the directory listing is refreshed, those drives are checked and their information updated if it has changed.

That takes time, and if there are drives that you do not frequently change to, and don't care about their remaining capacity, you can speed up general operations considerably by telling CT-Shell to ignore them.

You do that with an IgnoreDrives= entry in the [OPTIONS] section of your
CTSHELL.INI file. This example shows how you would limit the drives to just A, B, C, D and E, in a system that has many more available:

```
[OPTIONS]
IgnoreDrives=FGHIJKLMNOPQRSTUVWXYZ
```

The drive letters do not need to be in any particular order, though keeping them
alphabetical makes sense, considering that you may from time to time add or delete certain letters. In fact, you can easily change this configuration option during a CT Shell session, by editing your CTSHELL.INI file and pressing <F7> to reload your menu and options.

If your network operating system allows the use of other symbols, such as
numeric digits, to be used as drive designators, you may include them in the list as well. Most systems allow the use of drive letters only.

It is important to note that the use of this option in no way prevents you from
changing to another avaiable drive, even if you have told CT-Shell to ignore that drive. You may issue a drive-change command from the CT-Shell command line as usual, and that

drive will become the default drive. Its remaining capacity will be displayed in the status window. For example, to change to drive L, which has been excluded by the option above, you could enter the command:

L: <enter>

at the CT-Shell command line, as always. Asking CT-Shell to ignore this list of drives simply prevents it from displaying entries in the files list for them, it doesn't affect your using them in any way.

This section also contains two complete commands for user options that CT-Shell will execute when the user presses the <F9> and <F10> function keys, or clicks the mouse on their representation on the screen. You'll need to be sure there are 10 sets of braces here. They may be left empty if you prefer, but they must be in the file, or CT-Shell may become confused by the missing fields.

Here are the entries from the [OPTIONS] section in the supplied sample CTSHELL.INI file:

```
[OPTIONS]
user1 {Configure} {} {} {} {config}
user2 {Solitaire} {} {sol.exe} {} {}
```

As you might be able to tell from this, the user1 task is assigned to execute a special CT-Shell keyword called "config," which will display the file listing options you currently have in effect and let you change them. The user2 task is assigned to run the Windows Solitaire game. The format of these user entries is the same as for the menu entries, which are described later in this chapter.

These commands begin with a name to display on the screen, such as "Configure" and "Solitaire" as shown here. It can be followed by the name of a directory to change to, a program name, and any switches or other options needed when it is run, or alternatively a CT-Shell keyword in the last field. If you want, you can use some special characters here to include additional information in the final command, such as the name of the current file, the names of all the tagged files, the contents of an environment variable, or a value provided by the user in response to a prompt.

Color

The [COLOR] section allows you to control the color of the main window background in CT-Shell. There are three color components that you can set individually-red, green and blue-that together make up the one window background color.

Those colors can be assigned values from 0 to 255, and you'll probably be surprised at some of the combinations that can result. Would you have believed that red and green together create yellow? The sample CTSHELL.INI file has a [COLOR] section that looks like this:

```
[COLOR]
BKRED=0
BKGREEN=255
BKBLUE=255
```

That produces a blue-green (cyan) background color that many people consider pleasant. You will find a command in your *Utils* menu that invokes the special CT-Shell keyword COLOR. That command will reset the colors after you've edited them here, without reloading anything else from the initialization file. Alternatively, you can restart CT-Shell.

Printer

The [PRINTER] section controls certain options that affect the way CT-Shell prints a file (or a list of tagged files) when you press the <F8> key. It looks like this in the sample CTSHELL.INI file that is supplied:

```
[PRINTER]
LINESIZE=80
HEADINGS=1
PAGENUMS=1
24HOUR=0
LINENUMS=1
INDEPENDENT=0
DRAFT=0
TEXTFIXED=1
```

All of these settings are controlled from within CT-Shell, from a dialog box that is presented when you execute a popup menu entry that uses the CT-Shell keyword CONFIG (which is described later in this chapter). That entry is available in the *Utils* menu item in the sample configuration. When you invoke

the keyword CONFIG, here's the dialog box that CT-Shell uses to get your choices:

(Graphic not in shareware Write version of documentation.)

If you click on the [Accept] button, CT-Shell will accept the settings that are shown. If you have checked the box marked "Save settings to CTSHELL.INI file," that will be done as well, so that next time you'll start off with these as default settings.

If you click on the [Reset] button, you will cause CT-Shell to read in the current settings from the CTSHELL.INI file. They will replace whatever other settings you had in effect, and will be displayed immediately.

As usual, if you click on the [Cancel] button, you'll exit from this dialog box without changing anything. Although you don't need to edit your CTSHELL.INI file to change these, here's an explanation of what each one means:

Linesize

The LINESIZE entry will be 80, 110 or 132, if it was entered from within CT-Shell, and it specifies the width of text file lines, as you usually work with them. When printing text files, CT-Shell will choose the largest font that is available for your printer that will display at least that many characters on a single line, in addition to allowing room for borders and optional line numbers.

For example, if you write programs, and always make sure that your source file lines are 80 characters or less in length, you can select a line size of 80 and know that your printed listings will contain all your text between the borders. If you occasionally write on past the width of your terminal, you might want to select 110 to ensure that everything will print. CT-Shell does not provide linewrap, so lines of text that are too long will be truncated at the edge of your printing area.

Some people use special video hardware that provides them with 132-column text displays. They may use that full width when editing programs, so a selection is available for that size as well.

Although these three sizes are the most useful, and are provided for easy selection from within CT-Shell, you may edit this entry to contain values other than 80, 110 or 132. CT-Shell will attempt to use your value, providing the closest font that it can.

Note that there is another special CT-Shell keyword

called PRINTER, which is available from your *Utils* menu, that will invoke the setup function from the Windows printer driver for your printer. Depending on the type that you use, you may be able to change paper size, change orientation from portrait to landscape, and change other settings that will also be reflected in CT-Shell's choice of fonts for your listings. As an example, the next illustration shows what you'd see if you happen to use a Hewlett-Packard LaserJet Series II printer, and happen to have it assigned to LPT1.OS2.

(Graphic not in shareware Write version of documentation.)

One other issue related to line size and fonts is your printer's resolution. Often people are able to specify a graphic resolution that is less than the maximum possible, thereby speeding up printing of program listings, with an acceptable loss of print quality. If your CT-Shell listings take too long to print, and are of unnecessarily high quality, explore your options in the printer settings. With a LaserJet, for example, you might want to select a resolution of 150 dots per inch. That will still provide crisp, readable listings, but they will print *considerably* faster than if the resolution were left at 300 dots per inch.

The rest of the options shown here are simply TRUE/FALSE values, where the number 1 represents TRUE and 0 represents FALSE. Inside CT-Shell, your selections will be made by checking boxes for the options that you want, however in the CTSHELL.INI file, your choices are stored as 1s and 0s.

Headings

HEADINGS determines whether name/date/time headings will be printed at the top of your listings. If you enable this option, each file's name and creation date and time will be printed at the top left of the page in a bold font, and the date and time the listing was printed will be at the top right of the page. Thus it will be easy to compare two listings to see which is newer. You can print a few additional lines of text on each page if you turn off this option.

Pagenums

PAGENUMS determines whether your listings will have page numbers at the bottom of each page. Be sure to see the closely related INDEPENDENT option below. You can print a few additional lines of text on each page if you turn off this

option.

24Hour

This option lets you determine whether the times displayed in file listings will be in 24-hour "military" time, or in the conventional AM and PM format. With this option enabled, 9:30 in the evening displays as 21:30, and with this option disabled the same time displays as 09:30p.

Linenums

If the file being printed is a program listing, chances are you'll want the lines to be numbered. This option will cause them to be numbered from 1 to 99999, and separated from the text with a > and a space. Thus, such a listing might look in part like this:

```
51>       if( iLimit > iValue)
52>          foobar( iValue );
```

Turning the line numbering option off would make more sense when printing a listing of a program documentation file.

Independent

The INDEPENDENT option refers to page-numbering for multiple-file printing jobs. If you have a series of files tagged before you press <F8>, all of them will be printed, not just the current file. If INDEPENDENT is set to TRUE, each file listing will begin with page 1. If INDEPENDENT is set to FALSE, the whole series of files will be page-numbered consecutively, straight through.

If a number of files is all part of a single programming project, you might prefer setting INDEPENDENT to off.

DRAFT

DRAFT tells Windows whether to try to find a font for high-quality output or one that will print more quickly, with lower quality. If DRAFT is TRUE, lower quality will be allowed.

Note that the operational word here is *allowed*. CT-Shell does not force Windows to use a lower-quality font, it can only allow it to do so. How much effect this switch has may well depend on the kind of printer you use, and the number of fonts its driver is able to make available.

Depending on the printer type, you may have more success in changing to a lower quality (and faster printing) font if you use the CT-Shell keyword PRINTER in a menu

entry, to gain access to the printer driver's setup function. By setting the graphic resolution to a value that is less than the maximum, you may be able to trade some excess print quality for a desired increase in printing speed.

TEXTFIXED

Whether the text portion of your printout is printed in a fixed font or a variable (proportional) font is controlled by this one. If you need to print program listings, you'll want to set TEXTFIXED to true, so that spacing is preserved in your listings.
(This feature does not affect the headings, as there is no reason to print headings using a fixed font.)

Under other circumstances, you might prefer to turn off this feature, so that proportional spacing will be used instead. In particular, you might want to turn this feature off to print files that were created by a word processor using a proportional font.

Items

This part of your CTSHELL.INI file determines the contents for your menu, and the popup menus that its items invoke. There is no practical limit to the number of menu items, although most people prefer to keep the number small enough (and the item names short enough) to make the menu fit on one line. Likewise, there is no practical limit to the number of entries that each menu item may contain.

Since the sample CTSHELL.INI file provides so many examples of menu entries, you shouldn't have any trouble at all adding the ones you need to customize your system. A good idea is to add one or two new entries that will run programs you use often, and try them out. Make sure you're including the current file in the right place, and that you're changing to the right directory before executing the programs.

While reading through these following subsections, you should get plenty of ideas for custom entries. Ask yourself questions like, "What do I use the computer for most of the time..." and think what you'd like to be able to do with a click of the mouse or the press of a couple keys.

Menu Items

Each menu item is distinguished by the special word *Item* that appears first on its line, then a set of braces containing the item name as it should appear in the main menu. Since

braces are used as delimiters, the menu entries can contain quotation marks, if you want, as well as spaces and punctuation.

item {ItemName}

ItemName

The menu item name may contain embedded spaces, and it may contain the special ampersand character (&), which determines a letter that will appear underlined in the menu itself. So identifying a key letter in the name provides a way for that menu item to be selected with a combination of the <Alt> key and that underlined letter.

For example, <Alt+E> typed together would activate the menu item that was described in the CTSHELL.INI file as {&Edit}. The ampersand is optional, but provides a quicker way to invoke this item.

Popup Entries

Each popup entry contains five fields, delimited by braces, of which only the first is required.

If you haven't yet loaded your CTSHELL.INI file into your editor to take a look at it, you should do so now. The following descriptions will be most meaningful if you're looking at the sample menu entries as you read about their various parts. These fields determine what action is to be taken when the user selects that entry from a popup menu:

{EntryName} {DirPath} {ExePath} {Switches} {Keyword}

Entry Name

The entry name is displayed in the popup menu to allow the selection of this option. Like the menu item name, the entry name may contain an ampersand character to determine the character that will be underlined in the menu, thus providing easy access to this item with a keyboard command. The ampersand is optional, but provides a quicker way to invoke this entry.

Two other special characters may be used here, to provide separation
from other popup menu entries. If an entry name begins with a hyphen (-), there will be a horizontal bar in the menu, separating that item from the ones that preceded. If there is a plus sign (+) before the name, that entry will

begin a new column in the popup menu, with a vertical bar separating it from the preceding entries.

If you visualize the hyphen as a horizontal bar, and the plus sign as a vertical bar that separates two halves of something, it should be easy to remember these. There are also a couple of examples in the sample CTSHELL.INI file. The Exit entry in the File menu is separated at the bottom with a horizontal bar, as has become customary for Microsoft products, and in the Win menu, the two entries that allow you to edit Windows xxx.INI files are separated into their own column, distinguishing them from the Windows utility programs in the other column.

DirPath

The directory path is an optional field which, when provided, causes CT-Shell to change either temporarily or permanently to that directory.

If a directory path field is present and an executable path field is not, it is assumed that the purpose of this entry is to change directories permanently. In that case, CT-Shell will change to the specified directory, and will begin operating there. An example might look like this, where you want to be able to change to a word processing work directory on drive D:

```
{&WordProc} {d:\winword\letters\personal} { } { } { }
```

Since the directory field has a content, CT-Shell will change to that directory when this entry is invoked. Since the executable field does not have any content, the directory change will be a permanent one.

If both a directory path field and an executable path field are provided, it is assumed that the directory change should be temporary, for the purpose of executing the command only. Afterwards, CT-Shell will return to the directory where it was before the command was executed. Here's an example where the same directory change is made, but a session with Word for Windows is also started:

```
{&WordProc} {d:\winword\letters\prsnl} {winword.exe}
{ } { }
```

This time the directory change will be temporary, and CT-Shell will return to the previous directory when the session with Word for Windows is finished.

ExePath

As you saw in the previous example, the executable path is the path name for an executable file which is to be run when this entry is selected. Because the menu entries are processed by a command processor that can look throughout the DOS path for an executable file, programs whose names end in .EXE and that reside along the DOS path may be listed here without any qualifying path information.

However, if the program to be executed does not reside along the DOS path, you must include the entire path/file name here. If it has an extension of .COM or .PIF or .BAT, you must at least include the extension - even if the file does reside along the DOS path - as CT-Shell will default to .EXE for a filename with no extension. More information about path names and the DOS path is available in your DOS manual.

An example of an executable that lies along your DOS path is CALC.EXE, assuming that it is in its customary place in your Windows directory. Here's a menu entry that would run the Windows calculator utility:

```
{&Calculator} { } {calc.exe} { } { }
```

No directory change was required, and in fact, we could have gotten away without the .EXE extension, since that's the default. Many people prefer to include it anyway, for purposes of documentation.

Switches

Programs often require additional information on the command line when they are run. An editor, for example, can often be told what file to edit by including the file name as part of the editor command.

Sometimes too, the way a program runs can be affected by switches that turn on or off certain features. For example, the extended copy XCOPY command from DOS will copy entire directories if you follow the command with the switch "/s" and will even include empty directories if you include the switch "/e". If a menu entry were created to execute the XCOPY command, you might want to include

these switches as part of the entry.

The switches field is the one in which the CT-Shell special field characters are most often used (see below). Using object-oriented techniques, you are able to include the current file as part of your command, a list of all the tagged files, an environment variable, and more. You are even able to cause CT-Shell to prompt you for one or more values to be inserted as it runs.

As it operates, CT-Shell combines the switches field with the executable path field to form a command. Thus, it doesn't really matter whether a command argument or switch occurs in one field or the other. However, it is easier to visualize executable programs as separate from the arguments and switches that are used with them, so both fields are provided. If you prefer, you may put all the necessary entries into the executable path field and leave the switches field empty, but its braces must be left in place or, under some circumstances, CT-Shell might become confused by the missing field.

Here's an example for an XCOPY command that assumes the current file is a directory, and copies it and everything in it to the floppy disk in drive A:. Note that the ! represents the current file, and can be placed in the command right where the current file name would be placed if the command were being given at a DOS command line:

```
{&CopyDir} { } {xcopy.exe} {! a: /s /e} { }
```

Keyword

Many of the operations that CT-Shell performs are handled internally by CT-Shell itself. That's how it is able to improve on many of the DOS commands, rather than passing the commands along to DOS. A keyword command is always used in place of - rather than in addition to - an executable path. In fact, if there is a keyword in this last field, CT-Shell will process that keyword first, ignoring any entries that may be in the executable path or switches fields.

CT-Shell keywords are listed here, along with a brief description of what they each accomplish. Note that there are a few keywords that are duplicated for use with single and with tagged files. If you tag a group of files, for example, and select *Delete* from the *File* menu, you'll only delete the one current file. If you do the same and select *Delete* from the *Tagged Files* menu, you'll delete them all.

Examples of all of these keywords are already included in menu items in the sample CTSHELL.INI file, however you may want to rearrange them to suit you:

About
Displays information about CT-Shell, including the number of the version that you're using.

Attrib
Changes file attributes. Use this keyword in a menu item to let you change the attributes of the current file. There is another version listed below that changes the attributes for a group of tagged files.

Color
Resets the background color based on three color values stored in your CTSHELL.INI file. You need to modify that file first, changing the values for the background color, then execute a menu entry that contains this keyword to read those values from the file.

Command
Invokes the command processor that is associated with your COMSPEC environment variable. In most cases this will be COMMAND.COM, the command processor that is supplied with MS-DOS and PC-DOS.[13]

If there is anything at all that you prefer doing at an ordinary DOS command line rather than from within CT-Shell, this keyword provides you with the ordinary DOS session where you can do it.

Config
Configures the file listing options. These are the settings within CT-Shell that affect how file listings are printed. See also the PRINTER keyword for access to the printer driver itself, which can provide even more control over your installed printer.

Copy
Copies a file to another location. You are prompted for a destination for the current file, and it will be copied to that destination. Like the DOS copy command, you may supply a file name or a directory as a destination. Like the COPY command that you use at the CT-Shell command line, this uses a much larger copy buffer than

DOS does, for better efficiency.

Deldir

Deletes the currently-selected directory and all files in it. Be careful! This one is so powerful that there are *two* confirmations necessary to make it work (you're asked *twice* whether it's okay to delete the directory).

The entire subdirectory will be deleted, *including any files in it and any subdirectories under it*, even files that have the read/only attribute. This is a wonderful way to remove an outdated or unwanted directory during disk maintenance, but it requires you to be careful. Files and directories that have been deleted with this command cannot usually be undeleted.[14]

Delete

Deletes a file. This removes a file in a way that cannot usually be reversed. Be careful, and be sure that you mean it when you use this keyword. You are asked only once for confirmation.

DirSize

Displays a listing that shows you how big a directory is. It shows how many subdirectories it contains, how many files, and how many bytes they all add up to. This can give you a very good approximation of how much room must be available on a destination to which you plan to copy that directory, or how much additional room will become available on your drive if you delete it.

Exit

Shuts down CT-Shell. You can also do this by double-clicking on the system menu box in the upper left corner of CT-Shell's window, but some people find it easier to pick an exit command out of a menu. One other difference is that the menu entry does not ask for confirmation, giving you a choice as to how you prefer to handle it.

If you like being asked to confirm exiting, teach yourself to doubleclick the system menu to leave. If you prefer to bail out without having to find the [OK] button to verify your intentions, learn to use the *Exit* entry in the *File* menu instead.

Help

Runs the Windows help engine. This provides access to the online helpfile that explains what all these options do, and reminds users how to use CT-Shell. You can start at the index, and select the topic you want to review. Wherever feasible, the help file contains hyperlinks to other topics, making it easy for you to find all the information related to a subject.

Home

Changes the CT-Shell "home" directory to the current directory, so here is where you'll return when you press <F6>, rather than the directory in which you started the program.

Move

Moves a file to another location. This feature changes the directory information relative to a file without copying the file itself. Thus, a move takes only part of a second, no matter how big the file is that is being moved. No file data needs to be read or written, just the directory entry for that file.

Printer

Invokes your printer driver setup function. The exact set of features and options that are offered by this function depends on the printer driver for your particular printer. This is probably where you can change from portrait to landscape mode, determine how high your graphic resolution should be, download font software, etc.

Rename

Changes the name of a file or directory. This is actually implemented as the same low-level DOS function that MOVEs a file, and it can be used for the same purpose. If you provide a new pathname that includes a different directory than the current directory, your file will be not only renamed, but moved to that directory as well.

System

Displays system information. This is the same information you can get from the Windows Program Manager by clicking on its HELP/ABOUT option. You can find out what mode you are running, using what kind of processor and coprocessor (if any), and whether small-frame or large-frame EMS operation is in effect (if any).

This keyword does not display the amount of memory available, as CT-Shell displays that at all times anyway.

Tagged
Displays the number and size of tagged files. If you have tagged a set of files to be copied to a floppy disk, you might want to check to be sure that the number of bytes tagged does not exceed the number of bytes that are free on your disk.
Because of the way disks are sectored, you will actually need a bit more room than the number of bytes that are tagged, but you'll never need less room. Use the value provided here as an approximation.

Here are documented the special keyword versions that work with a group of tagged files, instead of just the current file. Note that any of these could be used to handle a single current file (unless it were explicitly untagged with <F1>), but the reverse is not true:

SetDate
Changes the date/time stamp that DOS has applied to a file or a set of files. This operation is easily reversed if an error is made, so only one version of this keyword is needed - one that will work for tagged files. Whenever this keyword is used, the new date and time that the user provides will be applied to all the tagged files. If you want to change the date/time for a single file, simply ensure that it's the only file that is tagged.[15]

Tattrib
Changes the attributes of tagged files. If you should want to change all the .EXE and .COM files in a directory to read/only status, to prevent unnecessary share violations with a network, you could tag those files, then use this keyword to give them all a read/only attribute.

Tcopy
Copies a set of files to another location. You must provide a directory as the destination. CT-Shell does not support file concatenation (combining several files into one) by copying multiple files to a single file.

Tdelete
Deletes a set of tagged files. You are prompted for

confirmation before the deletion is accomplished.

Tmove
Moves a set of tagged files to another location. Just as with the single file move keyword, these files are not physically copied to their new location, just their directory entries are changed.

These keywords have been provided so that you can have complete control over how your menus are crafted, rather than having CT-Shell contain a fixed menu that determines how you must access these features. For example, one user might think it makes good sense to have COPY and MOVE in a menu named *Utils*, whereas someone else might think they belong in one named *Claudia*..

These keywords are not case-sensitive: uppercase and lowercase work the same way.

Special Field Characters

Many times a command should contain the name of the current file, or a list of all the tagged files, or other additional information. Sometimes it is convenient to refer to an environment variable in the directory path field, so that the command doesn't need to be changed just because the directory has been changed. There are a number of special field characters that allow you to insert such information into a command in a convenient object-oriented manner.

The term "object-oriented" here means that you do not need to write special code using a programming language, or call a function or procedure to do these things. Certain objects (characters like ! and #) that you place in the command automatically take on values that represent file names or other information.

Here is a listing of all the special field characters that may be used in CT-Shell popup menu entries. Although any of them may be used in any of the fields except the entry name field and the keyword field, you will find that certain ones are likely to be used in the directory path field, and other ones are more likely to be of use in the executable path and switches fields.

In each of the following examples, a DOS command line is shown, to illustrate how the command would look if it were entered normally at a DOS prompt. After that, the CTSHELL.INI entry is shown that would produce that

command, substituting current information for the CT-Shell special characters:

!

The exclamation mark translates into the current file name. Here's an example that would use the Windows NOTEPAD.EXE editor to edit the current file:

Command line: notepad.exe filename.ext

CT-Shell entry: {&Edit} { } {notepad.exe} {!} { }

#

The hashmark translates into a list of files that are tagged, or as many of them as can be squeezed into the DOS limit of 127 characters on a command line. You might like to add all of the tagged files to an archive file named ARCNAME.LZH by using the LHArc program:[16]

Command line: lha a arcname file1 file2 file3 ...

CT-Shell entry: {&LHArc Add} { } {lha.exe} {a arcname #} { }

??

A pair of question marks surrounds the prompt you want CT-Shell to display when it asks you for a string of characters to put in its place. This is how you can supply variable arguments at the time an entry is executed.

For example, the LHArc command shown above will always create an archive called ARCNAME.LZH, because the name ARCNAME has been *hard coded*, or stated explicitly, in the command. It will require that the same archive be created or updated each time this popup entry is executed, although the currently-tagged file names may be different each time.

Compare that to this example, where CT-Shell will ask the user for an archive name each time the command is executed:

Command line: lha a arcname file1 file2 file3 ...

CT-Shell entry: {&LHArc Add} { } {lha.exe} {a ?Arc name? #} { }

When this popup entry is executed, CT-Shell will display a dialog box identifying the needed argument as "Arc name" and asking the user to supply a name. That answer will be inserted into the command line, replacing the ?Arc name? characters.

Although it is usually an error to use the **!** or the **#** special characters more than one time in a command, you may want to use several **??** pairs, to ask for multiple arguments for a command. Since each prompt specifies what information is needed, the user won't get them confused.

And since it does no harm to enter a blank answer, it is even practical to use a prompt for those times when you might, or might not, need input. If none is needed for a particular execution of the command, the prompt can be ignored by clicking on [Cancel] button or simply entering a blank answer.

%%

A pair of percent signs will cause CT-Shell to insert the value for a named environment variable into the command. This is consistent with the way environment variables can be accessed within a batch file, and the topic of environment variables is explained fully in your DOS manual.

Briefly, you set environment variables to a given value with a SET command like this:

SET ENVAR=contents of variable

Although that can be done at a DOS prompt, it is usually done in an AUTOEXEC.BAT file instead, so that your environment variables are established correctly each time you start your computer. Various programs obtain various kinds of information from environment variables, and the documentation for those programs must tell you how to set them, if any are needed.

It's very common for a language compiler to require an environment variable named LIB to contain the directory name where the compiler's runtime libraries - files that are used in creating programs - are stored.

When used in a CT-Shell popup entry, the two percent signs and the variable name that is between them are replaced by the value that DOS associates with that environment variable.

For example, a programmer might want an easy way to insert object modules that are being created (parts of programs) into a library named FOO.LIB, and which is located in the directory pointed to by the LIB environment variable. It is assumed that the current file will be an object module, a file ending with the extension .OBJ. This example assumes that the environment variable currently contains the value C:\LIB and that the current file name is BAR.OBJ:

Command line: lib C:\LIB\FOO +BAR.OBJ;

CT-Shell entry: {Add &Module} { } {lib} {%LIB%\FOO +!;} { }

A second example shows how you might use an environment variable in the directory path field, one of the rare uses of CT-Shell special characters in that field. Here a command is created that will change the working directory to the one in which a programmer's header files are stored, and pointed to by the environment variable called INCLUDE:

CT-Shell entry: {Change to &Headers} {%INCLUDE%} { } { } { }

Since there is no command to execute in this case, the directory change will be permanent (although you can still return to the original starting directory by pressing <F6>).

Another example shows how you might use the same environment variable to edit your PRG.H file, which is assumed to be located in that directory, using the QEdit editor:

Command line: q.exe C:\MSC600A\INCLUDE\PRG.H

CT-Shell entry: {Edit PRG.H} { } {q.exe} {%INCLUDE%\PRG.H} { }

Finally, there's even a special CT-Shell pseudo-environment-variable called WINDIR that you can use in

your command entries wherever you need to refer to the Windows "home" directory. Although such an environment variable is never set at DOS (there's no need for it - Windows already knows where its directory is, and so do Windows programs), it is used in these entries in the same way that an environment variable would be, so it follows the same syntax.

Thus, the sample menu entry that edits your Windows SYSTEM.INI file using NOTEPAD.EXE is always able to find it because of the %WINDIR% "environment variable" that CT-Shell replaces with the actual directory name. It looks like this:

```
{SYSTEM.INI File} { } {notepad.exe} {%WINDIR%\system.ini} { }
```

As in the case of the **??** special characters, it does no harm to use more than one environment variable in a command. They may even be used in more than one field in the same command, if appropriate.

[1] Of course, experienced users will understand that there are some programs that simply won't run in Windows at all, in their current versions, perhaps due to memory management conflicts or conventional memory requirements. CT-Shell is subject to the same limitations that Windows itself is, and it can't work any special magic with these hard cases. However, it's safe to say that if you've run it from Windows, you can almost certainly run it from CT-Shell.

Incidentally, CT-Shell has been designed to require as little memory as possible when it runs. Although the executable file is more than 60K in size, you'll find that the program actually requires about 20K or so to run, depending partly on your system.

[2] Be advised that the dialog box you use to enter your file spec will contain a default of *. , to which you can simply add an extension, if you want. To keep that original part of the prompt from disappearing when you type your first letter, you need to click the mouse one time where you intend to type, or press one of the arrow keys on the keyboard. The reverse-image prompt will change to a normal image, letting you add to it rather than replacing it with your input.

It is probably obvious by now, but if you want to specify a file spec that does *not* begin with *. , you can simply begin typing, and what you enter will replace that default prompt.

[3] Windows traps the <F10> key, by the way, and uses it to access the menu, duplicating what the <Alt> key is usually used for. That is because of IBM mainframe terminals that don't have an <Alt> key. In an attempt to standardize an interface that can be used across many diverse systems, <F10> was chosen to be the menu access key that exists on all terminals.

Still, anything you assign to this <F10> key can be executed by clicking the mouse on the *screen* representation of the key, as you can with all the others. Since most users will probably access these commands with the mouse, most of you probably won't notice - or care - that pressing the <F10> key itself accesses the menus.

[4] As a matter of curiosity, this function uses the same low-level DOS function that the CT-Shell MOVE command uses to move a file from one directory to another on the same drive without copying the contents of the file. Although it will most often be used for changing the name of a file or a directory within the current directory, you could also supply a complete path/file name here to move the file elsewhere. There's no compelling reason to remember this, however, as there is a MOVE keyword provided as well.

[5] Note that there is no guarantee that your directory will copy to a disk that has exactly that much room free - in fact, it probably won't, because most files require slightly more of a disk allocation than their size would indicate. (Disks are allocated in terms of *clusters*, not bytes, and the cluster size of your disk may be from 1K to 8K or more.) However, this value provides a good approximation of the minimum space that would be required. Clearly, this directory could *not* be copied to a disk that has only 2.5 MB free.

[6] The reason full seconds are not stored is an interesting matter of simply not enough room in the directory entry. The creation time is stored in a single 16-bit integer in the DOS directory on the disk. The Hours field requires 5 bits, to store numbers as high as 23. The Minutes field requires 6 bits, as it must store numbers as high as 59, and that leaves only 5 bits left for the Seconds field. The best approximation (that can be stored in 5 bits) is seconds divided by 2, and that's exactly what DOS does.

[7] Sometimes network administrators will assign the read/only attribute to executable files that are to be shared by several users. If such a file is accessed by more than one user at a time, having the read/only attribute will prevent the DOS SHARE program from complaining about a share violation. Since the files can't be modified by anyone, SHARE is content to allow multiple users to access it at the same time.

[8] Sometimes hidden files are used to provide copy protection for a program: files you can't see and don't know are there are required for the application to run. Since it displays all file attributes, and you can easily see that a file has this attribute, CT-Shell displays all hidden files.

CT-Shell, by the way, is not copy protected in any way. Computer Training respects the honesty of its customers, and doesn't want to make their lives any more complicated than they may already be!

[9] Not without using a special command to delete the directory. DOS provides an RD (remove directory) command, but it won't remove a directory that has any files in it, and in any case, the DEL command doesn't work with directories at all. CT-Shell has a DELDIR keyword that is documented in the CTSHELL.INI Reference section, that can be used to delete an entire directory and everything it contains.

[10] Because that's really the only purpose of this attribute, a little more explanation seems in order. A differential backup is one that copies only those files which have been changed since the last full backup, which cleared this attribute on all the files it copied. A differential backup does *not* clear the archive attribute, so you always have just two backup sets - the full set and the differential set. When you do a restore, there are never more than these two backup sets to replace.

Another type of backup, an incremental backup, differs by clearing the archive attribute whenever files are saved. Thus, with an incremental backup you create a new backup set every time you do a backup, and it always contains the files that were changed since the last incremental backup. When you do a restore, you may be required to restore a large number of backup sets.

[11] If you honestly want to use an ordinary DOS command like DIR from the CT-Shell command line or from a menu entry, you'll want to make a PIF file for that command that invokes your command processor and provides any options that you need. For example, if you run *any* command from within CT-Shell that produces screen output that you want to look at before returning, you'll want to be sure that the PIF file for that command does *not* have the option checked to close the window automatically when the program is finished.

The program to run will be your command processor, and you'll probably need to include a /c switch on its command line, otherwise you'll have started a DOS session with it. (The /c switch tells it you want to return immediately after executing the command.) Here's how you might handle this in a PIF file that uses COMMAND.COM to provide a CHKDSK command:

```
COMMAND.COM /c chkdsk
```

By the way, with current versions of DOS, it is very unsafe to use the /f switch with CHKDSK, when you're running under Windows (to "fix" disk errors). The reason is because you may have a number of files open for programs that you have running, and CHKDSK doesn't understand that. It will think the files are lost clusters, and will gather them together for deletion. Future versions of DOS will undoubtedly contain CHKDSK commands that can make this distinction and will be able to be used safely under Windows, but be sure before you use it!

[12] These extensions are automatically installed in your WIN.INI file by Windows during its setup process. You can also edit that section of your WIN.INI file to add other extensions that would be useful to you. Your Windows documentation has more information about the [Extensions] section of your WIN.INI file, but here are some examples that may be enough for you:

```
C=QFULL.PIF ^.C
SLC=TELIX S^.SLC
```

The first example shows what CT-Shell should do if you doubleclick a file name that ends in .C (a C language program source file). This implementation will run the QEdit editor using the Program Information File named QFULL.PIF, and pass it the current file name (^) and the extension .C as arguments.

The second example shows a way to start up the Telix communication program and pass it the name of a compiled script to run. Telix's command line may include an optional letter "S" which is followed by the name of a compiled script. Those scripts end with the extension .SLC.

Any such extensions that you set up in your WIN.INI file can be used both by the Windows File Manager and by CT-Shell. Be creative with them, and you can save a great deal of work. You can doubleclick on a database file and automatically start your database manager. You can doubleclick on a phone directory file and automatically start the communication program that uses it. The possibilities are nearly endless. This is a VERY powerful feature, and one that experienced users should not let pass by without experimenting a bit!

[13] If you are using a third-party replacement command processor, be sure that you have followed the manufacturer's directions regarding setting your COMSPEC variable. If you do not set this explicitly to match your substitute processor, DOS will set COMSPEC to COMMAND.COM in the root directory of the boot disk, as a default.

[14] The UNDELETE command for DOS 5.0, for example, won't find a file that was deleted by CT-Shell. You may be able to recover a deleted file using another utility, but there is no guarantee at all. It is best to assume that once CT-Shell has deleted a file, it's going to stay deleted, and to be very careful with this keyword. The same is true of the DELETE keyword documented next, and the DEL command when used at the CT-Shell command line.

[15] Programmers, in particular, enjoy this feature, because it allows them to follow the convention in which all the files for the release of a software package have the same datestamp, and have a timestamp that specifies the version number. Thus, a timestamp of 01:00 would mean the files are part of version 1.00, and a timestamp of 2:34 would make it version 2.34.

When a file is modified, DOS changes the date and time for that file, so you can easily determine later whether any of the release files has been modified in any way.

[16] LHArc is a popular freeware data compression program that is available from many sources. It creates archives, or libraries, of files that have been compressed much smaller than their original size. It was used to create the CTINSTAL.EXE program that you used to install this product. It makes an excellent example for these special characters, because it gets a lot of information from its command line when it is run.

Legalities

This software is copyrighted, and all rights are reserved by Computer Training. Individual personal users are granted permission to evaluate CT-Shell v1.00 for a period not to exceed thirty days, and are permitted to distribute the original archive to others for the purpose of evaluation. Those individuals who continue to use CT-Shell beyond the evaluation period become obligated to pay for the software, thereupon becoming *registered users* with all rights and benefits that pertain thereto.

Copying, duplicating, leasing, selling, or otherwise distributing this software in any other way than specifically stated above requires express written permission from a principle of Computer Training. USE OF CT-SHELL BY ANY BUSINESS, GOVERNMENT AGENCY, OR EDUCATIONAL INSTITUTUON IS NOT SUBJECT TO AN EVALUATION PERIOD, AND REQUIRES REGISTRATION IN ALL CASES. Volume users are encouraged to contact Computer Training for liberal site license terms.

Trademarks

Various product names that are mentioned in this manual are the trademarks or registered trademarks of their respective manufacturers. *CT-Shell* and *CT-Shell for Windows* are trademarks of Computer Training.

Limited Warranty

The diskettes and printed materials that are sent to registered users of CT-Shell are warranted for 90 days against physical defects. To obtain a replacement for a defective product, return it to the place of purchase with an explanation of the defect. If your copy was purchased directly from Computer Training, you may return it postpaid to:

Computer Training
7016 NE 137 ST
Kirkland, WA 98034-5010

Not covered by any warranty are materials that have been lost, stolen, or damaged by accident, misuse, or unauthorized modification.

COMPUTER TRAINING WILL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR OTHER SIMILAR

DAMAGES, EVEN IF WE OR OUR AGENT HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. ANY LIABILITY IS NOT TO EXCEED THE ORIGINAL PURCHASE PRICE.

We make no other warranty, express or implied, to you or to any other person or entity. Specifically, we make no warranty that the software is fit for a particular purpose. Any implied warranty of merchantability is expressly and specifically disclaimed.