INCOMMAND

Command Line Utilities
for DOS

Version 2.0

Manual Revised 1/14/93

from Inductive Logic

What Is Inductive Logic ?

In science, inductive logic is the process of inferring general principles from observed phenomena. In software, Inductive Logic means general purpose utilities, based on the observed needs of computer users. Inductive Logic solutions adhere to the principles of consistency, power, and flexibility. We believe in the intelligence of our users, so our software does exactly what you tell it to do. We dislike software that thinks it is more clever than we are, thereby introducing inconsistencies which are far more annoying than useful. The consistency of Inductive Logic products makes them easier to learn, automate (e.g. .BAT files), and interpret their results. Occasionally, we have been frustrated by the inconsistencies inherent in the operating system. Since we have no control over DOS, we have made some compromises to be consistent with existing DOS commands.

We hope you like Inductive Logic products, but there's always room for improvement. Let us know what you think. What do you like? What do you dislike? How's our documentation? What would you like to see in the future? Use the Reader Response form at the back of this manual, or write us:

Inductive Logic
P.O. Box 26238
San Diego, CA 92196
(619) 578-5146

We're Members of the Association of Shareware Professionals

Inductive Logic is a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware

principle works for you. If you are unable to resolve a shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products. Please write to the ASP Ombudsman at 545 Grover Road, Muskegon, MI 49442 or send a CompuServe message via CompuServe Mail to ASP Ombudsman 70007,3536.

Why Registering Is Important to You

For one thing, registering your software will give you a disk to eliminate the registration reminders on every command. It will also give you a nice printed manual. But in the larger view, remember that without software licensing/registration, no one would publish software, and you wouldn't be able to do anything with your computer. Like patents, software copyrighting encourages development in this critical technological area. So copyrighting is important to you as an individual, and to America, to keep us competitive in a world market.

DISCLAIMER OF WARRANTY

THIS SOFTWARE AND MANUAL ARE SOLD WITHOUT WARRANTIES AS TO PERFORMANCE OR MERCHANTABILITY, OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. BECAUSE OF THE VARIETY OF HARDWARE AND SOFTWARE ENVIRONMENTS INTO WHICH THIS PROGRAM MAY BE PUT, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. GOOD DATA PROCESSING PROCEDURE DICTATES THAT THE USER THOROUGHLY TEST ANY PROGRAM WITH NON-CRITICAL DATA BEFORE RELYING ON IT. THE USER MUST ASSUME THE ENTIRE RISK OF USING THE PROGRAM. ANY LIABILITY OF THE SELLER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF PURCHASE PRICE.

Table of Contents

GETTING STARTED

Another Utility Package ?

Yes, another utility package! Why? Because existing ones just aren't good enough, that's why. There are a number of good utility packages for various specialized functions, but none for normal, everyday command line use. For example, would you like to see all the files you've created since this morning? Or delete all the files of a certain type that are more than a year old? Or find out if a group of files will fit on your floppy disk? Would you like to be able to do any operation on a whole disk as easily as on one file? And would you like to do it without learning a whole new command structure? Inductive Logic's **InCommand** provides all this, and more:

- Multiple *'s in file and directory wildcards
- Select (and exclude) multiple wildcard patterns in a single command
- Process an entire disk as easily as 1 file
- Select files before, after, or on any date or time, or between any 2 times
- Select files bigger or smaller than any size, or between any 2 sizes
- Select files including or excluding any combination of attributes
- Move files or entire directories without copying and deleting
- Large numbers (like file sizes) are printed with commas for easy reading
- Utilities run as DOS commands. No menu walking to slow you down.
- Designed for both interactive and batch use
- All utilities return exit status codes (ERROR LEVEL)
- Incremental backup to multiple floppies
- Fast incremental directory updates

Users at all levels (beginners, intermediate, and advanced) will appreciate the command format that is a natural extension of the DOS format they already know. **Beginners** will like the feedback, because InCommand utilities display both individual and summary information about the files affected, so you know the commands did exactly what you expected. By default, potentially dangerous operations prompt with specific information for confirmation

before proceeding. **Intermediate** users will appreciate the total power and flexibility the file selection criteria provide. And because InCommand utilities are designed for both interactive and

batch users, advanced PC users will appreciate the ability to turn off confirmation prompts for any operation, and the exit status codes provided by each utility.

Why don't we use menus? Menus are good for many applications, and we like them. However, they do have some inherent limitations. We have specifically chosen to implement command line utilities because they are most flexible, they can be automated in batch files, and because they are usually much faster to use than menus.

The InCommand utilities are:

CHATT	Changes file attributes
DAYTIME	Prints the current day of the week, date, and time, or timestamps a message
DHELP	Instant on-line help for all DOS commands
DI	Makes a directory listing
EXECUTE	Runs your own program or batch file as if it had InCommand wildcard-filespecs
ICOPY	Copies files
IDEL	Deletes files
IHELP	Instant on-line help for InCommand
IRD	Removes (deletes) directories
IREN	Renames files
MOVE	Moves files to other directories
MOVEDIR	Renames subdirectories, or moves them to other directories
SEARCH	Search for text strings in files
TOUCH	Sets file modification times to any value
WHICH	Searches your PATH for the executable file for a DOS command
WIPEDIR	Deletes entire directory trees

How to Use This Manual

This document has 4 main sections: Getting Started , Syntax Reference, Utility Reference, and Applications. Getting Started is a quick, easy-to-read introduction to the basic operations. Everyone should read this section. The Syntax Reference describes the general command syntax, shared by all InCommand utilities. New users need not read this at first, but should refer to it to answer specific questions, and to learn more advanced operations. The Utility Reference lists all utilities

alphabetically, and fully describes each. This section is appropriate for all levels of users needing detailed information about a particular utility. The Applications section describes some typical tasks that PC users need to do, with descriptive examples of how InCommand helps you get them done.

This document assumes that you are familiar with some of the common DOS commands and utilities, and with DOS file and directory concepts. Refer to your DOS documentation, or the DHELP utility, for more information on these topics.

Notation

When describing command syntax in this document, we print all text which you would type literally (exactly as shown) in **BOLD CAPS MONOSPACE.** Parameters for which you would substitute your own text are given in italics (e.g., *filespec*). As usual, square brackets surround [optional parameters], but you don't actually type the brackets. Control characters are listed as **^X**, or as a keyname in angle brackets, **<ESC>**.

We're not sticklers for mathematically rigorous syntax specifications, so we use a less formal notation, because it's easier to understand. For example,

XYZ *filespec* [**/BEFORE:***date:time*]

means type "XYZ" followed by a space and a filespec of your choice, optionally followed by "/BEFORE" and your choice of dates and/or times. Even though you have many options for specifying a date, we don't clutter the syntax description with nested square brackets and such. The complete Date/Time syntax is described in the Syntax Reference section.

Because each of the InCommand utilities is a "command" to DOS, we use the terms "command" and "utility" interchangeably.

Also, the term "directory" means any directory (set of files and subdirectories) on a disk. A "subdirectory" is just a directory like any other. The prefix "sub" is used only to help you understand that the subdirectory lies below another directory in the hierarchy.

Installation

InCommand requires an IBM compatible PC, DOS 3.0 or higher, and 256 kbyte of RAM. You must install InCommand with the INSTALL command on the master floppy, because INSTALL does more than just copy the files from the floppy. INSTALL will put the InCommand utilities in a directory of their own, and (if you wish) place that directory in the PATH command of your AUTOEXEC.BAT. To install from the A: floppy drive, just type

```
A:INSTALL
```

and answer the questions that INSTALL asks. Of course, if your floppy drive is not A:, use your drive letter in place of the "A" in the above example.

Be sure to check the README.TXT file in the Inductive Logic directory for any late news that did not arrive in time for printing in the manual. You can read it on your screen by typing:

```
MORE <\IL\README.TXT      assuming "\IL" is your InCommand
                          directory
```

You can print it by typing

```
COPY README.TXT PRN:
```

Tutorial

In this section, we introduce you to some of the common InCommand utilities, and the general command syntax. The syntax is common to all of the InCommand utilities, and because it is similar to DOS, it is easy to learn and remember. We suggest you sit at your computer while reading this section, and try some of the examples as you go. Most of this manual is available on-line with the IHELP utility. IHELP allows you to browse through the help text, and instantly locate help on specific topics. To use it, just type

```
IHELP
```

and follow the directions that appear. All InCommand utilities provide on-line usage help, which can be seen by typing the command with the /? modifier.

InCommand provides commands to list directories (DI), delete files (IDEL), remove directories (IRD), copy files (ICOPY), move files without copying (MOVE), change file attributes (CHATT), search for text strings in files (SEARCH), change file modification times (TOUCH), and more. These commands use a familiar syntax similar to DOS, but with greater capabilities. Throughout the utilities, similar functions use the same command modifiers.

In this tutorial, we introduce you to the most common utilities: DI, IDEL, IRD, ICOPY, and MOVE. The DI (DIrectory) command is probably the most used of the InCommand utilities. This command lists the contents of directories, similar to the DOS DIR command. For example, to list the current directory, type

```
DI                      list the current directory, sorted
                        alphabetically
```

Right away, you notice how much better than DOS the directory listing is:

- the files are listed alphabetically
- file sizes have commas every 3 digits
- attributes are displayed with each directory entry
- complete file times are listed

Or, you can have the files sorted any way you want. For example,

```
DI /BYEXTENSION list the current directory, sorted by
                        extension
```

```
DI /BYTIME      list the current directory, sorted by file
                        modification time

DI /BYSIZE      list the current directory, sorted by file
                        size
```

Files and directories can be selected using wildcard characters. As with DOS, the "*" wildcard can be substituted for any number of any characters (except ".") in a file name:

```
DI *.OBJ                list all .OBJ files

DI A*.OBJ               list all .OBJ files beginning with
                        "A"
```

Better than the DOS "*" wildcard, the InCommand "*" wildcard can be anywhere in the filename:

```
DI *A.OBJ               list all .OBJ files whose names end
                        with "A"

DI A*B.OBJ              list all .OBJ files whose names
                        begin with "A" and end with "B"

DI A*B*C.OBJ            list all .OBJ files whose names
                        begin with "A", end with "C", and
                        have "B" anywhere in between
```

Another wildcard character is "?". The "?" can be substituted for any one character in a file name (except blanks or the "."):

```
DI ?A*                  list all files whose name has "A"
                        as the second character.
```

With InCommand you can also specify multiple wildcard filename patterns. You can only give a single path specifier, but you can include any number of filename patterns by seperating them with commas. For example, to see all of the *.BAK and *.TMP files in subdirectory A, type

```
DI A\.BAK,.TMP          list all *.BAK and *.TMP files in
                        subdirectory A
```

InCommand wildcards can also be used in directory names:

DI A*B\	list the contents of all the subdirectories of the current directory whose names begin with "A", and end with "B"
DI A?B\	list the contents of all subdirectories whose names are 3 characters, beginning with "A", ending with "B", and having any character as the second character.

Sometimes, you'll want to perform an operation on all files in a directory, and all files in all subdirectories below it (i.e., on a "directory tree"). We us the syntax ". . ." to specify a directory and all subdirectories below it (a directory tree):

DI ...	list the directory tree starting at the current directory.
DI A\...*.OBJ	list all the .OBJ files in the directory tree starting at subdirectory A.

An easy way to think of the ". . ." wildcard is to note that it is replaced by zero or more subdirectories. That is, the set of directories specified by ". . ." is first, the directory given preceding the ". . . ", then all its subdirectories, then all their subdirectories, and so on.

Notice also that in the preceding example, the backslash after the directory A could be omitted. We allow this abbreviation to reduce your typing:

DI A...*.OBJ	is the same as	DI A\...*.OBJ

Command modifiers can be used to select files by date and time:

DI /SINCE:12-23-89	list entries in the current directory that have been modified since December 23, 1989.
DI /BEFORE:1-27:16:10	list entries in the current directory that were last modified before 4:10 pm on January 27 of the current year.

DI /ON:MONDAY	list entries in the current directory that were last modified on Monday
DI ...*.OBJ /BE:1-1-90 /SI:12-1-89	list .OBJ entries in the tree starting at the current directory that were last modified in December, 1989.

Notice that in the last example the /BEFORE and /SINCE modifiers are used together to select a time interval. Also, they are abbreviated. All keyword modifiers can be abbreviated to the minimum number of characters necessary to uniquely identify them.

Other InCommand command modifiers allow you to select files by their size or file attributes (see the Syntax Reference section for more information).

To delete files, the IDEL command uses the same file specification syntax as the DI command (in fact, all of the InCommand utilities use the same syntax). Here are some examples, but **don't try them unless you really want to delete some files:**

IDEL ...*.BAK	delete all files with the extension .BAK in the current directory tree.
IDEL ...*.*	delete all the files in the current directory tree. (Be very carelist .ful of this!). This example strips all of the files from the current tree, but leaves the directory structure in place.

Sometimes, you want to pick and choose among a set of files, deleting some and keeping others. IDEL lets you do this easily, with the /INDIVIDUALLY modifier:

IDEL *.BAK /IND	prompt on each file and delete only if user replies "Y"

IDEL will list the first file matching *.BAK, and ask if you want to delete it. If you say "N", the file is kept. If you say "Y", the file is deleted. Either way, IDEL then shows you the next file matching *.BAK, and asks the same question. In this way,

you can step through all the files, and delete only the ones you want.

The IRD command removes (deletes) directories, but only if they are empty. For example, the IRD command can be used to remove a tree of empty directories:

```
IRD ...*.*          remove all empty subdirectories in
                    the current tree.
```

Note that you can use /INDIVIDUALLY on IRD, just like IDEL. For deleting directory trees, see also the WIPEDIR command in the Utility Reference section (page 84).

The **ICOPY** command copies files from any disk or directory to any other (or the same) disk or directory:

```
ICOPY SOURCE.TXT NEWDIR\DEST.TXT
                    copy the file SOURCE.TXT into the
                    file NEWDIR\DEST.TXT
```

If the destination directory path does not exist, ICOPY will ask if you want to create it.
You can copy a whole directory tree, in this example to the A: drive, with the following command:

```
ICOPY ...*.* A:   or   ICOPY *.* A: /S
                    copy the entire tree starting at
                    the current directory to A:
```

For those of you used to the DOS XCOPY/S command for copying directory trees, notice that we also support the "/S" modifier, for your convenience.

ICOPY can also flatten a tree of subdirectories into a single directory:

```
ICOPY A\...*.* B\ /FLATTEN
                    copy all the files in the tree
                    starting at A\ to the single
                    directory B\
```

A unique feature of the InCommand utilities is the MOVE command. The MOVE command moves files from one directory to any other on the same disk, without actually copying the file and deleting the original. Because the file is not copied (only directory entries are changed), the MOVE command is instantaneous no matter how big the file is. For example, if you have a file named X in

directory A\, you can move it out of directory A\ and into directory B\ with this command:

MOVE A\X B\	move the file X from directory A\ to directory B\

You can also change the name of the file when moving it. For example,

MOVE A\X B\Y	move the file X from directory A\ to directory B\, and change its name to Y

Even more powerful, you can move an entire directory to another directory on the same disk with the MOVEDIR command. Notice that by moving a directory, we mean that all of its contents are moved with it, including all lower subdirectories, i.e. the whole directory tree. Like the MOVE command, nothing is copied or deleted, only directory entries are changed. This means you can move hundreds of megabytes in no time. This is a more advanced topic, and you can read all about it in the MOVEDIR description in the Utility Reference section.

Aborting Commands

You can abort any InCommand utility while it is executing by pressing ^C (Ctrl and C simultaneously). It may take a few seconds to abort. You can make utilities abort faster by using the DOS BREAK ON command. See your DOS manual, or the DHELP utility, for more information.

Where Do I Go Now?

You're now ready to start being more productive every day with InCommand, the Inductive Logic DOS Command Line Utilities. All the utilities you've just seen have many more capabilities than described in this introduction. To find out everything a command can do, read its description in the Utility Reference section. There, you can also find out about other utilities which allow you to move directories (MOVEDIR), to modify file attributes (CHATT), to modify the time and date of a file (TOUCH), to display the current date and time (DAYTIME), to determine the

location of an executable file (**WHICH**), to run your own programs or batch files as if they had all the InCommand file selection capabilities (**EXECUTE**), to search for text strings in files (**SEARCH**), and more.

If you want to learn some of the more advanced features, such as complete wildcard filespecs, command modifiers, and using InCommand utilities in batch files, you should read the Syntax Reference section, immediately following. If you want to see examples of typical chores for which InCommand can save you time, read the Applications section of this manual.

Customer Support

Customer support is available by mail or phone. If you have any questions about our products, you can mail them to us, or call us. When you call, you may get an answering machine. If so, please leave a message with (1) your name, (2) the date and time you called, (3) your time zone, (4) phone number, (5) product registration number (from your license agreement), and (6) the best time to reach you (so we don't play extended telephone tag). Then (7) describe your question. We will get back to you. Please be sure to leave all 7 items listed here, or we may not be able to return your call. You can write/call us at this address:

Inductive Logic, Customer Support
P.O. Box 26238
San Diego, CA 92196-0238
619-578-5146

SYNTAX REFERENCE

Command Structure

InCommand command lines are comprised of **commands, filespecs,** and **modifiers.** The **command** is just the name of the utility (e.g., ICOPY). A **filespec** designates a file (or directory entry), or set of files (or directory entries). A filespec may use wildcard characters to identify a set of files. A filespec may be either a wildcard-filespec, or a destination-filespec. **Modifiers** alter the operation of commands in some way. Filespecs and modifiers are completely described in the following sections.

Wildcard File Specifications

Most InCommand utilities accept wildcard file specifications, shown in the syntax description as a wildcard-filespec. InCommand wildcards are much more powerful than DOS's, and they are a superset, so all standard DOS wildcards work with InCommand utilities.

Note: Directories vs. Their Contents

To be consistent in our wildcard file specifications, we have had to introduce one small difference between our **DI** directory command and DOS's **DIR** command. We are upward compatible with DOS in most ways, but we could not find a reasonable way to be compatible with DOS's inconsistency. The difference occurs when listing the contents of a subdirectory. In InCommand utilities, a subdirectory <u>entry</u> is specified by its name without a

"\" after it, and the subdirectory's <u>contents</u> are

specified by the subdirectory name with a "\" after it. In DOS, a subdirectory and its contents are often ambiguous because they use the same syntax, and are inconsistently interpreted by nearly identical utilities (try it with **COPY** and **XCOPY,** for example).

Therefore, in the InCommand DI command, if you type

```
DI SUBDIR
```

you get a single entry for the subdirectory named "SUBDIR":

```
C:\
SUBDIR          <DIR>      9-23-90  16:10:00
  1 directory,  0 files,  0 (0) bytes.
  31,922,654 bytes free.
```

To see the contents of this subdirectory, you must add a backslash:

```
DI SUBDIR\
C:\SUBDIR\
CHRIST.MAS            123  12-25-89   4:56:12
YEAR.NEW            2,048   1-01-90   0:00:02
BIRTH.DAY          15,432   1-27-90  22:05:44
  3 files,  17,603 (24,576) bytes.
  31,922,654 bytes free.
```

The Wildcards: ?, *, ...

First off, InCommand accepts the standard DOS "*" and " ?" wildcards. The InCommand "?" wildcard matches any non-blank character. (The DOS "?" matches blanks, also, but we have found it more useful not to match blanks.) For example, to list all files with two characters in their name, that start with A, type

```
DI A?
```

The "*" wildcard matches any number of any character (except dot "."). Unlike DOS, you can specify any number of "*" wildcards, and they work as expected. Therefore, A*BC* would match ABC, AXBXBC, and AXYZBCUV. Also, wildcards are allowed in both directory paths and filespecs. Thus you can type

```
DI ?B*\XYZ
```

to list all the entries for XYZ in subdirectories with "B" as the second letter.

InCommand utilities also have a third wildcard, which is only allowed for directories. It is ". . .", and it means "this and all subdirectories below this," or in other words a "directory tree". A directory tree includes all the subdirectories of the given directory, and all their subdirectories, and so on. To list all entries in the current directory tree, type

```
DI ...   or   DI ...*.*
```

You can only use ". . ." at the end of a directory path specification, but you may follow it with a wildcarded filename. Thus, to see all the files named "X" in the current directory tree, type

```
DI ...X
```

To see all the files with ".BAT" extensions in the current directory tree, type

```
DI ...*.BAT
```

Notice that there is a difference between ". . ." and " *\". Whereas ". . ." includes the current directory, and all subdirectories at any level below the current directory, "*\" does not include the current directory, and it means only the immediate subdirectories of the current directory, but none below them. For example, if your directory structure looks like this:

```
TOP\
   A\
      A1\
      A2\
   B\
```

If you set your current directory to TOP\, then the filespec ". . .*.*" includes all files in directories TOP\, A\, B\, A1\, and A2\. The filespec "**.*" includes only files in A\ and B\.

For most applications, the syntax is pretty intuitive, so type what you think, and it will probably work.

Note

Some file networking software uses the ". . ." syntax for their own purposes. Be aware, when using InCommand utilities, the syntax will always be as described here, and ". . ." will not have any network meaning.

Novell, for example, uses ". . . " to mean "the parent of the parent." When using InCommand on Novell networks, you must use the DOS standard syntax of ". .\. ." for that purpose.

Implied Wildcards

For commands which do not modify files (e.g. DI), you may omit the "*" wildcard specifier for either names or extensions. The omitted names or extensions are "implied wildcards." For example,

DI .BAT is equivalent to **DI *.BAT,** and

DI NAME is equivalent to **DI NAME.***

Commands which modify files do not allow implied wildcards. So if you want to delete all files with .BAT extensions, you must type

IDEL *.BAT

Note that directories never have implied wildcards. If you want to include all extensions of a directory name, you must explicitly include the ".*" wildcard. For example,

DI A.*
lists the contents of all directories with name "A" and any extension.

Multiple Wildcard Patterns

Any wildcard-filespec allows multiple wildcard filename patterns. You can only give a single path specifier, but you can include any number of filename patterns by separating them with commas. All files in the specified directories which match **any** of the wildcard filename patterns will be selected. For example, to see all the .OBJ and .EXE files in the current directory tree, type

DI ...*.OBJ,.EXE list all the *.OBJ and *.EXE files in the current directory tree

Wildcard Filespec Summary

*.X	all entries with any name and extension X
.X	implied wildcard equivalent to *.X for some commands
N.*	all entries with name N and any extension
N	implied wildcard equivalent to N.* for some commands
N.	all entries with name N and no extension
.	current directory
..	parent directory
...	current directory tree (this and all lower subdirectories)
...N	N anywhere in the current directory tree
D\...	directory tree starting at D (D and all lower subdirectories)
D...	an abbreviation for "D\. . ."
..X	illegal (same as DOS)
..\X	X in parent directory (same as DOS)
..\...X	X anywhere in directory tree starting at the parent

Some Subtleties About ". . ."

Unless you're really serious about all the details, you should just skip this section. We describe here some subtleties which we expect would only arise in unusual situations, such as computer programs which generate wildcard filenames. Such programs may benefit from knowing the details in this section.

Because DOS allows almost any character in a filename, we had to struggle a bit to come up with the syntax for ". . .". This leads to some weird looking cases, namely, four dots in a row, and five dots in a row. If you think about it, four dots can only be interpreted as dot (the current directory) followed by 3 dots (meaning it and all its subdirectories). This is equivalent to 3 dots, so it isn't very useful. But it is consistent, so we allow it.

More useful is 5 dots, which means "the parent and all its subdirectories", so it includes siblings of the current directory.

Weird Cases That We Don't Recommend

....X	X in current or any subdirectory (same as ".\. . . X")
.....X	X in parent or any of its subdirectories (same as ". .\. . . X")

Destination Filespecs

Some commands require 2 filespecs: a "source" filespec and a "destination" filespec. For example, ICOPY reads source files, and creates destination files. The source filespec is any wildcard-filespec and optional file selection modifiers (described later). A destination-filespec can be as simple as a filename, or it may contain wildcards. The destination-filespec

wildcards are "?" and "*", just as in DOS. A question mark in a destination-filespec is replaced by the character in the same position of the source filespec. For example,

```
ICOPY ABC.DEF ?X?.Y?Z          destination file is AXC.YEZ.
```

Unlike wildcard-filespecs, a "*" wildcard in a destination must be the last character in the filename (or in the extension). The "*" character in the destination filespec will be replaced by all the remaining characters of the name (or extension) of the source file. For example,

```
MOVE NAME.EXT X*.*             destination file is XAME.EXT.
```

The destination filespec may not include any directory wildcards. However, it may specify a directory path.

If the destination filename is omitted, it defaults to "*.*". This means that if you specify a destination directory and omit the filename, it is equivalent to specifying *.* for the filename.

When the source filespec includes subdirectory wildcards (i.e. uses ". . ." , "*", or "?"), the destination directory path will replicate the source directory tree. This is as you would expect, for example, for copying directory trees. You can override this feature with the /FLATTEN modifier. With /FLATTEN, the destination directory is the single directory you specified (thus "flattening" the source directory structure).

Command Modifiers

Most commands accept modifiers which alter their operation in some way. These command modifiers are used to select specialized features of the commands. There are 2 kinds of modifiers: **keyword modifiers** (/BEFORE, /SINCE, /ON, etc.) and **attribute modifiers** (+ADVSHR, -ADVSHR).

Some modifiers (+ADVSHR, -ADVSHR, /EXCLUDE, /BEFORE, /SINCE, /ON, /BIGGER, and /SMALLER) are used with wildcard filespecs to select files. Therefore, any command which accepts a wildcard-filespec also accepts these modifiers. When you specify more than one of these file selection modifiers, a file must match **all** of the given modifiers to be selected. These modifiers are described in detail in the following sections.

Other modifiers are specific to each command, and are described with the commands in the Utility Reference.

Keyword Modifiers

The keyword modifiers are words that invoke a specific feature of the command. Keyword modifiers always start with a "/", and may be given in any order. They may always be abbreviated to the minimum length required to uniquely distinguish them from other keyword modifiers in the same command. Because keyword modifiers begin with a "/", they do not need to be separated by spaces. For example, the following command is valid:

```
DI/BYTIME *.BAK/NOPAGE/SINCE:8:00
                          list all files with .BAK
                          extensions, that were last modified
                          after 8:00 am this morning, sorted
                          by time, with no screen pause
```

Notice that some keyword modifiers accept parameters after them (e.g., /ON takes a date, /EXCLUDE takes a filename). We use a colon ":" or an "=" to separate the modifier from its parameter. You might think other characters would be more natural than a ":", and you'd be right. However, DOS changes some characters on the command line before passing them to batch files (in particular, DOS changes "=" into a space). The colon is a safe character which DOS doesn't mess with, so we use it. But the "=" is more natural, so we allow it also. For example,

`DI /SINCE=MONDAY` is equivalent to `DI /SINCE:MONDAY`

Note also that when the colon isn't needed to clearly separate letters, you can omit it, as shown below in the modifier descriptions.

Exclude Files Modifier (/EXCLUDE)

In commands where you specify a wildcard-filespec, you can also exclude a file or set of files from being selected. The syntax of the /EXCLUDE modifier is

    /EXCLUDE:wildcard-namespec

Note that a wildcard-**name**spec is not the same as a wildcard-**file**spec. In the wildcard-namespec, you can specify a name, but not a directory path. The name can include any of the InCommand wildcards, and "*" wildcards are **always** implied if only a name or only an extension is given. As with wildcard-filespecs, you can give multiple wildcard patterns separated by commas. This will exclude any file which matches **any** of the patterns.

For example, to obtain a listing of all the entries in the current directory, except those with .OBJ or .EXE extensions, type

    DI /EXCLUDE:.OBJ,.EXE   (same as DI /EXCLUDE:*.OBJ,*.EXE)

To copy all files from the current directory to the directory NEWDIR\ except the files HERE.*, type

    ICOPY *.* NEWDIR\ /EX:HERE

Date/Time Syntax (/BEFORE, /SINCE, /ON)

InCommand allows you to select files based on their modification times. You can enter dates and times in either of two formats: full date/time, or day-of-week. The full date/time syntax is this:

    [:]MM-DD[-YY]:HH:MM[:SS]

You must use 24-hour time. Either the date, or time, or both may be omitted. The default date is the current date. You may specify the month and day, and omit the year. The default time is 0:00:00 (the start of the given day). You may specify hours and minutes, and optionally, seconds.

There are 3 file selection modifiers that use the date/time syntax: /BEFORE, /SINCE, and /ON. The /BEFORE modifier selects files that were modified before the given date/time. The /SINCE modifier selects files that were modified on or after a given

date/time. The /ON modifier selects files that were modified in a 24 hour period starting at the given date/time. The /ON modifier is typically used to select files which were modified on a given day (in which case you specify the day and omit the time). However, the /ON modifier does not restrict you to a 24 hour period beginning at midnight.

Examples:

/SINCE	files modified today
/SINCE9-23 or	
/SINCE:9-23	files modified on or after September 23 of this year at 0:00:00
/BEFORE:9-5-89	files modified before September 5, 1989 at 0:00:00
/ON9-5	files modified on September 5 of this year
/ON:10-18:11:30	files modified during the 24 hour period starting on October 18 of this year at 11:30 AM
/BEFORE16:10 or	
/BEFORE:16:10	files modified before 4:10 PM today
/BE:1-1-90 /SI:12-1-89	files modified in December, 1989.

Notice that you can specify both /BEFORE and /SINCE to select a time interval. Also, note that DOS records file times in 2 second increments. If you specify an odd number of seconds, it is truncated to the next lower multiple of 2 seconds. For example, 10:00:03 is equivalent to 10:00:02.

Alternatively, you can use days of the week (Monday, Tuesday, etc.) instead of a date (Month-Day-Year):

:DAY[:]HH:MM[:SS]

Again, you must use 24 hour time. The day is one of the following (and may be abbreviated to that shown in capitals):

Monday, TUesday, Wednesday, THursday, Friday, SAturday, or SUnday. If only a day is given, the time defaults to 00:00:00.

Examples:

/SINCE:Tuesday	files modified since last Tuesday
/BEFORE:Sun:9:00	files modified before last Sunday at 9:00 am
/ON:M	files modified on Monday
/ON:Th:10:00	files modified during the 24 hour period starting at 10:00am last Thursday

Note that if today is Monday, then **MONDAY** refers to 7 days ago. You can specify "today" by omitting the date completely (and optionally specifying a time). For example,

/BEFORE	files modified before 0:00:00 today

File Size Modifiers (/BIGGER, /SMALLER)

InCommand also allows you to select files based on their size. There are two modifiers that you use to select files by size: /BIGGER and /SMALLER. /BIGGER is used to select files that are bigger than or equal to the given size. /SMALLER is used to select files that are smaller than or equal to the given size. The syntax for these modifiers is shown below:

/BIGGER[:]size and **/SMALLER[:]**size
The file size is in bytes.

Examples:

/BIGGER:100	files bigger than or equal to 100 bytes
/SMALLER:100000	files smaller than or equal to 100,000 bytes

/BI:30 /SM:40	files bigger than or equal to 30 bytes and less than or equal to 40 bytes

Notice that you can use /BIGGER and /SMALLER together to select a size interval.

Attribute Modifiers

Attribute modifiers are used to select files (or other directory entries) based on their attributes. Attribute modifiers start with "+" or "-", and specify some combination of the 6 attributes:

A=needs-archiving	S=system
D=subdirectory	H=hidden
V=volume-label	R=read-only

The attributes may be given in any order (e.g., +SH is the same as +HS). The "+" modifier tells InCommand utilities to select only those files which have at least one of the given attributes. The "-" modifier says to exclude files which have any of the given attributes. As an example,

```
DI +D
```

displays only the subdirectory entries of the current directory.

Both "+" and "-" modifiers can be used together in a single command, but they cannot conflict. For example,

```
DI *.* +SH -R
```

displays all files with either the System or Hidden attributes, but not the Read-only attribute.

Unlike keyword modifiers, **attribute modifiers must be preceded by a space** (because the "+" and "-" characters are valid in other contexts). Also, because DOS allows the "-" character in a filename, filenames which begin with the "-" character could be confused with the "-" attribute modifier. We resolve this ambiguity the same way DOS does (though this is not documented for DOS). When you enter a filespec which starts with a "-", you must include an extension (i.e., it must have a "." in it). Without an extension, it will be interpreted as an attribute modifier, and probably generate an error. If you want no file extension, you must end the filename with the "." character. If

you want a wildcard file extension, then you must type it explicitly as "filename.*". For example,

DI AB	display files named "AB" with any extension (same as AB.*)
DI -B	is illegal because "B" is not a valid attribute
DI -B.	display files named "-B" with no extension
DI -B.*	display files named "-B" with any extension

System and Hidden Files

Advanced users will occasionally need to work with System and Hidden files (files with either the System or Hidden attributes). Because it may be dangerous to include these files in some operations, some InCommand utilities will not select them by default (just like some DOS utilities). Usually, InCommand utilities which modify the selected files will not select System or Hidden files by default. To operate on those files, you must explicitly use the +S or +H (or +SH) attribute modifier. Note that specifying this modifier will then select only files with at least one of the given attributes; other (normal) files will not be selected. This means that for those InCommand utilities that do not select System or Hidden files by default, there is no way to select normal and System or Hidden files in a single command.

If you want to operate on both kinds of files, you must issue 2 commands: one with a +SH modifier, and one without it. For example, to delete all files in a directory, including System and Hidden files, type

```
IDEL *.*

IDEL *.* +SH
```

The utilities which select System and Hidden files by default are DI and CHATT. All other commands do not select System and Hidden files by default .

Exit Codes

Many of the InCommand utilities are very useful in batch files. We have designed all utilities to return one of the following general purpose exit codes:

0	Normal	Successful execution, nothing of note occurred
1	Information	Successful execution, with additional information given to user
2	Warning	Successful execution, but may have produced unexpected result
3	Error	An error occurred, though some processing may have been successful
4	Fatal Error	Syntax error, or command failed. Nothing done

After command execution in a batch file, the exit code may be tested with the DOS **IF [NOT] ERRORLEVEL** commands, so that batch files may be programmed to respond to exit codes. For example, a batch file can test if an ICOPY is successful with this:

```
ICOPY *.EXE SOMEWHER\
IF ERRORLEVEL 3 ECHO The copy failed
```

UTILITY REFERENCE

CHATT [+ASHR] [-ASHR] wildcard-filespec

Changes file attributes

Modifiers:

+ASHR	Select only files with any of the given attributes. System and Hidden files are selected by default See text for an important distinction between this modifier and the attributes to be changed.
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/NOLIST	Do not list filenames and their new attributes

Exit codes:

0	Normal	All specified files' attributes changed
2	Warning	No files match specifications
3	Error	One or more files changed attributes, and one or more files attempted could not be changed

4	Fatal Error	Syntax error, or no specified files could have attributes changed

Files can be given any of the 4 file attributes:

A Needs archiving.	Usually used to indicate file has changed since last backed up.
S System file.	Used for operating system boot files. System attribute implies Hidden.
H Hidden.	Used for operating system files. DOS utilities do not find them in wildcard searches, and they are not listed by the DOS DIR command. Some InCommand utilities do not process System or Hidden files by default, but DI will list them.
R Read-only.	File may not be written to or deleted. Use this to protect files from accidental destruction.

CHATT has a somewhat unusual syntax because it needs to specify file attributes for 2 different things: first, you tell CHATT which attributes to add or remove from the specified files; second, you have the normal file selection attribute modifiers. Notice that you specify the attributes you want added or deleted **before** the wildcard-filespec, as in the DOS ATTRIB command. The attribute modifiers **after** the wildcard-filespec select which files will have their attributes modified (as in all InCommand file selections). They do not specify how a file's attributes will be changed.

By default CHATT lists the files and their new attributes when they are changed. If you do not want the files to be listed when their attributes are changed then specify **/NOLIST**.

Notice that System and Hidden files are selected by default. If you do not want to process them, you must explicitly exclude them with the -SH modifier **after** the wildcard-filespec.

Files may not be given the +D (directory) or +V (volume label) attributes, nor may these attributes be removed from directories or volume labels.

Examples:

To set the Archive attribute for all files in the current directory which were modified today, type

```
CHATT +A *.*/SINCE
```

To set the Read-only attribute for all of the .C and .DOC files in the current directory that were last modified before January 1, 1991, type

```
CHATT +R *.C,*.DOC /BE:1-1-91
```

To set the System attribute on all files in the current directory tree which have the Hidden attribute, type

```
CHATT +S ...*.* +H
```

To set the Archive attribute for all files in the current directory except files with the .OBJ and .EXE extensions, type

```
CHATT +A *.* /EXCLUDE:.EXE,.OBJ
```

To set the Archive attribute for all files in the current directory with two character file names and A as the second character, and any extension, type

```
CHATT +A ?A.*
```

DAYTIME

Prints the current day of the week, date, and time, or timestamps a message

Modifiers:

None

Exit codes:

0 Normal

When typed at the keyboard, DAYTIME simply prints the day of the week, date, and time.

DAYTIME has another feature intended for advanced users. DAYTIME behaves slightly differently when you pipe to it or redirect input to it (i.e., stdin is not the keyboard). In this case, DAYTIME reads a line from stdin, and prints one line, consisting of the day of the week, date, and time followed by the line read from stdin.

Examples:

DAYTIME is most useful for timestamping things from a batch file. For example, to automatically record every time that you run a batch file, put this line in the file:

```
DAYTIME >>BATRECRD.DAT
```

This will append a line containing the current date and time to the file BATRECRD.DAT.

To record when the batch file was run, and its first parameter for that run, include this line in the file:

```
ECHO %1 | DAYTIME >>BATRECRD.DAT
```

ECHO will pipe in the first batch file parameter to DAYTIME. DAYTIME will concatenate this string to the current date and time, and append this line to BATRECRD.DAT.

DHELP [topic]

Interactive help for DOS

Modifiers:

None

Exit codes:

0	Normal	Successfully executed
4	Fatal Error	Utilities not installed properly or not enough memory

DHELP is a quick reference for DOS commands. It is not a full tutorial, and is most helpful for users familiar with DOS basics. Currently, DHELP applies to DOS version 4.01. Note that DHELP includes several clarifications and corrections to the standard DOS documentation. DHELP for other versions of DOS will be released soon.

DHELP is a fast, self-explanatory, interactive help utility for DOS. DHELP saves the entire screen on entry, and restores it on exit, so that you can pick up right where you left off after getting help. For more information, just type

DHELP

and follow the directions that appear.

If you know what topic you want help on, you can jump right to that topic by including it on the command line with DHELP. For example,

DHELP DIR display help on the DIR command

If the topic you want help on has spaces in it, then enclose the topic in double quotes. For example,

DHELP "MODE PRINT" display help on MODE PRINTER commands

A prompt bar at the bottom of the screen gives you context sensitive help by listing the valid keystrokes at all times. Use the <up>, <down>, <PgUp>, <PgDn>, <End> and <Home> keys to move around in the help text. To exit DHELP, press <ESC>, Q, or q.

Valid keystrokes in the DHELP text window:

<down>	Scroll to next line
<up>	Scroll to previous line
<PgUp>	Page up a screenful
<PgDn>	Page down a screenful
<End>	Display end of file
<Home>	Display beginning of file

From the DHELP text window, press <F1> to see all available help topics. The Topic window will appear at the bottom of the screen. In the Topic window, you can type the name of the topic that you want help on, or use the <up>, <down>, <left>, <right>, <End> and <Home> keys to move between topics. The currently selected topic will be highlighted in the topic window. Also, the DHELP text window will display the text for the selected topic. To select the current topic, press <ENTER>. The Topic window will disappear.

In the Topic window, you can use <Backspace> to delete the last character typed. To clear the entire topic, press <ESC>. To exit the topic window without selecting a new topic press <ESC> again.

Valid keystrokes in the Topic window:

<down>	Select next topic
<up>	Select previous topic
<left>	Select topic 1 column to the left
<right>	Select topic 1 column to the right
<End>	Select first topic
<Home>	Select last topic
<Backspace>	Delete previous character
<ESC>	Clear current topic, or close topic window

Examples:

To get help on using DHELP, type

```
DHELP
```

To get help on the DOS FORMAT command, type

```
DHELP FORMAT
```

DI

DI [wildcard-filespec]

Makes a directory listing

Modifiers:

+ADVSHR	Select only entries with any of the given attributes. System and Hidden files are selected by default
-ADVSHR	Do not select entries with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/ABSOLUTE	List only the fully qualified filespec for each entry (see /RELATIVE)
/BYEXTENSION	Sort directory by extension first, then name
/BYSIZE	Sort directory by file size
/BYTIME	Sort directory by modification date and time
/EXIST	Test for existence of files

/NOPAGE	DON'T pause between screens on a CRT
/RELATIVE	List only the relative filespec for each entry (see /ABSOLUTE)
/SUBTOTAL	Display the file and directory counts individually for each subdirectory, and the /TOTAL information
/TOTAL	Display only the total file and directory counts and total size
/WIDE	List only the filenames, formatted in 5 columns (like DOS DIR/W)

Exit codes:

0	Normal	One or more files listed
1	Information	No entries matched given specifiers
4	Fatal Error	Syntax error, or invalid file specification

You may specify any combination of the 6 attributes (in any order) with the "+" or "-" attribute modifiers.

The DI command lists each directory entry with its size, modification (or creation) date, and attributes. Because we don't believe in hiding things from intelligent users, all hidden, system, and volume entries are listed. In fact, all entries of any kind (that meet the user given specifiers) will be shown.

By default, the directory entries are sorted alphabetically by name. You may change the sort key with the **/BYEXTENSION, /BYTIME,** and **/BYSIZE** modifiers.

DI puts a summary of the number of directories, files, and total file size at the end of the directory list. After the total size of all files, DI puts the **allocated** size in parentheses. The allocated size includes rounding up all file sizes to the next full cluster, the 32 bytes per directory entry for each file, and the cluster rounding of the directories themselves. The idea is to tell you how much real disk space the given files consume. Because of complex system details, the allocated bytes listed cannot always be exact, but they are very close. When you specify a network drive directly (i.e., without a virtual drive), DI cannot know the cluster size of the network disk, so it uses a

cluster size of 1 byte, effectively eliminating any cluster rounding from the allocated size total.

The **/TOTAL** modifier displays only the total number of directories, files, and bytes of the selected files. The **/SUBTOTAL** modifier displays these values for each subdirectory selected, and then the TOTAL information.

By default, DI pauses after each screenful of display. You can omit the screen pausing by specifying **/NOPAGE.** If the output is redirected to a file (or any non-CRT), there is no pausing.

You can exclude a file or set of files from being listed by specifying **/EXCLUDE:**wildcard-filename**.**

You can get a list of entries without blank lines or intervening directory lines by specifying **/ABSOLUTE** or **/RELATIVE.** With either option, DI lists each entry with its directory path, either **absolute** (fully qualified) or **relative,** and no other information. These options are useful for generating file lists for input to file processing programs. Note that usually you will want to specify **-D,** to omit the directory entries themselves, and get only the file names.

An **absolute** filespec (sometimes called a "fully qualified" filespec) is one that starts with a drive letter, and the root directory, and explicitly specifies every subdirectory in the path. A **relative** filespec specifies a path starting from the default directory for a drive. Therefore, a relative filespec includes any drive or directories that you give in the wildcard-filespec. As a result, if you specify a drive and directory from the root, the **/ABSOLUTE** and **/RELATIVE** modifiers are equivalent.

The **/EXIST** modifier is an advanced feature, mostly used in batch files to test for the existence of files or directories. With this modifier, DI produces no screen output, but returns the usual exit codes. You can test the exit code in a batch file with the DOS IF [NOT] ERRORLEVEL command. An exit code of 1 indicates no directory entries were found to match the specifiers. An exit code of 0 indicates a matching directory entry was found. When you use **/EXIST,** DI stops after finding a single matching directory entry. There is no need for it to find more than one entry, so it saves a lot of time on a large search to stop as soon as possible.

Because DOS (non-networked) subdirectories always have the "." and ".." entries in them, /EXIST automatically excludes them from the search, so that it only looks for directory entries over which you have control. /EXIST also excludes volume labels by default, but you can search for them explicitly by specifying the +V attribute modifier.

Examples:

To display all the entries with .OBJ extensions in the current directory, type

```
DI .OBJ
```

To list all the *.TXT and *.DOC files in the subdirectories of the current directory beginning with the letter A, type

```
DI A*\.TXT,.DOC
```

To display all the entries except those with .OBJ or .EXE extensions in the current directory, type

```
DI /EXCLUDE:.OBJ,.EXE
```

To display all files in the current directory modified last Sunday, type

```
DI /ON:SUNDAY
```

To display all files on the entire disk which are 1,000,000 bytes (1 megabyte) or larger, type

```
DI ... /BI:1000000
```

To display all the subdirectories in a directory named PARENT, type

```
DI PARENT\ +D
```

To see all the files needing archiving in the current directory tree, type

```
DI ... +A
```

To see only the total size of all those files, without listing each individually, type

```
DI ... +A /T
```

To make a list of the absolute filespecs of all the files in the current directory tree, and omit subdirectory names themselves, type

```
DI ... /ABS -D
```

To see the total number of files, and the total size of the files in each subdirectory of the current tree, type

```
DI ... /SUB
```

To see all of the files on the current disk (starting from the root directory) that have "2" as the second character in the filename, type

```
DI \...?2*.*
```

EXECUTE user-command [user-parameters] [wildcard-filespec enclosed in square brackets]
up to 5 destination-filespecs, enclosed in square brackets ("[ ]")

Allows any command, program, or batch file to use InCommand wildcard-filespecs

Modifiers:

+ASHR	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/BYEXTENSION	Execute files alphabetically by extension first, then name
/BYSIZE	Execute files in order of file size
/BYTIME	Execute files in order of modification date and time

/CREATE	Create destination directories if they do not already exist
/FLATTEN	Put all destination files in a single directory
/IFEXISTS	File is selected only if the first destination file already exists
/IFMISSING	File is selected only if the first destination file does not already exist
/IFNEWER	File is selected if the source file time is newer than the first destination file time, or if the first destination file does not exist
/IFOLDER	File is selected if the source file time is older than the first destination file time, or if the first destination file does not exist
/INDIVIDUALLY	Asks for confirmation before executing user-command on each file
/NOLIST	Do not list commands as they are executed

Exit codes:

0	Normal	Command executed with all specified files, though some may have errors
2	Warning	No files match specifications
3	Error	One or more commands executed, and one or more commands attempted could not be executed
4	Fatal Error	Syntax error, or no commands could be executed

EXECUTE is an advanced utility that allows any command, program, or batch file to be run as if it had full InCommand wildcard-filespecs and file selection modifiers. EXECUTE achieves this effect by executing user-command once for each file it finds matching your specifications. For each execution of user-command, EXECUTE inserts the actual filename in place of the wildcard-filespec. In this way, user-command never sees any wildcards; it only sees actual filenames that EXECUTE has found.

Besides a source filespec, you may give up to 5 destination-filespecs. These filespecs are formed according to the standard rules for destination-filespecs (see p. 20). Notice that you must put square brackets around wildcard-filespec and any destination-filespecs. The square brackets ("[]") tell EXECUTE where in the command line to find wildcard-filespec and destination-filespecs. Anything following user-command which is not inside square brackets is assumed to be parameters for user-command, and EXECUTE does not interpret it in any way; it simply passes it on to the user-command. You may freely intersperse user-command parameters anywhere between the filespecs that follow user-command.

Because of EXECUTE's wide capabilities, it accepts much more freedom of placement of its parameters than the above syntax summary can show. The wildcard-filespec, and also the first destination-filespec, may optionally precede user-command (still enclosed in square brackets). If they do, then the filenames they represent do not appear in the command actually executed, but destination-filespecs after user-command will be substituted normally. This allows you to use the **/IFxxx** modifiers to select files, without requiring you to have those particular filenames used in user-command. If you use any **/IFxxx** modifier, you must give at least one destination-filespec.

Note that the wildcard-filespec is required, but destination-filespecs are optional. Because any modifiers following user-command belong to user-command, **all the InCommand modifiers must come before** user-command. The first parameter which EXECUTE does not recognize is assumed to be user-command.

If user-command requires square brackets for some options, which might conflict with EXECUTE's use of square brackets for filespecs, try enclosing the user-command parameters in double-quotes. EXECUTE will ignore square brackets inside quotes, but the quotes will be passed on to user-command. For example, if you have a file named FIRST.C in the current directory, then the command

```
EXECUTE USERCOM "[USER-STUFF]" [*.C]
```

will execute the following command:

```
USERCOM "[USER-STUFF]" FIRST.C
```

If user-command requires square brackets and cannot accept the double-quotes around them, you can write a batch file which invokes user-command with the square brackets inside the batch

file. There will then be no square brackets on the user command line, and hence no conflict with EXECUTE. For example, if in the above example, USERCOM would not allow the double-quotes around "[USER-STUFF]", you could write a batch file named USERBAT.BAT, containing this line:

```
USERCOM [USER-STUFF] %1
```

And then you could use EXECUTE to run the batch file:

```
EXECUTE USERBAT [*.C]
```

EXECUTE processes files in alphabetical order by default. You can select a different order with the **/BYxxx** modifiers.

You can use the /INDIVIDUALLY option to pick and choose among the set of given files. EXECUTE will ask you to confirm each file before executing user-command. You may answer

N or **<ENTER>**	to not execute user-command on the file
Y	to execute user-command on the file
Q	to quit executing
A	to execute user-command on this and all subsequent files without confirmation

Examples:

To use EXECUTE to copy a number of files with the DOS COPY command, you could type

```
EXECUTE COPY/V [*.OBJ] [*.XYZ]
```

If you had files named A.OBJ, B.OBJ, and C.OBJ, this command is equivalent to typing

```
COPY/V A.OBJ A.XYZ
COPY/V B.OBJ B.XYZ
COPY/V C.OBJ C.XYZ
```

Notice that in either case, the DOS COPY command does not see any wildcards in either the source or destination filespecs. Notice also that the /V is a modifier to the DOS COPY command, and is not interpreted by EXECUTE.

To find out which files in a directory tree (DIR1. . .) do not exist in a parallel tree (DIR2. . .), type

```
EXECUTE/IFMISSING [DIR1\...*.*] REM [DIR2\*.*]
```

In this case, user-command is a REMark, which does nothing but list the filename missing from DIR2\.

To find out which files in a directory tree (DIR1. . .) are newer than their counterparts in a parallel tree (DIR2. . .), type

```
EXECUTE/IFNEWER [DIR1\...*.*] REM [DIR2\*.*]
```

In this case, user-command is a REMark, which does nothing but list the filename missing from DIR2\.

Suppose you have 2 directory trees which should be identical. To use the DOS FC command to compare all the .C and .ASM files in the 2 trees, type

```
EXECUTE FC [DIR1\...*.C,*.ASM] [DIR2\]
```

Notice that if DIR2\ has *.C or *.ASM files **not** in DIR1\, they will not be selected at all.

Assume you have a batch file named MYBATCH.BAT that processes its first parameter as an input file, and its second parameter as an output file. You can invoke that batch file on all the *.IN files in the current directory tree created since Monday into *.OUT files in the same directories, by typing:

```
EXECUTE/SI:MON MYBATCH [...*.IN] [*.OUT]
```

Alternatively, you could invoke the batch file on all *.IN files which are newer than their corresponding *.OUT files, by typing

```
EXECUTE/IFNEWER MYBATCH [...*.IN] [*.OUT]
```

To assemble all *.ASM files created since this morning into *.OBJ and *.LIS files, type

```
EXECUTE/SINCE ASM [*.ASM] [*.OBJ] [*.LIS]
```

ICOPY

ICOPY wildcard-filespec [destination-filespec]

Copies files, with better verification than DOS

Modifiers:

+ASHR	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/ATTRIBUTES	Copy source file attributes to destination file
/BACKUP[:namespec]	If the destination file exists, it is renamed to a backup name before the source file is copied
/BYEXTENSION	Copy files alphabetically by extension first, then name
/BYSIZE	Copy files in order of file size

/BYTIME	Copy files in order of modification date and time
/CREATE	Create destination directory if it does not already exist
/FLATTEN	Put all destination files in a single directory
/IFEXISTS	File is copied only if the destination file already exists
/IFMISSING	File is copied only if the destination file does not already exist
/IFNEWER	File is copied if the source file time is newer than the destination file time, or if the destination file does not exist
/IFOLDER	File is copied if the source file time is older than the destination file time, or if the destination file does not exist
/MODIFIED	Copies files with the +A attribute, and resets the attribute on the source files. /M is compatible with the DOS XCOPY/M command
/NOLIST	Do not list filenames as they are copied
/NOVERIFY	Do not verify destination against source after copying
/OVERWRITE	Overwrite destination files without prompting.
/SUBDIRS	Copy all subdirectories below the given source. Used only for compatibility with DOS XCOPY/S. Equivalent to the ". . ." wildcard
/TOUCH	Set the file modification times of the destination files to the current time
/VERIFY	After copying, both files are read back and verified identical. This is the default, but is allowed for compatibility with DOS

Exit codes:

0	Normal	All specified files copied normally
2	Warning	No files match specifications
3	Error	One or more files copied, and one or more files attempted could not be copied
4	Fatal Error	Syntax error, or no specified files could be copied

The source files are copied to the destination files. If the source filespec is an explicit directory (i.e., it ends in either ":" or "\"), then all the files in that directory are copied. In other words, the filename of "*.*" is implied if you omit the filename. If the source filespec is a directory, but you did not explicitly specify it as such, ICOPY will give the error "File not found". This function is consistent with the InCommand filespec syntax, and prevents inadvertently copying a whole directory .

The destination-filespec may not include any directory wildcards, but it may specify a directory path. Refer to the Syntax Reference (page 20) for a complete description of destination-filespec. If the destination-filespec is a directory, then each source file will be copied to a separate destination file of the same name in the given directory.

If you specify a destination directory that does not already exist, ICOPY will, by default, ask you if you want the directory to be created. You must put a backslash after it to tell ICOPY that it is a directory and not a file. For example,

```
ICOPY *.* X\
```

copies all the files in the current directory into a subdirectory named X. If you use the /CREATE modifier, ICOPY will create the destination directory (if it does not already exist) without prompting.

If you specify a single destination file, then all of the source files will be appended (concatenated) into the single destination file. For example,

```
ICOPY *.* D\X
```

will create a file in the subdirectory D, named X, and concatenate all the files in the current directory into it.

If the destination file already exists, and you have specified no /IFxxx or /OVERWRITE modifiers, then ICOPY asks you what to do. You have 4 choices:

O	Overwrite the destination file
S	Skip this file
A	overwrite this and All subsequent files, with no more prompting
Q	Quit

Unless you specify **/NOVERIFY**, both the source and destination files are read back and verified identical. This verify will catch any read error of the source file or write error of the destination file. This is expressly better than the DOS COPY/V command, which only reads back the destination file to verify that the disk medium is not defective, and XCOPY/V, which does not re-read the source file to protect against read errors. It is also better than the DOS VERIFY command, which does the same thing for all disk writes that the DOS COPY /V option does. You can save time by specifying VERIFY OFF to DOS when using ICOPY, and be confident that your copy succeeded.

When you specify directory wildcards ("?", "*", or ". . ."), ICOPY, by default, creates destination subdirectories to replicate the source directory structure. This allows you to copy entire directory trees with a single command. ICOPY creates new directories with the current date and time. Alternatively, you can use the **/FLATTEN** modifier to put all the destination files in a single directory (thus flattening the directory structure).

If the destination disk fills up during the copy, ICOPY will tell you, and give you 4 choices:

R	Retry the copy
S	Skip the file
D	go temporarily to a DOS prompt (shell)
Q	Quit

If the destination is a floppy, you can replace the diskette, and Retry the copy. This feature allows you to copy a set of files to a set of floppy diskettes, using as many diskettes as it takes to hold them. If you decide you don't want to copy the file that wouldn't fit, you can Skip it, and ICOPY will continue with the rest of the files. Or, you can temporarily return to the DOS prompt, where you can format a new diskette, delete junk files (even on the destination disk), etc. When you're done with DOS, enter the DOS EXIT command, and you will return to ICOPY, where you can again choose to Retry the copy, Skip the file, go to DOS prompt, or Quit.

The /IFNEWER modifier allows you to copy files only if the source file is newer than the destination file, or the destination file does not exist. This is useful for merging separate directories into a single, up-to-date directory. The /IFEXISTS modifier copies the file only if the destination already exists, which is useful for restoring files from backups that merged several sources. /IFNEWER and /IFEXISTS may be used together to copy files only if the destination already exists, and the source file is newer than the destination file.

The /IFMISSING modifier allows you to copy files only if the destination file doesn't already exist. This is useful for partial restoring of directories.

The /IFOLDER modifier is like /IFNEWER, except the source file is copied only if it is older than the destination, or if the destination does not exist. This is useful for restoring a directory to an older state. /IFOLDER may be used together with /IFEXISTS.

/BACKUP allows you to overwrite an existing file, but preserve the original by renaming it to a backup name. The default backup name is "*.BAK". For example, if you type

```
ICOPY FILEA.C FILEB.C /BACKUP
```

and if FILEB.C already exists, then it will be renamed to FILEB.BAK, and then a copy of FILEA.C will be made into FILEB.C.

You can use any destination file pattern you choose for the backup filename, by entering a destination-namespec after the /BACKUP. For example, if you want your backup files to be called *.OLD, you can enter

```
ICOPY FILEA.C FILEB.C /BA:*.OLD
```

The wildcard pattern can be any destination-namespec , but it may not include a directory. Backups are always made in the directory in which the original exists. As always with destination-filespecs, if you omit the base name, "*" is implied (e.g., ".BAK" is equivalent to "*.BAK").

Note that if the backup file already exists, it is deleted, and replaced by the current file.

The /MODIFIED modifier is useful for making incremental backups of files, and is identical to the DOS XCOPY/M function. Source files with the Archive attribute (and others selected by the "+" and "-" modifiers) are selected, and after each file is copied (and verified if appropriate), the source file's Archive attribute is cleared. To understand how this performs an incremental backup, recall that DOS sets a file's Archive attribute when the file is created, and any time it is modified. When you first use ICOPY/MODIFIED, all the files are backed up, and all Archive attributes are cleared. Thereafter, only files which have been created or modified since the last backup have the Archive attribute, so only those files will be copied. Thus the backup is fully up to date on all files, with a major time savings by not copying files which are already on the backup. See the Applications section for an example of an incremental backup process (page 88).

The /SUBDIRS modifier exists only for compatibility with the DOS XCOPY/S command. Because XCOPY is one of the few DOS commands which can process subdirectories, many people are used to using the /S modifier. For compatibility, we support the DOS syntax.

By default, files are copied in alphabetical order. This means that if you copy a set of files to a new directory, that directory will be physically sorted on the disk. You can choose to copy files in a different order with the /BYTIME, /BYSIZE, and /BYEXTENSION modifiers. With these, you can create directories sorted any way you like. But see the Applications section for a description of "Physically Sorting a Directory" (page 92).

By default, destination files are given the same date/time as their source files. You use the /TOUCH modifier to set the modification times of the destination files to the current date/time.

By default, ICOPY creates destination files with the DOS default attributes, which is only the Archive attribute. You use the

/ATTRIBUTES modifier to copy not only the file contents, but also the file attributes, of the source file to the destination file. This is useful for things like preserving the need for a file to be backed up (the Archive attribute), or preserving file protection (the Read-only attribute). It allows you to back up and restore a directory exactly as it was, including all file attributes.

When copying from hard disk to floppies, ICOPY is significantly faster than DOS XCOPY. Also, ICOPY works faster with more memory available to it, because it fills all available memory with file data. Therefore, a copy operation will require fewer disk head seeks with more memory.

Examples:

To copy all the files in the current directory to NEWDIR, with verify, type

```
ICOPY *.* NEWDIR\
```

If NEWDIR doesn't exist, ICOPY asks if you want to create it.

To copy all of the *.C and *.ASM files in the current directory to the directory \BACKUP\, type

```
ICOPY *.C,*.ASM \BACKUP\
```

To copy all of the files in the current directory except the .OBJ and .EXE files to the directory \LIBRARY\, and to have ICOPY create that directory without prompting, type

```
ICOPY *.* \LIBRARY\ /EX:.OBJ,.EXE /CREA
```

To copy all of the files in the current directory with "A" as the second character in the filename and an extension of .C into files with the same name but the extension of .SAV, type

```
ICOPY ?A*.C *.SAV
```

To perform an incremental backup, with verify, of the directory tree starting at \ROOT to drive B:, type

```
ICOPY \ROOT... B: /M
```

Assume you have a backup of your hard disk that takes several floppies. To update the files on one of those floppies (in the B: drive) with the current versions from the hard drive, type

```
ICOPY *.* B: /IFEXISTS/IFNEWER
```

This command copies files from the current directory to B: if there is already a copy of the file on B: and if the modification time is newer in the current directory. See the Applications section for more information about hard disk backups (page 88).

To add files from the PART\ directory which do not already exist in the ALL\ directory, type

```
ICOPY PART\ ALL\ /IFMISSING
```

To copy all the files in a tree starting at ROOT\ into a single directory ALL\, type

```
ICOPY ROOT... ALL\ /FLAT
```

Imagine you have the 6 chapters to a book named CHAPTERS\CHAP1 through CHAP6. To append all these files from CHAPTERS\ into a single file in the current directory, type

```
ICOPY CHAPTERS\CHAP? BOOK
```

Note that, unlike DOS copy functions, files are (by default) copied in ascending alphabetical (ASCII) order, so the chapters will be appended in the proper sequence.

To concatenate all the *.DAT files in the order in which they were created, type

```
ICOPY *.DAT DAT.ALL /BYTIME
```

IDEL wildcard-filespec

Deletes files

Modifiers:

+ASH	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/INDIVIDUALLY	Asks for confirmation before deleting each file. You may reply with **<ENTER>, N, Y, Q,** or **A**
/NOCONFIRM	Do not ask for confirmation when deleting with wildcards
/NOLIST	Do not list each file as it is deleted.
/PROMPT	Synonym for /INDIVIDUALLY, compatible with DOS 5

Exit codes:

0	Normal	All specified files deleted successfully
1	Information	User replied "no" to wildcard confirmation
2	Warning	No files match specifications
3	Error	One or more files deleted, and one or more files attempted could not be deleted
4	Fatal Error	Syntax error, or no files could be deleted

IDEL deletes the selected files. By default, System and Hidden files are not selected. You must select them explicitly with the +S or +H attribute modifiers.

If the wildcard-filespec includes any wildcard ("*" or "?"), and you have specified neither /INDIVIDUALLY nor /NOCONFIRM , IDEL will prompt you with the absolute pathname to be sure you really want to delete those files. You may reply Y(es) to delete without confirming, N(o) to abort the command, or I(ndividually) to confirm each file before deleting. Note that once in Individual mode, you may answer A(ll) to delete the remaining files without confirming each one.

You can use the /INDIVIDUALLY option to pick and choose among the set of given files. It will ask you to confirm each file before deleting it. You may answer

N or <ENTER>	to keep the file intact
Y	to delete the file
Q	to quit deleting
A	to delete this and all subsequent files without confirmation

If you always want to be prompted for confirmation before deleting any files, and want to avoid the drudgery of typing "/IN" each time, you can instead make a batch file which contains this one line:

```
IDEL %1 /INDIVIDUALLY
```

Then, using the batch file instead of IDEL, you will always be prompted before deleting the first file. If acceptable, you may reply A(ll) to delete all the files without further prompting.

If you specify /NOCONFIRM the files will be deleted without asking for your confirmation **even** if you use wildcards.

By default, **IDEL** lists each file's directory entry as it is deleted. You may omit this with the /NOLIST modifier. **IDEL** always prints a summary of the files and bytes deleted, including total bytes and allocated bytes. See the DI command for a description of allocated bytes (page 39).

We don't allow the +R modifier on this command, because you're not allowed to delete read-only files. To delete them, you must first remove the read-only attribute. You can use the -R modifier to avoid read-only files, and the errors that trying to delete them would generate.

Examples:

To delete all files in the current directory with one character filenames and any extension, type

```
IDEL ?.*
```

To delete all the *.TMP and *.BAK files in the current directory, type

```
IDEL *.TMP,*.BAK
```

To delete all of the files except the ones named CORRECT.C and RIGHT.C in the current directory, type

```
IDEL *.* /EXCLUDE:CORRECT.C,RIGHT.C
```

To individually ask to delete each .OBJ file in the root directory, type

```
IDEL \*.OBJ /INDIV
```

To delete all the .BAK files in each directory tree below the current directory, but not in the current directory itself, type

```
IDEL *\...*.BAK
```

IHELP [topic]

Interactive help for InCommand

Modifiers:

None

Exit codes:

0	Normal	Successfully executed
4	Fatal Error	Utilities not installed properly or not enough memory

IHELP is a fast, self-explanatory, interactive help utility. IHELP saves the entire screen on entry, and restores it on exit, so that you can pick up right where you left off after getting help. For more information, just type

IHELP

and follow the directions that appear.

If you know what topic you want help on, you can jump right to that topic by including it on the command line with IHELP. For example,

IHELP DI display help on the DI command

A prompt bar at the bottom of the screen gives you context sensitive help by listing the valid keystrokes at all times. Use the <up>, <down>, <PgUp>, <PgDn>, <End> and <Home> keys to move around in the help text. To exit IHELP, press <ESC>, Q, or q.

Valid keystrokes in the IHELP text window:

<down>	Scroll to next line
<up>	Scroll to previous line
<PgUp>	Page up a screenful
<PgDn>	Page down a screenful
<End>	Display end of file

<Home>	Display beginning of file

From the IHELP text window, press <F1> to see all available help topics. The Topic window will appear at the bottom of the screen. In the Topic window, you can type the name of the topic that you want help on or use the <up>, <down>, <left>, <right>, <End> and <Home> keys to move between topics. The currently selected topic will be highlighted in the topic window. Also, the IHELP text window will display the text for the selected topic. To select the current topic, press <ENTER>. The Topic window will disappear.

In the Topic window, you can use <Backspace> to delete the last character typed. To clear the entire topic, press <ESC>. To exit the topic window without selecting a new topic press <ESC> again.

Valid keystrokes in the Topic window:

<down>	Select next topic
<up>	Select previous topic
<left>	Select topic 1 column to the left
<right>	Select topic 1 column to the right
<End>	Select first topic
<Home>	Select last topic
<Backspace>	Delete previous character
<ESC>	Clear current topic, or close topic window

Examples:

To get help on using IHELP, type

```
IHELP
```

To get help on the InCommand ICOPY command, type

```
IHELP ICOPY
```

IRD wildcard-filespec

Removes (deletes) directories

Modifiers:

+ASH	Select only directories with any of the given attributes. System and Hidden directories are not selected by default
-ASH	Do not select directories with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/INDIVIDUALLY	Asks for confirmation before removing each directory. You may reply with **<ENTER>, N, Y, Q,** or **A**
/NOCONFIRM	Do not ask for confirmation when removing with wildcards
/NOLIST	Do not list each directory as it is removed.
/PROMPT	Synonym for /INDIVIDUALLY, compatible with DOS 5

Exit codes:

0	Normal	All specified directories removed successfully
1	Information	User replied "no" to wildcard confirmation
2	Warning	No directories match specifications
3	Error	One or more directories removed, and one or more directories attempted could not be removed
4	Fatal Error	Syntax error, or no directories could be removed

If the wildcard-filespec includes any wildcard ("*" or "?"), and you have specified neither /INDIVIDUALLY nor /NOCONFIRM, IRD will prompt you with the absolute pathname to be sure you really want to remove those directories. You may reply Y(es) to remove without confirming, N(o) to abort the command, or I(ndividually) to confirm each directory before removing. Note that once in Individual mode, you may answer A(ll) to remove the remaining directories without confirming each one.

You can use the /INDIVIDUALLY option to individually pick and choose among the set of given directories. It will ask you to confirm each directory before removing it. You may answer

N or <ENTER>	to keep the directory intact
Y	to remove the directory
Q	to quit removing
A	to remove this and all subsequent directories without confirmation

If you always want to be prompted for confirmation before removing any directories, and want to avoid the drudgery of typing "/IN" each time, you can instead make a batch file which contains this one line:

```
IRD %1 /INDIVIDUALLY
```

Then, using the batch file instead of IRD, you will always be prompted before removing the first directory, and if acceptable, you may reply A(ll) to remove all the directories without further prompting.

If you specify /NOCONFIRM the directories will be deleted without asking for your confirmation **even** if you use wildcards.

By default, IRD lists each directory entry as it is removed. You may omit this with the /NOLIST modifier. IRD always prints a summary of the directories removed.

Note that a directory must be empty of all files and subdirectories to be removed. IRD prints an error if a specified directory cannot be removed. Note also that to remove a (parent) directory and all its subdirectories (a directory tree), you must first remove all the files, then all the subdirectories, and then the parent directory. The InCommand utility WIPEDIR (see page 84) does these operations with one command, and it will prompt you before deleting anything.

Examples:

To remove all directories with one character filenames in the current directory, type

```
IRD ?
```

To remove all the empty subdirectories in the current directory tree, type

```
IRD ...*.*
```

To remove the two subdirectories TEMP and SAVE from the current directory, type

```
IRD TEMP,SAVE
```

To remove the empty subdirectories in the current directory, except those beginning with "A" and those beginning with "B", type

```
IRD *.* /EXCLUDE:A*,B*
```

To individually ask about removing all the empty subdirectories in the current directory, but not below it, type

```
IRD *.* /IND
```

IREN wildcard-filespec new-filename

Renames files

Modifiers:

+ASHR	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/NOLIST	Do not list new filenames as they are renamed

Exit codes:

0	Normal	All specified files renamed normally
2	Warning	No files match specifications
3	Error	One or more files renamed, and one or more files attempted could not be renamed
4	Fatal Error	Syntax error, or no specified files could be renamed

The source files are renamed according to new-filename. Note that new-filename is required. New-filename may contain the usual destination wildcards ("*" and "?"). See the Syntax Reference for a complete description of destination filename formation (page 20). New-filename may not contain any drive or path. Since renamed files remain in their current directories, there is no reason to specify a new path. (If you want to move files to a different directory, use the MOVE utility.)

IREN is similar to the DOS RENAME command, but it allows the full InCommand file selection capabilities. Unlike DOS, IREN won't rename Read-only files (files with the Read-only attribute). You must first remove the Read-only attribute (with CHATT) before renaming them.

If the destination filename is only an extension, the base name defaults to "*". For example, a new-filename of ".XYZ" is equivalent to "*.XYZ".

When a file is renamed, its modification date is unchanged.

Examples:

To rename all the .BAK files in the current directory to .BKP, type

```
IREN *.BAK .BKP
```

To rename all the *.C and *.ASM files to have a .BAK extension, type

```
IREN *.C,*.ASM *.BAK
```

To rename all the files in the current directory, except those named STD*.* or XYZ*.*, to have .OLD extensions, type

```
IREN *.* *.OLD /EX:STD*,XYZ*
```

If the current directory tree contains .C and .H files, then to rename all the .C files to .CX and all the .H files to .HX extensions, type

```
IREN ...*.? .?X
```

Note that this renames all files with one-letter extensions to add an "X" to the extension.

To change the first character of all the .DAT files in the current directory to "A", and leave the 2nd through 8th characters unchanged, type

```
IREN *.DAT A*.DAT
```

MOVE wildcard-filespec [destination-filespec]

Moves files to other directories

Modifiers:

+ASH	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASH	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/BACKUP[:namespec]	If the destination file exists, it is renamed to a backup name before the source file is moved
/BYEXTENSION	Move files alphabetically by extension first, then name
/BYSIZE	Move files in order of file size
/BYTIME	Move files in order of modification date and time

/CREATE	Create destination directories if they do not already exist
/FLATTEN	Put all destination files in a single directory
/NOLIST	Do not list filenames as they are moved
/OVERWRITE	Overwrite destination files without prompting.

Exit codes:

0	Normal	All specified files moved normally
2	Warning	No files match specifications
3	Error	One or more files moved, and one or more files attempted could not be moved
4	Fatal Error	Syntax error, or no specified files could be moved

The source files are removed from their directories, and put into the destination directories and filenames. MOVE does not create new files; it just moves existing ones to new directories and/or names. The files are not copied; only their directory entries are changed. This makes MOVE very fast, no matter how big the files are.

The destination filespec may not include any directory wildcards, but it may specify a directory path. If that path does not exist, MOVE will not, by default, create it. If you want MOVE to create destination directories as it moves files, you must specify the /CREATE modifier.

The destination filename may include wildcards, just like ICOPY. See the Syntax Reference (page 20) for a complete description of destination wildcards.

Because MOVE can only move files within a disk partition, the source and destination drive letters must map to the same disk partition. The drive letters may be different if you use SUBST or network mappings that put them on the same disk partition.

If the destination filename is omitted, it defaults to "*.*". This means that if no destination filespec is given, files will be moved with their current names into the current directory. If the destination filespec is a directory, then each source file

will be moved to a destination file of the same name in the given directory.

If the destination file already exists, and you have not specified /OVERWRITE, then MOVE asks you what to do. You have 4 choices:

O	Overwrite the destination file
S	Skip this file
A	overwrite this and All subsequent files, with no more prompting
Q	Quit.

MOVE will not move files with the Read-only attribute.

When you specify directory wildcards ("?", "*", or ". . ."), MOVE, by default, expects the destination subdirectories to exist which replicate the source directory structure. If the destination directory structure does not exist, you can use the /CREATE modifier to have MOVE create the destination directory structure. This allows you to move entire directory trees with a single command. But note that MOVE does not move directories (although it may create them with the /CREATE modifier). If you MOVE all the files out of a directory, the empty directory is left behind. To actually move a directory (including any files or subdirectories it may have), use the MOVEDIR command.

Alternatively, you can use the /FLATTEN modifier to put all the destination files in a single directory (thus flattening the source directory structure).

When you MOVE a file, its modification date is unchanged.

If the source directory is the same as the destination directory, MOVE will simply rename the files (though the syntax is necessarily different from the IREN or DOS RENAME commands). With IREN or RENAME, you cannot specify a destination directory path, and files are never moved to other directories.

/BACKUP allows you to overwrite an existing file, but preserve the original by renaming it to a backup name. The default backup name is "*.BAK". For example, if you type

```
MOVE FILEA.C FILEB.C /BACKUP
```

and if FILEB.C already exists, then it will be renamed to FILEB.BAK, and then FILEA.C will be MOVEd into FILEB.C.

You can use any destination file pattern you choose for the backup filename, by entering a destination-namespec after the /BACKUP. For example, if you want your backup files to be called *.OLD, you can enter

```
MOVE FILEA.C FILEB.C /BA:*.OLD
```

The wildcard pattern can be any destination-namespec , but it may not include a directory. Backups are always made in the directory in which the original exists. As always with destination-filespecs, if you omit the base name, "*" is implied (e.g., ".BAK" is equivalent to "*.BAK").

Note that if the backup file already exists, it is deleted, and replaced by the current file.

By default, files are moved in alphabetical order. This means that if you move a set of files to a new directory, that directory will be physically sorted on the disk. You can choose to move files in a different order with the **/BYTIME, /BYSIZE,** and **/BYEXTENSION** modifiers. With these, you can create directories sorted any way you like. See the Applications section for a description of "Physically Sorting a Directory" (page 92).

Examples:

To move all the .BAK files in the current directory to the subdirectory BACKUP\, type

```
MOVE *.BAK BACKUP\
```

To move all of the files with 2 character names starting with A, to the subdirectory A\, type

```
MOVE A?.* A\
```

To move all the *.C and *.H files in the A\ directory tree to an existing similar directory tree named B\, type

```
MOVE A\...*.C,*.H B\
```

To move all the files in the current directory, except the .OBJ and .EXE files, to the subdirectory SAVE\, type

```
MOVE *.* SAVE\ /EX:.OBJ,.EXE
```

To move all the .C files in the current directory tree, and create a new, sibling directory tree named NEW\, type

```
MOVE ...*.C ..\NEW\ /CREATE
```

To move all the .OBJ files in the current directory tree to the directory \OBJ\ (i.e., OBJ\ in the root), type

```
MOVE ...*.OBJ \OBJ\ /FLAT
```

To effectively move a directory tree (say, on a network that doesn't support MOVEDIR), type

```
MOVE OLD\...*.* NEW\ /CREATE       this leaves a bare tree
                                behind in OLD\. . .
WIPEDIR OLD
```

MOVEDIR

MOVEDIR wildcard-filespec [destination-filespec]

Renames subdirectories, or moves them to other directories

Modifiers:

+ASH	Select only directories with any of the given attributes. System and Hidden directories are not selected by default
-ASH	Do not select directories with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/CREATE	Create destination directories if they do not already exist
/FLATTEN	Move all directories to a single directory
/NOLIST	Do not list directory names as they are moved

Exit codes:

0	Normal	Directories successfully moved.
1	Information	User replied "no" to wildcard confirmation
2	Warning	No directories match specifications
3	Error	One or more directories moved, and one or more directories attempted could not be moved
4	Fatal Error	Syntax error, or no directories could be moved

MOVEDIR moves a directory (or set of directories), and therefore all its contents including all lower subdirectories, to a new directory and name. If the specified source and destination directories are siblings, then the source directory is simply renamed. As always, you can use wildcards to move many directories with one command.

MOVEDIR is an advanced function, and its operation may seem somewhat confusing at first. Like the MOVE command, no files are copied or deleted. Only directory entries are changed. This means you can move hundreds of megabytes of data instantly, even if you have no free space left on your disk.

The destination-filespec may specify a directory path, and if it does not exist, MOVEDIR will not, by default, create it. If you want MOVEDIR to create destination directories as it moves directories, you must specify the **/CREATE** modifier. The destination name may include wildcards, just like ICOPY. See the Syntax Reference (page 20) for a complete description of destination wildcards.

Warning

> You should not use MOVEDIR in a multitasking environment (e.g. Windows in 386 enhanced mode) to move directories, but you may use it to safely rename them. Because MOVEDIR directly manipulates the disk structure, and most versions of DOS provide no way to lock the disk during these operations, a conflict between MOVEDIR and concurrent tasks could cause corruption of the disk, and a loss of data.

Because MOVEDIR can only move directories within a disk partition, the source and destination drive letters must be the same.

MOVEDIR will not move directories with the Read-only attribute.

When you specify directory wildcards ("?", "*", or ". . .") in the source filespec, MOVEDIR, by default, expects the destination subdirectories to exist which replicate the source directory structure. If the destination directory structure does not exist, you can use the **/CREATE** modifier to have MOVEDIR create the destination directory structure. Alternatively, you can use the **/FLATTEN** modifier to put all the destination files in a

single directory (thus flattening the source directory structure). Note that **/FLATTEN** does not flatten the directories that are moved. It simply moves all the selected directories from an existing tree into a single directory.

MOVEDIR allows a full wildcard-filespec, which can lead to unexpected things if you are not careful. For example, you can use the ". . ." directory wildcard, but if all the first level subdirectories are moved, then there is, by definition, no more directory tree left to move. However, notice that ". . ." may be useful with filespecs other than "*.*".

Because DOS does not directly support moving directories, MOVEDIR chooses one of several methods to effect its function. In some cases (such as network drives), MOVEDIR uses DOS functions to create the destination directory tree, move each file into it, and remove the old tree. In this case, the newly created directories will have the current date/time stamp, and will retain only the standard DOS attributes. Any extended network attributes on any directory in the tree being moved will be lost.

Because DOS does not record directory sizes, the modifiers **/BIGGER** and **/SMALLER** have no meaning to MOVEDIR.

Examples:

If you have a directory path (from the root) of \LEVEL1\DEEP, you can move it to the root directory and name it SHALLOW with this:

```
MOVEDIR \LEVEL1\DEEP \SHALLOW
```

Move DEEP to the root, and change its name to SHALLOW.

If you want to move a directory from one level into another, you can type

```
MOVEDIR \LEVEL1\SHALLOW \LEVEL1\LEVEL2\  /CREATE
```

This moves LEVEL1\SHALLOW to \LEVEL1\LEVEL2\SHALLOW. Notice that in this case the destination is a directory (i.e. it ends in "\"), so the subdirectory SHALLOW was moved into \LEVEL1\LEVEL2\, and kept its name.

SEARCH wildcard-filespec text-string

Searches files for a text string

Modifiers:

+ASHR	Select only files with any of the given attributes
-ASHR	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/CASE	Case sensitive search
/FILES	List only filenames containing a match
/LIST[:window-size]	List a window of lines around each matching line
/LOG[:window-size]	Disables all CRT specific aspects of the output (e.g. highlighting, browsing)
/NONUMBER	In list mode, list lines without line numbers

/WORD	Match whole words (or C identifiers) only, not fragments of words

Exit codes:

0	Normal	One or more files found containing text-string
1	Information	No files found containing text-string
2	Warning	No files matched given specifications
4	Fatal Error	Syntax error

SEARCH scans through the selected files looking for text matching your text-string. Unless you specify /LIST or /LOG, or redirect the output, SEARCH uses "browse mode". When a match is found, you get a full screen view of the lines around the match, and you can freely browse back and forth anywhere through the file. SEARCH highlights the current match on the screen, and displays the line number and file name of the match in the prompt bar at the bottom of the screen.

While browsing, use the <up>, <down>, <left>, <Ctrl-left>, <right>, <PgUp>, <PgDn>, <End> and <Home> keys to move around in the text. The prompt bar at the bottom of the screen lists the valid keystrokes at all times:

<F1>	Help, including current file directory information
N or **<space>**	Find the next occurrence of text-string
P	Find the previous occurrence of text-string (in the current file)
S	Skip to the next file containing text-string
C	Center the current match on the screen
Q or **<ESC>**	Quit SEARCH
<down>	Scroll to next line
<up>	Scroll to previous line
<left>	Shift view 8 columns left
<Ctrl><left>	Return to column 1
<right>	Shift view 8 columns right
<PgUp>	Page up a screenful
<PgDn>	Page down a screenful
<End>	Display end of file
<Home>	Display beginning of file

By default, SEARCH flashes on the screen the name of each file being searched, so you can follow its progress.

If you specify **/LIST** or **/LOG** (or if the output is redirected to a file), SEARCH prints each line that contains a match for text-string. Optionally, you can list several lines surrounding each line matching text-string, by entering the "window size" with the **/LIST** or **/LOG** modifier. For example, to see a window of 5 lines (2 above and 2 below) around each matching line, you could use **/LIST:5**. Note that you can give a window size of zero, and get a

list of filenames and the total number of occurrences of text-string.

The /LOG modifier additionally removes any CRT specific formatting of SEARCH output. SEARCH will **not** flash the file names on the screen as the files are searched, and will **not** highlight the text-string in the output lines. Though SEARCH automatically recognizes when its output is redirected to a file, you can use the /LOG modifier for applications where the output is not redirected, but CRT specific output is not appropriate.

If your text-string contains any spaces, commas, or double quotation marks, you must use a special syntax for text-string. First, enclose the text-string in double quotations. Then you can include any comma or space between the double quotes. To include a double quote in such a text-string, enter a backslash before the double quote. Any backslash interprets the next character literally, so to include a backslash in a text-string that is enclosed in quotes, enter 2 backslashes. For example:

```
SEARCH * "a b,c"          search for the string "a b,c"
SEARCH "\"abc\""          search for the string "abc"
                          enclosed in quotes
```

In list mode, SEARCH displays lines with their line numbers. Use /NONUMBER to display lines without line numbers.

/FILES lists only the filenames containing text matches. This is faster than list mode, because the search stops in each file after the first match.

By default, searches are case blind; that is, SEARCH treats upper- and lower- as the same. If searching for "abc", then "ABC" will also match. /CASE specifies a case sensitive search. Text must exactly match the case you typed in text-string.

/WORD specifies that the text in a file must begin and end at a word boundary to be a match. In other words, the text must be preceded and followed by non-word characters. For example, without /WORD, if you search for "and", then it would be found in the word "command" (the last 3 letters). With /WORD, SEARCH would only find the word "and" as a whole word, and not embedded within another word. Programmers can use this to find C language identifiers, because "word" characters are all the letters, digits, and the underscore ("_").

Examples:

To find all occurrences of the string "XYZZY", in any combination of upper- or lower-case, in your *.C and *.H files, type

```
SEARCH .C,.H XYZZY
```

To find only the occurrences in all capitals, type

```
SEARCH .C XYZZY /CASE
```

To find all the files in the current directory which contain "inductive", without listing the matching lines, type

```
SEARCH * INDUCTIVE/FILES
```

To see a window of 5 lines (2 above and 2 below) around each matching line for "logic", type

```
SEARCH * LOGIC/LI:5
```

To find all occurrences of the string "A,B" in the files in the SOURCE\ directory, excluding the *.BAK and *.TMP files, type

```
SEARCH SOURCE\ /EX:.BAK,.TMP "A,B"
```

To search for the text string XYZ in all the files in subdirectories with two character names starting with A, type

```
SEARCH A?\ XYZ
```

TOUCH wildcard-filespec

Sets the file modification times to any value

Modifiers:

+ASH	Select only files with any of the given attributes. System and Hidden files are not selected by default
-ASH	Do not select files with any of the given attributes
/BEFORE[:date:time]	Select files modified before a given time
/SINCE[:date:time]	Select files modified on or after a given time
/ON[:date:time]	Select files modified in the 24 hour period beginning with the given time
/BIGGER[:size]	Select files bigger than (or equal to) the given size
/SMALLER[:size]	Select files smaller than (or equal to) the given size
/EXCLUDE:namespec	Do not select files which match the wildcard-namespec
/TIME[:date:time]	Set the file modification time to the given time.

Exit codes:

0	Normal	All specified files touched
2	Warning	No files matched given specifications
3	Error	Some files touched, some attempted but failed

4	Fatal Error	Syntax error, or no files could be touched

TOUCH sets the file modification time of the given files. If the **/TIME** modifier is not used, all the files are set to the current date and time. In this case, TOUCH sets all specified files to the same time, even if the touch operation takes several seconds to complete.

If the **/TIME** modifier is used, the syntax is exactly the same as for **/BEFORE**, **/SINCE** and **/ON**. The default dates and times are the same also: if the date is omitted, it defaults to the current day. If the time is omitted, it defaults to 0:00:00.

Examples:

To set all the files with .C and .H extensions to show modified as of now, type

```
TOUCH *.C,*.H
```

To set all the files except those with .EXE and .OBJ extensions to show modified as of now, type

```
TOUCH *.* /EXCLUDE:.EXE,.OBJ
```

To set the modification times of all the .EXE files with A as the second character of the filename, to midnight this morning, type

```
TOUCH ?A*.EXE /T
```

To set all files in the current directory tree that have the Archive attribute to show modified just before midnight tonight, type

```
TOUCH ...*.* +A /T23:59:58
```

To set all files modified last Monday to show modified as of last Sunday, type

```
TOUCH *.* /ON:MON /T:SUN
```

WHICH

WHICH filespec or **WHICH /?**

Searches your PATH for the executable file for a DOS command

Modifiers:

None

Exit codes:

0	Normal	Command was found in path
1	Information	Command not found in path
4	Fatal Error	Syntax error, or no filespec given

WHICH finds a file which is executable from the DOS command line. WHICH uses the same search rules as DOS, and locates the executable file corresponding to the given filespec.

If no directory is given in filespec, then your current directory and PATH will be searched. If a directory is given, only the given directory is searched.

If a drive is given in filespec but no directory, WHICH searches the current directory on the given drive for the executable. If the filespec isn't found, WHICH completes the search using your PATH.

If a drive and a directory are given in filespec, WHICH searches for the executable only in the given directory on the given drive.

If no extension is given, then in each directory WHICH searches for filespec.COM first, then filespec.EXE, and lastly filespec.BAT. The first file found is the one that DOS would execute if the given filespec were typed at the DOS command line.

Note that DOS 3.x and earlier ignores any extension given in filespec. DOS 4.x and higher properly handle filespecs with extensions, and if given, will only execute files which have the given extension. When an extension is given in the filespec parameter of WHICH, WHICH properly finds the executable file for the DOS version you are using (i.e. if you are using DOS 3.x or

earlier, WHICH ignores the given extension. If you are running DOS 4.x or later, WHICH searches for a command with the given extension.).

Examples:

To find which executable will run from typing XCOM, type

```
WHICH XCOM
```

This command causes WHICH to search the current directory on the current drive, then the directories in your PATH, in order, for the command named XCOM.

To find which file extension will be used when typing \PATH\XCOM, type

```
WHICH \PATH\XCOM
```

This causes WHICH to only search the directory \PATH\ for the executable XCOM.

To find the executable which will run when you type the command A:XCOM, type

```
WHICH A:XCOM
```

WHICH searches the current directory on the A: drive first for the executable XCOM. If XCOM isn't found, WHICH continues the search using your PATH.

To find which executable will run from typing XCOM.EXE, type

```
WHICH XCOM.EXE
```

but note that in DOS 3.x or earlier, the filespec found might not have an extension of ".EXE".

WIPEDIR wildcard-filespec

Deletes entire directory trees

WIPEDIR is a simple batch file which uses the IDEL and IRD commands to erase an entire directory tree. The wildcard-filespec should specify only directories, not files. WIPEDIR first deletes all the files in the directory tree, then removes all the subdirectories, and then removes the given directory. WIPEDIR accepts no modifiers, but you can use IDEL and IRD manually for deletions that require special modifiers.

WIPEDIR lets IDEL and IRD give you the usual confirmation prompts before deleting anything, so you are protected from mistyping this command. Note that because WIPEDIR is a simple batch file, it relies on the error reporting capability of the underlying IDEL and IRD utilities.

You should be very careful when using WIPEDIR with wildcards, because it can delete many directory trees with just one command.

WIPEDIR is a batch file based on these 3 commands:

IDEL %1\...*.*	delete files in given and subdirectories
IRD %1\...*.*	remove all subdirectories from given directory
IRD %1	remove the given directory

WIPEDIR.BAT also has additional commands to avoid undesirable messages.

Examples:

To remove an entire directory tree named DEAD, type

```
WIPEDIR DEAD
```

To remove all the directory trees starting with the letters DEAD from the root directory, type

```
WIPEDIR DEAD*
```

Pay close attention to the confirmation prompts when you use this command, because it can delete a large amount of data in a very short time.

APPLICATIONS

Finding a File

Sometimes, you know the name (or part of the name) of a file you want, but you don't know where (in what directory) to find it. To find in what directory a file (or other directory entry) exists, simply use the DI command to search the whole disk by specifying the directory tree starting at the root directory. For example, if you are looking for a file that has "LOST" somewhere in it's name, type

```
DI \...*LOST*
```

Cleaning Up Your Disk

After a while, disks often become cluttered with old, useless files. These files not only waste space, but also slow down your disk by making utilities search through needless data to find what you really want. Disk management is a large topic, but the InCommand utilities provide several features to make it a lot easier.

For example, suppose you have a simple word processor that leaves a lot of annoying temporary files lying around all throughout your disk directories. If you know that those junk files are always named ~WRIxxxx.TMP (where "xxxx" are random digits), then you can clean your whole disk of them with one simple command:

```
IDEL \...~WRI*.TMP
```

As another example, suppose you edit a lot of files, and for safety, your editor creates a backup copy of each file you edit in a file of the same name, but with an extension of ".BAK". After a while, you may find your disk littered with scores of .BAK files. You want to get rid of old ones, say, more than a month old, but you'd like to keep the recent ones, in case you need them. That's easy, with InCommand. If you want to delete all the .BAK files created before May anywhere on the disk, type

```
IDEL \...*.BAK /BE:5-1
```

Why Doesn't My Computer Work?

Did you ever turn on your computer one day and find that, suddenly, some software doesn't work any more? Something you just used yesterday, but now its broken. It may be that you changed some configuration or setup files, but you can't really remember what all you did yesterday (or worse, someone else used your computer and changed something). You may have lots of files in many different directories that all have to be set just right for everything to work properly. Since you know it worked yesterday, can you find out what's changed since then? You can with InCommand! If today is Tuesday, then to scan your entire disk for all the files changed since yesterday, type:

```
DI \.../SINCE:MONDAY
```

Look through the file list for things like AUTOEXEC.BAT, CONFIG.SYS, *.INI, and all your other configuration files. Then examine those recently modified files to make sure they're still correct.

Backing Up Hard Disks

Everyone should know that it is important to have backup copies of all files on your hard disk, because it is a certainty that every hard disk will crash sooner or later. When it happens to you, you want to be prepared. Amazingly, DOS gives you no reasonable way to make backups, despite their importance to everyone.

The DOS BACKUP and RESTORE utilities are largely unusable because they don't let you make reasonable "incremental" backups. Incremental backups are those where you make a full backup of your disk once, and thereafter, you only back up those files which have changed since your original full backup. Since you probably want to backup fairly often, incremental backups save you countless hours by copying only the small fraction of files you've changed since your last incremental backup.

A major point here is that most people must use floppy disks to backup their hard disks, and a single floppy can hold only a fraction of the files that a hard disk can hold. So most backups span many floppies. Though the DOS BACKUP utility allows you to make backups over several floppies, and the DOS XCOPY command allows you to make incremental backups to a single floppy,

neither DOS utility can make incremental backups across multiple floppies. (We should note that the BACKUP utility will let you copy only your recently modified files to a new set of floppies, but BACKUP will not update your original set of backup floppies. The result is that now you have 2 sets of floppies, and next week you'll have 3 sets, and so on. Not a very useful capability.)

Making the Backups

With InCommand, your problems are solved. The ICOPY utility can easily perform incremental backups across any number of floppies. You do it with 2 steps:

1. Make a full backup of your disk, and then
2. After a while, make an incremental backup of only those files that need it.

Step 1 can be performed once, and step 2 can be repeated as often as you want.

1. First, you create your original full backup of all your files. Let's say you want to backup your C: hard disk to floppies in your A: drive. To do this, first set the Archive attribute on all files on the disk, to indicate that every file needs to be backed up (archived):

   ```
   CHATT +A C:\...*.*
   ```

   Next, ICOPY all the files to floppies by typing

   ```
   ICOPY C:\...*.* A: /MODIFIED
   ```

   When the first floppy fills up, ICOPY will tell you that the disk is full, and ask if you want to retry the copy operation. Just replace the full floppy with a blank one, and answer "R" (retry) to continue copying. Repeat these steps for as many floppies as it takes to backup your hard disk. Mark your last floppy, as this one will likely have the most free space on it.

2. To make an incremental backup, put one of your full backup floppies (that you made in step 1) in A:, and type

   ```
   ICOPY C:\... *.* A: /MODIFIED/IFEXISTS
   ```

This will bring up to date any files on that floppy. If the floppy fills up, ICOPY will ask you if you want to retry the file that was too big. Answer "S" (skip), and ICOPY will continue with the rest of the floppy. **Do not change floppies until this ICOPY command completes successfully.**

Repeat this command for each of your backup floppies, except the "last" one that you marked. When all but the "last" floppy are updated, update the "last" one with

```
ICOPY C:\...*.* A: /MODIFIED
```

This will update any files on the last floppy, and also add any new files that did not exist on your backup floppies. Eventually, this "last" floppy will fill up, and you will not be able to fit all the new files on it. No problem. Simply insert a blank floppy, and answer "R" (retry) to ICOPY's asking to retry. From now on, this new floppy will be your "last" backup floppy, until it fills up and you create another one.

Notice here that if you need to format a new floppy to finish the second ICOPY command, you can do that by answering "D" (go to DOS) to ICOPY's retry question. Then format a new floppy, type "EXIT" to return to ICOPY, and enter "R" to continue to copy files. You can repeat this for as many new floppies as you need to format.

So in summary, you can make multi-floppy incremental backups by 1) making a multi-floppy full backup, and then 2) as often as you like, make incremental backups across the set of floppies. You can add more floppies as you need them at any time.

You may wonder what happens when a file already on the first floppy grows so large that it won't fit on the floppy any more. In that case, the file is automatically deleted from the floppy when ICOPY discovers that the new version won't fit. Later, it will be copied to your "last" floppy when you finish the incremental backup.

Restoring From Your Backups

To restore your hard disk from this set of floppies, you simply copy each floppy in turn to your hard disk:

```
ICOPY A:\... C:\
```

The floppies may be copied in any order. After restoring your hard disk, the floppies are still a valid backup set, and may be used for future incremental backups of the now-restored hard disk.

Comparing Two Directories

To compare the dates/times of files in a directory with the files in another (presumably related) directory, you can use EXECUTE/IFNEWER. For example, suppose you have a directory, DIR1, of *.C files, and a parallel directory, DIR2, of archives of DIR1 named *.LIB. You'd like to know which files in DIR1 are newer than their last archive, and therefore need archiving. Just type

```
EXECUTE /IFNEWER [DIR1\*.C] REM [DIR2\]
```

In this case, the REM command does nothing when executed, so you simply get a list of file names.

Fast Updating of a Directory

Sometimes, you may have a directory (or directory tree) of the latest versions of some files. Periodically, some or all of those files are updated in a master directory, and you must update your directory to match the master. You could simply copy all of the files from the master to your directory, but if only a few of the files have changed, you will waste a lot of time copying over those files that have not changed (and were still up-to-date in your directory). With ICOPY, you can easily avoid copying the files that have not changed by simply using the /IFNEWER modifier. For example, suppose you want to update your directory SLAVE\ to the latest version of directory MASTER\. Simply type

```
ICOPY MASTER\ SLAVE\ /IFNEWER
```

Only those files in MASTER\ which have changed since your copy of SLAVE\ was last updated will be copied. Your SLAVE\ directory will be up-to-date, and you will have saved time.

Physically Sorting a Directory

Normally, the files in a directory are stored in a first available, first filled order. This means that usually, over time, the files are in random order. You've probably noticed how annoying that is with the DOS DIR command, which doesn't even sort the files before listing them. With DI, of course, there's no problem, because the listings are always sorted.

Sometimes, though, that's not enough. For some applications, you need the files in a directory to be physically sorted on the disk, so that when DOS runs through them, it does so in a predictable way. InCommand gives you a fast way to physically sort a directory. As described in the Utility Reference, the MOVE command moves files in sorted order (alphabetically by default, but you can choose sorting by time, size, or extension). To sort an existing directory, simply MOVE all the files to a new, temporary directory, then move them back to the original directory. For example, to sort all the files in directory CITIES, do this:

```
MOVE CITIES\*.* CITIES\~TEMP~.~\ /CREATE
MOVE CITIES\~TEMP~.~\ CITIES\
IRD CITIES\~TEMP~.~
```

We used an odd name for the temporary directory to reduce the chance that we would pick the name of an existing directory. Note that because the files are not copied, the MOVE is very fast, even for large files. **This procedure also works for most network directories.**

You can generalize this procedure in a batch file. InCommand includes the batch file SORTDIR.BAT as an example. You can use SORTDIR as a general utility, but because SORTDIR is a simple batch file, its error detection and messages are very simple.

File Existence Testing in Batch Files

You can test for the existence of any files (or directories) in a batch file with the DI command. You can avoid any display output with the /EXIST modifier. This is much more powerful than the DOS IF EXIST commands, because of the full InCommand file selection capabilities. DI returns an exit code which your batch file can test. For example, to test if any file (not any directories) named TESTFILE.* exist in the current directory, you could include these lines in your Batch file:

```
DI TESTFILE.* -D /EXIST
IF ERRORLEVEL 1 GOTO NOFILES
     ...instructions if one or more files exist
NOFILES:
```

Similarly, you could test for the existence of a directory by using +D in place of -D in the above example. See the InCommand utility file WIPEDIR.BAT for another example. DI/EXIST is fast because it stops after finding the first match.

DOS Facts and Foibles

The 52 legal filename characters in DOS are

```
     ! # $ % & ' ( ) - 0 1 2 3 4 5 6 7 8 9 @ A B C D E F G H I J
K L M N O P Q R S T U V W X Y Z
     ^ _ ` { } ~
```

Lower case letters are translated into uppercase by DOS. The remaining 16 printable ASCII characters are illegal in a file name:

```
     " * + , . / : ; < = > ? [ \ ] |
```

The DOS batch processor removes all commas, semicolons, and equal signs ("=") from the batch parameters, and replaces them with spaces. Thus

```
     MYBATCH A=B C,D
     MYBATCH A,B,C;D
     MYBATCH A B C D
```

are all equivalent. This substitution does not occur on external (.COM or .EXE) commands.

DOS 3.3 has a bug in its processing of destination wildcards that does not exist in DOS 4.0 or 3.2. Forms like the following do not work in the RENAME command:

```
     RENAME *.DEF X*.DEF
```

If you have a file named ABC.DEF, DOS 3.3 incorrectly names the file XABC.DEF instead of XBC.DEF. Therefore, if you use a version of DOS with this bug, InCommand destination wildcards may produce different results than your version of DOS. The DOS COPY and XCOPY commands seem to work properly. Note also that some vendors have customized versions of DOS, so the version number alone may not be enough to tell if you have this bug or not.

DOS 4.0 now correctly processes extensions you specify on external commands, instead of ignoring them as in earlier versions. Therefore, if you type

```
     MYBATCH.BAT
```

and a file named MYBATCH.COM exists first in the path, DOS 4.0 will still run the batch file. WHICH always finds the actual command your DOS version will run.

Some DOS systems have a problem with ROM BIOS shadowing (into RAM). When shadowing, if you "shell out" of Windows to DOS, you

cannot get a directory listing of a floppy disk. In fact, you cannot do anything that computes the number of free bytes on a floppy disk. If you try, Windows reports a protection error, and terminates your DOS session. This is true of DOS's own DIR command, and there is nothing DI can do to work around the DOS bug.

Revision History

Version 1.0 was released in February 1991.

Version 1.1 was released in May 1991, and included the following changes:

- Added /TOUCH to the ICOPY command.
- Allowed ICOPY to continue copying to another diskette when one fills up.
- Added /BIGGER and /SMALLER file selection modifiers.
- Sped up DI/TOTAL by not sorting the directory internally.
- Allowed network drives with more than 1 letter to be used in wildcard-filespecs
- Added the /EXIST modifier to DI.
- Fixed occasional 1 hour error in DAYTIME
- Added /BYxxx to MOVE and ICOPY
- Added free bytes display to IDEL

Version 1.2 was released in August 1991, and included the following changes:

- Added subdirectory subtotals and the /SUBTOTAL modifier to DI
- DI/ABS and DI/REL now sort properly
- Standardized all the /HELP output
- Added /? as a synonym for /HELP (for DOS 5 compatibility)
- Added /PROMPT as a synonym for /INDIVIDUALLY in IDEL and IRD (for DOS 5 compatibility)
- Fixed an occasional lockup after using MOVEDIR
- Added SEARCH

Version 1.2a was released in September 1991, and included the following changes:

- Added DHELP
- Fixed ICOPY /MODIFIED to not change the attributes if the file is not copied
- Fixed directory detection on PC/NFS networks

Version 1.2b was released in October 1991, and included the following changes:

- Added left/right scrolling to SEARCH
- Added options to ICOPY when a destination disk is full

Version 1.3 was released in December 1991, and included the following changes:

- Added multiple filename wildcard patterns
- Added find (P)revious to SEARCH
- Added full file directory information to SEARCH <F1> screen
- Added /IFOLDER to ICOPY and EXECUTE
- MOVEDIR works on network drives
- Added /OVERWRITE to ICOPY and MOVE

Version 1.3e was released in October 1992, and included the following changes:

- Added /BACKUP to MOVE and ICOPY

Version 2.0 was released in January 1993, and was the first Shareware release.

Index

InCommand User Response

Name: Date:

Organization:

Street:

City, State, Zip:

What do you like about the utilities or the manual?

What do you dislike about the utilities or the manual? Please make suggestions for improvement.

For what applications do you use your computer?

Please mail your comments to:
Inductive Logic
P.O. Box 26238
San Diego, CA 92196
(619) 578-5146

Inductive Logic Order Form

The following prices and order form are for INDIVIDUAL licenses only. The software governed by this license may legally be used by only 1 person at a time. It is specifically prohibited from being installed in any network directory available to more than one user. Network and site licenses are available at discount prices. Call us for details.

Please mail me (postage paid by IL) the following Inductive Logic products:

Description	Qty.	Unit Price	Extended Price

InCommand: Command Line Utilities for DOS.

Single user license ____x $35.00 each = _____

California deliveries add 7.25% sales tax _____

Shipping and handling add $5 (USA), $15 (foreign) _____

For COD add $5 _____

Total _____

System Requirements: IBM compatible PC, DOS 3.0 or higher, and 256K of RAM. A hard disk is strongly recommended.

Please make check or money order payable to "Inductive Logic". Do not send cash through the mail.

Check one: __ 5.25" 1.2 Mbyte floppy
__ 5.25" 360 kbyte floppy
__ 3.5" 1.44 Mbyte floppy

Name _____

Company _____

Address_____

Phone (_____)_____-_____ Ext._____

InCommand is a trademark of Inductive Logic.

How did you hear about our product ?

→