CT-Shell for Windows

from Computer Training of Kirkland, WA

"Making Windows Easy"

C:\WINWORD\CTSHTOC.DOT

Copyright

Trademarks

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 (Licensee) is granted a non-exclusive personal license to use this Software under the terms stated in this Agreement. The Licensee may transfer his license to a subsequent purchaser by sale, provided that 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 the Software or this 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.

Computer Training is concerned about the unreasonable cost of software used by individuals on small LANs, particularly in small businesses. One user of the Software, who may use it on several workstations in an office, should not have to purchase a license for each workstation. Several people in one location who use one copy of the Software that is on a single computer should likewise not be required to own multiple licenses. Multiple users of the Software on multiple computers should be able to purchase a site license with reasonable discounts. Our License Agreement addresses this issue in the following paragraph.

In environments where one or more users may access the same licensed copy of this Software, the required number of licenses shall be the LOWER of: (a) the number of users, or (b) the number of computers on which it is used at a single location. To clarify: one user of this Software who uses it on three computers in an office requires one license. Three users who access the same copy of this Software on a single computer require one license. Thirty users at one location who use three copies of this Software on three computers require three licenses, and they should contact Computer Training for information about site licensing. For the purposes of this paragraph, *computer* is taken to mean any computer on which the Software is actually used, whether a workstation or a standalone computer. A file server on which the Software is stored, but not used, does not require an additional license.

Limited 90-day Warranty

The diskettes and printed materials that are part of this product are warranted for 90 days against physical defects. To obtain a replacement for a defective product, return it to Computer Training within the 90-day period, with an explanation of the problem.

Computer Training 13215 NE 123 St, Ste 114
Kirkland, WA 98034 (206) 820-6859

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 the problem before returning your software to your vendor 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.

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 governed by the laws of the State of Washington.

CT-Shell for Windows
Table of Contents

(This page intentionally left unused...)

Chapter 1 CT-Shell Installation

This chapter describes the process of installing CT-Shell onto your computer. It is a very simple process, and because it is much easier to understand how CT 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.

The CT-Shell Rationale

CT-Shell was intentionally designed to be a program that does *not* use quite all the Windows bells and whistles. It was intentionally designed to be small, powerful, configurable and easy to use. But above all, CT-Shell provides more functionality than many of its larger colleagues.

CT is barely over 100K in size, and if it needs to, it can run in considerably less memory than that, using Windows' ability to swap program segments. Yet it provides all the functions you need in a program to replace the PROGMAN, FILEMAN and MSDOS.EXE that come with Windows. Because of its small size and efficient non-Windows way of reading most of its own INI file, CT starts up very quickly.

Rather than representing programs as graphic icons, CT represents them as text entries in menus that you can create and modify yourself. That requires far less memory, does the job every bit as well, and you'll notice right away that CT takes a much smaller bite out of your system resources than other alternatives!

We're convinced that you'll be sold on CT-Shell from the first day you try it. At the beginning of the next chapter, *Overview*, is a brief summary of the major features. Congratulations on showing such excellent taste in Windows system software!

Compatibility and Hardware Requirements

This version of CT-Shell has been created for Windows v3.1, however it has been extensively tested for compatibility with Windows v3.0 and found to work well under that environment as well. Both Windows v3.1 and CT-Shell v2.10 assume a 286-or-better computer running Standard Mode or Enhanced Mode.

This version of CT has been compiled for use in protected mode, and it will not run on an 8088 or 8086 machine at all. It *should not* be run on any computer in Real Mode, even if it can be started that way under certain circumstances.

The *absolute* minimum hardware requirements for CT are the same as the published minimum requirements for Windows. However, *practical* minimum requirements for using Windows v3.x and nearly any software include at least the following:

- 386 SX or better computer
- 4 megabytes or more of RAM memory
- Microsoft or compatible mouse

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 CTSHW210.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.TXT 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

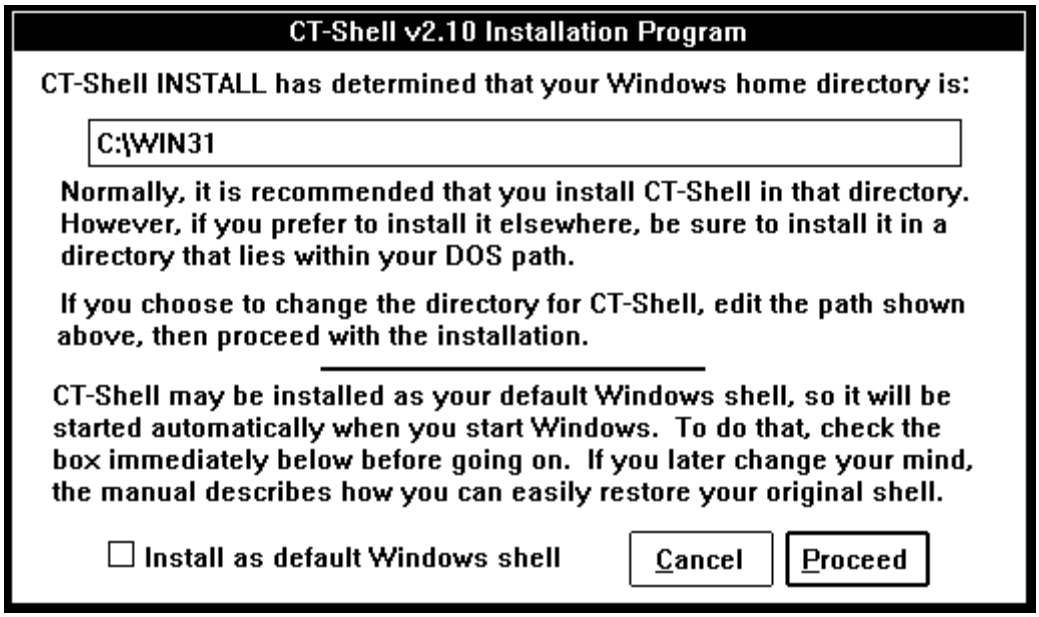
There is an installation program provided, named CTINSTAL (its executable file name is CTINSTAL.EXE) which can be run from within the Windows environment using any means you usually use to run Windows programs. In other words, you can doubleclick on CTINSTAL.EXE while using the Windows File Manager, 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 about 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.

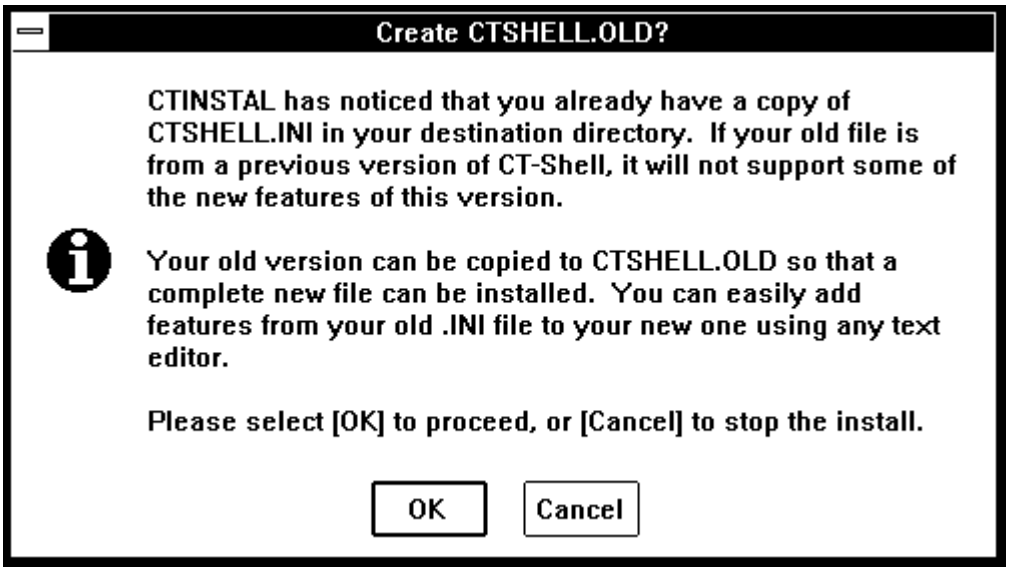
Likewise, CT can be installed as the default shell by replacing whatever is currently being used with CTSHELL.EXE. In fact, CTINSTAL will make the change for you automatically, if you check the box marked *Install as default Windows shell* near the bottom of the installation screen. To install the CT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



Because *so much* of what your old CTSHELL.INI file contains is likely to be your own customizing, it just isn't feasible for CTINSTAL to try to figure out what should be retained and what should be replaced. Fortunately, copying your custom [ITEMS] entries from the old file to the new one is quite simple with most editors.

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.

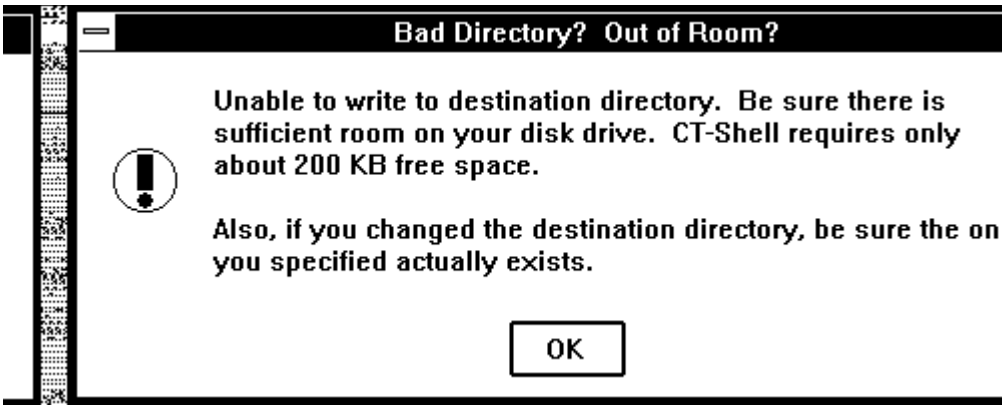
File Not Found

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.

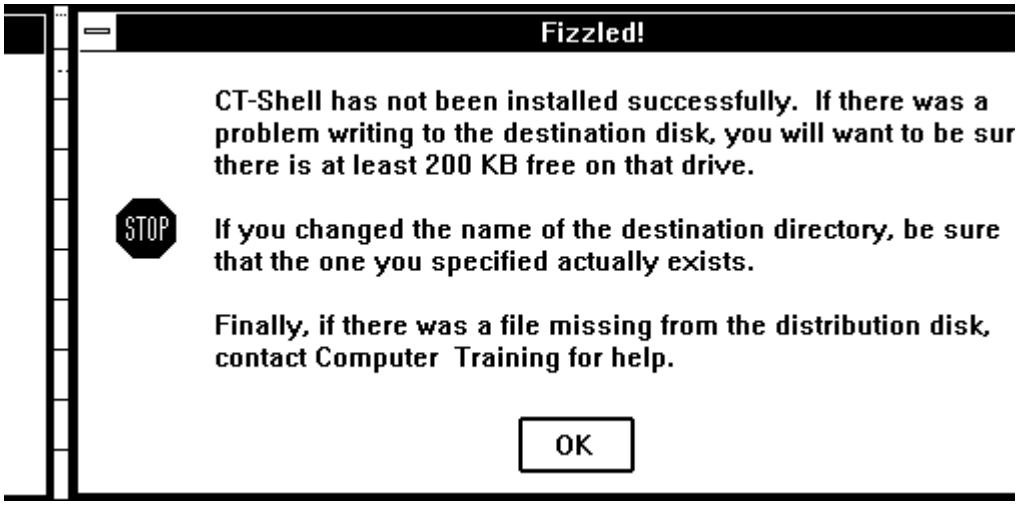
Likewise, if you received CT in an archive from a BBS or a friend, you will have extracted the contents of the archive into a directory *somewhere*. It is only necessary for you to change to that directory–make it your default directory–before proceeding to run CTINSTAL.

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 256 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 lets you exit from Windows and restart it, if you have elected to install CT as your default Windows shell. If you chose not to install CT as your default Windows shell, you will be advised to start it as you would any ordinary Windows application. 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 or accounting tasks. 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:

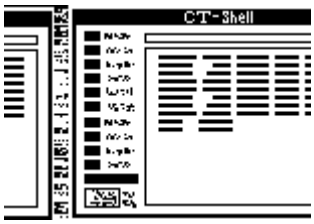
```
CTSHELL  c:\win\wordproc.ini
```

Of course, either example would require that you first create the new xxx.INI file. You might begin by copying your original CTSHELL.INI to the new file name, then modifying it to suit your purpose.

This handy feature also makes it easy for multiple people to use a computer on which one copy of 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. Each xxx.INI file can even contain its own caption and its own color specifications, so that each instance of CT can easily be identified. (See the later section about the Preferences Dialog for more information about setting these options.)

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 a third copy each require even less. (Additional copies typically require only an additional 1% in system resources!) 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 menus.

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.

Capabilities

CT is a small, convenient and fast:

- Program launcher (to replace ProgMan)
- File manager (to replace FileMan)
- Command line interface and DOS command processor (to replace the MS DOS executive)
- Event scheduler and alarm clock
- Phone database and automatic dialer
- Menuing system, user configurable

- Autoexec facility for Windows
- (Source) file listing utility
- String search utility
- File finding utility
- Programmer's helper for all languages

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. Likewise, all the function key assignments that you see at the left side of the main CT window are user-definable. You can reassign any or all of them, so you can start your most frequent tasks just by pressing a key.

It is important for you to realize that your menu and your function key assignments need *not* stay the way they are, 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 that you can accomplish them simply by clicking on a menu entry or by pressing a function key. 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: {EntryName} {DirPath} {ExePath} {Switches} {Keyword}

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

In this first example, the name that will show up in the menu is *NOTEPAD Editor*. Running that program doesn't require changing 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.exe*. The fourth set of braces contains any switches, or arguments that you need to pass to the program when it runs. Here, the exclamation point 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, below.

To edit a specific file every time–such as to create a menu entry that allows you to edit MYFILE.TXT 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 MYFILE.TXT} {} {notepad} {c:\windows\myfile.txt} {}

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 a pound sign in that field, such as {#}. (Some people call that a *number sign*–it's the <Shift+3> key on your computer keyboard. Musicians like the author call it a *sharp*, and wonder why there aren't any *flats* on the 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 are often used instead of running external programs. 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. The one shown below activates the dialog that allows you to change your preferences–those settings that affect how CT does certain things:

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

CT keywords may be entered in uppercase, lowercase, or mixed case. They are all converted internally to uppercase for evaluation. There are 66 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.[2]

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 may 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 easy to spot. 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.

Throughout this manual those two terms——*item* and *entry*——will be used to refer to the main menu items and the individual entries in each one. Because of the way Windows works, the items will sometimes also be referred to as menus, themselves.

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.[3]

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:\WINDOWS\PIF, perhaps a place where you store all the PIF files for your system.

Example 5: {PIF Directory} {C:\WINDOWS\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 when CT is started. Because any number of entries can be placed here[4] , 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}

When you've finished examining your CTSHELL.INI file, you can select the *Restart Windows* entry from your *File* menu. When Windows 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.

Note that this will happen only when you start your MAIN copy of CT. If you start any additional copies of CT, they will bypass the *Autoexec* process, so you don't inadvertently start additional unwanted copies of those programs.

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 most-often-needed applications whenever you start CT. If you understand it well enough already, go ahead and install another program or two in the Autoexec section while you're here.

Load= and Run=

In addition to its own——more powerful——*Autoexec* feature, CT-Shell will process the LOAD= and RUN= lines that you may already have in your WIN.INI file. Thus, if you install CT as your primary Windows shell (in place of ProgMan, for example), you'll find that those existing programs are still run at startup, just as before.

In fact, CT enhances those features by allowing up to 512 characters per line, compared to the original 80-character limit. Thus, if you need to start several programs that require long path designations, you'll have more room for them when they're processed by CT instead of ProgMan.

Just in passing, note that CT also allows you to install a MAXIMIZE= line and a HIDE= line in your WIN.INI file if you want them. They work like RUN=, but will start your Windows program in its maximized state or start it as a hidden process.

(More on this issue later.)

For obvious reasons, you will want CT-Shell to process your LOAD= and RUN= entries *only* when it is the primary Windows shell (also called the ROOT shell), not when it is started from another shell like ProgMan. In fact, it works just that way. CT is able to determine whether it is the root Windows shell when it is started, and it will process these entries only if that is true.

Incidentally, you can always tell whether a copy of CT-Shell is the root shell, by the [ROOT] that you'll see in its caption bar, at the top of its main window. If a copy of CT has been started from another program, but is still the first instance of CT that is run, that caption will show it as your [MAIN] shell. Any additional instances that are started will show up as [ADDL] shells. The type of shell affects certain options, such as whether you can restart Windows upon leaving that shell.

User Entries

The third place where you'll find entries that look almost exactly like the ones you've been working with so far is in the [OPTIONS] section, in the lines that begin with User1= , User2= , etc. These are user-supplied function key assignments, and the twelve that you see here become assigned to the function keys F1 through F12, respectively.

The only real difference between these entries and the ones you have in your [ITEMS] section 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 some of those entries look like in the default configuration.

Example 7:

```
User1={Toggle Current} {} {} {} {TOGGLE}
User2={Tag All} {} {} {} {TAGALL}
User3={Untag All} {} {} {} {UNTAGALL}
User4={Invert Tags} {} {} {} {INVERT}
. . .
```

These user-supplied settings allow someone to press F1, for example, to change whether the current file name (in the list box to the right) is selected (also called "tagged"), or to press F2 to tag all the files in the directory.

Note that any of these choices can be made available in the ordinary menu entries as well, and that you could replace any of these with other entries that you want activated at the press of a 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 options that control up to five event timers that CT provides. You can configure CT 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 preset time. Normally you would change these settings by selecting a menu entry that invokes the special CT ALARM keyword, but you could also 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 that 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.

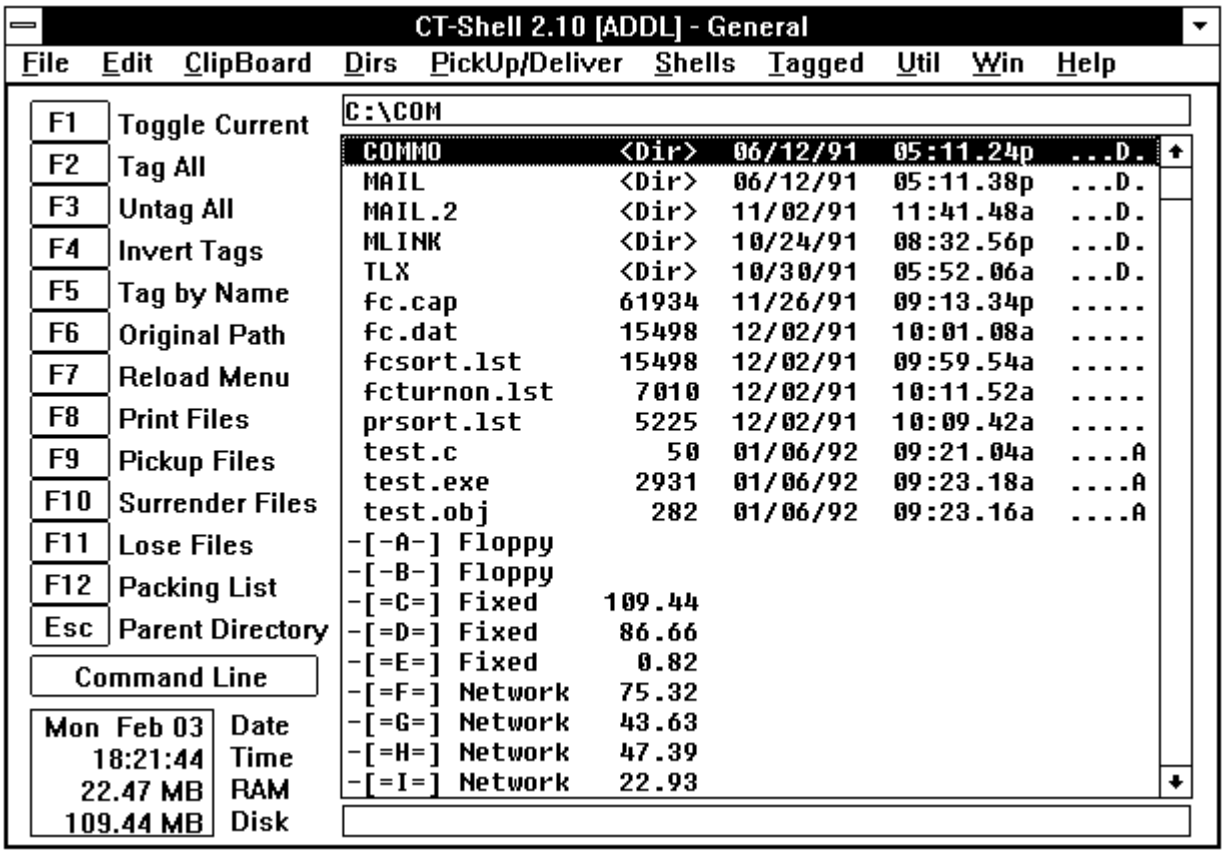
At this time you should finish looking through CTSHELL.INI and close the file. If you are still in NOTEPAD, open the *File* menu and select *Exit*, which will cause NOTEPAD to ask whether you want to save your changes. Answer *yes* or *no*, as appropriate, and you'll be returned directly to CTSHELL. 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, assuming that the default function key assignments are still in effect. Of course, any changes that you've made will take effect the next time you start CT, even if you don't reload the menu at this time.

(This page intentionally left unused...)

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, the caption at the top of the main window, and the numbers in the status window in the lower left corner. If you are not a registered user, the word *Unregistered* will appear in the caption.

Menu

When CT is run, at the top of its main window is a menu that shows several items, such as *File*, *Edit*, *Clipboard*, *Dirs*, and more. This menu is based on the entries in the CTSHELL.INI file, and you will later want to revise the sample file to include your own program choices. As you add entries to your CTSHELL.INI file, those new options will appear in the CT 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 setup that the sample initialization file creates, you'll find a number of things in the *File* menu that you can do with the current file (the one file that's selected with a dotted outline in the files list window), and a directory that you can change to in the *Dirs* menu. The *Edit* menu contains a number of choices, one of which you have probably used already to edit your CTSHELL.INI file. (Remember that you can reload a modified menu by pressing <F7>.)

The *Clipboard* menu provides a number of operations that you can perform on the Windows Clipboard, that place where programs can temporarily store data for later use in other programs. CT lets you see whether there is text in the clipboard, copy it to a file, and more. You can also move a copy of your files list into the clipboard, to be inserted into a document you're editing with a word processor.

The *Pickup/Deliver* menu contains a number of options for easily copying or moving files from one place to another. You can *Pickup* files from one or more places by selecting them in the current file list and using the *Pickup* entry.

Having done that, your caption bar will change to show that you're carrying files. Locate the directory where you want them to be copied or moved, and once there, the *Deliver* entry will copy them to the new current directory. If you want them somewhere else as well, you can then change to that other directory and *Deliver* them again. They remain in your *Packing List* and can be delivered as many times as you want. When you're finished with them, you can *Lose* the files, which means you no longer carry them around with you.

Alternatively, you can use the *Surrender* entry to copy them to the current directory and remove them from your packing list with only one command. Or use the *Move 'em Here* option to move the files to the current location, *removing them from their original directory*.

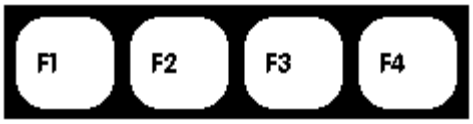
The *Shells* menu offers an ordinary DOS session, and an additional CT window if you ever need a second one (or a third one, for that matter). There are also entries available for most of the CT functions. The menu labeled *Tagged Files* contains many of the same options that are provided for the current 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.)

Util contains a command that will allow you to reset your computer's date and time from within CT-Shell, without needing to invoke DOS through its command processor.

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.

It's worth saying one more time before we go on: all these entries are here because they are in your CTSHELL.INI file. You are free to reorganize your menu system any way you want to, and to add any other entries that you'll find useful.



Function Keys

All of the function keys have pre-assigned meanings, based on entries in the [OPTIONS] section of the default CTSHELL.INI file that comes with CT. All of them may be reassigned by the user, to allow you to customize your CT setup as you see fit.

In fact, since you can start additional instances of CT using other xxx.INI files, each instance can map the function keys in its own way. You change the settings by editing your CTSHELL.INI file, or by invoking the *Preferences* dialog from the *Shells* menu. In that dialog is a combo box of user commands, which you can edit from within CT.

The following sections describe the default settings for the function keys, based on entries in the supplied sample CTSHELL.INI file.

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, by default.

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.

Toggle Current File

Toggles the tagged/untagged condition of the current file. If, for any operation, you want to be sure that *no* files are tagged, you can turn off the tag for the current file by pressing <F1>.

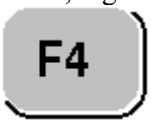
You might want to do something with all the files in the directory *except* the current file, for example (such as deleting them all, copying them all, or moving them all somewhere else). You could do that by selecting the one file you want excluded from the command, then press <F2> to tag all the files, and finally press <F1> to untag the current file.

Tag All Files

Tags all the files (but not directories or drives) that are in the files listing to the right. You can perform a variety 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.

Untag all Files

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

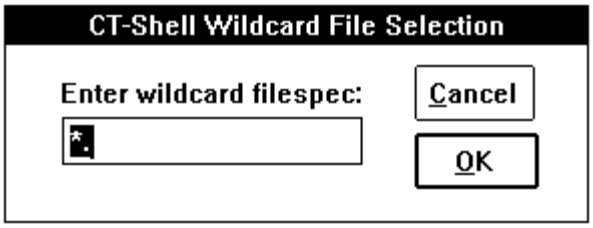


Invert Tags

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>.

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.[5]
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.

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 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.

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 restart CT (and possibly also Windows).

Print File(s)

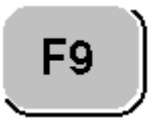
Will print a formatted and line-numbered listing of the current or tagged files. (Note that this description uses the term *listing* as it is commonly used in computer jargon, to refer to a printed *hard-copy* version of the file contents. It does not mean a list of the files in the directory.) If you 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 filespec, 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.

Incidentally, if you *do* want a printed listing of the files in your current directory, you can select them with <F2> or with mouse or keyboard operations, then look to your *Clipboard* menu for an entry that will copy the files list to the clipboard. Only those entries that are selected will be copied, allowing you to make a list of all your files that fit a particular category.

Pickup/Deliver Files



Pickup Files

Pickup files. Places the currently selected files (possibly a single file) into the Packing List, which allows you to carry those files to another directory. You have the option to copy them to that destination and elsewhere, or to move them there, which also deletes them from their original location.

You can tell that you're carrying these files, because the caption bar at the top of the main window will change to a reminder of how many are included. When you either *Lose* those files or *Surrender* them, that caption bar will return to its previous condition.

You can pickup files from multiple locations, each time adding them to the packing list, allowing you to collect an assortment of files from any number of directories on various disk drives.

Surrender Files

Surrenders the files you are carrying to the current directory. This removes them automatically from your packing list, and assumes that you don't want to copy them to any additional locations.

If you have in mind placing the same collection of files in more locations, instead use the *Deliver* entry in your *Pickup/Deliver* menu. An assumption is made in the default setup that most people will want to surrender files most of the time, in the course of ordinary disk maintenance.

If it turns out that you more often want to deliver them to multiple locations instead, simply replace the *Surrender* keyword in User10 entry in your CTSHELL.INI file with the *Deliver* keyword instead. You probably know how to do that already, using an editor, and you'll soon see how to do it from your Preferences dialog, without leaving CT-Shell or needing to reload your menu.

Lose Files

Loses the files currently in your packing list. Use this option to clear out your packing list after delivering files to all the locations where you want them, or simply to clear it out if you change your mind about moving the ones you've picked up.

Packing List

Displays the current contents of your packing list. While the caption bar at the top of your main CT-Shell window will tell you how many files you're carrying around, and this option will present a list of those files.

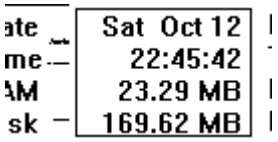
Parent Directory

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.
Note that this key is not user-assignable, but it provides somewhat the same functionality in CT as it does in the dialog boxes that Windows presents. Most people think of the <Esc> key as a way to back out of the current location, and that's what it does here. If you need to back out of the current directory to its parent, a simple press of <Esc> is all that's needed.

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. You can also get the same results by clicking on the text description of any of the function keys. Thus, whether you prefer using the keyboard or prefer using the mouse, you can have it your way.

Status Display

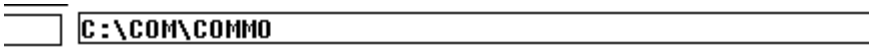


Below the listing of the function keys is a small window that displays the current date and time, the amount of RAM that is available (including virtual memory if you're running Windows in Enhanced 386 mode) and how much room is left on the current disk drive. The latter two measurements are displayed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

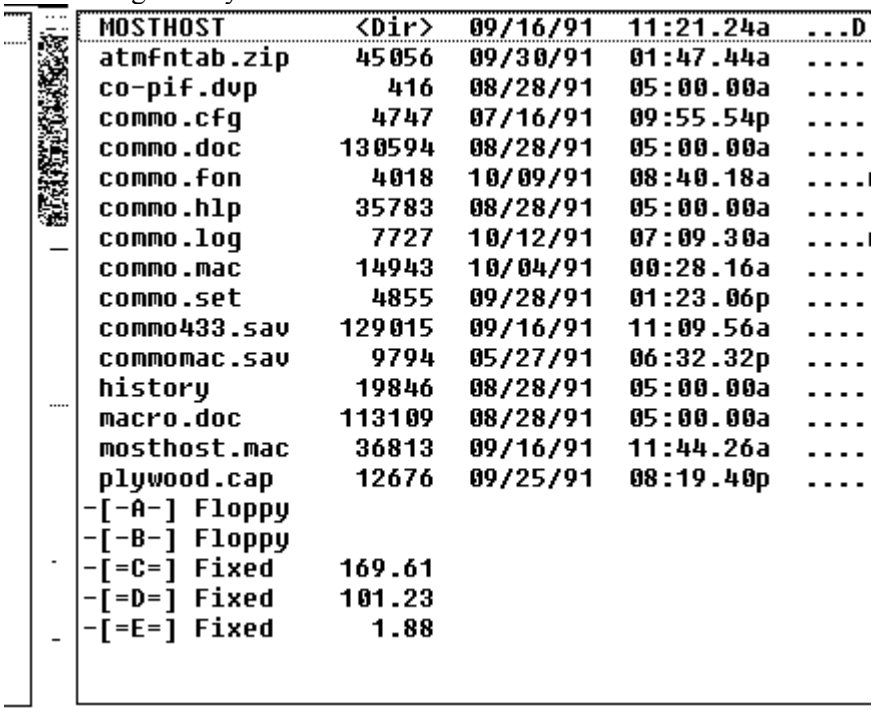
The display of files contains considerable information that is always conveniently visible. One of the biggest advantages of a visual shell over an ordinary command line is that so much more information can be made available at all times.

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.

Having the files all readily visible (or able to be scrolled, at least), makes it easy to perform operations on single or multiple files. You can visit various directories, for example, and pick up a few files here and there to be delivered to one or more destinations. You can mark a set of files in one directory, and copy just the ones that are newer to a second directory, thereby updating that directory. Whenever you perform an operation that results in a change to the information that is displayed in your files window, CT will sense that, and update the listing for you. If you do something that does not affect any of the information that's displayed here, your files window is left unchanged——any selected files will be left selected, for example.

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> , one or more tagged files can be deleted. You are prompted for confirmation before that happens, of course! There are additional deletion options, including a couple of CT keywords called *Remove* and *Deldir*. (See Chapter 6 for a reference to CT keywords that you can use in your menu entries.) *Remove* is implemented n the sample CTSHELL.INI file in the *File* menu item, in an entry called *RmDir*, like the system command of the same name. It requires that a directory be empty of files and subdirectories before it will remove that directory. *Deldir* is implemented in the *File* menu as the entry called *Kill Directory*. It does not require the directory to be empty first.

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 *Kill Directory*. 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 will delete files is *Shred*. This deletes a file after writing a pattern of bits to every byte of the file that prevents it from being used for anything even if someone manages to undelete it. Shredding a file takes longer than simply deleting one, so you will want to use this keyword only when data security is important, and unauthorized access to the file cannot be tolerated..

The sections that follow each describe one of the components of a line in the files display window, and explain 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.

<Volume> VolumeName

Each disk drive may have one (and only one) optional volume name that identifies that disk in operations such as a DOS DIR command. If there is a volume label present, it must be in the root directory of that drive.

Most times when you see a volume label, it will be on a floppy disk. The label name is displayed at the top of the list of files, and it cannot be used in any operations, such as copying or deleting. It is not a file, nor a directory, simply a name for your disk.

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*.

Note that the files may be ordered in the list based on your choice of three available criteria:

- file name
- file extension
- file date/time

The sorting method is an option that you can select from your *Preferences* dialog. If you choose to sort based on file extension or based on date/time, files will be displayed within each group in file name order.

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*.

Since the file size is a measurement of the file, and not a user option, there is no way for you to change the size of a file with CT-Shell.

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.[7]

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 *File* and *Tagged Files* 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 most of the 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.[8]

You might want to change important files to read/only status to protect them from being deleted or overwritten accidentally. Remember, however, that some files *need* to be overwritten, for them to be useful. If you apply this attribute, you will need to remove it before you can intentionally update those files.

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.[9]

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, and you can still 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.

This is an attribute that you can't change with CT-Shell——nor would you want to——because directories are a very special kind of file that is created and maintained by DOS. An ordinary text or program file cannot be converted to a directory, or vice-versa.

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.[10]

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 and WNBACKUP from Central Point Software, Inc., use this attribute to determine which files need to be processed when a differential backup is done.[11]

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

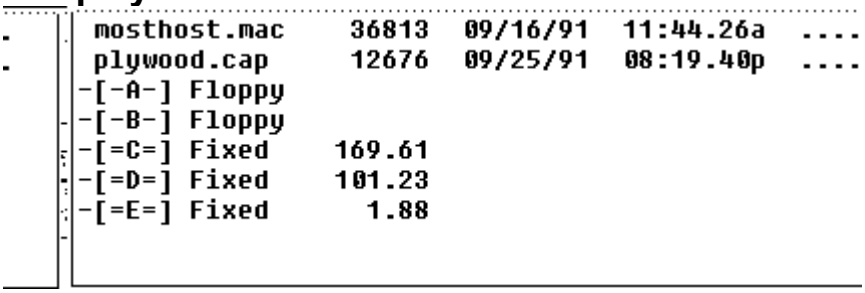
Besides displaying the attributes, CT makes it easy for you to change most of them. You can't turn a program into a directory, but you might want to make a file read/only, for example, to prevent its being accidentally deleted or overwritten. You can alter the attributes for a single file or for a group 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 a floppy) 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 to check on the remaining space. If you want to know how much room is left on a floppy disk, simply change to that drive by doubleclicking on its entry or by holding down the <Shift+Control> keys and typing the letter of the drive. You can easily change back to the current drive the same way afterwards, or by pressing <F6> to return to your starting drive/directory.

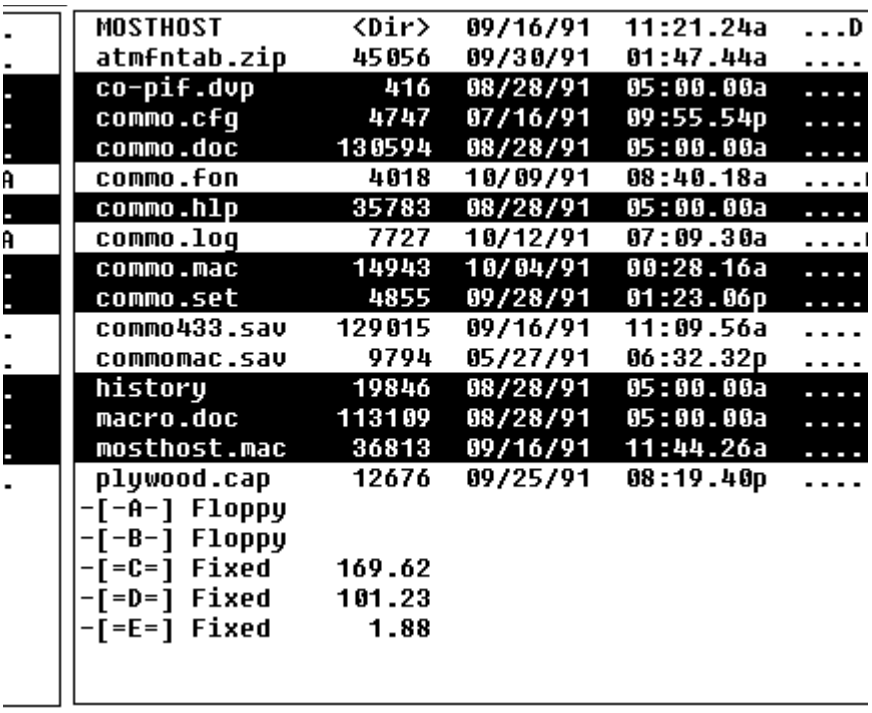
CT always displays the free space on the current drive, even if it is a floppy disk, so you'll want to make sure there is a disk in the floppy drive before making that your current drive. If you try to change to an empty floppy drive, DOS/Windows will present an error message that seems to offer you a chance to [Cancel] the operation, but it doesn't really work. The simplest way out of such an error situation is to put *any* formatted diskette in that floppy drive, thereby allowing the change to that drive, after which you can return to your original 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>. In the following discussion, when you see a reference to doubleclicking one of these entries, remember that pressing <Enter> will do the same.) What happens when you do either of these things 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 additional 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.

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 association 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]

CT provides a convenient way for you to modify and add extensions to your WIN.INI file. From the default *Windows* menu, select the *WIN.INI Extensions* entry to invoke a dialog that you can use to discover whether an extension is known, and if so, what program it is associated with. You can also change an existing association or add a new one for an extension not previously found there.

New Windows users, take note: this is a *very* powerful feature! If you often work with files for which the extension indicates what program uses that file, you will certainly want to add associations for those files. See the fuller description of this process in Chapter 6, where CT-Shell keywords are described in greater detail.

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 nearly everyone needs to do 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.

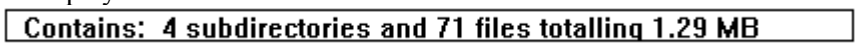
If the drive you want to change to is visible on the screen, the doubleclick method just described here is quite convenient. But the easiest of all, especially if your hands are on the keyboard, is to type <Shift+Ctrl> plus the letter of the drive you want to change to. For example, to change to drive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

Status Line

C:\WINWORD\CTSHTOC.DOT

Not to be confused with the *Status Window* to the left, the *status line* just below the files listbox contains a changing stream of information depending on what operation you are currently involved in. Whenever CT does something for you, such as copying a file or changing directories, you'll be notified in the status line. Even with beeping turned off (more details about that in the next chapter), you can often tell when a process has finished by looking here. At the end of most operations, the status line will be cleared.

Under certain circumstances, this line contains information about the number of subdirectories in the current directory, and the number of files and their combined size. That's the case when you have first moved to a new directory and haven't yet told CT to do anything else for you. If the status line has been cleared and you would like to see the file size information once again, use the *Update Directory* entry in your *Dirs* menu. That display looks like this:



It's worth mentioning here that when logging is turned on (another explanation in the next chapter!), most of the entries you'll see in the status line are used as entries in your CT-Shell log file. Such directory information as above is not logged, however most operations are.

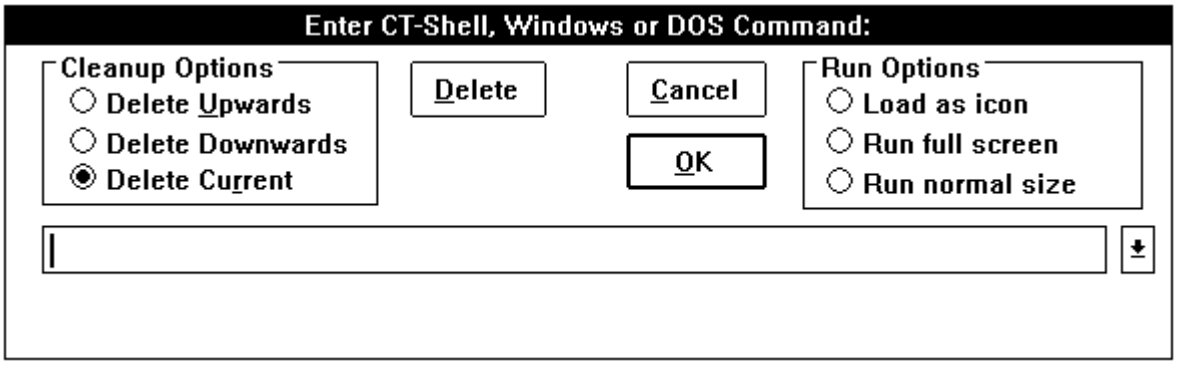
(This page intentionally left unused...)

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
```

will create four subdirectories in the current directory. You won' t need to issue a separate MD command for each directory you want to create, as you would at DOS. 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 when you're not at the command line–by pressing a key combination that includes <Shift+Ctrl> plus the drive letter.)

Persistent Command Line

CT offers an option, available through the *Preferences* dialog, that allows you to keep the command line open following a command, so that you can continue to process DOS, Windows, or CT-Shell commands without reopening it.

If this option is selected, the command line will go away when you select the [Cancel] button. If this option is turned off, the command line will close after each time you use it, and you will need to reopen it for any subsequent commands. Remember, that's easily done with <Shift+Enter>, so the command line is always available quickly whenever you need it.

Note that you can also open the CT command line from the system menu——that little bar-in-a-box in the upper left-hand corner that you can doubleclick to close most Windows programs. If you first minimize CT into an icon, you'll find that you can leave it in that condition and still open a command line by clicking once on the CT-Shell icon to open the system menu, then selecting the *CT Command Line* entry from that menu.

For a long series of command-line operations, this gives you a workplace without the main CT window, but with the command line window. If you decide to try this method, be sure to use the *Preferences* dialog to turn on the "remain open" option.

The CT-Shell command line, when used as an independent part of the program, is a very powerful tool. Like other powerful tools, it must be used carefully, and with a little common sense.

You are able to use other CT features while the command line is open. Because of this, you must give some thought to any commands that you give that will be running at the same time. In particular, be cautions about starting a process that uses a file in the current directory, and then changing CT to another drive or directory. Also be careful to avoid giving the same command with different data. Don't start a file copy, for example, while CT is in the middle of copying another file.

Usually it can do no harm to start a number of different processes that are external to CT-Shell——other programs, that is. But be sensible about what other things you ask CT to do at the same time.

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.

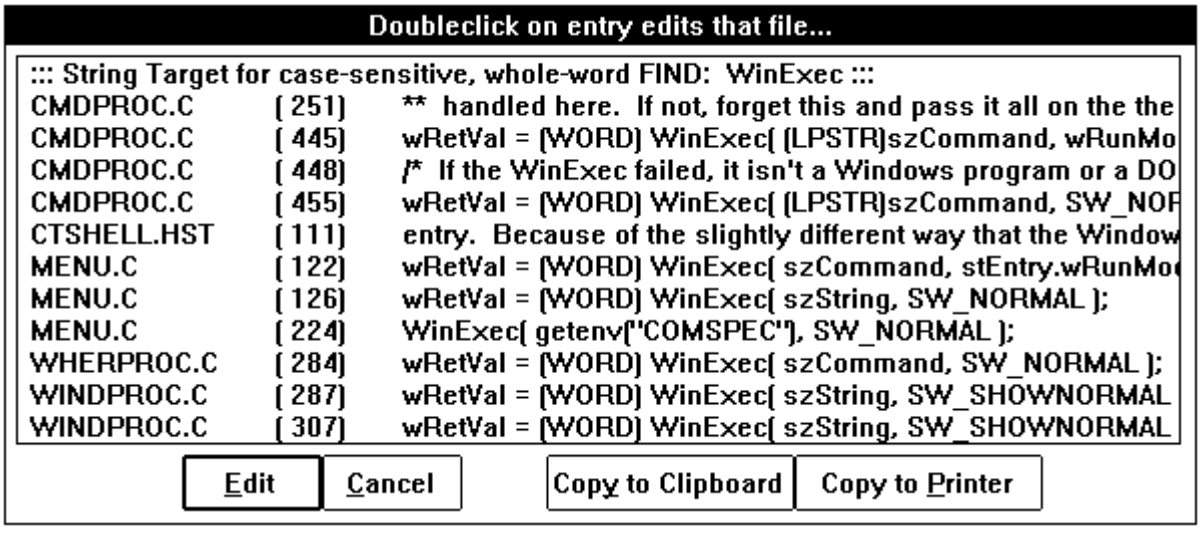
This command deletes all the files in the directories you ask to delete. *Even if those files have the read/only attribute, they will be deleted.* CT will remove that attribute if necessary, in order to delete the files.

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* command is designed to find a string (of characters) wherever it may occur within any of the text files in the current directory. Use quotation marks to enclose strings that include embedded spaces.

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 *may be* case sensitive, depending upon your choice in the *Preferences* dialog. You'll want to be sure your <CapsLock> is not on when you look for a string that contains lowercase letters if you have selected case-sensitive finds!

In that same dialog you'll also find a choice for whole-word searches. If you do not use whole-word searches, CT will consider it "a match" if it locates your string anywhere in the file, even inside another word. Thus, a programmer who looks for the function name *printf* will be shown all the places where that occurs, also all the places where *sprintf* occurs, and *wsprintf*, etc. If the whole-word option were in effect, only those lines that contain the word *printf* as a word by itself would be considered to be a match. CT considers spaces and punctuation to delimit words, to allow programmers to search for a function name or a keyword, even if it is followed by some form of punctuation without a space between.



This list box is created to show you 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 more easily view the significant parts of that line. The first line reminds you of the target that you asked to find, and also the condition of the two options that affect how the searching is done.

Up to the first 256 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. If there are more lines than will fit at one time, a vertical scroll bar will appear that allows you to move easily among the lines. If any lines are longer than can be displayed in the list box, a horizontal scroll bar will appear that allows you to move the contents sideways.

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 probably want 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. As a reminder, in the default configuration, the *Preferences* entry is found in the *Shells* menu.

This listbox also has buttons available that allow you to copy its contents to the printer or to the clipboard. It's obvious why you might want to print the listbox contents——for example, a programmer may want to find all the places where a function or procedure name was used in a big project, then edit them all to use a different routine instead. Having a printed list of all the places where that name was used makes the job a lot easier!

However, you may also want to copy the contents of the listbox directly into a file that you're editing, or into a document that you're creating with a word processor. The easiest way to do that is usually to copy the listbox content to the Windows clipboard, from where you can paste it into your other program.

FormFeed

It is 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 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 command instead of looking for the menu entry.

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.

Shred

This one deletes a file more thoroughly than the *Del* command, making it meaningless even if someone is able to undelete it. *Shred* takes longer than *Del*, so it should be used only when data security is important.

Because the effect of *Shred* is not reversible, it does not work with wildcards, but you are able to *Shred* multiple files with a single command. You will need to explicitly name each of the files, however.

Where

If you want to know where a file is on the disk, you can use the *Where* command. The customary DOS wildcard characters are acceptable here, so you could search for files such as:

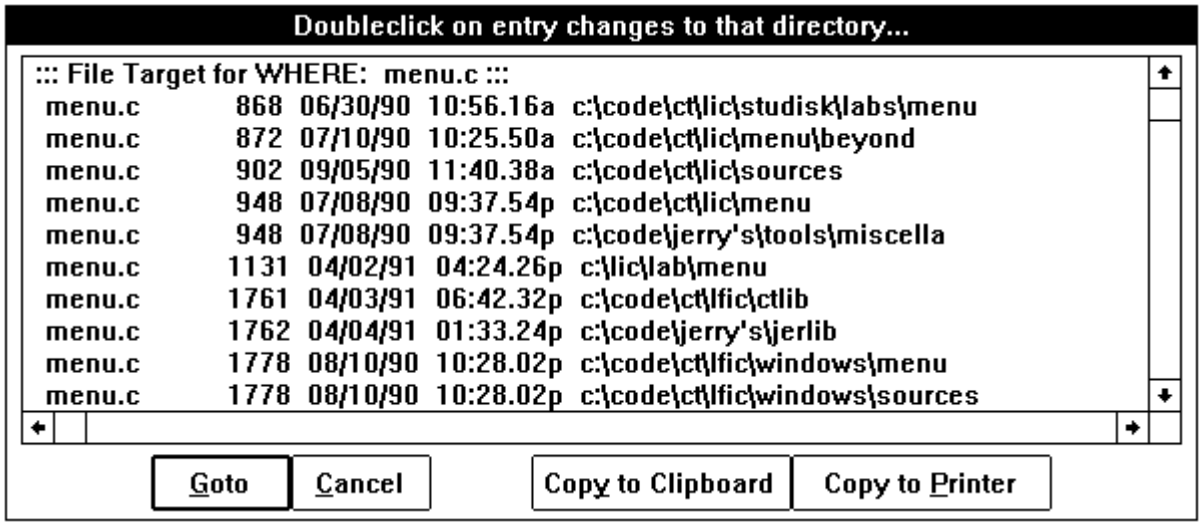
```
where *.dbf
```

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 may 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:



The first line reminds you which file name you asked for., and all the other lines show the various places where a file of that name is located. Note that the *Where* command searches the current disk drive. If you'd like to look for occurrences of the same file name on other drives, you'll need to change to those drives and do the search again.

You 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.

Like the listbox presented for the FIND command, this one has options to print the contents or copy them to the clipboard. That makes it easy to create a printed listing of all the file matches, or to insert such a listing into another document via the clipboard.

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. The 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 thing, with certain enhancements, of course.
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.

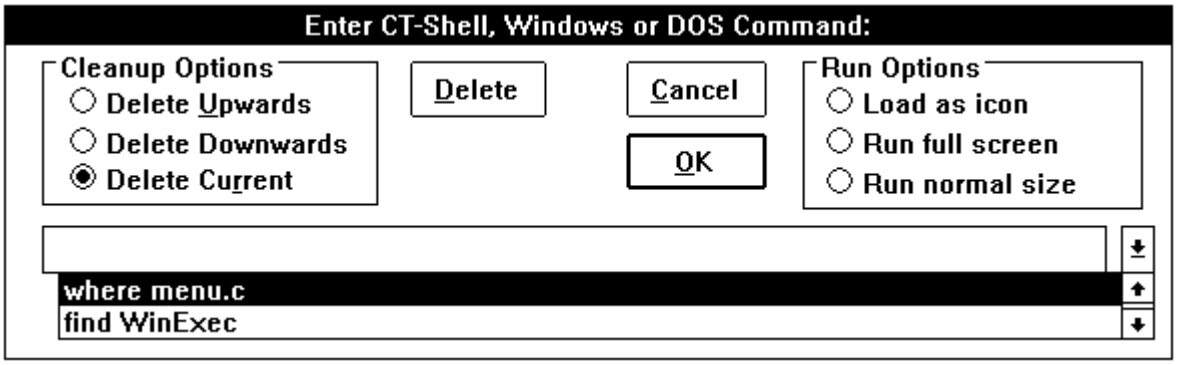
Run Mode

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, with the one exception noted next. DosApps 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. If that PIF file specifies that the DosApp is to be *run in a window*, then the CT-Shell run mode options will be able to affect it. Otherwise, only the entries in the PIF file will have any meaning for it.

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, so 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 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.

(This page intentionally left unused...)

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. Nearly all of these modifications can be done from the Preferences *dialog without leaving CT. 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 menu entries. 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 the Windows NOTEPAD isn't a powerful editor, it is fine for the light-duty work of customizing your initialization file. Editing is necessary to change or add menu entries; all other options can be changed or added either by editing CTSHELL.INI or by making selections from *Preferences* and other dialogs.

Regardless of where your CTSHELL.INI file is stored, or even what the file is actually named, each instance of CT instinctively knows the name of its own initialization file. 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.

Once you've modified and saved your initialization file, you can put your changes into effect by pressing the <F7> function key (provided you haven't changed the default configuration of the function keys), 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 large number of preferences, all of which will be explained in the sections that follow.

These 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 was new for version 2.0 of CT-Shell, and improved somewhat for version 2.1. If you've only used version 1.x before, be sure to study this feature carefully. This section of your file contains the controlling data for up to five 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.[13]

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
Enabled5=0
Time1=00:00
Time2=00:00
Time3=00:00
Time4=00:00
Time5=00:00
Message1=
Message2=
Message3=
Message4=
Message5=
Event1={} {} {} {} {}
Event2={} {} {} {} {}
Event3={} {} {} {} {}
Event4={} {} {} {} {}
Event5={} {} {} {} {}
```

These entries affect the way that up to five timers can be set. Those timers can be used to start programs or simply to display messages at predetermined times. The timers are all reset each day, so regular events can easily be programmed.

Although you can edit this part of the file manually, 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, along with a more complete explanation, 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. Other keywords that can also be used here are *Run* (or *Normal*), *Maximize* and *Hide*.

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.

The default CTSHELL.INI file contains a line with a set of five empty fields here, to make it easy for you to fill in your entries. With most editors, you can easily copy that set of braces to as many lines as you need, then simply fill in the required fields.

[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=225
```

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.

Each instance of CT that uses its own xxx.INI file can have its own color settings, letting multiple instances appear in distinct colors. All instances that use the same xxx.INI file, however, will appear in the same color.

[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:

```
[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 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 is 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. Remember, 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[14]. 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=
```

In addition to these settings, dialing capability is affected by the AllowDial entry later in your [OPTIONS] section, which is changeable from the *Preferences* dialog. Thus, you can easily turn dialing on or off without needing to change these settings.

If you want to activate the dialing feature for your version of CT, you must have a modem connected to your computer, you must provide appropriate settings in this

section, and you must turn on *AllowDial* in your options. 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 handshaking 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. If your phone system requires additional characters to be dialed to access a line (such as many business phone systems that require a 9 first), you may add them to the end of the prefix string. The dial suffix is usually not required at all, 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.

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. 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, and they are discussed in the following paragraphs:

```
[OPTIONS]
AllowDial=1
Beep=1
CTCaption=General
FindCaseSensitive=1
FindWholeWord=1
FlexSize=1
IgnoreDrives=
KeepOpen=0
LogActive=0
LogFilePath=
Mail_In=
Mail_Out=
RequireConf=1
```

```
ShareTime=1
ShrinkOnRun=0
SortMethod=1
StartX=
StartY=
User1={Toggle Current} {} {} {} {TOGGLE}
User2={Tag All} {} {} {} {TAGALL}
User3={Untag All} {} {} {} {UNTAG}
User4={Invert Tags} {} {} {} {INVERT}
User5={Tag By Name} {} {} {} {BYNAME}
User6={Original Path} {} {} {} {ORIGPATH}
User7={Reload Menu} {} {} {} {RELOAD}
User8={Print Files} {} {} {} {PRINT}
User9={Pickup Files} {} {} {} {PICKUP}
User10={Surrender Files} {} {} {} {SURRENDER}
User11={Lose Files} {} {} {} {LOSE}
User12={Packing List} {} {} {} {PACKING}
VisualBeep=0
WinClose=1
```

The entries that use a 1 or a 0 (such as for KeepOpen and RequireConf) are on/off switches, where 1 represents on and 0 represents off. There will be other examples of this type of entry in the sections that follow.

AllowDial

Some people may not have a modem connected to the computer on which they use CT-Shell, and others may simply not want to use CT to dial the phone for them. This option lets you turn on or off the [Dial] button that appears in the *Phones* dialog.

With dialing turned off, you can still use the CT phone directory——you just won't have the option to automatically dial a number from it. The default is on, on the assumption that most users will probably want this useful feature activated.

Beep

CT is able to provide a beep sound at the end of lengthy operations, letting you know audibly that your task has been completed. The default is on, as this may be your only clear indication that a process has finished.

If you are annoyed by the sound of the beep, or if you want to use your computer in an environment where it may annoy others (who may be sleeping, for example), you can turn off the beep from *Preferences*. See also the *VisualBeep* option, below.

CTCaption

At the top of the main CT-Shell window is a caption bar that contains various information, such as whether the current instance of CT is your root Windows shell, an additional window, or an unregistered copy. The string contained in CTCaption is added to the end of that caption line and separated from it with a hyphen, if it is present. This provides a way for you to visually identify any of various instances of CT that you may have started using alternative xxx.INI files.

The default for this option is *General*, and it can be changed from *Preferences*. Examples of other useful captions might be *Programming*, *Accounting*, etc. Also, multiple users might use their names here, such as *Jerry's* or *Polly's*.

FindCaseSensitive

When you ask CT to find a string within the text files in the current directory, you are able to specify whether that search will be case-sensitive. If on (the default), you

will need to enter a search target using uppercase and lowercase in the way it actually appears in the files. If off, you can enter a search string using any case. This option can be changed from *Preferences*.

Use of this option is likely to depend very much on what kind of searches you ordinarily do. C language programmers, for example, will likely turn on case sensitivity, as C is a case-sensitive language. Most others will likely turn off case sensitivity.

FindWholeWord

Another *Find* option, this one lets you determine whether to report success for a find that matches part of a larger series of characters, or for a find that matches a whole word only.

If you turn on whole-word searching, and ask to find the string *verse*, you'll be shown only those lines where *verse* appears as a word by itself. If you turn off whole-word matching, you'll be shown any lines that contain the words *reverse*, *obverse*, and so forth. The default is to find whole words, and it can be changed from *Preferences*.

FlexSize

There is ordinarily no reason to resize CT-Shell's main window, since all the data that is presented there is fixed in size and in relation to the other components. Ordinarily, if you were to enlarge the window, it would only allow you to see unusable areas, and if you accidentally decreased its size, you would cover up parts of CT that you really do need to see.

There are some video adapters and drivers available for Windows that are only partially compatible, however, and yours may not display all of the CT-Shell window properly. If that's the case, you may be able to use CT more comfortably by resizing the window after opening it.

If this *FlexSize* option is enabled, CT will do two things differently. First, the CT window will be made one line taller, which may be all it takes to solve a video driver problem. Second, the window will become resizeable, so that if that extra line isn't enough, you can enlarge the window even more.

The default for this one is off, and hopefully you'll be able to use CT-Shell without a problem that way. Turn it on from *Preferences* only if you find that you need to.

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 it 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, and the default is an empty string, which tells CT not to ignore any drives.

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 a 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. The default is off.

Remember that the command line can be started from CT-Shell even if it has been minimized to an icon. Some users prefer to minimize CT, then open a command line for a lengthy session with a relatively clean workplace. For that option to be really useful, *KeepOpen* should first be turned on.

Log Active

CT-Shell is able to keep a log of most of your activities in a text file. The best way to tell whether you want a log is to use one for a while, then take a look at it. No harm can result from that, unless you have a particularly slow disk drive, since a log entry is made for nearly every operation.

To try this out, you'll need to turn on *LogActive* from *Preferences*, and you should probably also provide a different path/name for the log file (see explanation below). There are entries in your *Edit* menu that make it easy to edit (to view) your log file, and to delete it if it becomes too big.

If you find, after trying logging for a while, that either it slows things down too much for you, or that you never need the information that CT stores for you, you can easily turn logging back off. You do not need to change the log file path name when you do that.

LogFilePath

The default for the log file is simply CTSHELL.LOG, since there is no way for us at Computer Training to know what subdirectories you have on your disk, or where you would like a log file to be stored. With that as the file name, CT will create a new log file in every directory where you work.

If you change that to a full path/file name, then CT-Shell will instead store all log information in that one file, no matter where you're working. For example, assuming that the WINDOWS directory exists on your drive C:, you might want to set it to something like this:

```
LogFilePath=c:\windows\ctshell.log
```

Whatever your choice, when you ask CT to edit your log or to delete your log, it will be the current log file that it affects. Note that you can provide a log file path from your *Preferences* dialog and still turn off logging using the *LogActive* entry shown above.

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 this 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* options. It then presents a message box that states in simple *Yes/No* terms whether mail is waiting to be sent (the *MailOut* file exists), or mail is waiting to be read (the *MailIn* file exists).

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 simple 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 user or not.

These path strings may both be modified using the *Preferences* dialog. The defaults are no file names.

RequireConf

RequireConf refers to closing CT by doubleclicking on the system menu box, or selecting one of the options that closes CT, restarts Windows, or reboots your computer from your *File* menu or the system menu. Remember, the system menu is the bar-in-a-box that's located in the upper left corner of the window in which CT runs.

Some users 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, however restarting Windows or rebooting your computer might come at an inappropriate time. Thus, the default for this option is on, to require confirmation. This option may be changed from the *Preferences* dialog.

ShareTime

Windows provides a non-preemptive multitasking system for Windows applications, which means that when a Windows application gets control of the computer, that application retains control until it gives it up. Under some circumstances, that might mean that a program could tie up the computer at the expense of other programs that need some attention, such as a communications program that is trying to stay synchronized with a remote host.

If *ShareTime* is turned on, which is the default, CT-Shell will be a "good neighbor" to other Windows applications, and it will pause from time to time during lengthy operations (like file copies, disk searches, etc.) to let other programs get some attention. If it is turned off, then operations like a file copy will be done from start

to finish before releasing control to Windows or to another application.
As an example, the author has used CT-Shell to safely copy files larger than a megabyte to a network drive, while a file was being downloaded from CompuServe by another program in the background.
Generally, this option should be turned on, unless you are sure that none of your other Windows applications is sensitive to being ignored for long periods of time. If other applications do not need attention, having *ShareTime* turned on does not slow down CT-Shell operations much at all. If CT *is* noticeably slowed, it means that other programs are indeed benefiting from this sharing.
ShareTime can be turned off from *Preferences* if you find that *other Windows applications* are being greedy, and not, themselves, sharing enough time with CT.

ShrinkOnRun

Some users have asked whether CT-Shell could shrink itself into an icon automatically when another application is run, and this feature is in response to those requests. If *ShrinkOnRun* is turned on from *Preferences*, CT will do just that. The default is off.

SortMethod

The listing of files can be sorted according to three keys. The number stored in your *SortMethod* option corresponds to one of these keys. Possible values and the method they represent are: (1) sort by name, (2) sort by extension, and (3) sort by date/time. The method you prefer can be set from the *Preferences* dialog.
Note that whenever (2) or (3) is selected, files will be displayed in file name order within each group. The default is to display entries in file name order, which is slightly faster than the other options, which require sorting on two keys..

StartX and StartY

If these two settings are left out entirely–i.e., do no appear in the CTSHELL.INI file at all–or if they are both set to zero, CT will automatically start in a centered position on the screen.
Some users 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* (from the *Reset Position* entry in the *Dirs* menu), these entries will be made in your CTSHELL.INI file for you. If they exist and are nonzero, 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 CT is in its normal centered position on an 800x600 SVGA monitor, *Position* will record these values:

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

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.

User1 through User12

As you know from the previous discussion in the section on function keys, the *User1* through *User12* tasks are all assigned by keywords, so that those operations may be performed from any function key you want to assign them to.

These settings can 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*... entry might need to differ from a regular menu entry–a menu can accommodate a much longer name[15].

VisualBeep

Some users who do not hear well at the frequencies used by a PC speaker may prefer a visual indication that a task has finished, in place of the audible beep that CT normally provides. If *VisualBeep* is turned on, the audible beep will be replaced by a flashing of the caption bar at the top of the main CT-Shell window. This flash inverts the color of that bar for a full quarter-second, so it is easily distinguished from various other——hopefully briefer——flickers of the main window.

This visual indicator may also be preferred by those who hear perfectly well, but find the flashing of the caption bar to be less annoying than the audible beep, or for some other reason prefer that their computer remain quiet.

If you turn on just the *VisualBeep* option in *Preferences*, the ordinary *Beep* option will be assumed automatically.

WinClose

If you're using CT-Shell as your root Windows shell, you will probably want to close Windows whenever you exit from CT. However, there's nothing about Windows that requires you to do that. If you want to, you can exit from CT in a way that leaves any existing applications running. After all of those have been closed, *then* you will exit from Windows.

The default for this option is on, which means that when you exit CT, you exit Windows as well.

[Phones]

This section contains a single entry in the distribution configuration:

```
[PHONES]
Phone0=Computer Training          1-(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 that 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 set of tagged files) when you press the <F8> key. It looks like this in the sample CTSHELL.INI file that is supplied:

```
[PRINTER]
24Hour=0
Draft=0
Headings=1
Independent=0
LineNums=0
LineSize=80
PageNums=1
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, next). 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 optionally 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 wordwrap, so lines of text that are too long may be truncated at the edge of your printing area. (CT sends the whole line, and what your printer does about the extra text depends upon it and its driver.)

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 in your CTSHELL.INI file 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 the *Printer Setup* entry in your *Utilities* menu——that will invoke the setup function from your Windows 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 *LineSize* setting affects only the printing of files, and the printing of your Windows clipboard. When you print the contents of the various listboxes (such as your FILES list, or the results of a *Find* or *Where* search), an appropriate size is used for each type of listbox. The default for files is 80 characters.

The rest of the options shown here are simply off/on values, where the number 1 represents on and 0 represents off. 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.

24Hour

This option lets you determine whether the times printed 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 prints as *9:30pm*. This option defaults to off.

Draft

Draft tells Windows whether to try to find a font for high-quality output or one that will print more quickly, with lower quality. If *Draft*=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 invoke 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. This option defaults to off.

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.

If you print the contents of any of your listboxes or the clipboard, an appropriate identifier will be used in place of the file name and date/time. The default for this one is on.

Independent

The *Independent* option refers to page-numbering for multiple-file printing jobs. If you have a series of files tagged before you press <F8>, all of them will be printed, not just the current file. If *Independent* is turned on, each file listing will begin with page 1. If *Independent* is off, 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. The default is 0, or off.

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. This option defaults to off.

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, above. You can print a few additional lines of text on each page if you turn off this option, as the default is on.

TextFixed

Whether the text portion of your printout is printed in a fixed font or a variable (proportional) font is controlled by this one. If you need to print program listings, you'll want to set *TextFixed* to on, 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. The default for this option is on, providing fixed fonts for file text.

This is the second of two options that may be different for printing the contents of the various listboxes. They are always printed with a fixed font, whereas files and the clipboard are printed according to the setting found here.

[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 arbitrary 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.

Also, since menu entries are so enclosed, CT is able to ignore comments *outside of* the braces.

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}, and displayed in your menu as *E*dit. 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 multiline 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 that allows you to edit CTSHELL.INI.

EntryName

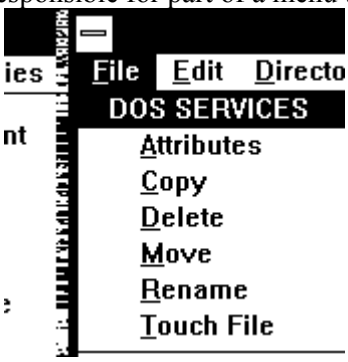
{EntryName} {DirPath} {ExePath} {Switches} {Keyword}

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 used only as a *label* for a section in the menu. (Since a label is not an executable entry, any fields——sets of braces——that follow the first field are ignored, up to the end of the line.) 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 in the name field, either in label entries or ordinary executable entries. They 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 OPTIONS* 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 to be 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 they are embedded into an entry name, they are 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} {} {lha.exe} {a $archive name$ !} {}
```

...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 (such as in Add), 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.

By now you know that such a *permanent* change usually means *until you press <F6> to return to your original path*. There is a special accommodation, however, for any directory change that's done like this one, but during the *Autoexec* processing at the beginning of a CT-Shell session.

Any such change is assumed to be for the express purpose of establishing a startup directory for your session, and will set your original path in addition to moving you there. Thus, you can use an *Autoexec* entry to provide a starting directory for all your sessions, and it will take effect no matter where you are when you run Windows. If you inadvertently include more than one such directory change in your *Autoexec* section, it will be the last of them that is used.

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 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. 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. However, by providing the extension we are able to make it faster and easier for CT-Shell to find and run the program.[16] 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 subdirectories 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 in 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 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 it 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-Shell 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.

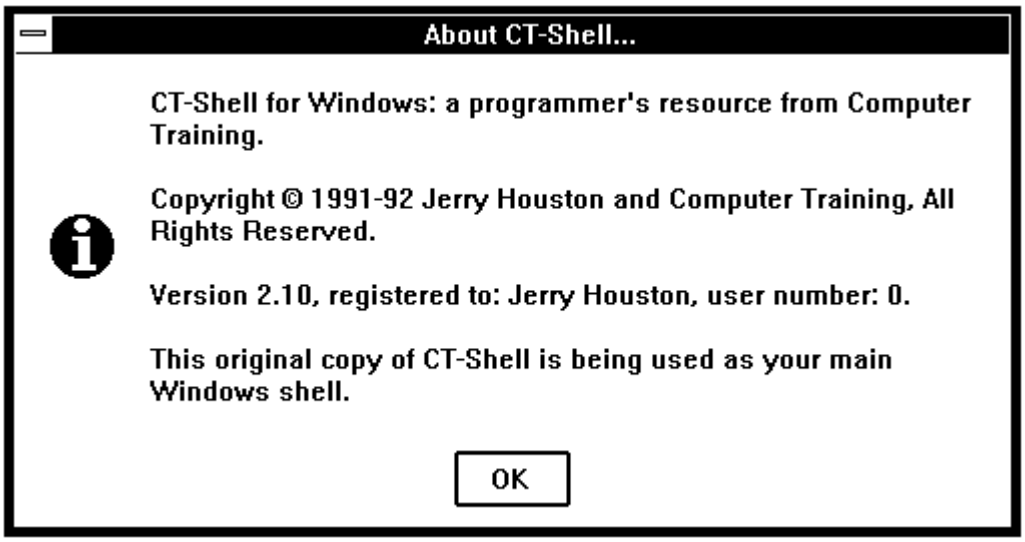
Single and Tagged Files

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[17]. 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

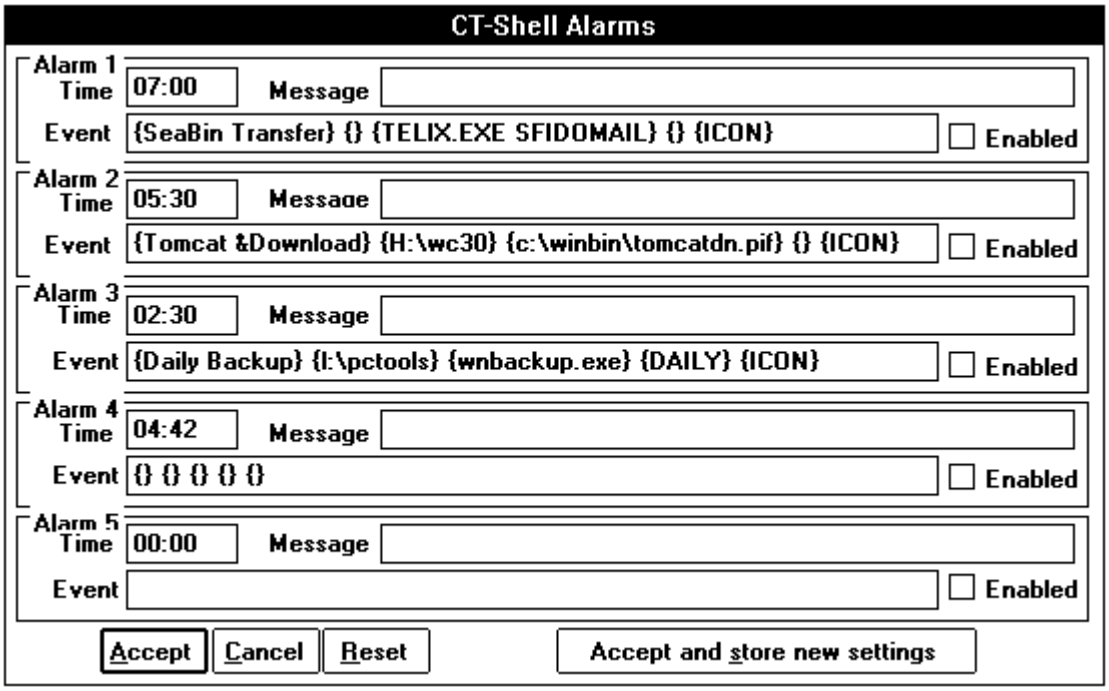
Displays information about CT, including the number of the version that you're using. The dialog that it invokes looks like this:



Alarm

Allows the user to modify the settings used by the five 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. Your settings will be used for this session only. If you select [Accept and store new settings], your changes will be both used and stored.

Once 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.

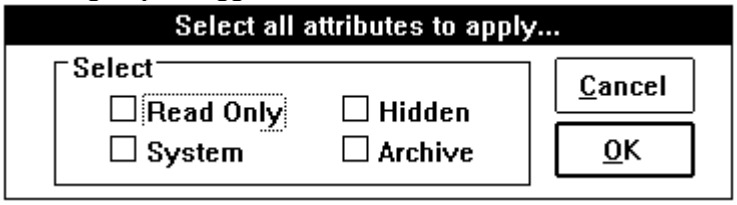
If you have made some changes and find that you need to return to the settings that were previously recorded in CTSHELL.INI, you can click on the [Reset] button.

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

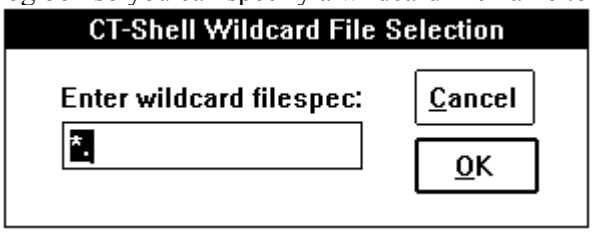
Changes file attributes. Use this keyword in a menu item to let you change the attributes of the current file. There is another version listed below that changes the attributes for a group of tagged files.



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.

ByName

Presents a dialog box so you can specify a wildcard file name to use for tagging files.

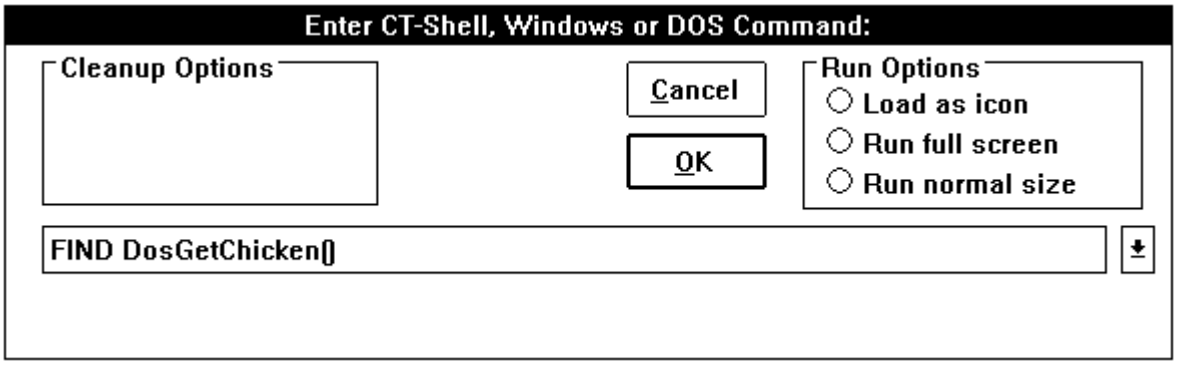


You might want to tag all of the xxx.DOC files in the current directory, so you can copy just those to another place for safekeeping. If you begin to type right away, your entry will replace the default *. that is provided. If you press a cursor-right key first, or click the mouse where the flashing cursor is first, you only need to enter the file name extension, in most cases.

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.[18]

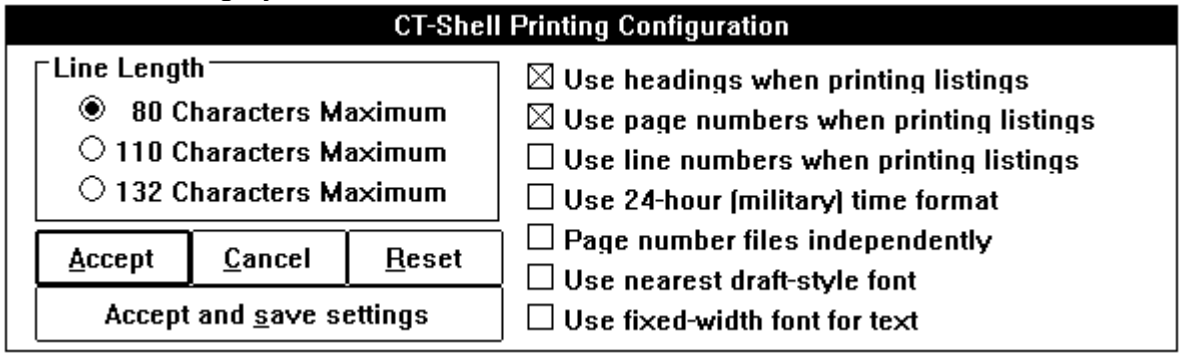
If there is anything at all that you prefer doing at an ordinary DOS command line rather than from within CT, this keyword provides you with the ordinary DOS session where you can do it.



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:

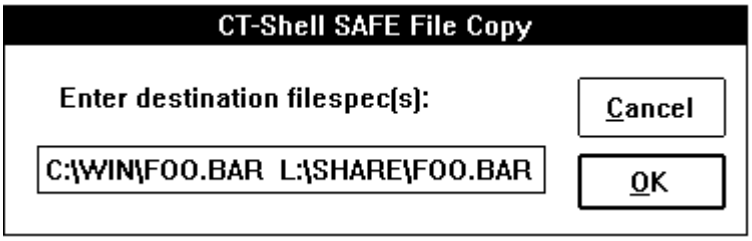


If you click on the [Accept] button, CT will accept the settings that are shown. If you click on the button marked [Accept and save settings], your new choices will be put into effect right away, and also stored in your CTSHELL.INI file, so you'll start off with those as default settings the next time .

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. This dialog provides a way to control options that were thoroughly explained in the previous chapter, so no more will be said about the individual settings here.

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.



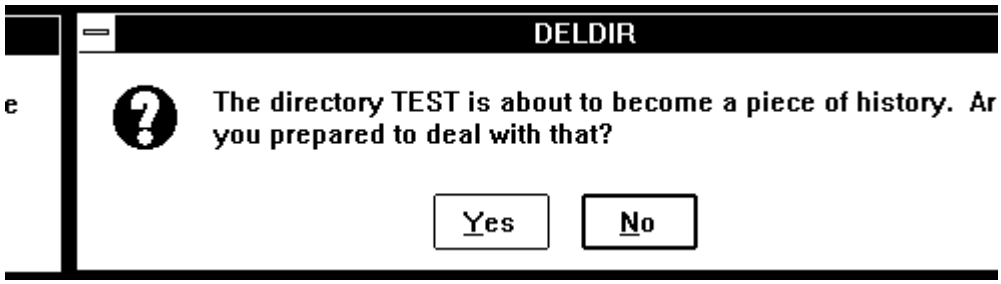
Note that *you may provide multiple destinations* when this dialog asks you for a destination. The example shown here would copy the current file to both the C:\WIN\FOO.BAR pathname and the L:\SHARE\FOO.BAR pathname. If the filename is to remain the same in the destination directory as it is in the source directory, of course only the directory (or directories) needs to be given.

Note that this dialog box shows that it is doing a SAFE file copy. That's because the *ShareTime* option was turned on at the time this copy was started. CT will do a little slower copy that may be safer for other programs that are depending on getting some attention. If *ShareTime* is turned off, the dialog will indicate that a FAST copy is being started——where CT will retain control until the copy is finished, no matter what other Windows programs are running.

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.[19]



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:

Deliver

This keyword is part of the CT-Shell Pickup/Deliver functionality, and is useful only after you've picked up one or more files using the Pickup keyword. It will copy any files from your current packing list to the current directory. *Deliver* does not remove those files from your packing list, so you are free to deliver them to any number of destinations.

If you want to deliver files to the current directory and also *Lose* those files from your packing list, you might want to use the *Surrender* keyword instead. *Surrender* simply invokes the *Deliver* routine followed by the *Lose* routine, saving you from doing both operations separately.

Delete

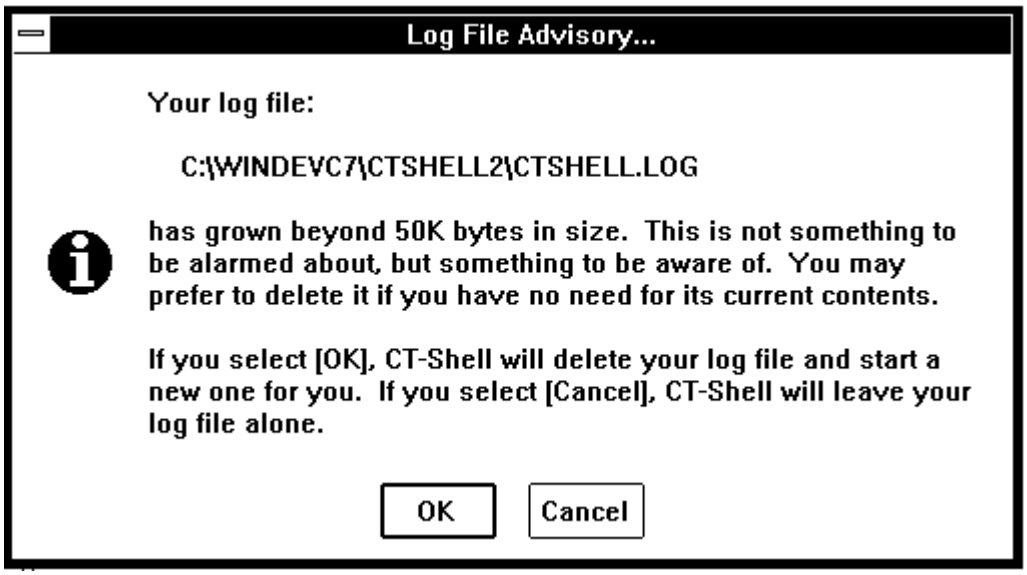
Deletes a file. This removes a file in a way that often cannot be reversed. Be careful, and be sure that you mean it when you use this keyword. You are asked only once for confirmation:

Because file maintenance so often involves deleting files, there is a keyboard shortcut for this operation. Note that either a single file or a list of tagged files can be deleted by pressing the <Del> key, then confirming. If you are deleting a *very* long list of tagged files, this confirmation will name the first several, followed by *etc...* If you are in doubt about any of them, check your files listbox carefully to be sure that you have tagged only those you really want to get rid of.

DelLog

The CT-Shell log file is a convenience that some appreciate, but there's little reason to let it grow too large. Usually there's no need to refer to operations you've done more than a day or so ago, and the log file could potentially eat a lot of your disk space if it is allowed to grow forever.

When your log file becomes larger than 50 KB in size, CT will call your attention to this fact the next time it enables logging. That occurs each time you start CT, or when you turn logging back on after having shut it off. You are given the option to delete the log file on the spot, or to leave it alone and continue.

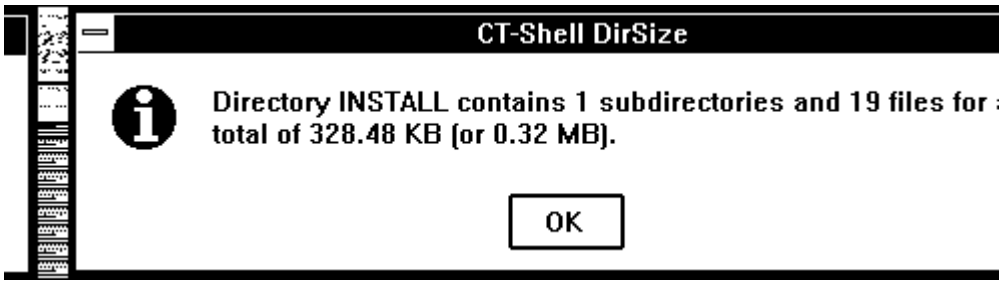


To make it easy to delete your log file at other times, this entry occurs in the default *Edit* menu, just under the entry that allows you to edit that file.

This operation will delete the currently used log file. If you have chosen to create a different log in each directory that you use, you will only delete the one in the current directory. If you have chosen a global location for your log file, then that file will be deleted regardless of where you are when exercise this option.

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.



EditIni

If you have provided a default editor name (in your *Preferences* dialog), CT will be able to edit your current CTSHELL.INI file wherever it may be, without your needing to change to that directory and execute your editor explicitly. Note that CT assumes a default editor name of NOTEPAD.EXE if you have not changed it, so this feature should work for everyone, right from the start. If you do change the default editor name, CT will thereafter use your editor instead of NOTEPAD.

Note that CT is always aware of which xxx.INI file that is currently being used, so if you have started an additional instance of CT with an alternative initialization file,

it will be that file that you'll edit by invoking this keyword. Thus, each instance of CT can safely use *EditIni* to modify its own initialization file, without worry that the wrong file might be changed.

EditIni is installed in the *Edit* menu by the default CTSHELL.INI file.

EditLog

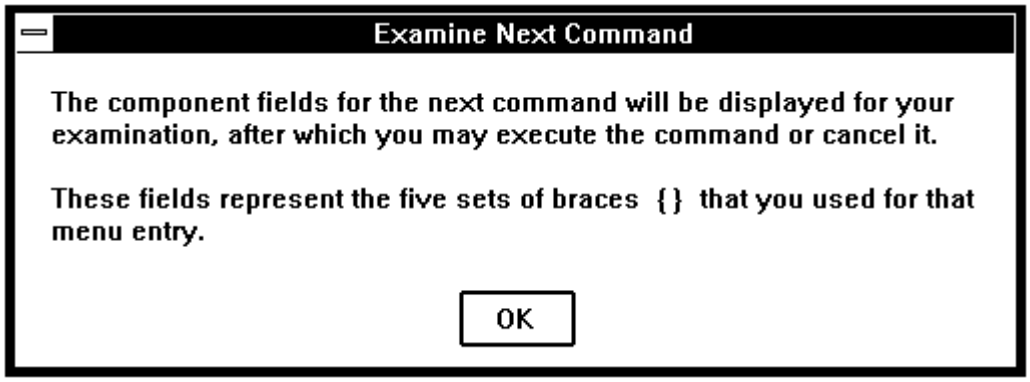
Use this keyword to invoke your default editor to view or change your current log file. As in the case of the initialization file, each instance of CT is aware of the name of its own log file, so you needn't worry about editing the wrong one.

The previous explanations about the log file path apply here as well. E.g., if you are using individual log files, the one in the current directory will be edited. If you have elected to use one global log file, that one will be edited, no matter from where you invoke this operation.

Examine

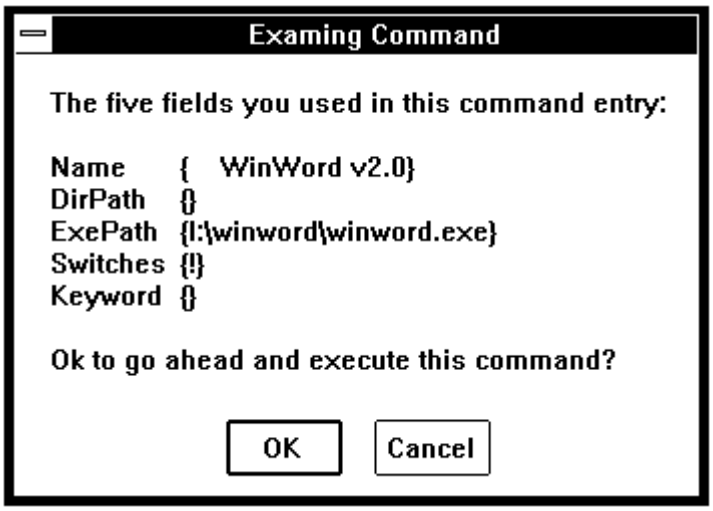
There may be times when you need to diagnose a menu entry that isn't working quite right. Perhaps you found yourself executing a different program than you thought you would, or editing a different file than you intended. The *Examine* keyword gives you a chance to see just what contents are in your entry fields at the time CT-Shell is just ready to execute your instructions.

There are two parts to an examination: first you set up CT so that it will intercept the next set of instructions instead of performing them, then you go ahead and give it a command normally. When you run the command, CT will first display a message box that shows you what it has to work with, then give you the option to go ahead and execute the instructions or cancel:



Note that this works with any operation that runs a program from the menus or from the command line——it does not diagnose operations that are strictly internal to CT, such as copying a set of files or deleting a directory. In short, it can diagnose any operation that involves an *entry*——including menu entries, alarm entries, or even *Autoexec* entries[20]——plus command-line operations that run programs, because CT internally prepares the same kind of entry for those as you would put into your menus.

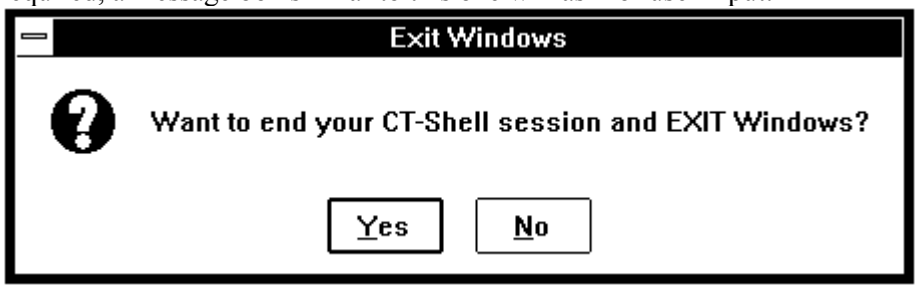
Here's an example of what an entry examination looks like:



Exit

Shuts down CT-Shell. You can also do this by double-clicking on the system menu box in the upper left corner of CT-Shell's window, but some people find it easier to pick an exit command out of a menu instead.

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, a message box similar to this one will ask for user input:



The exact wording of the message box will vary somewhat, depending on the state of several options that you may have selected from the *Preferences* dialog. Also, you can exit Windows only from an instance of CT that is your *root* Windows shell. If you invoke this keyword from within an additional copy of CT, you'll be asked whether you want to end this CT session only, and not given the option to exit from Windows.

Note that similar confirmation is required before restarting Windows or rebooting your computer, two other ways that you can exit from CT.

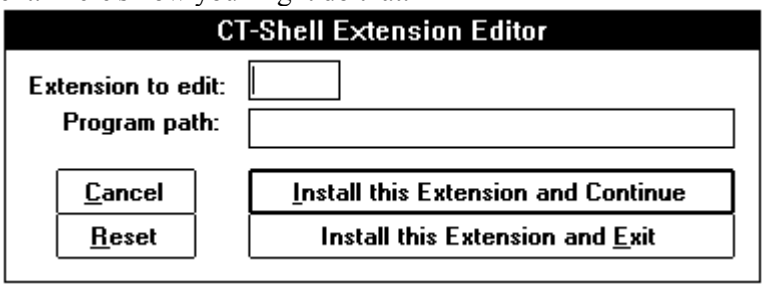
Extensions

File name extensions often provide a convenient way to identify the applications that work with that particular type of file. For example, .CRD files are usually used with the Windows CARDFILE utility. By default, .TXT files are associated with the NOTEPAD editor. You may use other applications, such as dBASE, which creates and uses .DBF files, or Excel, which creates and uses .XLS files.

CT-Shell makes it easy to start a program by doubleclicking on an executable filename in the files listbox, and it extends this capability to non-program file types that have an executable program associated with them. Your Windows WIN.INI file has a section called [Extensions] that lists the file name extensions you want associated with particular executable programs, and CT-Shell provides an easy way for you to view and

modify those associations without leaving CT.

Assume for a moment that you are a C programmer, and the most common thing that you do with individual .C files is to edit them. You can establish an association for the extension .C such that when you doubleclick on a C source file in your files listbox, your programming editor is started, and it is fed the name of that file as a command-line argument. Here's how you might do that:



Invoke the *Edit Extensions* entry in your *Shells* menu to bring forth the above dialog box, which invites you to type in the extension you want to check or modify. If you type in the extension C (no dot needed, since it is assumed) and press <Enter>, CT will display the program path/file name that is associated with that extension in the larger box just below. If there is no current association for that extension, the box will remain empty.

At that point, you can type in a new program path/file name or modify the one that is shown. You can continue to view/add extensions so long as you select the option to install and continue, or you can select the option to install and exit. CT-Shell sends a message to all Windows programs that are currently running, letting them know that this section of WIN.INI has been changed.

If you simply wanted to view an existing association——and not change it——the [Cancel] button lets you quit without changing anything. If you have already typed in something wrong, decide not to change it, and want the original value returned, you can select [Reset].

After establishing such an association for files that end in .C, you will be able to doubleclick on a filename like FOO.C in your files listbox, and automatically start your editor with FOO.C as its file to edit.

It bears repeating at this point that the extensions capability is a *very* important one, and that CT-Shell makes it particularly easy to use. You can make your sessions with the computer particularly enjoyable by setting up associations for all the file extensions that are most often used with particular programs.

FileClip

This keyword creates a file of the current clipboard data, if there is text data there. You are asked for the file name to use, and you may provide a simple filename or a complete path/file name. If you use a simple filename, the file will be created in the current directory.

Note that *FileClip* works pretty much like the DOS copy command——that is, it will copy the current clipboard contents over anything that was previously in the file you name, without prompting you for permission to overwrite that file. Be sure that you provide a name that won't cause it to overwrite a file you intended to keep.

You'll find this one implemented in the *Clipboard* menu by the default CTSHELL.INI file.

FileInfo

Copies the descriptive information (from the files listbox) 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 *Clipboard* 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  .....
```

Find

The *Find* keyword is an implementation of the same logic that was described in the earlier section about the *Find* command, which can be used at the CT-Shell command line. With it you can find all the occurrences of a string of characters within the files in the current directory, edit one of the entries, print a list of the matching files, or copy a list of matching files to the clipboard.

This one is for those users who would rather pick an entry from a menu, than open the command line and type the command.

FullScreen

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode.

This keyword causes a program to be run the full size of the display screen, rather than its normal default size. Some programs, such as CT-Shell when used without the *FlexSize* keyword, limit themselves to a fixed size, and will ignore this option.

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.

Hide

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode. In particular, DOS applications must have a PIF file that runs them in a window, or this will have no effect.

This keyword causes the program to be run in a hidden window, which means that it produces no visual output at all, not even an icon on the desktop. Those of you who might want to experiment with *Hide* should do so only after reading the following explanation carefully!

When a program is run in a hidden window, there is no way for it to display

output to the user, and no way for you to answer its questions. Therefore, a candidate for this mode of execution must be a program that *does not ask any questions*.

If a DOS program is to be run this way, its PIF file must specify that it is to be run in a window, and that it can run in the background. Remember, it must be a program that runs to completion without asking the user for any input.

WARNING: The following example is for the technically advanced user who is curious about this feature. If you find the explanation confusing, just ignore the *Hide* keyword as you use CT-Shell. There's nothing you can accomplish *with* this keyword that you can't accomplish *without* it.

The author, while experimenting with running hidden programs, developed a program-building batch file that can run quietly in the background, with no visual indication that it is running. That batch file runs a MAKE utility to build the program, redirecting its output (syntax errors, etc.) to a text file. At the end of the build, that text file is copied to the printer and a formfeed sent to eject any partial page.

The programs that are part of this process are all run with option switches that keep them from prompting the user for anything. If they encounter any errors, they simply report them and continue to completion. The batch file itself is run from a PIF file that specifies windowed operation and allows it to run in the background. The PIF file is run from a menu entry like this one, using the *Hide* keyword in the last field:

```
{   &BUILD.BAT} {} {c:\winpif\build.pif} {} {HIDE}
```

If you're the pioneer type, enjoy experimenting with *Hide*. It can be a lot of fun, but you need to be ready to deal sometimes with unexpected results.

Along these same lines, remember that CT-Shell will process a line that begins with HIDE= in your WIN.INI file, as well as the more ordinary ones that begin with RUN= and LOAD=. It should be obvious by now that CT runs any programs found there with the *Hide* keyword as the RunMode.

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.

As mentioned in an earlier section, you can reset the original path with a directory-change entry in your *Autoexec* processing, allowing you to establish a working directory for you session, no matter where you are when you start Windows.

Icon

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode.

This one causes the program to be run as an icon, rather than its normal size. It is a synonym of *Load*, and is the RunMode that is used to execute any programs that CT-Shell might process from the LOAD= line in your WIN.INI file.

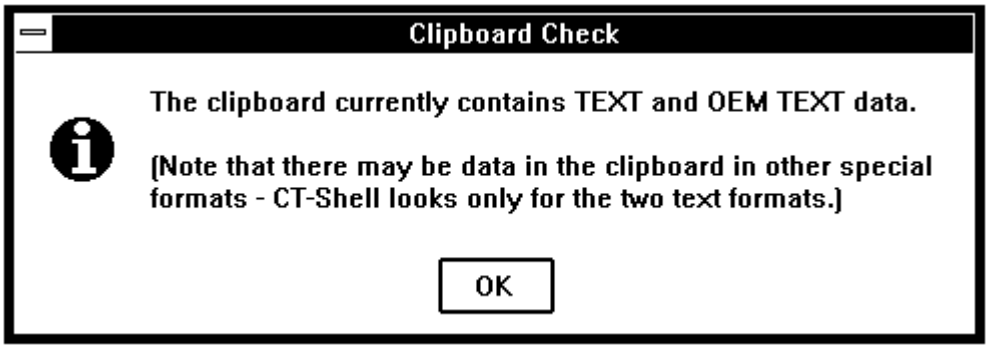
Invert

Inverts the condition of all the tagged files. Those that were selected become unselected, and vice versa. Thus, you can tag some of the files and do something with

them, then invert the tags and do something else with all the others. This is assigned to the <F4> key in the default configuration.

IsClip

Is there text in the clipboard? If you want to be sure before printing the clipboard, or before starting another application to use that data, you can test with this keyword. It presents a message box that looks like this:



As the message reminds you, IsClip tests only for text, either ANSI text or OEM text. It does not check for metafiles or other special graphic data formats, so don't assume that the clipboard is necessarily empty just because IsClip says there's no text there.

Load

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode.

This one causes a program to be run as an icon, and is a synonym for the *Icon* keyword (above).

Lose

When you have used *Pickup* to collect some files to be delivered elsewhere, they will remain onboard until you either *Surrender* them to a directory or *Lose* them. This keyword removes all the files from your packing list, so that you will no longer carry them around with you.

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 actually just comments on the existence of the two files identified by path names in the *MailIn* and *MailOut* entries in the [OPTIONS] section of your 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 other two files, possibly serving some other useful purpose instead.

Maximize

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode.

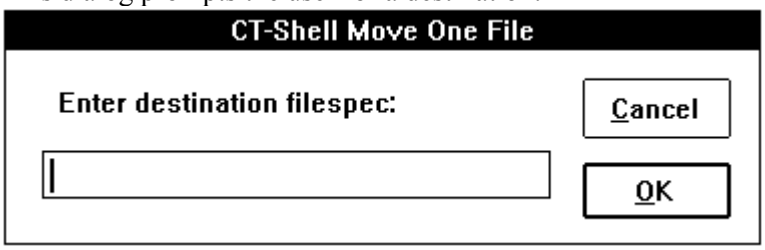
This one causes a program to be run full-screen, and is a synonym for the *FullSize* keyword (above).

Move

Moves a file to another location. If the destination is on the same drive as the source file, this feature changes the directory information relative to a file without copying the file itself. Thus, a move from one directory to another on the same drive takes only part of a second, no matter how big the file that is being moved. No file data needs to be read or written, just the directory entry for that file.

If this keyword is used to move a file across drives, CT will copy the file, verify that it arrived safely, and delete the original. Although that takes longer than moving a file to another directory on a single drive, the functionality remains the same.

This dialog prompts the user for a destination:



If the move is done on a single drive, it is done in one quick operation anyway. If it is done across drives, so that a copy is actually required, that copy is performed in accordance with your *ShareTime* option setting, explained under the *Copy* keyword, above.

MoveHere

You may want to *Pickup* files from one or more locations, then *Move* them to the current directory, rather than copying them to here with *Deliver* or *Surrender*. This keyword provides that service. It also removes those files from your packing list (since the original path/file names are no longer valid), so there is no need to *Lose* them after *MoveHere*.

This keyword is implemented by the *Move 'em Here* entry in your *Pickup/Deliver* menu by the default configuration.

Normal

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field.

This option is the default, if no other RunMode is specified in the last set of braces. As such, it is never actually needed, though some users may like to use it to document their entries explicitly. It is a synonym of the keyword *Run*, and it is the RunMode used whenever CT-Shell processes programs found in the RUN= line in your WIN.INI file.

OrigPath

Returns to your original path, or one that you have explicitly changed your original path to be. This is invoked by <F6> in the default setup, and provides a way easily and quickly to return to the drive/directory from where you started CT-Shell.

Remember that the *Home* keyword can be used to make any current directory the original path, as far as CT is concerned. Thus, if you start working in one path, then after a while you intend to be working somewhere else on your disk drive, you can invoke the *Home* keyword from your *Dirs* menu to mark that new place as your original path.

Also the original path can be set with a directory-change entry in the *Autoexec* portion of your CTSHELL.INI file.

Packing

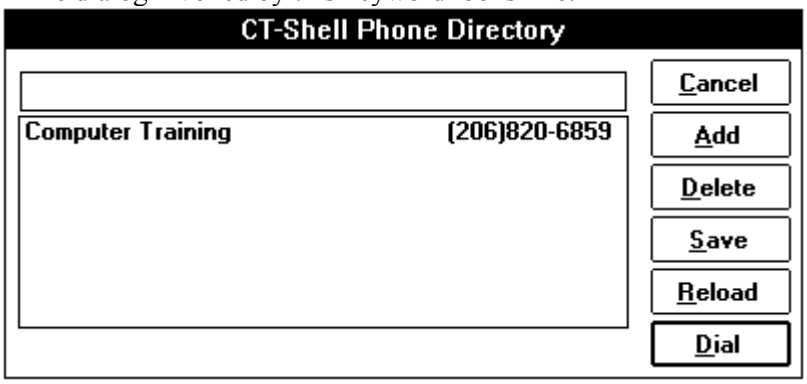
Displays a packing list, showing you all the files that you've picked up with the *Pickup* keyword, and haven't yet gotten rid of with the *Surrender*, *Lose* or *MoveHere* operations. You'll know that you're carrying files around with you by the appearance of the caption at the top of your CT-Shell window. To find out exactly what files they are, invoke this keyword.

You'll find it implemented in the *Pickup/Deliver* menu in the default configuration.

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 pretty 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 double-check 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 (especially if your hands are already on the keyboard), 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.

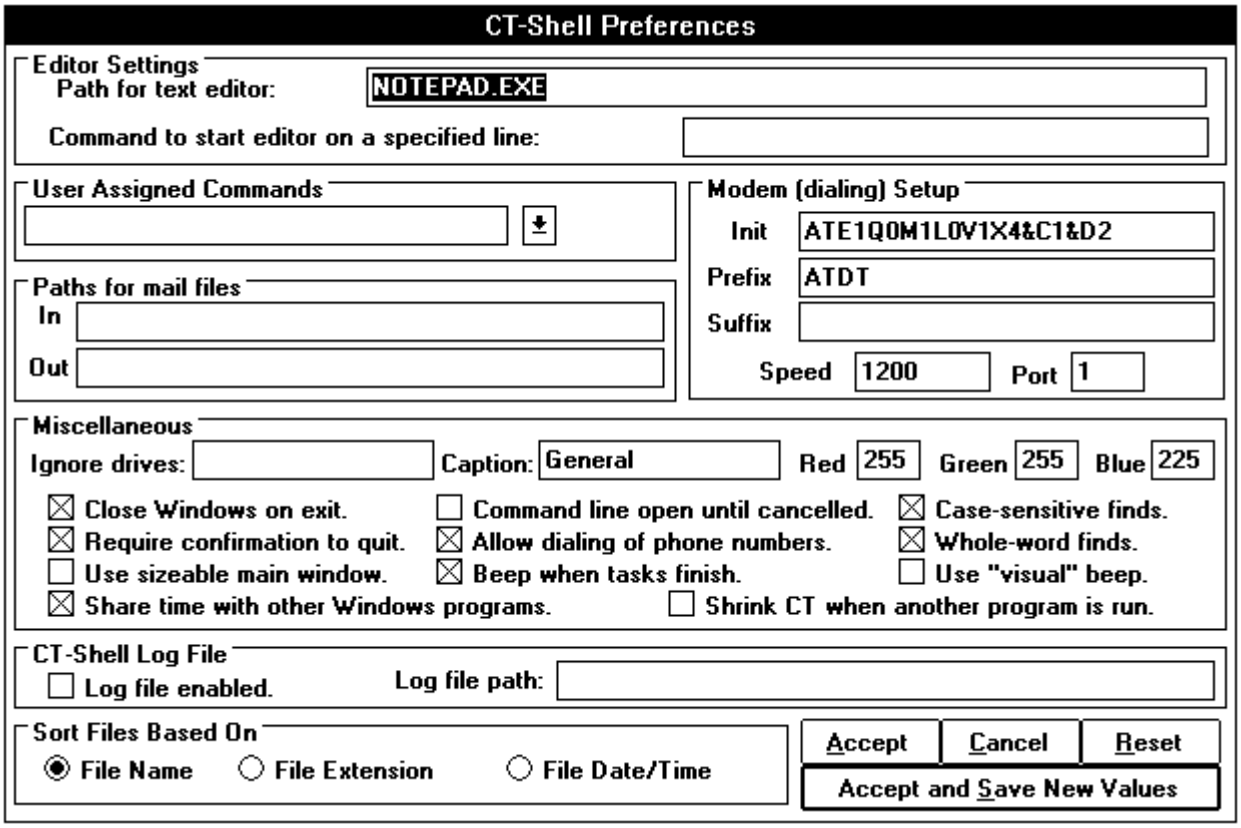
Position

CT-Shell 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 CT. 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 position for CT.[21]

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 will want to click on [Accept and Save New Values].

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.

Print

This begins the process of printing one or more selected files. Printed copies of text files are usually called *listings* when they do not refer to documents, such as letters.

CT offers a wide variety of options to control the way those listings are created, allowing you to choose headings, page numbers, line numbers, and more. See the description of the *Config* keyword, above, and the discussion of the printer options in the last chapter for more details about these options.

PrintClip

If there is text currently in the clipboard (see the *IsClip* keyword), you can print a copy of it with this option. Line size and whether to use fixed-width text is determined by your current printing options, which you can change by invoking the *Config* keyword.

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.

Since that display will be different for every printer driver, we won't show an example here. You'll recognize it, however, as the same dialog that is presented when you change your printer settings from the Windows Control Panel, or from within other applications, such as a Windows-based word processor.

An 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.

PrintList

Prints a copy of the *selected* files in the CT-Shell files listbox. Note that this is handled differently than the *Where* and *Find* listboxes, where all the entries are included when you ask to print them. By printing a list that includes only the selected files, you are able to use this function to create reminders of temporary files that you want to delete at a later time. Or you could make a list of all the files in a directory in which you are about to delete some files you consider unnecessary. That way, if it turns out later that something you deleted *is* really needed, you'll have a reminder of which ones you hosed.

This one is implemented in the *Clipboard* menu in the default configuration, although it doesn't really have anything to do with the clipboard. It is similar in design and operation to the *FileInfo* function, which copies the same information to the clipboard, so that seemed like the best place to put it. If you disagree, remember that it's easily moved!

Reboot

There may be times when you have made changes to a system configuration file (such as CONFIG.SYS or AUTOEXEC.BAT), and you need to reboot your system for those changes to take effect. The *Reboot* keyword does that from within CT-Shell, and is different than pressing <Ctrl+Alt+Del> in an important way. By invoking this keyword, you give Windows applications a chance to shut down in an orderly way before the rug is pulled out from under them, so to speak.

This operation sends a message to Windows, telling it to shut down, which lets you close any pending operations first. Windows will take care of notifying each process that it's time to quit, allowing an orderly shutdown of your system.

In the default configuration, this is implemented in your *Files* menu, along with *Exit* and *Restart Windows*.

Reload

If you have edited your CTSHELL.INI file manually, rather than changing its contents with the various dialogs that CT-Shell provides, you will need to let CT know that you want it to re-read the setup information and reconfigure itself. *Reload* does just that. In the default configuration, this is assigned to the <F7> key.

Remove

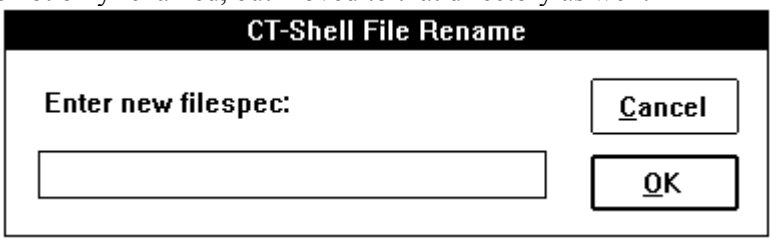
Removes the current directory, essentially duplicating the operation of the DOS command RD (or its synonym, RmDir). Like the DOS command, *Remove* requires that

a directory be empty before it can be removed.

This is found in the *Files* menu as the entry *RmDir* in the default configuration. If you want to delete a directory *and* all its files and subdirectories (and all their files, etc.), you can use the *Kill Directory* entry in that same menu. It invokes the *Deldir* keyword, explained above.

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.



After making major changes to any of the Windows configuration files (such as WIN.INI, SYSTEM.INI, etc.), you may want to close CT-Shell and restart Windows, so those changes can take effect. This keyword will restart Windows, after giving your applications a chance to shut down normally. (See the discussion of *Reboot*, above.)

The default configuration places this in the *File* menu, along with *Exit.*

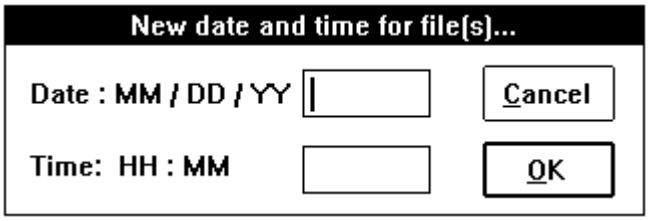
Run

One of the RunMode keywords that specifies how a program is to be run. This is used in the last field of a menu entry, the *Keyword* field, and affects only programs that can be run in this mode.

This is a synonym for the *Normal* keyword, and as such, it is the default if no other RunMode is specified. This is the mode that CT-Shell uses if it processes any programs in the RUN= line of your WIN.INI file.

SetDate

Displays a dialog box that allows you to change the date/time for one or more selected files.



Software developers, in particular, like to assign the same date/time to all the files in a release of a product, and quite often the "time" actually represents the version number of that release. You may want to do the same with all the individual document files that make up a large manuscript, such as a book or a play.

Just select all the files that you want changed, provide a new time and date via the above dialog, then select [Ok] to change them all or [Cancel] to quit without changing anything.

times are valid, however it doesn't check for leap years. It will allow you to enter 02/29/xx as a valid date for any valid year 19xx.

Although the same version of the routine is used in both places, for your convenience the default configuration puts this in both the *Files* and the *Tagged Files* menus.

Shred

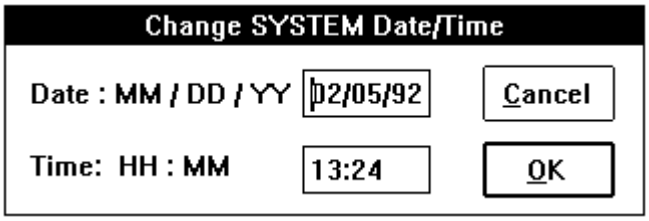
Destroys a file so that it can't be read even if it is somehow undeleted. This is analogous to feeding a document into a paper shredder, for security reasons. The confirmation dialog looks almost exactly like the one for the *Delete* keyword.

Shred works by first overwriting the file with all zero bytes, then deleting the file. Even if someone were to "undelete" it later, using a utility designed for that purpose, the file they end up with will be useless.

This is implemented in the *Files* menu by the default configuration, and there is an equivalent version that will handle multiple tagged files in the *Tagged Files* menu.

SysDate

This one lets you reset the system date and time, that is, the computer's main clock. It works like the DOS commands DATE and TIME, but it does it from within CT-Shell.



When first invoked, its dialog shows you the current date/time according to the computer's main clock, and allows you to change those entries. You can then click on [Ok] to reset the clock at that time, or select [Cancel] to exit without changing anything.

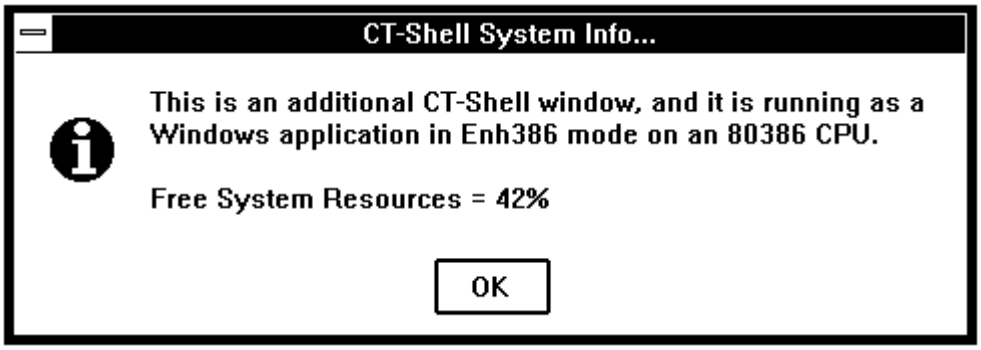
Surrender

Having picked up one or more files from one or more directories using the *Pickup* keyword, the commonest subsequent operation is to copy them somewhere else, and empty them from the packing list of files that you were carrying with you. The *Surrender* keyword handles both these tasks——the copying and the emptying. It is a combination of the routines *Deliver* and *Lose*, which are both available to be used separately.

Look for this one in your *Pickup/Deliver* menu, if you haven't changed the default configuration. It is also assigned by default to <F10>, because it is used so often.

System

Displays system information. This is similar to the 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 numbers. CT reports information that is obtained directly from Windows, and any difference should be so small as to be ignored.

The fact that resources are reported at 42% in the above example shows that a number of large and hungry programs were being run simultaneously as this manual was created. CT-Shell, by itself, requires very few resources.

TagAll

Tags (selects) all the files in the current directory. By default, this is assigned to the <F2> key.

Touch

Makes the date/time 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.

Untag

Untags all the files in the current directory. The default configuration assigns this operation to the <F3> key.

Where

The *Where* keyword is an implementation of the same logic that was described in the earlier section about the *Where* command, which can be used at the CT-Shell command line. With it, you can locate a file anywhere on the current disk drive, go to that location, print a list of all the matching files, or copy a list of matching files to the clipboard.

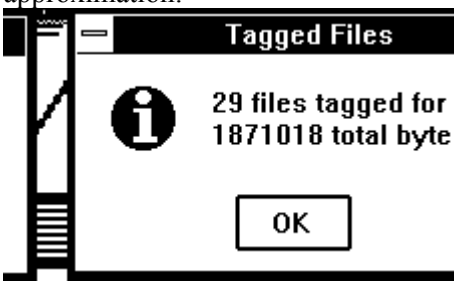
This one is for those users who would rather pick an entry from a menu than open the command line and type the command.

Below are documented the special keyword versions that work with a group of tagged files, instead of just the current file. In the default configuration, all of these are implemented in the Tagged Files *menu. Note that any of these could be used to handle a single file (unless it were explicitly untagged with <F1>), but the reverse is not true:*

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:



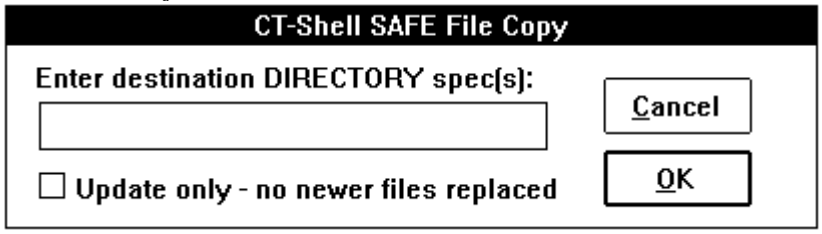
Tattrib

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[22]. 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 backup 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.

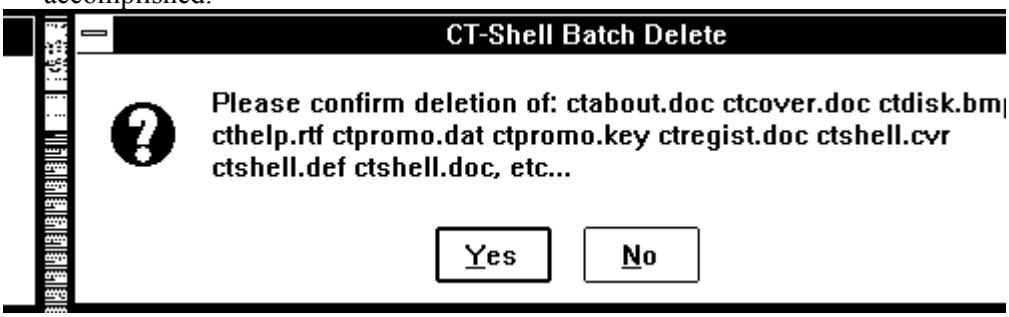
This feature is really convenient if you have a main directory on a server for the storage of document files or program source files. You can work with copies of those files in a local directory, and easily update the server with your revisions, without taking a chance that you'll accidentally overwrite any newer files.

As with the dialog for the single file *Copy* keyword, this one indicates that a SAFE copy will be done (pausing occasionally to let other Windows programs run),

because the *ShareTime* option has been turned on. Otherwise, it would indicate a FAST copy, whereby CT would copy all the files before giving up any time to another program.

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, if the move is from one place to another on the same disk drive, these files are not physically copied to their new location, just their directory entries are changed. If the move is done across drives, a copy is first done, and the original file removed after verification that the copy was successful.

Tshred

Shreds a set of tagged files, so that their data is unrecoverable even if the files themselves are undeleted. The dialog presented for this keyword is nearly the same as the one presented for *Tdelete*.

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 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 point 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:[23]

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

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

If you give CT a command this way that turns out to be longer than the allowable 127 characters, it will let you know about it and offer an opportunity for you to abort the operation or truncate the command to less than 127 characters and go ahead with it.

If asked to truncate the command and go ahead with it, CT will try to shorten it to a place where there is a space. That way, if you are sending a number of files to an editor, for example, you won't inadvertently create a new file whose name is part of a legitimate name.

However, after such a truncation, be sure to check how many files you actually *did* edit, or pack into an archive, etc. You will almost certainly need to select the remaining files and repeat the operation.

@

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, then 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?

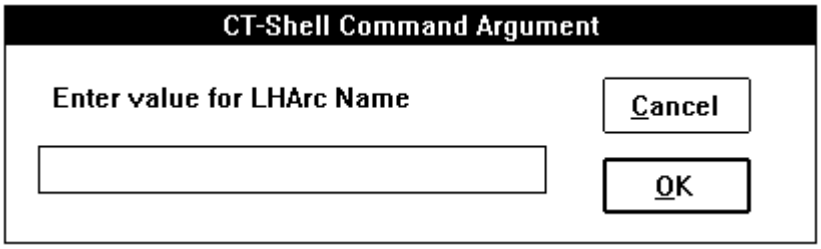
A pair of question marks surrounds the prompt you want CT to display when it asks you for a string of characters to put in its place. This is how you can supply variable arguments at the time an entry is executed.

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 ?LHArc 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 ?LHArc 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 usually 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. Note that some programs that you run this way may take exception to the missing argument, but most will handle it gracefully.

%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 variable called WINDIR that you can use in your command entries wherever you need to refer to the Windows "home" directory. Although you don't set this one in your AUTOEXEC.BAT file, Windows sets it whenever you start a Windows session.[24]

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.

; Comments

The semicolon is not a character that takes on a special meaning inside a field, but a character that can be used to temporarily turn an entry into a comment. To be used that way, it must appear in the first position on a line.

Normally, any text that is not enclosed in braces is considered to be comments, and is ignored by CT-Shell. Sometimes, however, you might like to turn an entire line into a comment, fields and all, so that you can leave the line in your CTSHELL.INI file and still have CT ignore it.

This is sometimes a useful way to temporarily disable an *Autoexec* entry, for example, making it easy to re-enable it at a later time. Or you could use this method to temporarily remove an entry from a menu, letting you decide later whether to remove it permanently by deleting the line, or put it back by removing the semicolon.

(This page intentionally left unused...)

[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] Users of previous versions of CT should note that this means 42 new keywords version 1.xx. Be sure that you include all of them in your CTSHELL.INI file if you are upgrading it manually from a previous version!

[3] 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 100K in size, you'll find that the program actually requires about 50K or so to run, depending partly on your system.

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

[5] 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.

[6] The *Preferences* dialog is available from the *Shells* menu in the default configuration.

[7] 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.

[8] 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.

[9] 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!

[10] 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.

[11] 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.

[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 that everyone should try out as soon as possible.

[13] However, the events are not checked each time the status display is updated (once a second). They are only checked one time every minute, to see whether one or more of them should be started.

[14] 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.

[15] 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.

[16] CT starts out with the assumption that a command needs to be passed on to the DOS command processor if it is a .BAT or .COM file, since Windows executables are all .EXE files. It also passes on to DOS any commands that include redirection symbols, such as <, >, >> or the pipe symbol, |. Finally, any commands that do not have a file extension are first assumed to be internal DOS commands, like DIR, and are also sent to DOS.

If a command has an extension of .PIF or .EXE, it is processed instead by the Windows command execution function, WinExec, which can run a DosApp just fine, too. The only complication arises when a Windows program that ends in .EXE is run without providing the extension. It is first assumed to be a DOS command, so it is passed to the DOS command processor (usually COMMAND.COM). Only when that fails is it passed to the Windows execution function. Although the program *is* run correctly *eventually*, this results in an annoying flash of FILE NOT FOUND in the DOS box, and an unnecessary delay.

[17] 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.

[18] 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.

[19] 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.

[20] To diagnose an entry that runs from your *Autoexec* section, simply add an entry just before it that invokes this *Examine* keyword. To diagonse an entry started from the *Alarms* processing, you can either schedule an *Examine* entry to be run the minute before the one you want to test, or simply select *Examine* from your menu just prior to the end of your active session. Remember, it will be the *first* operation after the exam setup that will be trapped.

[21] 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= (or change them both to zeros). 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.

[22] 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, thereby saving some program size and complexity.

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.

[23] 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.

[24] For those users who are technically inclined, we should mention that CT-Shell doesn't get this environment variable from the DOS environment——it gets it from Windows, itself. Thus, if the environment variable should be changed by some other application for its own purposes, CT will still know where the Windows home directory is. For the purposes of using this in CT menu entries, however, it may be considered to be an environment variable.