Chapter 0 Shareware and License Information

CT-Shell is distributed as shareware, which means that you are encouraged to try it out before buying it, and you are welcome to pass it along to friends and others, provided that no charge is made for it, and you distribute all the files intact. The following sections provide more details regarding copyright and other legal matters.

Trial Period

Non-registered users may try the evaluation version of CT-Shell for a period not to exceed 30 days, for the purpose of determining whether it is suited to their needs. A license must be purchased to continue using CT-Shell beyond that time, or its use discontinued and all copies of it removed.

Copyright

This software is copyrighted and all rights are reserved by Computer Training. The distribution and sale of registered versions of this software are intended for use of the original purchaser only. Lawful users of this software are hereby licensed only to read the software in the asso-ciated files solely for the purpose of executing it. Copying, duplicating, renting, leasing, selling, or otherwise distributing registered versions of this software is against the law, with the excep-tion that the original purchaser may sell the original license to another person, keeping no copy of the software for himself.

Licenses Required

Licenses are required on a per-user basis, with a maximum number of licenses not to exceed the number of machines that CT-Shell is installed on.. Thus, if two people use CT-Shell in an office where it may be installed on five different computers, only two licenses are required. If five dif-ferent people use CT-Shell on a single computer, only one license is required. CT-Shell may be installed on and from, and it may be executed on and from, a server that is part of a network, provided that sufficient licenses are purchased, as provided above.

Trademarks

Various product names referred to in this manual are the trademarks or registered trademarks of their respective manufacturers. CT, CT-Shell,

and CT-Shell for Windows are trademarks of Computer Training.

License Agreement

Carefully read the following terms and conditions. Use of this Software constitutes your acceptance of these terms and conditions, and your agreement to abide by them:

The original purchaser of a registered version of CT-Shell (Licensee) is granted a nonexclusive personal license to use the Software under the terms stated in this agreement. The Licensee may transfer his license to a subsequent purchaser by sale, provided tht the original Licensee does not retain any copy of the Software. Except for that provision, any attempt to sublicense, assign, or transfer any of the rights, duties, or obligations hereunder is void. The Licensee may not copy, modify, alter, electronically transfer, or lease any registered copy of the software, or the printed manual. The license is effective until terminated. The Licensee may terminate it at any time by destroying the Software. The license will also terminate if the Licensee fails to comply with any term or condition of this Agreement. The Licensee agrees upon such termination to destroy the Software.

Limited 90-day Warranty

The diskette(s) and printed materials that are part of a registered copy of this product are warranted for 90 days against physical defects. To obtain a replacement for a defective product, return it within the 90-day period with an explanation of the problem to:

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

Voice: (206) 820-6859 Data: (206) 823-2831

This product is also warranted against significant errors in the software that render it unusable for you. However, you may find it more useful to call Computer Training and discuss a problem before returning your software for replacement. Often what appears to be an error in the software is actually a misunderstanding about how it is to be used, and an immediate workaround or solution to the problem is possible.

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 PAID.

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 limited to the 90-day duration of the limited warranty, and is otherwise expressly and specifically disclaimed.

This warranty gives you specific legal rights. You may also have other rights which vary from state to state. Some states do not allow the exclusion of incidental and consequential damages, or the limitation on how long an implied warranty lasts, so some of the above may not apply to you. This agreement will be goverened by the laws of the State of Washington.

Chapter 1

Installing

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 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 before going beyond this chapter.

Preparing for Installation

CT is distributed in two ways. If you obtained a copy of it from a friend or from a Bulletin Board System (BBS), you probably received an archive file named something like CTSHW200.LZH (or perhaps .ZIP or .ARJ, depending on the source). In any event, that archive will require an extraction program of some kind, depending on the type of compression program that was required. But then, you already have found that out if you have gotten to the file containing this manual, as it was stored within that archive.

If you obtained a registered-user copy of CT directly from Computer Training, you will have a disk with the necessary files in an uncompressed form. In that case, your disk should contain at least the following files:

- CTSHELL.EXE the executable program

- CTSHELL.INI sample menu/initialization file
- CTSHELL.HLP Windows help file
- CTSHELL.KEY SoftKey to unlock your copy
- CTINSTAL.EXE Installation program

and possibly some other files, such as

- CTSHELL.NOT Distribution notes

Having extracted the files from a shareware version of the program, you would have the same selection of files with two exceptions: (1) rather than a printed copy of the manual, you will have a Windows Write document file named CTSHELL.WRI, and (2) you will *not* have the SoftKey file CTSHELL.KEY. That one is provided to registered users only.

Once you've arrived at this point, installation of the program is exactly the same, whether the files came in a shareware archive, or the files came on a registered distribution diskette.

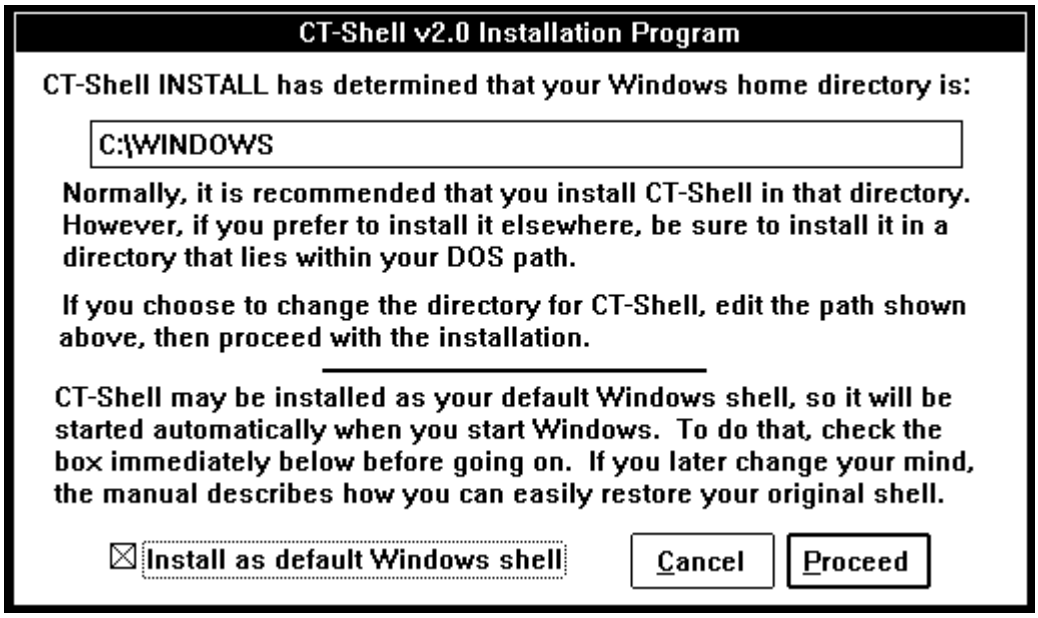
Installation

There is an installation program provided, named CTINSTAL (its executable file name is CTINSTAL.EXE) which can be run from within the Windows environment using any means you usually use to run programs. In other words, you can doubleclick on CTINSTAL.EXE while using the Windows File Manager, or you can select the *Run* entry from the *File* menu while using the Windows Program Manager, and specify CTINSTAL as the program to run. You can probably even start up Windows and run the install program all at once, with the DOS command:

```
C:\> WIN CTINSTAL
```

Install Disk/Directory

Before running CTINSTAL by any means, be sure to change to the drive and/or directory where the distribution files are stored. Everything else about the installation is automatic, but CTINSTAL makes the assumption that it is being run from the directory where it and the rest of the distribution files are stored. When run, the CTINSTAL program produces a screen that looks like this:



Decisions

There will be two decisions for you to make–both of them easy ones. You need to agree to install CT in your Windows home directory, or provide another place for it to be kept. Ordinarily there is no reason why you would want it anywhere else, but CTINSTAL offers that option anyway. Most people will choose to install it where CTINSTAL wants to put it. You will also need to decide whether CT is to be your default Windows shell.

Directory

If you have any compelling reason to install CT in another directory, it is recommended that it be one that lies along your DOS path. That will allow Windows to find CTSHELL.EXE to run it without requiring full path information, and it will allow CT to find its initialization and help files easily when it needs to. If you're not sure what is meant by the *DOS Path*, you can find out more about that from your DOS manual.

The installation process will put several files into your Windows directory, if you accept the default, and they're easy to recognize because they all begin with CTSHELL.... If you later decide to remove the files for any reason, there will be no question which ones they are.

If you do decide to install CT somewhere other than your Windows home directory, edit the path that is shown before going on with the installation. When you click the button that is marked [Proceed], CTINSTAL will copy your distribution files to the directory path shown in the setup window.

Default Windows Shell

The second decision is a little more complicated, but should be just

as easy to make as the first one. Most people will want to use CT as their replacement Windows shell–in other words, the program that starts up when Windows starts, and gives them control over what other programs to run. That's what CT was designed to do, from the start.

When Windows was first installed on your computer, it was almost certainly the Windows Program Manager program (PROGMAN.EXE) that was used for this purpose by default. Some Windows users decide to install the Windows File Manager (FILEMAN.EXE) in its place, preferring the file-oriented rather than icon-oriented approach to program management.[1]

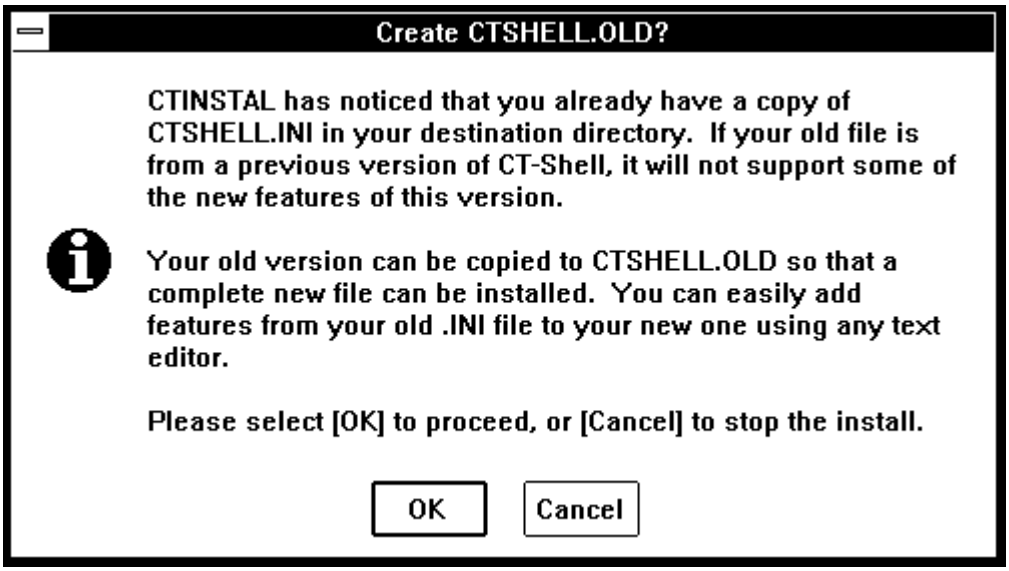
Likewise, CT can be installed as the default shell by replacing whatever is currently being used with CTSHELL.EXE. In fact, CTINSTAL will make the change for you automatically, if you check the box marked *Install as default Windows shell* near the bottom of the installation screen. To install the CT files but *not* modify your SYSTEM.INI file, leave that box un-checked. If you do it that way, you will need to start CT manually, as you would any other Windows application.

Note that Program Manager and File Manager may both be started quite easily from CT, so there's really no reason to be concerned about making CT your default Windows shell, and besides, CT uses less of your Windows resources than does Program Manager. However, if you later change your mind and decide you would like Program Manager–or any other program–back as your default shell, you can simply edit your SYSTEM.INI file to restore the original name. There's even a CT menu entry that makes it easy to edit any of your system files.

If you did decide to install CT as your default Windows shell, simply exit from Windows when the installation is complete, then restart Windows. CT will appear on your screen in place of the shell you used to use. If you decided not to install CT as your default Windows shell, you can start it in the way you normally start Windows applications. You can even install it as an icon in one of your Program Manager groups. See your Microsoft Windows User Guide for more information about installing a group item, if you haven't done that before.

Previous Versions

Many who install CT will already have installed an earlier version, and will probably have an older copy of CTSHELL.INI in their destination directories that they do not want to erased by copying the new version over it. CTINSTAL will spot the old version if one exists, and will prompt you for the action to take:



Summary

Here are the steps for installing CT properly, listed without a lot of extra descriptive information. Rather than going back over the preceding paragraphs, you may want to refer to this list as you install the program:

1. If you are installing from a registered user diskette, place the diskette into any disk drive and change to that drive as your default. In other words, if you put the disk into A:, make that your default drive before going on. CTINSTAL can easily locate your Windows directory, but it will only look in the *current directory* on the *default drive* for the files it is to install.

If you received CT as an archive from a BBS, you probably already have it somewhere on your hard drive, and have already extracted the files it contains, since this document is in one of those files. It doesn't matter to CT where you install it from, but you may want to create a temporary directory somewhere and copy the distribution files to there. Make that your default directory, then proceed.

2. Using your usual means for running a Windows application, execute the program named CTINSTAL.EXE, which is one of the distribution files.

3. Unless you have a good reason to change it, let CTINSTAL install CT into your Windows home directory. If you do have a good reason to want it elsewhere, edit the path name that's shown, but be sure to put the files into a directory that's along your DOS executable path.

4. Unless you prefer not to use CT as your default Windows shell

(what it was actually designed to do), check the box at the bottom of the install window, and CTINSTAL will modify your SYSTEM.INI file automatically to install CT as your default shell. If you decide not to make CT your default shell, you will need to start it manually from your other shell.

5. Finally, click the [Proceed] button to go ahead with the installation. Depending on whether it is a registered user version, CTINSTAL will copy three or four files to the destination directory, and if all went well, will say so.

Headaches

There are two potential problems that should not occur during an installation. If all didn't go well, you will see a message box that tells you a certain file couldn't be found, or that the destination directory couldn't be written to.

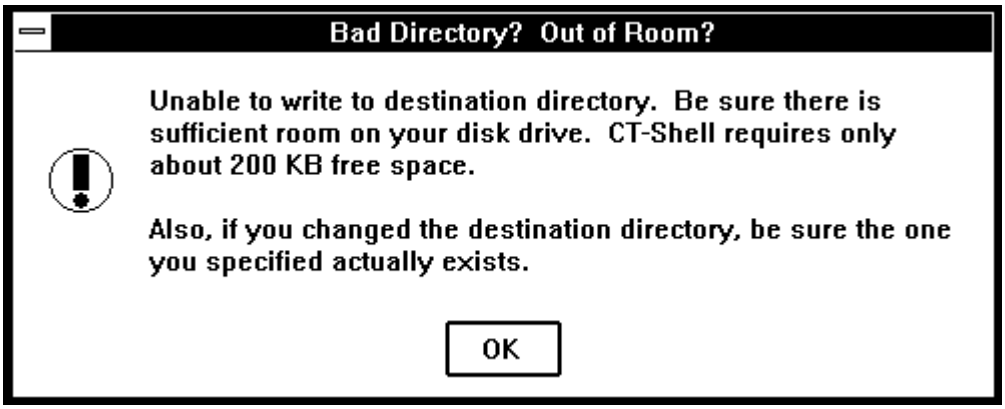
File Not Found

About the only reason you would see this message is if you have *not* changed to the directory where the distribution files are stored before beginning your install. CTINSTAL is able to locate your Windows directory easily, so it knows where to put the files it copies for you. The only way it knows where to find the files, however, is if they are in the current directory.

CTINSTAL was designed this way, so it can be run from any drive or directory. There is no need to install from drive A: only, although that's the commonest arrangement. If the type of disk you received fits into drive B:, feel free to install from there.

Likewise, if you received CT in an archive from a BBS or a friend, you will have extracted the contents of the archive into a directory *somewhere*. It is only necessary for you to change to that directory–make it your default directory–before proceeding to run CTINSTAL.

Cannot Write to Destination



Although it's technically possible for you to get this message if the disk drive is full and CTINSTAL can't find room for the files it needs to copy, it is unlikely to be caused by that. CT only requires about 200 KB of space on your hard drive for all the files that are installed, so it is unusual for that to be an issue.

What is more likely the case is that you have changed the destination directory name to a path that does not exist. Perhaps you intended to create the directory before beginning your installation, but forgot to do it. That is sufficient to cause the installation to fail.

Installation Fizzled

If your installation didn't go well, you'll be told in no uncertain terms. Also, even if you have elected to install CT as your default Windows shell, that change won't be made in your SYSTEM.INI file if your installation failed for any reason.

Success!

Likewise, CTINSTAL doesn't keep you wondering if you had a successful installation. This message tells you everything went okay, and reminds you to exit from Windows and restart it, if you have elected to install CT as your default Windows shell. Congratulations!

Multiple Configurations

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

Once you have learned 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 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 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
```

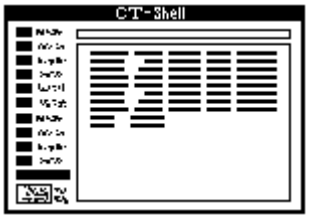
Of course, either of these would require that you create those xxx.INI

files. You might begin by copying CTSHELL.INI to the new file name, then modifying it to suit your purposes.
This handy feature also makes it easy for multiple people to use a computer on which CT is installed. Each of them can create an initialization file that suits the way they individually work, and change to it when they're the one using the computer.

One last point should be made here. As a Windows application, multiple instances of CT 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 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 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.

Origins

The original DOS version of CT was developed for use in advanced computer 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 needed to work, and made routine commands a simple matter of selecting them from its menu.

CT 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 Menu Basics

You won't need to reconfigure your menu system to try out CT, so we won't go into all the details here. The following few sections provide an overview to let you know how simple that system is to set up the way you want it.

CT is quite configurable; in fact, *every entry* in the menu that you see when you run the program 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 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.

Throughout this manual there will be many mentions of this default CTSHELL.INI file. It is designed to make available all the CT features that are built into the program, and to serve as a guide to creating your own menu entries. You'll find much more discussion about customizing your CTSHELL.INI file in later sections of this manual.

Besides menu entries, the CTSHELL.INI file contains values for many options that CT relies on for its various features. Phone numbers are stored there for its directory/dialer, preferences are stored there, and configuration options such as those that specify how program listings are printed. However, the most significant features in CTSHELL.INI are the entries that define the menus, the AUTOEXEC section, and the USER options. They are all similar in construction, they all contain five sets of braces, and look like the following examples:

Generic form: {Entry Name} {DirPath} {ExePath} {Switches} {Keyword}

Example 1: {NOTEPAD Editor} {} {notepad} {!} {}

In this first example, the name that will show up in the menu is *NOTEPAD Editor*. Running that program doesn't require changing first to a different directory, so the second set of braces is left empty. The third set of braces contains the name of the program to execute–in this case, it's *notepad*. The fourth set of braces contains any switches, or arguments that you need to pass to the program when it runs. Here, the exclamation mark indicates that we want to edit the *current file*, the file that is highlighted in CT-Shell's list of all files in this directory. Finally, the fifth set of braces is left empty, because this

command doesn't require a special CT keyword. You'll see one of those in the third example.

To edit a specific file every time–such as to create a menu entry that allows you to edit the CTSHELL.INI file any time you want to–you would simply replace that exclamation point with the full path name of the file you want to edit. For example, it might be done like this:

Example 2: {Edit CTSHELL.INI} {} {notepad} {c:\windows\CTSHELL.ini} {}

In addition to what you've seen so far, you can send a list of all the *tagged files* to a program that accepts multiple filename arguments by putting an pound sign in that field, such as {#}. (Some people call that a *number sign*–musicians call it a *sharp*. It's the <Shift+3> key on your computer keyboard.

Finally, if you need your command line to include the base filename (of the current file) without its extension, you can specify that with the commercial at-sign {@}. Some users will appreciate that they can create an expression that has the same name as the current file, but a different extension.

Keywords

Many of the features that CT makes available are based on special keywords that take the place of program names. They are implemented this way to give you complete freedom in redesigning your menu system. Most entries that use a keyword do not use any of the other fields, except the first. This one would activate the dialog that allows you to change your preferences–those settings that affect how CT does certain things:

Example 3: {Set Preferences} {} {} {} {PREFER}

The special CT keywords may be entered in uppercase, lowercase, or mixed case. They are all converted internally to uppercase for evaluation. There are 32 CT keywords in this version of the program; they are all described in detail in Chapter 6 of this manual, and in the CT on-line help file.

There are just a few CT keywords that are, in fact, used with executable programs. They include ICON and LOAD, which are synonyms and would cause the program to be run as an icon, as it would if it were listed in the LOAD= entry in your WIN.INI file. FULLSIZE causes a program to be run in the maximized–full size–state. RUN causes a program to be run normal size, as in the RUN= entry in your WIN.INI file. For completeness, the keyword NORMAL is also included, although that's the default if none of the others is used. This next example shows how you could create a menu entry that would load NOTEPAD, feed it the current file, and install it as an icon, ready to be used at any time:

Example 4: {Edit CTSHELL.INI} {} {notepad.exe} {!} {ICON}

Two Kinds of Menu Entries

To summarize, CT pop-up menu entries can contain two different kinds of commands: commands that run programs that are not part of CT (such as the programs that you use every day), and commands that are internal to CTSHELL. The latter are implemented using a special set of *keywords* that CT recognizes. The rest of this section will probably mean more to you if you take a look at your original copy of CTSHELL.INI as we speak. There's a menu entry in the *Edit* menu called *CTSHELL.INI*, and selecting that entry will run NOTEPAD to edit that file. Go ahead and select that entry using the mouse or standard Windows keyboard methods, but be careful not to change anything you're not sure about. For now, you'll want to "look but don't touch."

Towards the beginning of the file are several sections that contain options and preferences. Move down in the file until you get to the section where the menus are defined–that should be clearly marked. It's a section with lines that begin with the word *Item* that name individual menus (*items* in the main menu), and which are each followed by one or more entries that look like the examples you saw in the previous section. The following paragraphs describe what you'll see in that part of the file.

Items

Most of the entries in the sample CTSHELL.INI file use CT keywords, since they will work exactly the same way on everyone's computer. Commands that run programs will usually vary from computer to computer, depending on what programs each user has installed.

However, there are a few entries in the sample CTSHELL.INI file that run external programs. There are some programs that we can count on all Windows users having, so the default menu contains entries that run some of the Windows utilities. Since command-line arguments can be supplied for the programs that are run, we can have entries to allow us to use the NOTEPAD editor to edit various files, such as the configuration and system files that Windows and CT use. Using the same techniques, you'll be able to create entries that run any of the programs you use on a day-to-day basis from the menu in CT, 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 CTSHELL. You'll be able to make any program that you can run from within Windows a part of your menu, including DOS applications.[2]

Directories

Besides running programs, CT 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 arbitrary limit to the number of entries you can have in such a menu!

As you would expect by now, an entry that is intended just to change to another directory usually has an entry name in the first field, a directory designation in the second field, and nothing in the remaining three fields. Here's another example, which would change you to a directory with the path C:\PIF, perhaps a place where you store all the PIF files for your system.

Example 5: {PIF Directory} {C:\PIF} {} {} {}

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 menu, so you can use it whenever you need to modify a text file.

Autoexec

If you look back towards the beginning of your CTSHELL.INI file while you have it loaded in the editor, you'll see an area marked [AUTOEXEC], that contains an empty set of braces, just like the ones you've seen used in the menu items section. This area allows you to create entries that will load or run programs automatically as CT is starting up. Because any number of entries can be placed here[3] , and because command-line arguments may be provided using all of CT-Shell's powerful features (many of which you haven't seen yet), this feature of CT far exceeds the usefulness of the LOAD= and RUN= lines in your WIN.INI file.

These autoexec entries are just like the menu entries you saw earlier. In fact, people often just copy-and-paste from the lower section when they want to start a program automatically when CT starts. You don't really *need* an entry name for anything used here in the Autoexec section, but it doesn't hurt to leave it there as a reminder for you. CT will simply ignore the first field.

Here is where you're most likely to use the ICON or LOAD keywords, which you saw earlier. If you have your CTSHELL.INI file still open to this section, experiment by entering the following example into this Autoexec section (just the five sets of braces, of course):

Example 6: {Calculator} {} {calc.exe} {} {ICON}

Now when you restart Windows, and it restarts CT, you'll see the Windows calculator become an icon in the lower left-hand corner of your screen even before the CT window appears. You'll be able to doubleclick on that icon to make the calculator available whenever you need it.

People often work with a particular set of applications when they use their computers, and if that applies to you, you'll probably want to customize this section to load your main applications whenever you start up. If you understand it well enough already, go ahead and install another program or two in the Autoexec section while you're here.

User Entries

The third place where you'll find entries that look almost exactly like the ones you've been working with is in the [OPTIONS] section, in the lines that begin with User1= , User2= , etc. These are user-supplied function key assignments, and the four that you see here become assigned to the function keys F9, F10, F11 and F12. The only real difference here is the way the assignments are made to the specific functions, and this is one of only two places where you'll use entries like this. For anyone who is reading this without the CTSHELL.INI file open in the editor, here's what those entries look like in the default configuration.

Example 7:
```
User1={Alarm Settings} {} {} {} {ALARM}
User2={Listings Config} {} {} {} {CONFIG}
User3={Preferences} {} {} {} {PREFER}
User4={Printer Setup} {} {} {} {PRINTER}
```

These user-supplied settings allow someone to press F9, for example, to set the CT alarm clock, or press F11 to set preference options. Note that all these entries are *also* available in the ordinary menu entries (in the *Shells* menu), but are duplicated here for convenience. You could replace these with any other entries that you want to be activated by the press of the key. Are you a big solitaire fan?

Alarm Entries

Moving all the way back to the beginning of the file, you'll see a section called [ALARMS], which contains a lot of options that control up to four event timers that CT provides. You can configure it to run a backup program each morning at 3am, for example, or call a remote system and download a packet of email or database files late at night, when the phone rates are lowest.

You will notice that this section contains some entries that look a lot like the ones for user entries–but they're used here as events that are to be executed at a set time. Normally you would change these

settings by selecting a menu entry that invokes the special CT ALARM keyword, but you could change them here with an editor, if you preferred that. More details are provide about this in Chapter 5 and Chapter 6 of this manual.

Summary

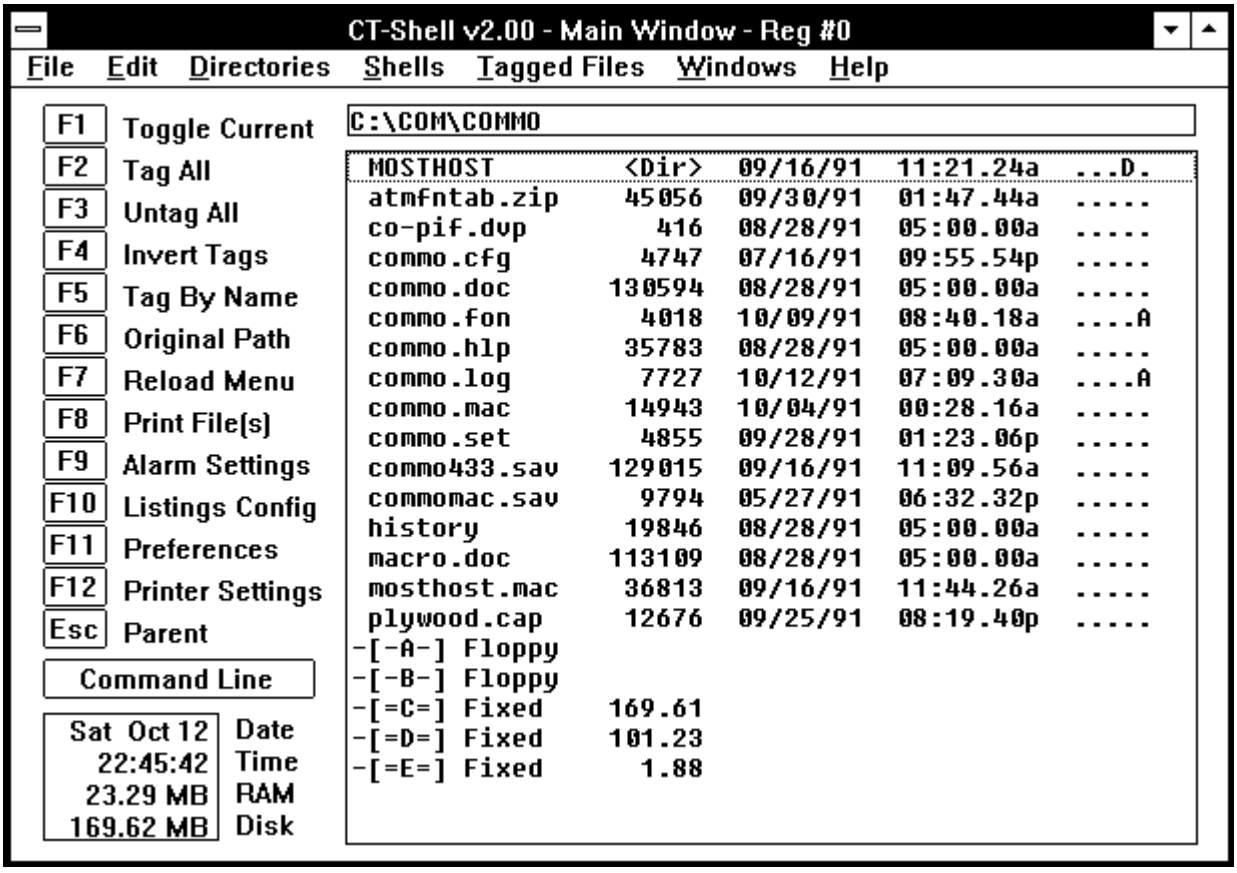
This overview was meant to introduce you to the CTSHELL.INI file, which configures and controls all the features that CT is able to offer you. There is much more in that file that will be described and defined in later sections, but what you've seen already should give you an idea of the way CT works, and the many things it can do for you. It should also have left you with an appreciation of the degree of configurability CT offers. You will be able to customize your installation–easily and quickly!–to make it just what you need for all your daily work.

At this time you should finish looking through CTSHELL.INI and close the file. Use the file menu and select *Exit*, which will cause NOTEPAD to ask whether you want to save your changes. Answer *yes* or *no*, as appropriate, and you'll be returned directly to CTSHELL. Whenever you want to edit this, or any of the other "system"-type files that are listed in the *Edit* menu, you'll know how easy it is to do!

If you have made and saved any changes in your CTSHELL.INI file, you can reload the menu–and put those changes into effect–by pressing the F7 function key. Of course, they'll take effect the next time you start up CT, even if you don't reload the menu at this time.

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.



Note that everything you see above should look the same on your system, except details like 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. If you are a registered user, your number will appear in the caption line. If you are not, an *Unregistered* message will appear there.

Menu

When CT is run, at the top of its main window is a menu that shows several items, such as *File*, *Edit*, *Directories*, 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 menu, and you'll be able to select them from within the program.

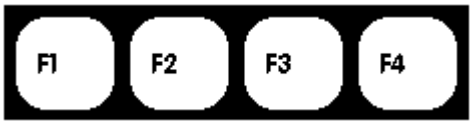
You've already seen an overview of the menu system, and this manual contains much more information about menu entries in later sections, with instructions about how you can customize yours. For now, realize that you'll have almost unlimited freedom to create such a menu with selections that are perfect 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 *Directories* menu. The *Edit* menu contains a number of choices, one of which you have probably used already to edit your CTSHELL.INI file. Remember that you can reload a modified menu by pressing <F7>.

The *Shells* menu offers an ordinary DOS session, and an additional CT window if you ever need a second one (or a third one, for that matter). There are also entries available for most of the CT functions. 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 list of files that you have tagged, rather than just the one current file. (The next section in this chapter will cover file tagging in much greater detail.)

Windows is where you'll see some utility options, applications, and system services that are part of Windows. If you ever feel you need them, here is where you can find the Windows Program Manager and the Windows File Manager. The *Help* menu provides access to the CT help file, and to some other options that provide you with information about CT and about your system.

Function Keys



There are pre-assigned meanings for many of the function keys, but <F9> through<F12> have been kept available for the user to define. In the sample configuration file, they are assigned to several configuration options, as you saw when you edited that file. You can reassign these four keys to any entries you want. Perhaps you would prefer them to start up Backgammon, Chess, Taipei and Solitaire.

Here are the default settings for the user function keys, based on entries in the supplied sample CTSHELL.INI file:

F9 **Alarm Settings**

*A*llows you to set up to four internal timers that can pop up reminder messages at preset times, or even execute programs according to an established schedule. You can even arrange programs to run at a given time every day–even if you're working on something else at the time! More details are available in a later section of the manual that describes CT keywords. Check the ALARM keyword when you need to know more.

F10 **Listings Config**

This will allow you to set all of the options that have to do with printing file listings. You are able to set preferences like the number of characters that a line must accommodate, whether to use a fixed or proportional font, whether headings are to be printed, line number and page numbers, and more.

F11 **Preferences**

Here is where you can choose many of the options that control CTSHELL. Many of the entries in the CTSHELL.INI file can be set by activating this keyword. For more details, see the keyword PREFER in the later section on keywords.

F12 **Printer Setup**

Pressing this one executes the "setup" function from the printer driver that you have installed. This is the same setup function that is invoked by your word processor when you change printer

settings there, so it will look familiar to you at first sight. More details are available in the later section on keywords, where PRINTER is explained.

Current File/Tagged Files

Much of what CT 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 the tag for 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

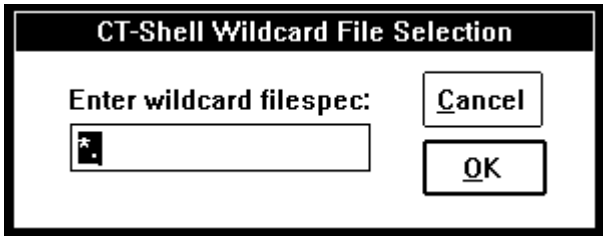
down the list of files (with the left button held down) to tag a few of the files, then press <F4> several times and see what happens. You can finish your experiment with <F3>.

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 asks you for a wildcard filespec, finish the process, then do it 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.[4]

The dialog box that's provided for you to specify a filespec looks like this:



The default entry begins with *. on the assumption that you will be tagging all files that end in a particular extension (the most common use for this function). However, you can type any legal DOS wildcard here. If you don't click on any entries–thereby changing your tags–you can use this function several times to tag any odd assortment of related files.

F6 Original Path

Returns you to the original path where CT 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 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 keyword called HOME that makes the current directory C:\WINWORD\CTSHTOC.DOT

your CT "home" directory, rather than the one where you started the program. If you work centers around one directory for a while, then changes to another location, you might want to change the place CT calls home. The HOME keyword is implemented in the *Directories* menu, in the default configuration.

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 you have probably already found out. 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 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 Chapter 6, for more information about setting up your printer driver and changing the format of your file listings.

F9 through F12

Are reserved for the user. In the sample CTSHELL.INI file they are assigned 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. [5]

Esc **Parent**

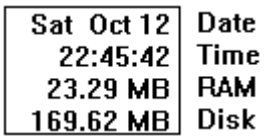
The escape key is used to change to the parent directory. So that the same operation is easy to accomplish with the mouse, the <Esc> key is represented on the screen along with the function keys.

Command Line

Not a function key, this is a screen-oriented way to open up the CT command line, a place where you can type commands to be run by CT, Windows, or the DOS command processor. Another way to open the command line is with the <Shift+Enter> keypress.

The display of these keys at the left of the CT 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



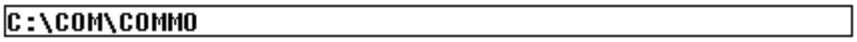
Below the listing of the function keys is a small window that displays the current date and time, the amount of RAM that is available (including virtual memory if you're running Windows in Enhanced 386 mode) and how much room is left on the current disk drive. The latter two measurements are displayed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

There is a problem with constantly monitoring the disk space on removable and network drives, as every time the remaining space is checked the disk has to be accessed. That wastes time, and results in floppy disk drives never shutting off completely. CT uses some clever programming to update such drives at all the right times, such as when a file copy or file move has been done that might change the remaining space. Otherwise, CT leaves those drives alone.

That works great *unless some other program changes the drive capacity*. On a local hard drive, CT will even take *that* in stride, properly updating the remaining space. On a floppy drive or a network drive, however, such access would escape CT-Shell's attention. If you ever suspect that another program might have changed the remaining space on a network or floppy drive, you can force an

update of this field by doubleclicking the mouse on the word "Disk", just to the right of this display.

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.

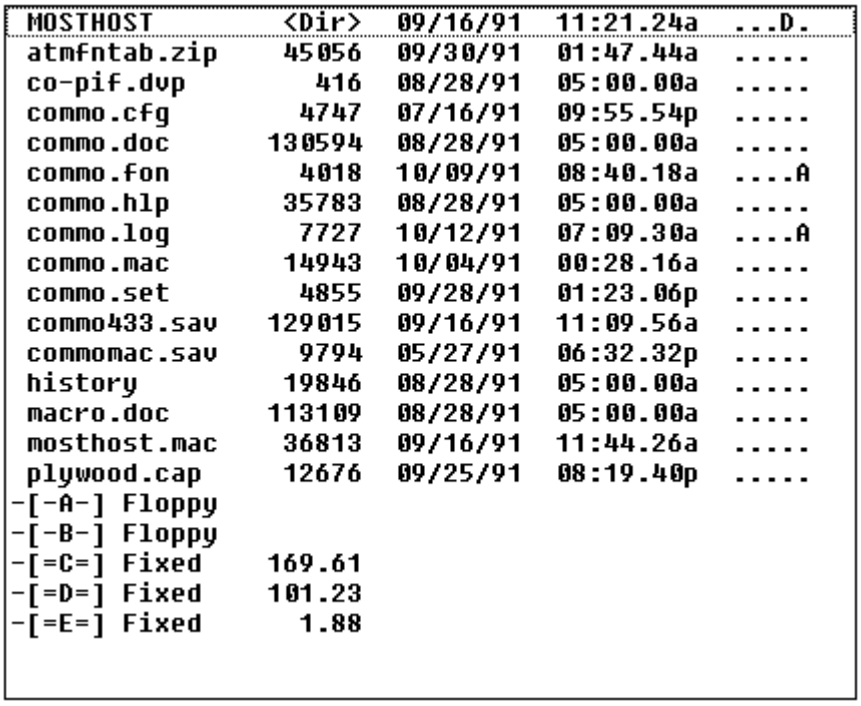
Directory changes made this way are "permanent" in the sense that CT will stay, and continue to work, in the new directory that you've chosen. However, you are still able to press <F6> at any time to go directly to the original drive and directory where CT was started. Changing directories this way requires only a single click of the mouse, however if you forget and doubleclick instead, no harm will come of it.

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:



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 keyword called DELDIR. (See Chapter 6 for a reference to CT 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 Directory*.

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 which is worded to get your attention.

Another keyword and command that can delete files is SHRED. This one first writes a pattern of bits to the file that renders it useless even if someone manages to undelete it. Thus, SHRED should be used in place of DEL where data security is important, however, it is slower than DEL because fo the extra work involved.

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 of 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 RENAME keyword. (See Chapter 6 for a reference to CT 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*.

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 keywords, TAGGED, which is explained in a Chapter 6. This keyword is implemented in the sample CTSHELL.INI file in the *Tagged Files* menu item, and called *Files Tagged*.

You are also able to discover how many subdirectories, files, and bytes a directory contains, using the CT DIRSIZE keyword. This keyword is implemented in the sample CTSHELL.INI file in the *File* menu item, and called *Size of Directory*.

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 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 provides a place for you to enter a new date and time.

Time

CT 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 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 pop-up 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 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 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 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 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 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 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.

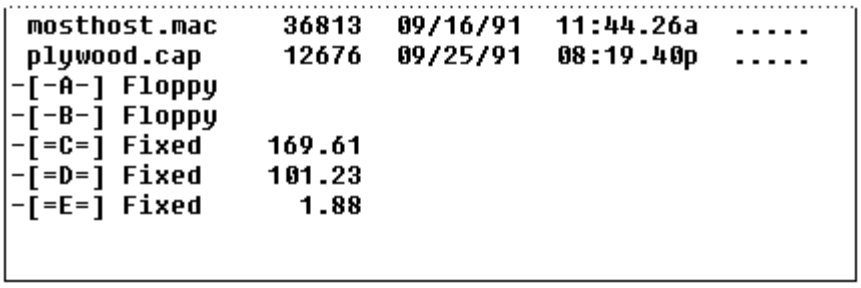
You can alter the attributes for a single file or for a group of tagged files if your CTSHELL.INI file contains a menu entry that uses the keyword ATTRIB (see the later section on CT keywords for more information about ATTRIB and other CT keywords). The sample CTSHELL.INI file contains entries for both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

When you invoke that menu entry, CT will present a dialog box that lets you determine which attributes are to be turned on and which ones are to be turned off. The attributes that you select are not added to the existing ones, but replace the existing ones. Thus, be sure you select *all* the ones that should apply. You can turn off all the attributes by leaving them all unchecked, and selecting [OK].

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 <Del> 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 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 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.

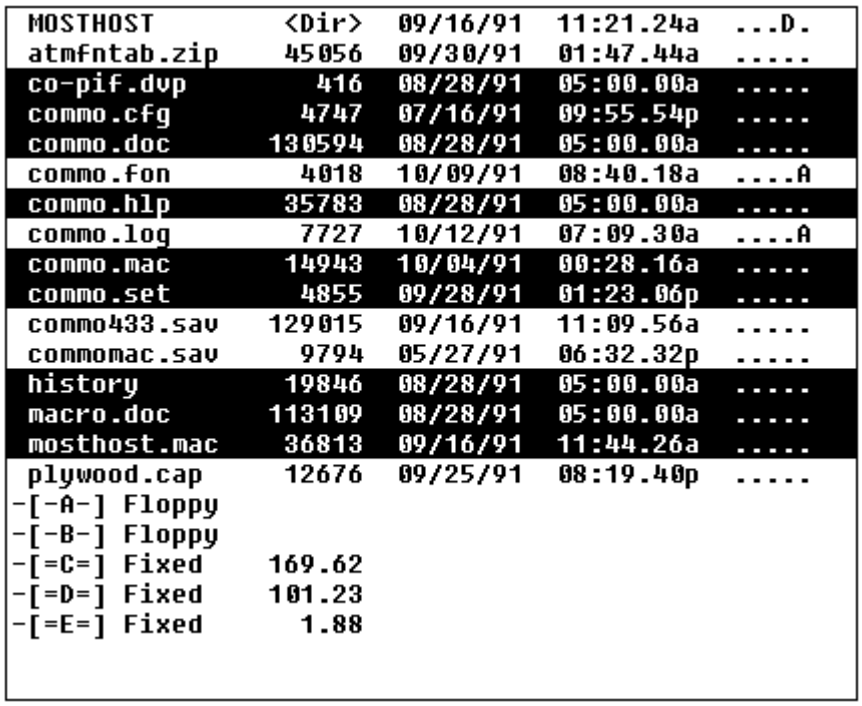
If you have a complex computer system–perhaps one with a lot of network drives in addition to the local ones–you may not want CT to monitor all the drives all the time. For one thing, whenever you change directories, all those drives have to be queried, to find out how much space remains, so that can be displayed here. There is an option available through your PREFER keyword that allows you to declare a series of drive letters that you want CT to ignore. You can also modify the IgnoreDrives= setting in the [OPTIONS] section of your CTSHELL.INI file, if you want. To ignore all drives higher than drive E:, for example:

```
IgnoreDrives=FGHIJKLMNOPQRSTUVWXYZ
```

Note that this does *not* keep you from accessing those drives. They simply will not be listed in this display. You can still change to a drive by typing its letter at the CT command line the same way you would do it from the DOS command line. Also, you can always change to another drive by pressing a key combination that includes <Shift+Ctrl> plus the letter of the drive you want to change to. To change to drive D:, for example, you could press

<Shift+Ctrl+D>.

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>.) When you do either, 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 CT-Shell'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 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, 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 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 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>.

It's worth repeating that there are three ways to change to another drive, since that's something everyone needs to do so often. If you have the command line open (see Chapter 4, next), you can change to another drive the same way you would at a DOS prompt–by entering the drive letter, followed by a colon, as a command.

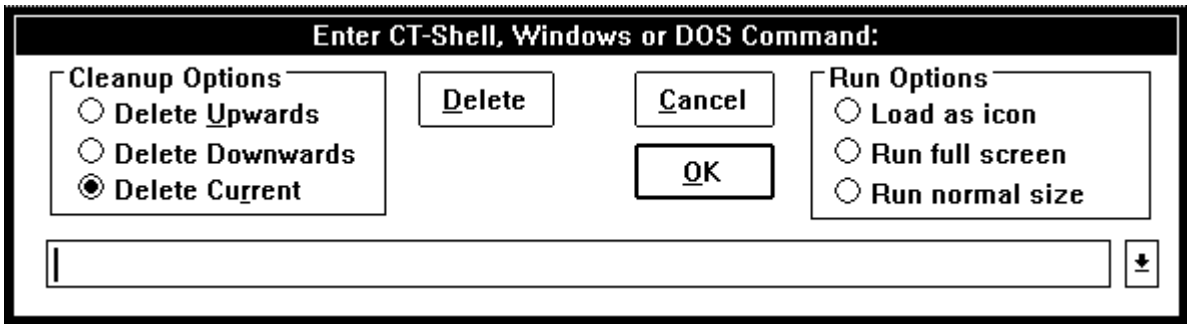
If the drive you want to change to is visible on the screen, the doubleclick method just described here is quite convenient. But the easiest of all, especially if your hands are on the keyboard, is to type <Shift+Ctrl> plus the letter of the drive you want to change to. For example, to change to drive D:, you could simply type <Shift+Ctrl+D>, and CT-ShelI will instantly change to that drive.

(This page intentionally left unused...)

Chapter 4 CT-Shell Command Line

An extremely important CT 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>.

The CT-Shell Command Line



DOS Commands

Most common DOS commands like CD, RD, MD, COPY, and DEL are handled internally in CT, without using the DOS command processor at all. Most of the DOS commands that CT 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. RD, MD and DELDIR all allow you to specify multiple arguments. Thus, the command:

```
MD bin init include lib
```

would create four subdirectories in the current directory, with one command.

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 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, without involving the DOS command processor at all.

However, if a command involves the DOS redirection operators (<, >, >>) or the pipe operator (|), the whole command is passed to DOS immediately, as is any command to run a .BAT file or a .COM file. (Windows executables are all .EXE files.)

Many users will 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. That's the way many Windows programs let you change to another drive, such as to locate and process a file. However, you are also free to enter a drive letter, followed by a colon, as a command on the CT command line, just as you would at a DOS command line. Such a drive change is handled internally by CTSHELL. (There is an even easier way to change drives–by pressing a key combination that includes <Shift+Ctrl> plus the drive letter.)

CT Commands

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

Deldir

You can delete a directory, and all the files in it with this command. In fact, it's one of the CT commands that accepts multiple arguments, so if there are a number of subdirectories that you want to remove, you can handle them all with one command.

This command actually invokes the same internal process that is used for the DELDIR keyword, which has already been described. You are prompted two times for *each* subdirectory that is to be deleted, to be sure you don't delete one by accident.

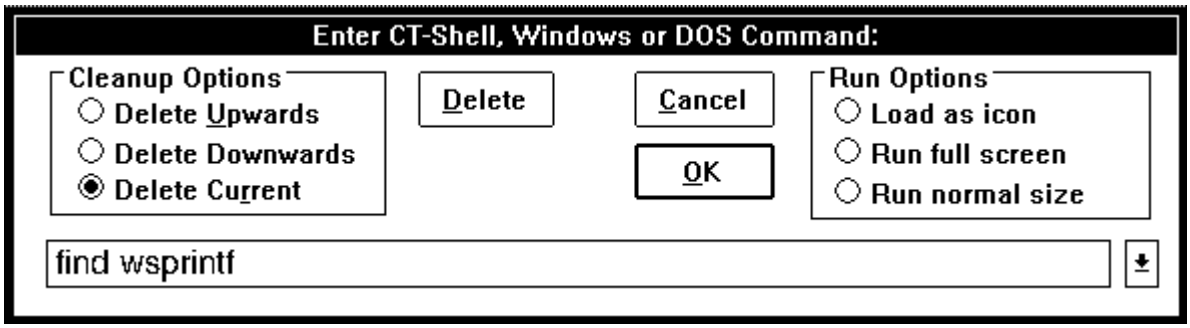
This command deletes all the files in the directories you ask to delete. *Even if those files have the read/only attribute, they will be deleted.* CT will remove that attribute if necessary, in order to delete the files.

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.

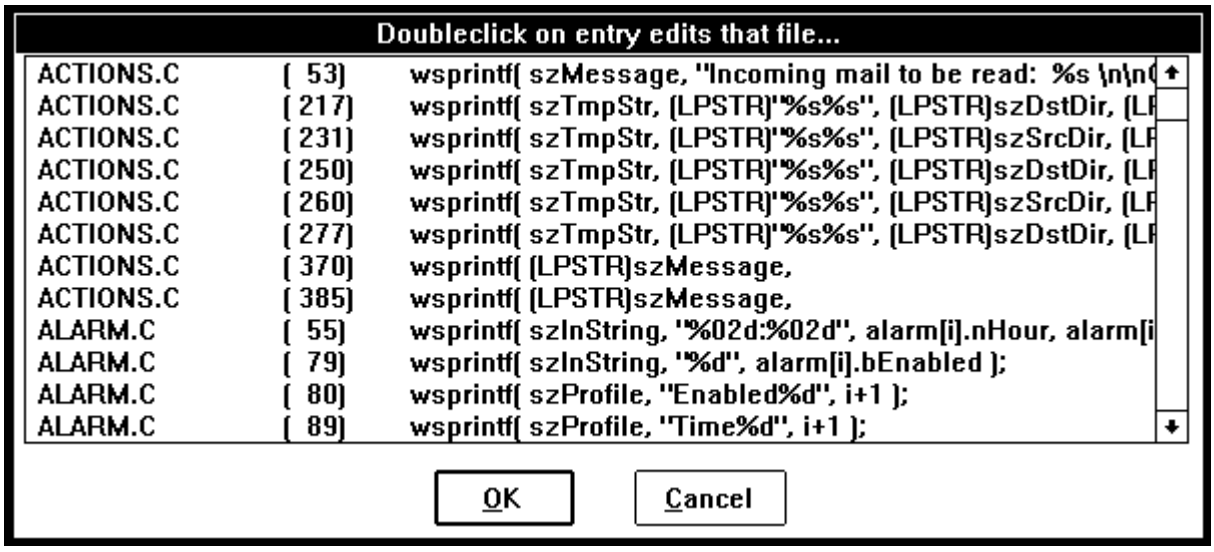
Use quotation marks to enclose strings that include

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":



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 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 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 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 5 of this manual, which is the reference to the entries in CTSHELL.INI.

Note that you can also provide these settings (and many others) in the *Preferences* dialog box that is presented when you invoke the PREFER keyword. In the default configuration, remember that PREFER is bound to the F11 function key for easy access.

FormFeed

It is quite often useful to send a command to the printer that tells it to eject the current page. For example, if you use the COPY command to send a text file to your printer, it will probably stop printing somewhere in the middle of the last page. With the FORMFEED command, you can easily tell your printer to eject that page.

There is also a FORMFEED keyword, so this feature can easily be implemented from a menu item. If you happen to have the command line open, however, and have just copied a file to the printer, it may be more convenient for you to type the FORMFEED command instead of looking for the menu entry.

Shred

This command first overwrites every byte in a file with a pattern

that leaves it unusable, then deletes the file. Thus, even if someone manages to undelete a file that has been shredded, it will be useless. This takes longer than simply deleting a file, so it should be reserved for those times when data security is an issue.

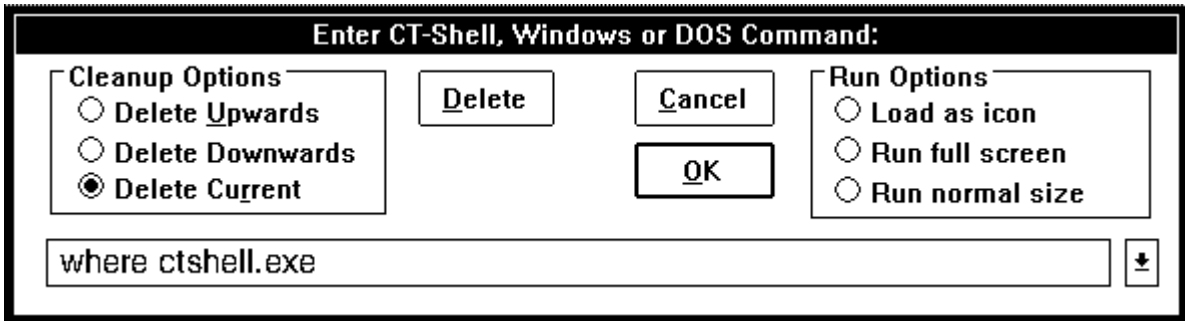
Because the effect of this command cannot be reversed, it does not accept wildcard arguments on the command line. You may shred multiple files with one command, but you will have to explicitly name each file that is to be shredded.

Move

If you want to move a file quickly from one place to another, rather than copying it, you can use the CT 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:



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
```

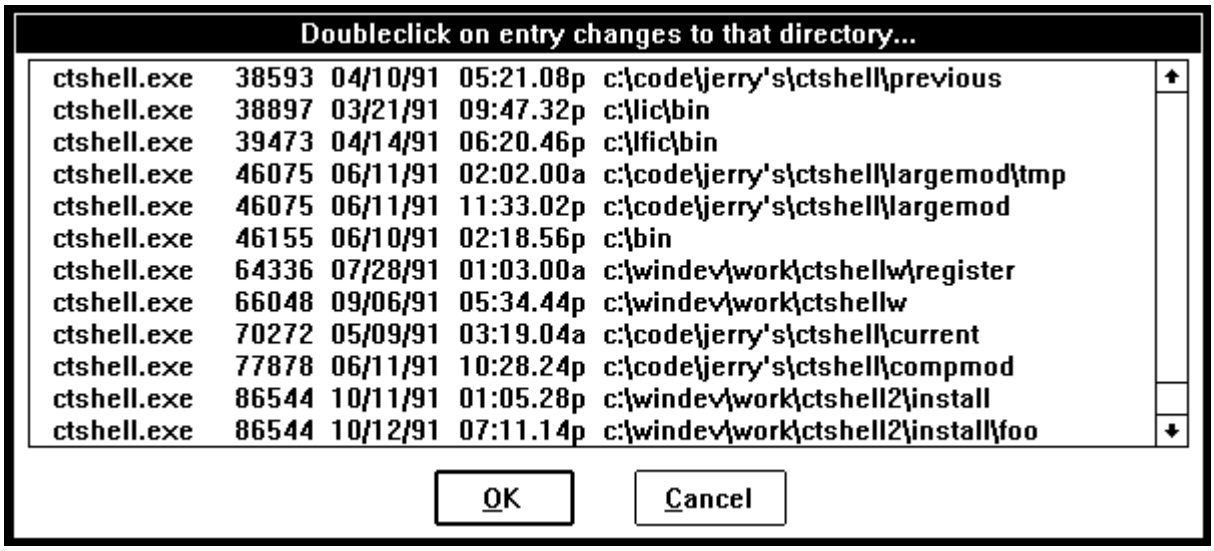
or

```
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 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 not to add the .* to the end.

After you've provided the name, the mouse cursor will change

temporarily, letting you know that CT is busy as it searches the current disk drive for that file. CT will display information in a list box about all the matching files that it finds:



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 WHERE command.

Run Mode

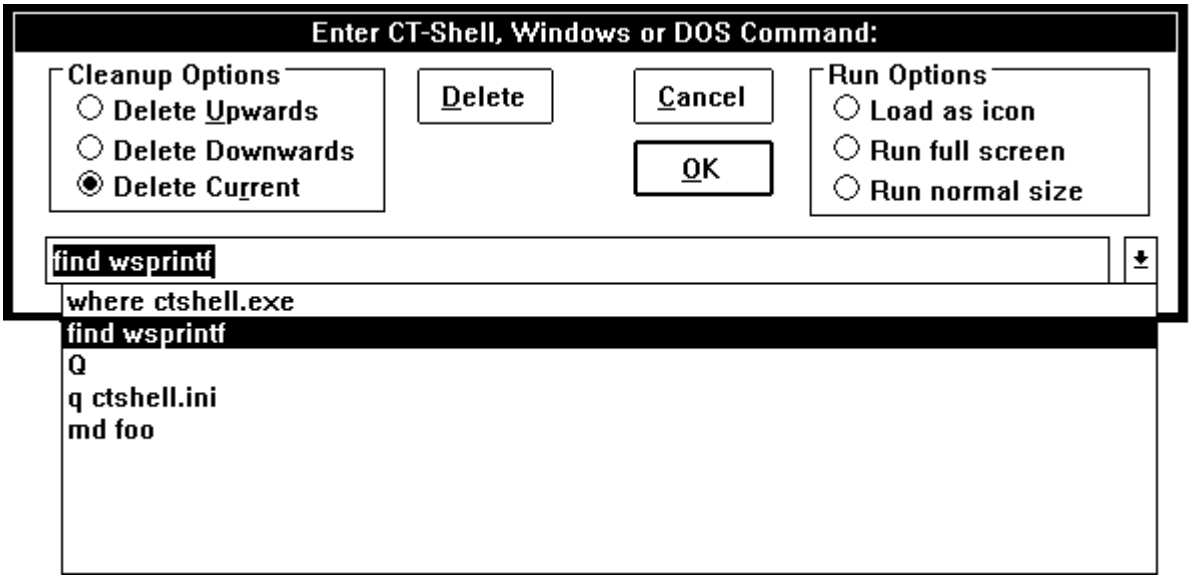
If you are issuing a command to run a Windows application, you can select from the options in the upper right-hand corner that allow you to run that program as an icon, or full-screen, or normal size. (Actually, the default is normal, if you don't select either of the others.)

This set of options has no effect on DOS commands, or when you run DOS programs. The latter are run according to their PIF files, if such exist, so the place to determine whether a DosApp is

to be run in a window or full screen is in the PIF.

Command Recall

CT 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.



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 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 5

CTSHELL.INI Reference

The real power of CT-Shell is unleashed when 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 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.

Accessing CTSHELL.INI

Far removed from the programming that is required to customize some similar products, CT-Shell lets you work with simple objects (characters like ! and #) that take on special meaning when you use them in your commands. Even those who have never written a program, a script, or a macro are encouraged to give this a try.

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, allowing you to make changes in the menu system and add new entries. If you have stored it somewhere else, simply change to that directory, select the file (make it the *current 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 CTSHELL.

Be aware that there are several CT dialogs that also affect portions of your CTSHELL.INI file, especially those sections that affect the way options are implemented (as opposed to menu items and entries). Note that any of these other option settings can always be changed by editing CTSHELL.INI, but usually the dialogs provide the easier way to make such choices.

For example, the way your printed listings are formatted is affected by a dialog that is invoked by the CONFIG keyword. The alarm/event settings are affected by a dialog that is invoked by the ALARM keyword. The PREFER keyword invokes a dialog that allows you to set a great number of preferences, all of which will be explained in the sections that follow.

The following sections describe the various parts of your CTSHELL.INI file, and what changes you might want to make to each of them. Where those changes can be made with a dialog, it will be identified, along with the CT keyword that invokes it.

[Alarm]

This section is new for version 2.0 of CTSHELL. It contains the controlling data for up to four event timers that work like alarm clocks. Timers are a limited resource in Windows, with only 16 available for all programs that need to use them. You might be concerned that CT uses more than its share of timers, but don't worry– it uses only one. The event timing is handled by the same part of the program that updates the time in the status display.

The default CTSHELL.INI file does not contain data for any events or timed messages. What you'll see at the beginning of this file is an [ALARM] section that looks like this:

```
[ALARM]
Enabled1=0
Enabled2=0
Enabled3=0
Enabled4=0
Time1=00:00
Time2=00:00
Time3=00:00
Time4=00:00
Message1=
Message2=
Message3=
Message4=
Event1=
Event2=
Event3=
Event4=
```

The easiest way to set up a timed event, or to have CT display a

reminder of some kind at a predetermined time, is to use the ALARM keyword to invoke its dialog. There is a graphic of that dialog later in Chapter 6, which is about CT keywords.

[Autoexec]

This section has already been thoroughly documented in earlier sections of this manual. It contains a series of entries (in the usual menu entry format), that determine the programs that will be started automatically when CT is first started. Remember that you can use the ICON or LOAD keywords *in addition to* the name of a Windows executable, which causes that program to be loaded as an icon, instead of being run in its normal size.

This section is unaffected by any of the dialogs that change various settings. The only way to place entries here is to enter them with an editor. Many people find it easy to do a copy-and-paste operation, to copy entries from another section of the file, such as from the later menu section.

[Color]

There are three entries here, named as shown below, which each contain one component of the background color that's used for CT-Shell's main window.

```
[COLOR]
BkRed=255
BkGreen=255
BkBlue=235
```

These are the three primary additive colors, and the intensity of each color ranges from 0 to 255. If all three were 0, the background would be black. If all three were 255, the background would be white. Various combinations of other values will create a whole spectrum of background colors, subject to built-in limitations of the video driver in use.

The default combination used in the distribution CTSHELL.INI file creates a background that is pale yellow (red + green = yellow). These values are among those that can be changed via the *Preferences* dialog, which is invoked by the PREFER keyword.

[Editor]

The section marked [EDITOR] contains two settings that tell CT 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 to find for you. These settings look like this in the sample CTSHELL.INI file:

C:\WINWORD\CTSHTOC.DOT

```
[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 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 will simply load the selected file into the editor, but not attempt to start at a particular line number. If EditLine contains any characters, they 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 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 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. These values are also among those that can be changed via the *Preferences* dialog, which is invoked by the PREFER keyword.

[Modem]

One of CT-Shell's functions is to store phone numbers in a dialing directory, and optionally to dial phone numbers for you, using a modem[13]. Nearly any kind of modem can be used for this, including the cheapest ones. There are several entries in this section that can be changed, if necessary, to accommodate unusual or nonstandard modems, or phone systems that require additional characters to be

dialed to access an outside line or to charge the calls to a credit card.

The sample CTSHELL.INI file includes most of these settings, which will be satisfactory for a large number of users:

```
[MODEM]
ModemInit=ATQ0M1L0V1X4&C1&D2
ModemSpeed=1200
ComPort=1
DialPrefix=ATDT
DialSuffix=
```

The one that is *not* included is the modem initialization string–the first field shown above. If you want to activate this feature for your version of CT, you must have a modem connected to your computer, and you must provide an appropriate modem initialization string. If you're not sure what your modem needs in an initialization string, the one shown here is a good place to start experimenting.

Assuming the very popular and common Hayes "AT" command set, the modem initialization string shown here turns quiet mode off (Q0), the modem speaker on (M1) at a low volume (L0), and asks for verbose answers from the modem rather than numeric result codes (V1). The extended result codes (X4) let many modems wait a reasonable length of time for a dial tone. The &C1 and &D2 control the way the modem handles some of its handshake lines.
The default file sets the modem speed to 1200, which is pretty much a lowest-common-denominator among modern modems. Since no communication is going to take place at this speed, it doesn't really matter that the modem may have higher speeds available. We're only passing a dialing command to the modem.

The default communication port is COM1, however, if your modem is connected to a different port you'll want to change this. If your phone uses tone dialing–as most of them do these days–the ATDT dial prefix is probably all you will need. If your phone system uses pulse dialing (a rotary-dial phone), you'll want to change that to ATDP instead. The dial suffix is usually not required, but some may need to add some more numbers here to accommodate the special needs of a particular private exchange.

Note that there is no need to specify carriage returns at the end of the strings that are sent to the modem. Those are added automatically by CT, when the modem initialization string is sent, and when a dialing command is sent.

The settings you see above are all defaults that are built into CT internally. In other words, you could leave them out of your CTSHELL.INI file entirely, and they would still be the values used. The one exception is the modem initialization string, which defaults to the empty string if it is not provided specifically.

That's important, because that string of characters determines whether CT will try to dial the phone for you, or just display phone numbers for you to dial yourself. The PHONE keyword invokes a dialog box that displays phone numbers that are stored in your

CTSHELL.INI file. If CT has a valid (not empty) modem initialization string, it will offer a [Dial] button in the PHONE dialog that does not show up if that string is empty. For more details, see the description of the PHONE keyword in Chapter 6.

[Options]

Here is a place to set a variety of options that affect the way CT works. Nearly all of these entries may be changed via the *Preferences* dialog, which is invoked by the PREFER keyword. The default CTSHELL.INI file contains the following entries in its [OPTIONS] section, which will be discussed in the following paragraphs:

```
[OPTIONS]
KeepOpen=0
RequireConf=0
IgnoreDrives=
Mail_In=
Mail_Out=
user1={Alarm Settings} {} {} {} {alarm}
user2={Listings Config} {} {} {} {config}
user3={Preferences} {} {} {} {prefer}
user4={Printer Settings} {} {} {} {printer}
```

The entries for KeepOpen= and RequireConf= are TRUE/FALSE type entries, in which a 1 represents TRUE and a 0 represents FALSE. There will be other examples of this type of entry in the sections that follow.

KeepOpen

KeepOpen refers to the command line, and whether it should be kept open following a command, so that additional commands may be entered without reopening the command line dialog. Some may prefer to keep the command line open, particularly if they regularly issue a lot of commands in series. Others will prefer to let the command line close after each command, since it can easily be reopened with <Shift+Enter>. With this entry–which can be selected from the *Preferences* dialog–each user can make that choice.

RequireConf

RequireConf refers to closing CT by doubleclicking on the system menu box. That's the bar-in-a-box that's located in the upper left corner of the window in which CT runs. Some will prefer a message box that asks, *Are You Sure?* and requires a keypress or a mouse click to confirm. Others will want to exit from CT without further ado. It is unlikely that an abrupt exit from CT could harm anything. This option may be changed from the *Preferences* dialog.

IgnoreDrives

The IgnoreDrives setting has already been discussed in earlier sections. If you have a system in which there are a great number of drives available–such as in a network–and you do *not* need a continuing display of the available space on those drives, you may instruct CT to ignore certain drives. The example shown here would cause CT to ignore drives from F: on up:

```
IgnoreDrives=FGHIJKLMNOPQRSTUVWXYZ
```

The drive letters in the string that follows the equal sign do not need to be in any particular order, though keeping them alphabetical makes maintaining the list a little easier for the user. Uppercase and lowercase letters may be used. If a drive does not exist for the current system, including its letter will have no effect, and will cause no harm.

Note that telling CT to ignore certain drives *does not* make it impossible to work with those drives. It simply means that those drives won't have to be queried every time the files listing is updated. You can still change to any of the "ignored" drives in the usual way from the command line, or by pressing a key combination that includes <Shift+Ctrl> plus the drive letter. For example, even given the IgnoreDrives option shown above, you could easily change to drive L: (if it exists on your system) by pressing <Shift+Ctrl+L>.

The current drive is always updated in the status window, even if it is one of the drives that is to be ignored. This setting may be changed from the *Preferences* dialog.

MailIn and MailOut

These two entries will not be useful for everyone, but for those who regularly work with files that contain network mail or email of any kind, the CT feature that uses them can be quite helpful. Because of the way this mail notification feature works–by reporting on the presence or absence of two specified files–users may find applications for the feature that have nothing to do with network mail.

One of CT-Shell's keywords is named MAIL, and invoking that keyword in a menu entry causes CT to look for the presence of files that are identified by these MailIn= and MailOut= entries. It then presents a message box that states in simple *Yes/No* terms whether mail is waiting to be sent (MailOut), or mail is waiting to be read (MailIn).

The obvious application is for those who work with a network mail system, or who download mail packets from a remote service like CompuServe or a BBS. Often these systems create a file of mail that is waiting to be read, and the fact that the file exists implies that there is mail waiting. Those systems that do, usually also create a file of answers, which are then transferred–such as by telephone or network connection–with the remote "host" system.

Again, the existence of a file containing replies often implies that there are answers that have not yet been sent.

Users who engage in this type of mail activity will be probably be able to figure out, based on this description of the process, how they can use the MailIn= and MailOut= path names to their advantage. It may also have occurred to others that it would be useful to be able to check quickly on the existence of two specific files for totally unrelated purposes. Anyone who is able to utilize these settings to their advantage should feel free to do so, network mail or not.

These path strings may both be modified using the *Preferences* dialog.

User1 through User4

As you know from the previous discussion in the section on function keys, the User1= through User4= tasks are originally assigned to the special CT keywords, ALARM, CONFIG, PREFER and PRINTER. Doing that provides quick and easy access to these configuration dialogs.

These *are* user-defined options, nonetheless, and any entries that are suitable for a menu are suitable here. Thus, four function keys–F9 through F12–can be used to invoke commands that you would normally select from a menu.

These settings may be modified through the *Preferences* dialog, however users may find it easier to use the copy-and-paste capabilities of their editors to copy entries from the later [ITEMS] section in the CTSHELL.INI file to this section.

Because of limited space to display the entry name next to the representation of the function key, entry names should be modified, if necessary, to be no longer than 18 characters or so. Even shorter names may be required if there are many wide characters, such as capital letters. This is the only way in which a User*x*= entry might need to differ from a regular menu entry–a menu can accommodate a much longer name[14].

StartX and StartY

These two settings are not part of the default options, for a good reason. If they are left out entirely–i.e., do no appear in the CTSHELL.INI file at all–CT will automatically start in a centered position on the screen.

Some may have reason to want it to appear in another position, so CT is able to memorize its current position and use that whenever it is started. If you invoke the CT keyword POSITION, these entries will be made in your CTSHELL.INI file for you. If they exist, CT will use them as the position of the upper left corner when the program is started.

The values will vary depending on what type of video monitor is in use. For example, if centered on an 800x600 monitor, POSITION will record these values:

```
StartX=92
StartY=76
```

If these two entries do exist in the file, but no values are provided for them, they are interpreted as 0 and 0, placing CT in the upper left corner. That's why the entries do not appear at all in the default configuration. If you have recorded a starting position and later decide to return to the centered default position, you will need to remove these lines from your CTSHELL.INI file entirely–just removing the values won't be good enough, as missing values are interpreted as zero values.

Naturally, if you are going to record a new starting position for the program, you will want to move it into your preferred position before you invoke this keyword. You can do that by "grabbing" the caption bar at the top of the window (move the mouse cursor there, then press and hold the left mouse key) and "dragging" it to the position you want.

Unlike the other settings in this section, these are not affected by the *Preferences* dialog. They are changed only by the POSITION keyword, or by editing the CTSHELL.INI file itself.

[Phones]

This section contains a single entry in the distribution configuration:

```
[PHONES]
Phone0=Computer Training          (206)820-6859
```

The entries here are made available to you in the dialog that is invoked with the PHONE keyword. There is a menu entry for that purpose in the default configuration, in the *Shells* menu.

A graphic of that dialog and more details pertaining to its use are in the later section on CT keywords (Chapter 6), but here are a few observations about the entries that may be used here:

The approximate size of the entries will become apparent when you first invoke that menu item. Subsequent phone numbers are put into place as Phone1= , Phone2= , Phone3= , etc. Legitimate characters for the phone number include the digits, hyphen, parentheses, and the period. A phone number must be the last entry on the line, and it may not contain embedded spaces. (CT figures out which is the phone number by moving to the end of the line, then backing up as long as it continues to find legitimate phone number characters. It will stop at the first space it finds.)

You may feel free to edit this section directly in your CTSHELL.INI file, and that may be the easiest way to enter a long list of phone numbers. You may also modify this section from the dialog that is invoked by the PHONE keyword.

[Printer]

The [PRINTER] section controls certain options that affect the way CT 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, from a dialog box that is presented when you execute a pop-up menu entry that uses the CT keyword CONFIG (which is described in Chapter 6, which follows next). In the default configuration, that keyword is bound to the <F10> function key. 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, and it specifies the width of text file lines, as you usually work with them. When printing text files, CT 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 does not provide linewrap, so lines of text that are too long may 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, you may edit this entry to contain values other than 80, 110 or 132. CT will attempt to use your value, providing the closest font that it can.

Note that there is another special CT keyword called PRINTER– which is available from your *Utilities* menu, and is assigned by default to the F12 key–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 of the output from one of those functions is shown in the section on CT keywords in Chapter 6.

The rest of the options shown here are simply TRUE/FALSE values, where the number 1 represents TRUE and 0 represents FALSE. Inside CT, 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 ones and zeros.

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=1, each file listing will begin with page 1. If Independent=0, 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= 0.

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=1, lower quality will be allowed.

Note that the operational word here is *allowed*. CT 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 keyword PRINTER 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=1, 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 main menu, and the pop-up menus that its items invoke. There is no arbitrry 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 arbitrary 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 sections, 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, parentheses, and most other 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.

Item names may be any practical size. Keeping them brief, however, helps to keep the main menu on a single line, and many people find a single-line main menu easier to use. Note that CT will automatically detect a multi-line main menu and adjust its window size accordingly.

Pop-up Entries

Each pop-up 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. To load CTSHELL.INI into the NOTEPAD editor from the default menu, select the *Edit* menu and choose the entry to edit CTSHELL.INI.

Entry Name

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

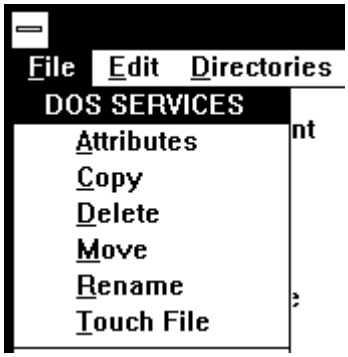
The entry name is displayed in the pop-up menu to allow the selection of this option. Like the menu item name, the entry

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 if it is used, it provides a quicker way to invoke this entry.

The entry name may also be preceded by a dollar sign ($), which has a special meaning. Such an entry name is always followed by four more sets of empty braces (or at least, CT ignores anything in them), and is used only as a *label* for a section in the menu. Thus, a number of commands to invoke various built-in DOS services might be grouped together under such an entry as in this example from the default configuration:

```
{$DOS SERVICES} {} {} {} {}
{    &Attributes} {} {} {} {ATTRIB}
{    &Copy} {} {} {} {COPY}
{    &Delete} {} {} {} {DELETE}
{    &Move} {} {} {} {MOVE}
{    &Rename} {} {} {} {RENAME}
{    &Touch File} {} {} {} {TOUCH}
```

That example is responsible for part of a menu that looks like this in use:



Two other special characters may be used here, to provide separation from other pop-up 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 pop-up 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 *Windows* menu, the *Utilities* entries are separated into their own

column, distinguishing them from the applications and system programs in the other column.

Note that spaces are considered ordinary characters, and can be used to provide indentation for a list of entry names that come under a label, as they were in the example just above. The location of the - and + symbols is important, as they are considered special characters only if they appear in the first or second columns. If embedded into an entry name, they would be considered ordinary characters.

In contrast, the $ must always be used as the first character in the entry name, but it can follow a - or a +, if those are used to provide separation. And, of course, the & retains its special meaning wherever it might appear in the entry name. Here are some practical examples of these rules and considerations:

{-$UTILITIES} {} {} {} {}

...starts a new section in the current column, using UTILITIES as the label for a group of entries.

{+$COMPRESSION} {} {} {} {}

...starts a new column, using COMPRESSION as the label for a group of entries that have to do with data compression.

{- LHArc &Add} {} {} {} {}

...starts a new section in the current column, separating it from previous sections with a horizontal bar. This entry will be indented by four spaces. The 'A' character will be underlined, and recognized as the significant letter to access this menu item.

The default configuration contains a number of examples that may guide you in developing new entries for your own programs.

DirPath

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

The directory path is an optional field which, when provided, causes CT to change either temporarily or permanently to that directory, before executing any program that may be part of this entry.

If a directory path field is present and an executable path field is not, it is assumed that the explicit purpose of this entry is to change directories permanently. In that case, CT will change to the specified directory, and will continue operating from 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 path has a content, CT will change to that directory when this entry is invoked. Since the executable path 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 will return automatically to the directory where it was before the command was executed. Here's an example where the same directory change is made, but where a session with Word for Windows is also started:

{&WordProc} {d:\winword\letters\personal} {winword.exe} {} {}

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

ExePath

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

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 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. If a full path were needed, such as to run a program that does not reside in a directory along the DOS path, its entry might look like this:

```
{&Calculator} {} {d:\foo\bar\progcalc.exe} {} {}
```

Switches

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

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 special field characters are most often used (see Chapter 7). 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 to prompt you for one or more values to be inserted as it runs.

As it operates, CT 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 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} {}
```

An exclamation point used in this way gets replaced by the current file, which we would assume to be a directory in this

usage. The a: is a destination, indicating that the copy should be done to drive a:. The /s tells XCOPY to copy all files in subdirectories, creating those subdirectories as needed on the target disk. The /e switch tells XCOPY that it should even preserve empty directories during the copying.

Keyword

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

Many of the operations that CT performs are handled internally by CT 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 almost always used in place of–rather than in addition to–an executable path. In fact, if there is a keyword in this last field other than those few that specify the run mode for a Windows executable, CT will process that keyword first, and ignore any entries that may be in the executable path or switches fields.

Because CT-Shell keywords provide so much of the functionality of the program, the whole next chapter is devoted to an explanation of them.

Chapter 6 CT-Shell Keywords

CT-ShelI keywords are listed here, along with a brief description of what they each accomplish, and an example of any dialogs that are invoked by them.

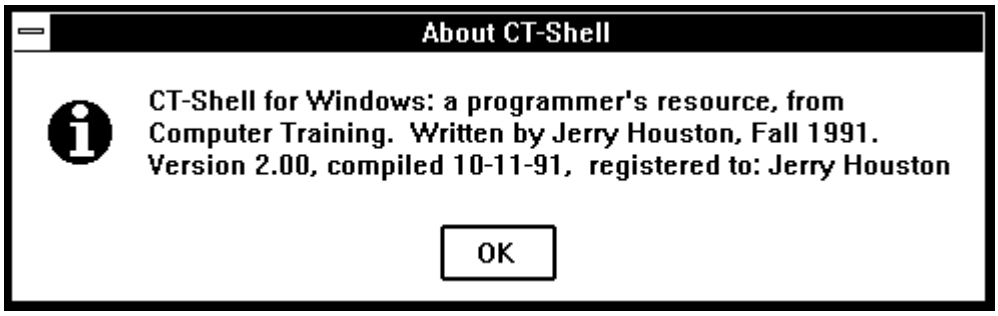
Single and Tagged Files

When a keyword is designed to affect a single file, CT needs only to inquire which file is the *current file* in the file list, and apply that operation to that single file. When a keyword is designed to affect a list of tagged files, CT needs to go through *all* the entries in the file list and ask of each, "Is this one tagged?" Then it applies the operation to the ones that are, skipping the ones that are not. Obviously, it is easier and quicker to affect only the single current file.

Keywords that are intended for use with tagged files will almost always work with the current file, which is almost always considered to be tagged[15]. Examples of most 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:

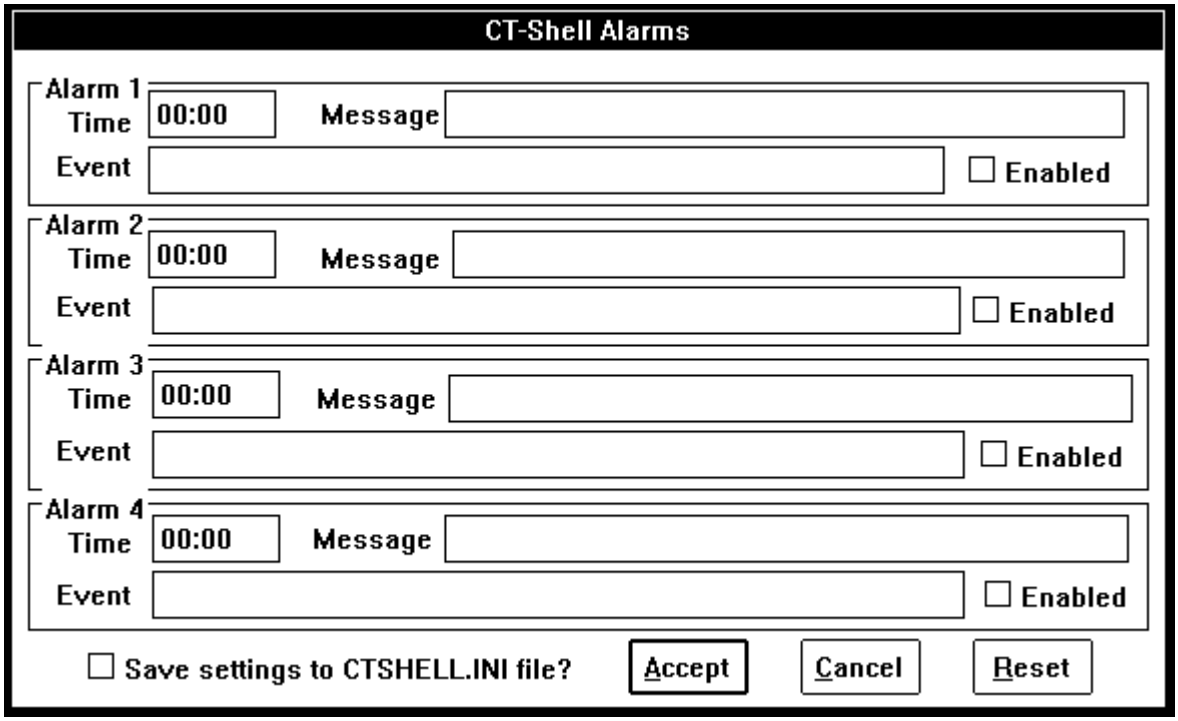
About

Displays information about CT, including the number of the version that you're using. The dialog that it invokes looks like this:



Alarm

Allows the user to modify the settings used by the four event timers that CT provides. If a time is established and that timer is enabled, a message entered for that timer will be displayed at the specified time, and any menu-type entry placed in the event field will be executed at the stated time. Note that messages can be displayed without starting an event, and events can be started without displaying a message. The dialog presented by ALARM looks like this:



Most of these fields should be pretty obvious. For example, to create an event you must enter a time for it to happen, and then enable it by clicking on the *Enabled* check box. If you only want to be reminded of something, you can enter a message that you would like displayed in a message box at that time. Enter the time in HH:MM format, and use 24-hour time. In other words, 9am would look like 09:00, while 9pm would look like 21:00.

If you would like to execute a program at the time you specified, you'll need to type in an entry for the *Event* field. This entry takes the same form as the ones you've already seen used in the menu section of your CTSHELL.INI file, with five sets of braces (and which you've seen in other examples in this manual). You might want to include an entry name in the first set of braces as a reminder, but CT won't need it.
Your event can execute DosApps and WinApps, or invoke a CT keyword. In short: you can do anything here that you could do in a menu entry.

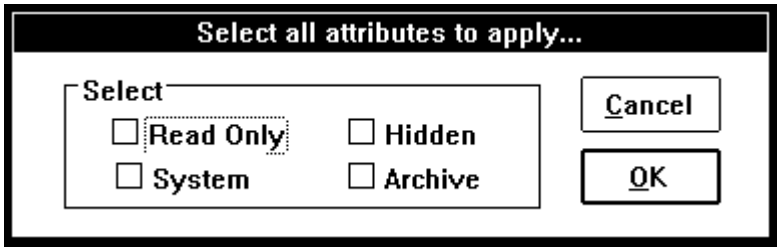
Having created an event or scheduled a reminder message, you can click on [Accept] to accept those settings and close the dialog. If you have also checked the box marked *Save settings to CTSHELL.INI file?*, the event will be recorded for future use, and reloaded whenever CT is started. If an event is enabled, it will be triggered the next time that time comes up, and will be triggered again the next day at the same time. In this way, an event can be established that will be automatically done every day, so long as CT is running when the time comes.

If you have made some changes and find that you need to return to the settings that were previously recorded in CTSHELL.INI, you can click on the [Reset] button. Clicking on [Cancel] exits from the dialog without changing anything.

The limit of four events is somewhat arbitrary, as the data required to set up four events fit nicely into a dialog box that fits on any screen. However, it isn't difficult to create more events if they are needed. Remember that you can run multiple instances of CT, and that all use the same copy of the program, but each has its own private set of data. That means that one of the events could start another instance of CT, which invokes three other events *plus* another instance of CT, which invokes three other events *plus*...

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.



Remember that these attributes *become* the attributes for the file that you are modifying, they're not added to the existing attributes. Be sure to set all the ones you want to apply, then click OK.

Command

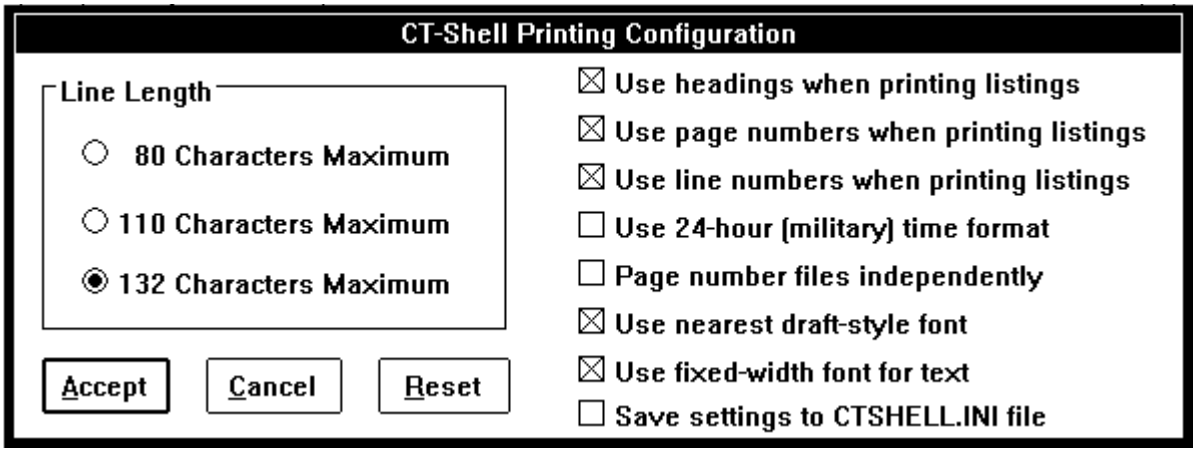
Invokes the command processor that is associated with your COMSPEC environment variable. In most cases this will be COMMAND.COM, the command processor that is supplied with MS-DOS and PC-DOS.[16]

If there is anything at all that you prefer doing at an ordinary DOS command line rather than from within CT, 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 that affect how file listings are printed. See also the PRINTER keyword for access to the printer driver itself, which provides control over your installed printer.

The menu entry that invokes this keyword is in the *Shells* menu in the sample configuration. When you invoke the keyword CONFIG, here's the dialog box that CT uses to get your choices:



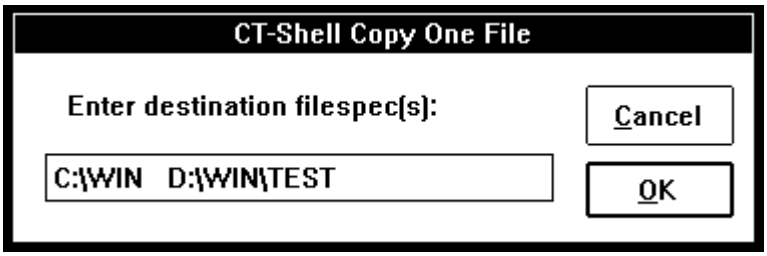
If you click on the [Accept] button, CT will accept the settings that are shown. If you have checked the box marked "Save settings to CTSHELL.INI file," that will be done as well, so you'll start off with those as default settings the next time .

If you click on the [Reset] button, you will cause CT 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.

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 command line, this uses a much larger copy buffer than DOS does, for better

efficiency.



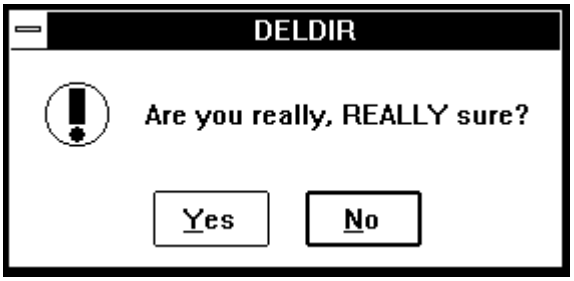
Note that you may provide multiple destinations when this dialog asks you for a destination. The example shown here would copy the current file to both the C:\WIN path and the D:\WIN\TEST path.

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).

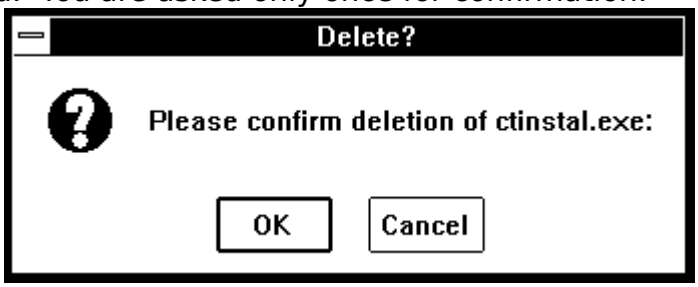
The entire subdirectory will be deleted, *including any files in it and any subdirectories under it*, even any files that have the read/only attribute. This is a wonderful way to remove an outdated or unwanted directory during disk maintenance, but it requires you to be careful. Files and directories that have been deleted with this command can sometimes not be undeleted.[17]

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:



Delete

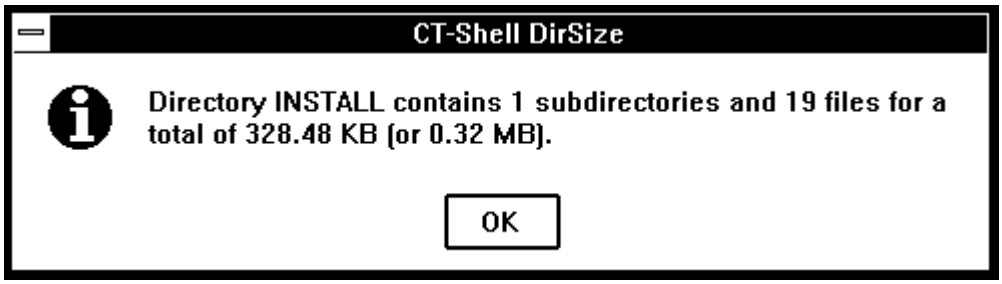
Deletes a file. This removes a file in a way that cannot usually be reversed. Be careful, and be sure that you mean it when you use this keyword. You are asked only once for confirmation:



Note that either a single file or a list of tagged files can be deleted by pressing the <Del> key, then confirming.

DirSize

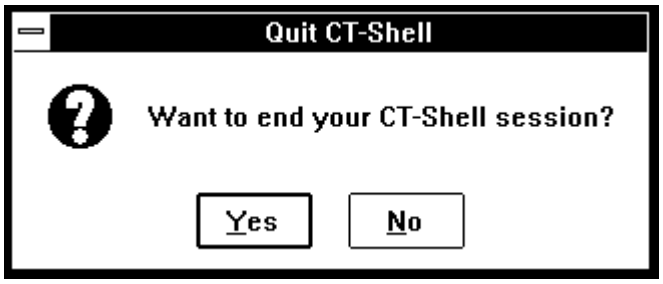
Displays a listing that shows you how big a directory is. It shows how many subdirectories it contains, how many files, and how many bytes they all add up to. This can give you a very good approximation of how much room must be available on a destination to which you plan to copy that directory, or how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



Exit

Shuts down CTSHELL. 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.

Users can choose whether a confirmation will be necessary to exit CT by modifying the RequireConf= setting via the *Preferences* dialog. If confirmation is required:



FileInfo

Copies the descriptive information (from the file list window) to the clipboard, from where it can easily be pasted into a document with nearly any editor, or used in other programs for which it is appropriate. This keyword is implemented in the *Tagged Files* menu. When it is invoked, information such as the following is copied to the clipboard for any files that are currently tagged:

```
ctcover.doc      5030  07/14/91  05:02.14a  .....
ctdisk.bmp     224918  07/14/91  06:38.44a  .....
fkeys.bmp      224918  10/06/91  04:04.14p  .....
header           1650  07/02/91  08:08.10a  .....
regcard1.doc     3481  07/19/91  05:28.38a  .....
```

FormFeed

When you have copied something to the printer, and it has not advanced the last sheet, you can force a formfeed with this keyword. It is invoked by a menu entry in the *Edit* menu of the default configuration.

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 "home" directory to the current directory, so that when you press <F6> this is where you'll return to, rather than the directory in which CT was started.

This is something you might want to do if you know that you must leave the current directory to do some things, but that you need to return afterwards. Having made the current directory "home," the return is easy.

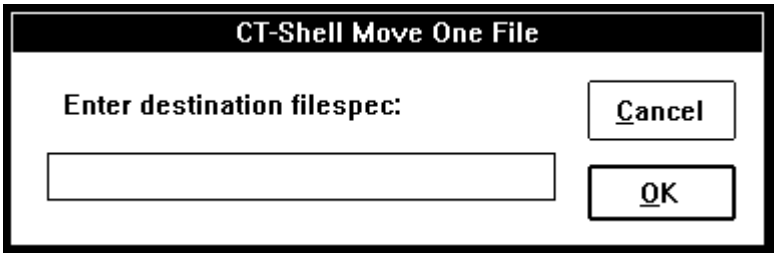
Mail

Displays a message box that tells whether there is mail waiting to be read, or waiting to be sent to a destination. This feature is actually only commenting on the existence of the two files identified by path names in the MailIn= and MailOut= entries in [OPTIONS] section of the CTSHELL.INI file. If the user does not participate in network mail or email, this feature could be used to determine the existence of any two other files, possibly serving some other purpose instead.

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.

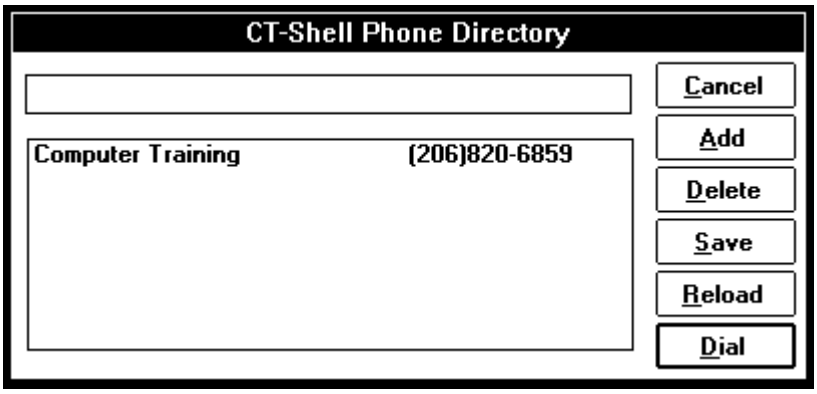
The destination is provided in response to this dialog:



Phone

The built-in phone directory can be used to recall numbers for manual dialing, and on a computer that is equipped with a modem, it can be used to automatically dial a listed number. Although a large number of phone numbers are more easily added to the CTSHELL.INI file with an editor, the dialog that is invoked by the PHONE keyword can also be used to add and delete entries. As well, it is used to look up numbers and optionally to dial them.

The dialog invoked by this keyword looks like:



The large box with the phone number for Computer Training is the list box in which will appear all your entries, sorted in alphabetical order. The small box above it is an edit field, in which you can type new entries to be added, or make changes to existing entries. (CT can also dial a number from the edit field.)

Doubleclicking on one of the entries in the list box will copy that entry to the edit field, making it easy to begin with one entry and modify it into others. (Perhaps one person or company you know has more than one phone?)

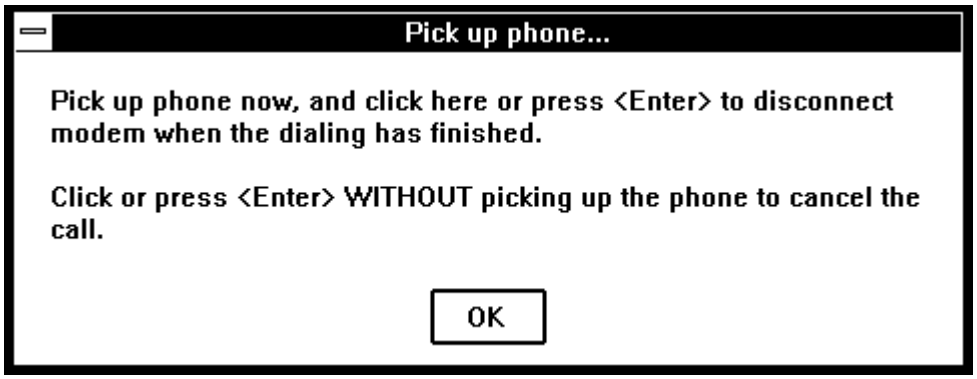
The controls at the right are all self-explanatory. The pushbutton for [Add] refers to the contents of the edit field, and adds that new entry to the list box. The pushbutton for [Delete] refers to the currently-selected entry in the list box, and provides a way to delete an existing entry.

To keep matters simple, there is no pushbutton to do a change. To change an existing entry, copy it to the edit field by doubleclicking on it, modify the entry, then select [Add]. Finish by deleting the old version.

Once you have made any changes, you will want to select the [Save] pushbutton to commit those changes to your CTSHELL.INI file. If you don't, they will disappear when you exit from this dialog. If you have made any changes you regret, and want to reload the original contents of the list box, you can do that with the [Reload] pushbutton.

The [Dial] pushbutton will appear only if the configuration includes a value for the ModemInit= setting. If that is the case, CT assumes that a modem is connected to the computer and that the rest of the settings in the [Modem] section of the CTSHELL.INI file are correct.

You can select one of the entries in the list box, then press the [Dial] pushbutton to have CT dial the number for you, so long as the edit field is empty. If there is an entry in the edit field, that entry will be dialed by default, even if one of the entries in the list box is also selected. When the dialing has been completed, CT will present a message box that looks like this, where the instructions are self-explanatory:



If there is a problem opening the port to dial the phone, you'll see a message that tells you so:

That means you should doublecheck the settings in your ModemInit= string in the [OPTIONS] section of your CTSHELL.INI file. That option can be changed via the *Preferences* dialog.

The fact that CT dials by default from the edit field makes it easy to enter and dial numbers without even adding them to your phone list. Simply type the number, which may be easier than using the phone in the ordinary way, and select the [Dial] pushbutton.

CT provides the additional benefit of handling long and complicated dialing prefixes and dialing suffixes with ease. Such additional numbers may be required to access lines from a complicated private business exchange, or to charge phone calls to a credit card or *calling card* number. Once set up properly, dialing from within CT can be a lot easier than dialing from the phone.

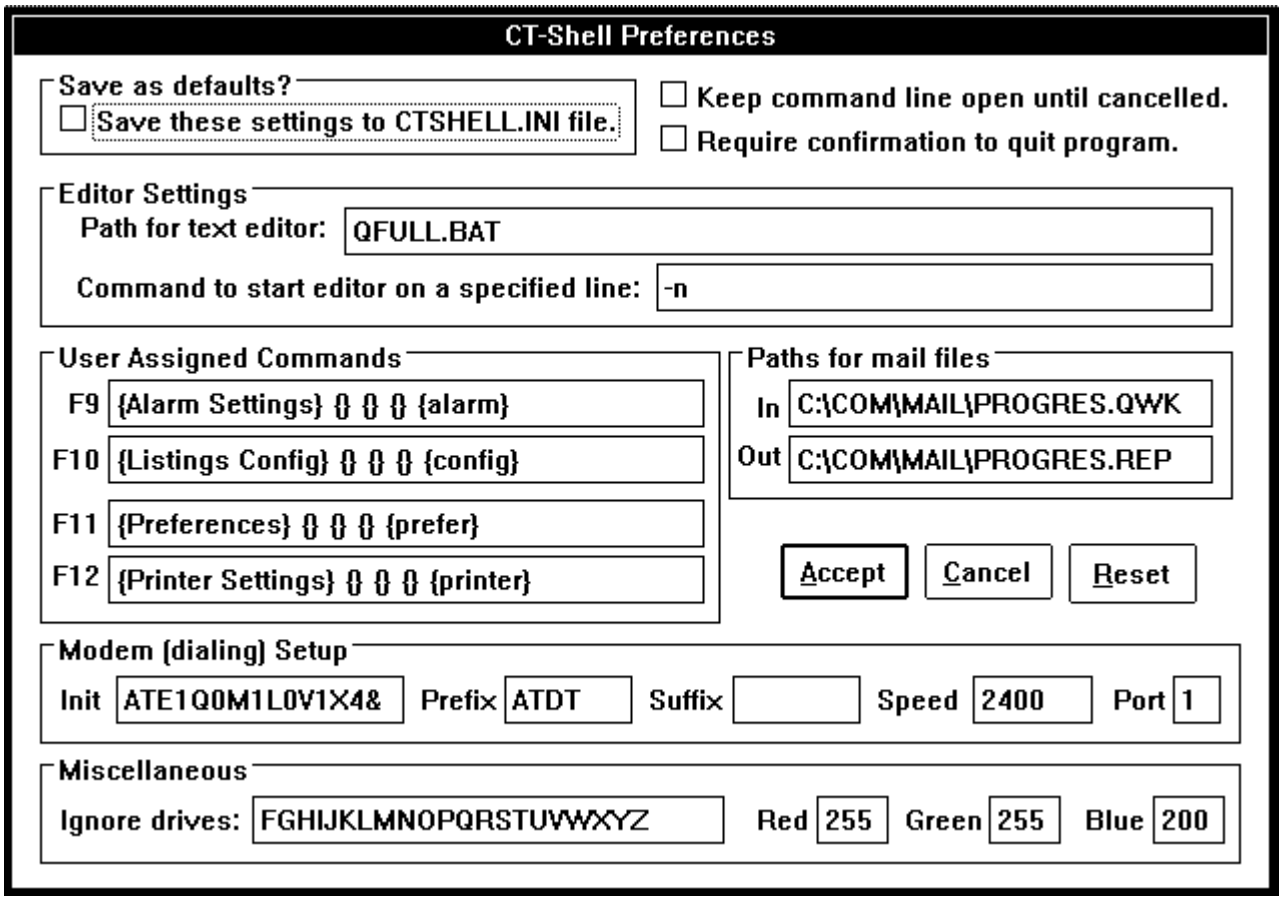
Position

CT defaults to displaying itself centered on the screen. Its main window is small enough to fit well using the resolution of any monitor that can be used with Windows.

Those with higher-resolution monitors, such as 800x600 and above, will have considerable choice regarding where to locate CTSHELL. By grabbing the caption bar at the top of the window with the mouse cursor, you can hold down the left button and drag CT to the location you prefer. Having done so, you can invoke the POSITION keyword by selecting the appropriate entry from the *Shells* menu, and thereby establish a new starting point for CT whenever it starts.[18]

Prefer

This one invokes the *Preferences* dialog that makes it easy for you to change many of the settings that are stored in your CTSHELL.INI file, without needing to load and edit that file with an ordinary editor. All of the entries have been explained in detail in the earlier section that described the contents of the CTSHELL.INI file:



If you make changes in any of these settings and click on the [Accept] pushbutton, those changes will immediately go into effect for the duration of the current session, or until you change them again. To make those changes permanent by storing them in the CTSHELL.INI file, you must check the box that indicates you want the changes saved. Then [Accept] will not only put the changes into effect, but will store them in CTSHELL.INI as well.

Note that some of the settings in the graphic just above were used as examples in other places in this manual, but are not necessarily used in the default configuration.

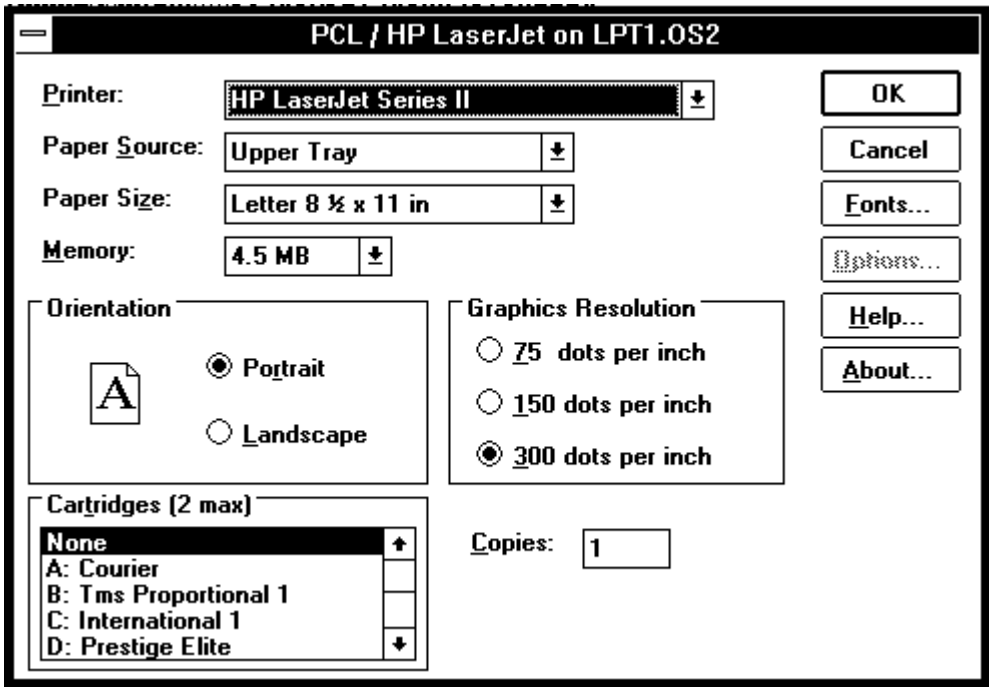
If you begin to make changes, and decide that you really need the original values back, you can select [Reset] to restore everything from the settings stored in CTSHELL.INI. Any changes made since the last

time your settings were saved will be lost.

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 that you use. This is probably where you can change from portrait to landscape mode, determine how high your graphic resolution should be, download font software, etc.

This 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:



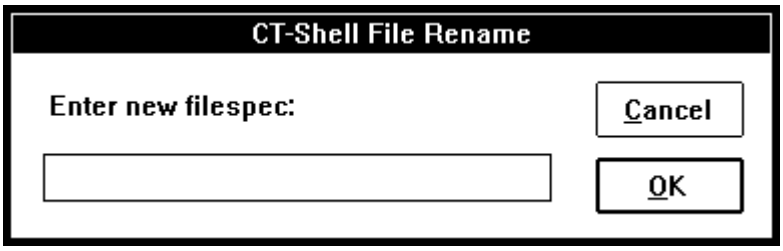
One other issue related to line size and fonts is your printer's resolution. Often printer drivers allow a graphic resolution that is less than the maximum possible, thereby speeding up printing of program listings, with an acceptable decrease in print quality.

If your CT 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.

Rename

Changes the name of a file, and unlike the DOS REN command, CT will change the name of a directory as well. 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.



SetDate

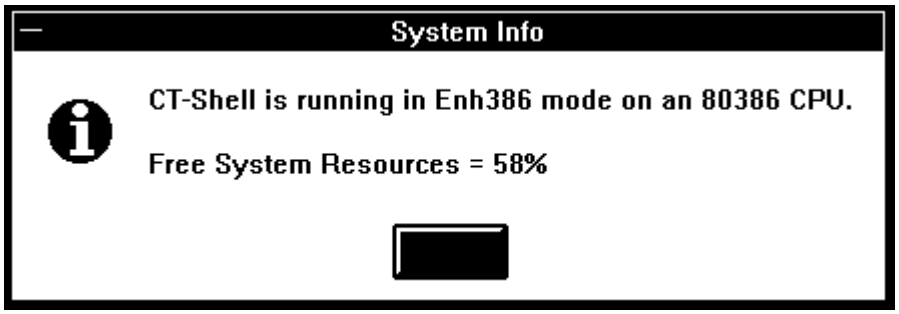
See the description of this one in the following section on keywords that affect tagged files. For this feature, the same function is used for single or tagged files.

Shred

This keyword first destroys the contents of a file, and then deletes it. In this way, the file is rendered useless, even if someone later manages to undelete it. Because the additional work takes extra time, this operation is slower than a simple DELETE, and should be used only when data security is an issue.
The dialog for this keyword looks almost exactly like the one for DELETE.

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 displays that at all times anyway:



Note that the percentage of system resources shown available here may differ by a small amount from that shown in the About box from

Program Manager, due to different ways of rounding the available numers. CT reports information that is obtained directly from the Windows kernel, and any difference should be so small as to be ignored.

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:

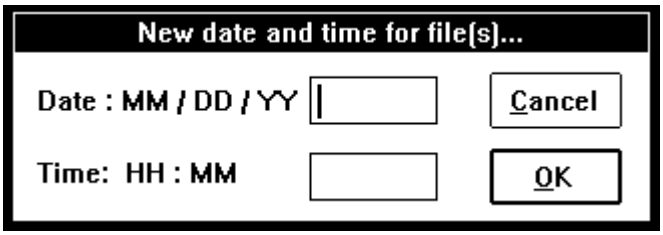
Touch

Makes the date/timestamp of the current file reflect the current date/time. This is mainly of interest to programmers, who sometimes need to adjust a date in this way while using a MAKE type program maintenance utility.

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.[19]



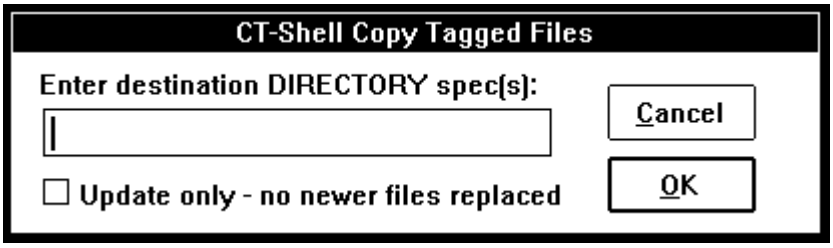
Tattrib

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.

The dialog for this keyword looks exactly like the one that is used to change the attributes of a single file.

Tcopy

Copies a set of files to another location. You must provide a directory as the destination. CT does not support file concatenation (combining several files into one) by copying multiple files to a single file[20]. However, CT does support multiple destinations for a multiple file copy. Put another way, you can tag an assortment of files that you want to copy to two places, then when the dialog asks for the destination(s), you can provide both. When the first series of copies has been made, the second will be done automatically.



Notice that there is a check box to allow you to specify an update only. If you do so, you are assured that only older files are overwritten by the ones you're copying now. In this way, you can safely copy a set of documents from a directory on your workstation to a directory on a server, knowing that you are doing just an update– that anything newer on the server remains unchanged.

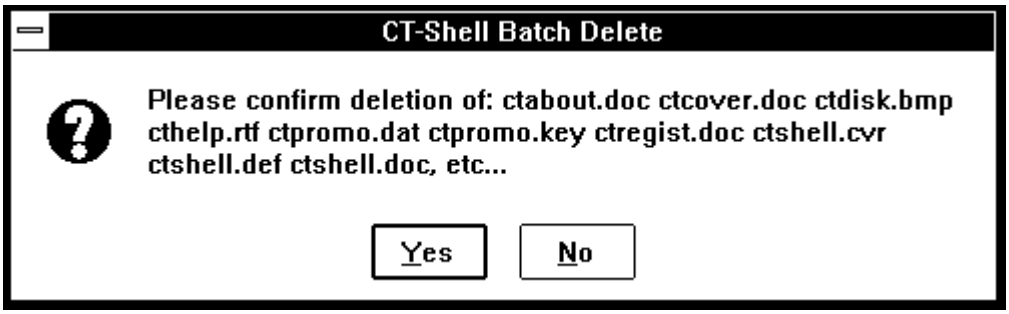
If you do an update from location A to location B, then do an update from location B to location A, you will end up with two identical directories, each of which contains the same set of the newest files from both.

This feature is really convenient if you have a main directory on a server for the storage of document files or program source files. You can work with copies of those files in a local directory, and easily

update the server with your revisions, without taking a chance that you'll accidentally overwrite any newer files.

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.

Tshred

Shreds a set of tagged files, first destroying the data in the file, then deleting it. In that way, its contents are rendered useless, even if someone later manages to undelete the file. This operation takes longer than simply deleting the files, so it should be used when data security is an issue.

Ttouch

Changes the date/timestamp of all the tagged files to the current date/time. Used mainly by programmers who use a MAKE type program maintenance utility.

These keywords have been provided so that you can have complete control over how your menus are crafted, rather than having CT 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. CT keywords are not case-sensitive: uppercase and lowercase work the same way.

(This page intentionally left unused...)

Chapter 7 Special Field Characters

Many times a command should contain the name of the current file, a list of all the tagged files, or other additional information. CT provides a powerful and easy way to generate complete commands, that doesn't require any programming knowledge or experience.

Object-Oriented Substitution

Sometimes it is convenient to refer to an environment variable within 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 pop-up 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 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 entry: {&Edit} {} {notepad.exe} {!} {}

#

The pound sign 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:[21]

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

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

@

The commercial at-sign translates to the root filename of the current file, without any extension. For example, if the current file were named FOO.EXE, the the {@} in a command would become FOO.

There aren't many cases when the root filename is needed, and programmers will probably find this one more useful than most other users. For example, if the current filename is FOO.EXE, the expression {@.C} would translate to FOO.C, and the expression {@.EXE} would translate to FOO.EXE.

If you do store all your .PIF files in a special subdirectory, you can create a menu entry that will allow you to edit the PIF file for any .EXE that happens to be the current file:

Command line: pifedit.exe \pif\filename.pif

CT entry: {&PIFedit an EXE} {} {pifedit.exe} {\pif\@.pif} {}

?argument?

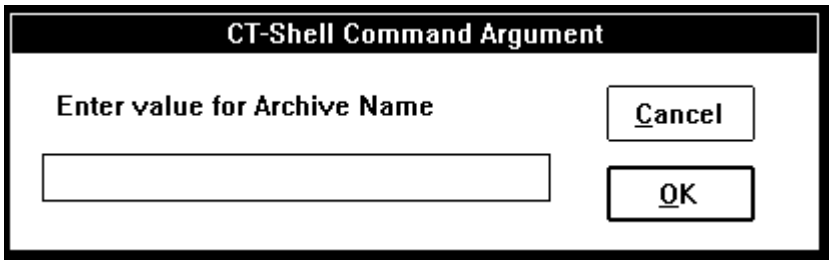
A pair of question marks surrounds the prompt you want CT to display when it asks you for a string of characters to put in its place. This is how you can supply variable arguments at the time an entry is executed.

For example, the LHArc command shown earlier 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 pop-up entry is executed, although the currently-tagged file names may be different each time. Compare that to this example, where CT will ask the user for an archive name each time the command is executed:

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

CT entry: {&LHArc Add} {} {lha.exe} {a ?Archive Name? #} {}

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

%variable%

A pair of percent signs will cause CT 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. Programs can obtain various kinds of information from environment variables, and the documentation for those programs will 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. Many programs use the TEMP environment variable to tell them the best place to create a temporary file.

When used in a CT pop-up 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 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 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 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 entry: {Edit PRG.H} {} {q.exe} {%INCLUDE%\PRG.H} {}

Finally, there's even a special CT 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 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] They make that change by replacing PROGMAN.EXE with FILEMAN.EXE in a SHELL= entry near the beginning of their SYSTEM.INI file, in the Windows directory.

[2] 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 85K in size, you'll find that the program actually requires about 35K or so to run, depending partly on your system.

[3] Subject to memory limitations, of course. Windows has to RUN all these programs!

[4] 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.

[5] 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.

[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 resolution available (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] A modem is a device that allows a computer to talk to another computer over ordinary phone lines. In this case, it isn't being used for that, but just to dial a phone number for a voice call.

[14] The names used in menu entries may be any practical length. Technically, CT-Shell limits them to 127 characters, but few will find any reason to use names that large.

[15] The current file is tagged if it is blackened. That's almost always the case, if you've clicked on any files in the list. The only way for the current file *not* to be tagged, is if you have used one of the function keys to untag it, either F1–Toggle Tag or F3–Untag All.

[16] 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.

[17] The UNDELETE command for DOS 5.0, for example, may not find a file that was deleted by CT-Shell. You may be able to recover a deleted file by using that or 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.

[18] If you ever want to reestablish the central position, go into your CTSHELL.INI file and in the [Options] section, delete the entries for StartX= and StartY= . The first of those determines the starting position of the upper left corner of the CT main window along the X-axis, that is, horizontally. The second determines the starting position along the Y-axis, or vertically.

[19] 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.

[20] DOS command processors like COMMAND.COM provide that capability. It is a feature that most people use so seldom, that a design decision was made to leave it out.

If you ever need to perform this unusual operation, you can easily select the *Default COMSPEC* entry in the *Shells*

menu, and start a copy of your default command processor. Then issue a copy command that looks like this: COPY file1+file2+file3+file4 file5 . That will create one file5 that contains all the text from the first four files.

If you ever need to do the same with non-text files, be sure to use the */B* switch to force a binary mode copy.

[21] 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 makes an excellent example for these special characters, because it gets a lot of information from its command line when it is run.