Price: $2.50

TYMSHARE MANUALS

REFERENCE SERIES

EXECUTIVE

January 1971

TYMSHARE, INC.
525 UNIVERSITY AVENUE, SUITE 220
PALO ALTO, CALIFORNIA 94301

REGIONAL OFFICES

Inglewood, California ■ Los Altos, California ■ Englewood Cliffs, New Jersey ■ Chicago, Illinois

DISTRICT OFFICES

Inglewood, California ■ Los Altos, California ■ Newport Beach, California ■ Oakland, California
Chestnut Hill, Massachusetts ■ Englewood Cliffs, New Jersey ■ Dallas, Texas ■ Houston, Texas
San Diego, California ■ Darien, Connecticut ■ Chicago, Illinois ■ Portland, Oregon
Arlington, Virginia ■ Seattle, Washington ■ Toronto, Ontario ■ Paris, France

CONTENTS

Page

TYMSHARE REFERENCE MANUAL
SYMBOL CONVENTIONS

The symbols used in this manual to indicate Carriage Return, Line Feed, ALT MODE/ ESCAPE, and the Emergency Exit Key are as follows:

Carriage Return: ↵

Line Feed: ↴

ALT MODE/ESCAPE: ⊕

Emergency Exit Key (EEK!): ☆

Control Characters

Control characters are denoted in this manual by a superscript c. For example, D^c denotes Control D. The method of typing a control character depends on the type of terminal. Consult the literature for your particular terminal.

The Emergency Exit Key

The keyboard position for the Emergency Exit Key varies among terminals. It is usually a Control Back Arrow ($\leftarrow^c$) or a Control Underscore ($\_^c$). The character has ACSII code 159 (237 octal) and the internal code 127 (177 octal). The use of the Emergency Exit Key is discussed on Page 9.

Action At The Terminal

To indicate clearly what is typed by the computer and what is typed by the user, the following color code convention is used:

Computer: **Black**

User: **Red**

INTRODUCTION

The EXECUTIVE is the key to the Tymshare system. All the languages and applications programs are called into the computer by the EXECUTIVE as they are needed. In addition, the EXECUTIVE commands implement the effective and flexible file security system.

Certain functions which do not require a subsystem can be performed directly by EXECUTIVE commands. These functions include:

- Entering and leaving the Tymshare system.
- Creating, deleting, and renaming files.
- Setting access controls for files and file directories.
- Sending and receiving mail.
- Determining terminal connect time, machine number, etc.

The main purpose of the EXECUTIVE commands is to make using the Tymshare system convenient and versatile. The commands are simple, straightforward, and easy to learn with many optional features which can be learned in a few minutes.

Whenever the EXECUTIVE is ready to accept a command, it types a dash (—). If the EXECUTIVE does not understand the command, it prints a question mark (?), returns the carriage, and types another dash. The user then retypes the command correctly. Other error messages state explicitly which part of the command is in error and request the user to retype that part only. The WHY command will explain any error message in detail. If an error is detected before typing the Carriage Return, abort the command by pressing the ALT MODE/ESCAPE key.

In this manual, you will encounter the concept of a file. Files are an important part of the Tymshare system. Simply, a file is a program or data stored on a large disk type storage device. Files are permanent storage; that is, they stay on the system until they are explicitly removed even though you terminate your session. Files are identified by the name you give them when you create them.

Some commands require a file name be specified. If the user types a Carriage Return without the file name, the EXECUTIVE will prompt him.

NOTE: EXECUTIVE commands may be abbreviated to their first three letters.

SECTION 1

USING THE TYMSHARE SYSTEM

ENTERING THE SYSTEM

Three steps are required to enter the Tymshare system: calling the computer, identifying the type of terminal, and identifying the user. This process is commonly called logging in.

Calling The Computer

The specific procedure for contacting the computer depends on the terminal arrangement. Two typical communication devices are the data modem (acoustic coupler) and the Data Phone. The procedures for using these two devices are described here. Tymshare operates terminals in the full duplex mode. Check your terminal for a Full Duplex/Half Duplex mode switch. In addition, some terminals have an Upright/Inverted code switch. This switch should be in the Upright position. If you have any questions about calling the Tymshare computer, contact your Tymshare representative.

Data Modem

1. Put the terminal in the line, or compute, mode.
2. Be sure that both cords from the terminal are plugged into the modem and that the modem is plugged into a standard wall outlet.
3. Using a regular telephone, dial the Tymshare computer number.
4. When the answer tone sounds, place the telephone handset into the modem in the orientation indicated on the coupler; push the modem's ORIGINATE button.

Data Phone

1. Put the terminal in the line, or compute, mode.
2. Depress the TALK button.
3. Dial the Tymshare computer number.
4. When the answer tone sounds, depress the DATA button and replace the handset.

Identifying The Terminal

As soon as the connection to the Tymshare computer is made, the system turns on the terminal and sends a message. This message is sent at 10 characters per second and is readable only on a 10 cps terminal. On other terminals, a sequence of characters will print, and then the terminal will pause.

Type the identification character for your terminal. This character tells the system which code and which transmission speed to use to communicate with your terminal.

Then the system will return the carriage and display

PLEASE LOG IN:

The table below lists the identification characters. If you have a question about which one applies to your particular terminal, contact your Tymshare representative.

Identification Character	CPS In/Out	Examples	Comments
D	10	Model 33 Teletype	
B	15	Model 37 Teletype (without parity)	ASCII
Carriage Return	15	IBM 2741, Datel, Dura, Novar	Correspondence Code
J	15	Model 37 Teletype (with even parity)	ASCII (even parity)
F	15/30	Tymshare 1030, Execuport, Gulton, Syner-Data	Recommended over C where telephone line quality is marginal.

Identification Character	CPS In/Out	Examples	Comments
A	30	CRT Terminals	The particular character used depends on the Carriage Return speed of the terminal.
C	30	Tymshare 1030, Execuport, Gulton, Syner-Data	
G	30	Memorex, GE TermiNet 300	

Logging In

The log in procedure requires typing a user name and password, both of which are registered with Tymshare. The system checks both the user name and the password before admitting the user to the system. An optional project code can be typed during the log in. The project code is included in the billing information sent to the customer; therefore, the project code can be used to assign costs.

After the system types PLEASE LOG IN:, the user types a Carriage Return. The system replies with a request for the user name. The user types his user name followed by a Carriage Return. The system next requests the password. The user types the password followed by a Carriage Return. For security, the computer does not print the password on the terminal. The system then requests the final information, the optional project code.

PLEASE LOG IN: ↵
USER NAME: JONES ↵
PASSWORD: ↵
PROJ CODE: ↵

The user types a project code and a Carriage Return. If no project code is wanted, he types a Carriage Return only. Any character typed in error in a project code may be deleted with Control A (A^C). When A^C is typed, a ← prints on the terminal and the preceding character is deleted. Repeated use of A^C deletes several characters. For example, the following project code

PROJ CODE: NERA^C←T3 ↵

is accepted as NET3 by the EXECUTIVE.

After the user has entered the requested information correctly, the system will type the time and date. For example,

TYMSHARE 12/24 11:20
–

The dash indicates that the user is in the EXECUTIVE, and the system is ready to accept an EXECUTIVE command.

Only one terminal at a time can be logged in under each user name. If a terminal is already using a name and an attempt is made to enter another terminal under the same name, the message

ALREADY ENTERED

will print. Then the computer will again type

PLEASE LOG IN:

If there are typing errors while logging in, the system replies with ERROR, TYPE: followed by another request for that information. In the following example, the user types an unacceptable user name, corrects his error, and continues to log in.

PLEASE LOG IN: ↵
USER NAME: ALFROD ↵
ERROR,TYPE:USER NAME: ALFRED ↵
PASSWORD: ↵
PROJ CODE: K-123-K ↵

TYMSHARE 12/24 11:25
–

Once the user is thoroughly familiar with this log in procedure, an alternate and faster method can be used as follows:

PLEASE LOG IN: ALFRED;;K-123-K ↵
↑Non-printing password here.

TYMSHARE 12/24 11:25
–

A semicolon must be typed between the user name and the password and between the password and the project code. If no project code is needed, the user types a Carriage Return after the second semicolon.

The error diagnostics are the same regardless of which log in procedure is used. When the system indicates an error, the user can correct the error and type the rest of the log in information in the normal way.

The user is allowed two minutes to log in. This time limit is set for security to prevent an illegal user

from accessing the system. If the log in is not completed within the time limit, the system prints a disconnect message and hangs up.

If you have any questions about the status of the system or the log in procedure, call the local Tymshare office. The telephone numbers in your area are usually on a label directly above the terminal keyboard.

LEAVING THE SYSTEM

To leave the system, type the command

– LOGOUT↵

or, simply,

– LOG ↵

The system replies with

CPU TIME: n SECS *Number of computing seconds used*

TERMINAL TIME: hours: minutes: seconds *Connect time*

PLEASE LOG IN:

Either disconnect the terminal or let another user log in.

NOTE: It is unnecessary to retype the terminal identifying character to log in again.

If the terminal remains connected to the computer and another log in is not made within two minutes, the system will type the disconnect message and hang up.

To change project codes without logging out and logging in again, use the PROJECT command as follows:

– PROJECT new project code ↵

CPU TIME: n SECS. *The system prints the time charged to the previous project code.*

TERMINAL TIME: minutes: hours: seconds

–

Typing errors in the new project code can be edited with Control A. See *Logging In* on Page 4.

If the user turns off the modem, disconnects the line, or otherwise is disconnected before giving the LOGOUT command, the entire machine contents will be saved on a file providing the fail-safe feature has been activated. An explanation of the fail-safe feature follows below.

Any command can be aborted by typing an ALT MODE at any time before the Carriage Return or Line Feed is typed.

If you need a reminder of the valid commands, the HELP command can assist you; it lists all of the EXECUTIVE commands with a brief description of each.

The WHY command can help you if the EXECUTIVE has printed an error message. Simply type WHY and the EXECUTIVE will describe the error.

THE FAIL-SAFE FEATURE

SETFAILSAFE and FAILSAFE are Tymshare programs which protect the user from losing time and effort due to line disconnect or terminal failure. If the line is disconnected before the LOGOUT command is given, the system will save the program, variable values, and subsystem references on a file. When the user logs in later, his working space can be restored to the exact status at the time of the unusual disconnect. This means that if he must leave the terminal quickly to attend to other matters, or if the line should be disconnected accidentally, he can be confident that his work has been saved. The work is saved on a special file called the fail-safe file. *NOTE: The fail-safe feature works only if the user has created a fail-safe file.*

To create a fail-safe file, type

– SETFAILSAFE ↵
NEW FILE

–

It is necessary to execute this program only once under each user name. It is then set until it is removed.

Because of the proprietary nature of the Tymshare Library, library programs cannot be saved on the fail-safe file.

SETFAILSAFE creates an empty file named /$/ in the user's directory.[1] If a premature disconnect occurs, the entire contents of the computer's memory are dumped on the /$/ file. This file can be used for temporary storage like any other file. Of course, the previous contents of /$/ will be erased if a premature disconnect occurs.

The contents of the /$/ file are recovered by the FAILSAFE program.

– FAILSAFE ↵
OLD FILE
HAS BEEN CLEARED. YOU MAY PROCEED

1 - The /$/ file can be created directly by the user with the COPY or RENAME command. In these cases it is unnecessary to use the SETFAILSAFE command. COPY and RENAME are discussed on Pages 13 and 14, respectively.

The computer loads all of the data stored on the /$/ file into the user's working area and empties the /$/ file. FAILSAFE will return the user to the language he was in at the time of the disconnect. The name of the language will print as a reminder.

To remove the fail-safe feature, use the command

– DELETE /$/ ↵
–

The fail-safe feature will not work after the DELETE /$/ command is used until the SETFAILSAFE program is run again, or the /$/ file is created directly.

As another fail-safe feature, every night Tymshare stores on magnetic tape the contents of all files that have been created or altered that day. Furthermore, at least once a week the entire contents of all files are written on magnetic tape. Therefore, if you should accidentally erase or destroy a file, just call your Tymshare representative and he can have it restored to your file directory as it existed at the end of the previous day.

UPPER AND LOWER CASE TERMINALS

The Tymshare system will accept terminals with upper and lower case. A user on such a terminal has a choice between using the full 126 character set[1] or restricting the system to recognize only 64 different characters (no lower case). Two programs are available for these terminals. ONLC turns on the lower case (full character set), and OFFLC turns off the lower case.

Example

– ONLC ↵
– "I can use Lower Case" ↵ *The user types the comment in both upper and lower case.*

or

– OFFLC ↵
– "I CANNOT USE LOWER CASE" ↵
The user can type the comment in upper case only.

When the user first logs in, lower case is turned off. In this mode, lower case letters typed on the terminal are converted to upper case. To use lower case, give the ONLC command.

All EXECUTIVE commands are in upper case characters.

1 - Two additional characters have been reserved for system use. These characters have internal codes 91 (133 octal) and 93 (135 octal) and correspond to the left and right braces.

SECTION 2
TRANSFERRING INTO AND OUT OF THE EXECUTIVE

CALLING LANGUAGES AND LIBRARY PROGRAMS

All Tymshare languages and Tymshare applications programs are called from the EXECUTIVE. In addition, some User Program Library (UPL) programs are called directly from the EXECUTIVE.

Tymshare Languages

Tymshare offers a variety of languages. For example, SUPER BASIC and SUPER FORTRAN are powerful, user-oriented computation languages with complex numbers, matrix commands, and string functions; BATCH FORTRAN IV is a fast, efficient batch type language; EDITOR is a text editing language.

All languages are called from the EXECUTIVE by typing the name and a Carriage Return. For example,

– **SBASIC** ↵	– **BFORTRAN4** ↵	– **EDITOR** ↵
>	+	*

Many of the Tymshare languages display a special character to request input just as the EXECUTIVE displays a dash. The table below lists some of the more popular Tymshare languages and their identifying characters. Most of the applications programs display a colon.

Language	Minimum Call Letters	Identifying Character
SUPER BASIC	SBA	>
SUPER FORTRAN	SFO	>
BATCH FORTRAN IV	BFO	+
FORTRAN II	F2C	+
	F2OS	+
BATCH FORTRAN	FTC	+
	FOS	+
EDITOR	EDI	*
CAL	CAL	>
XCAL	XCAL	>
DCAL	DCAL	>

To leave any language type QUIT.

Tymshare Library

The Tymshare Library offers user-oriented programs covering diverse applications.

Tymshare Library programs are called by typing the program name in the EXECUTIVE. For example,

– **STATPAK** ↵ *STATPAK is a general purpose statistical analysis package.*

The Tymshare Library includes many large applications programs such as

STATPAK	Statistical analysis.
RETRIEVE	Information retrieval.
LINPROG	Extended linear programming.
ECAP	Linear circuit analysis.
CODED-CAP and CIRC	Non-linear circuit analysis.
MICAP	Two-port device network analysis.
LOGSIM and DIGILOG	Logic circuit analysis.
LOGMIN	Logic circuit minimization
CSMP and ANALOG	Continuous system modelling.
LAPLACE	Laplace transform inversion.

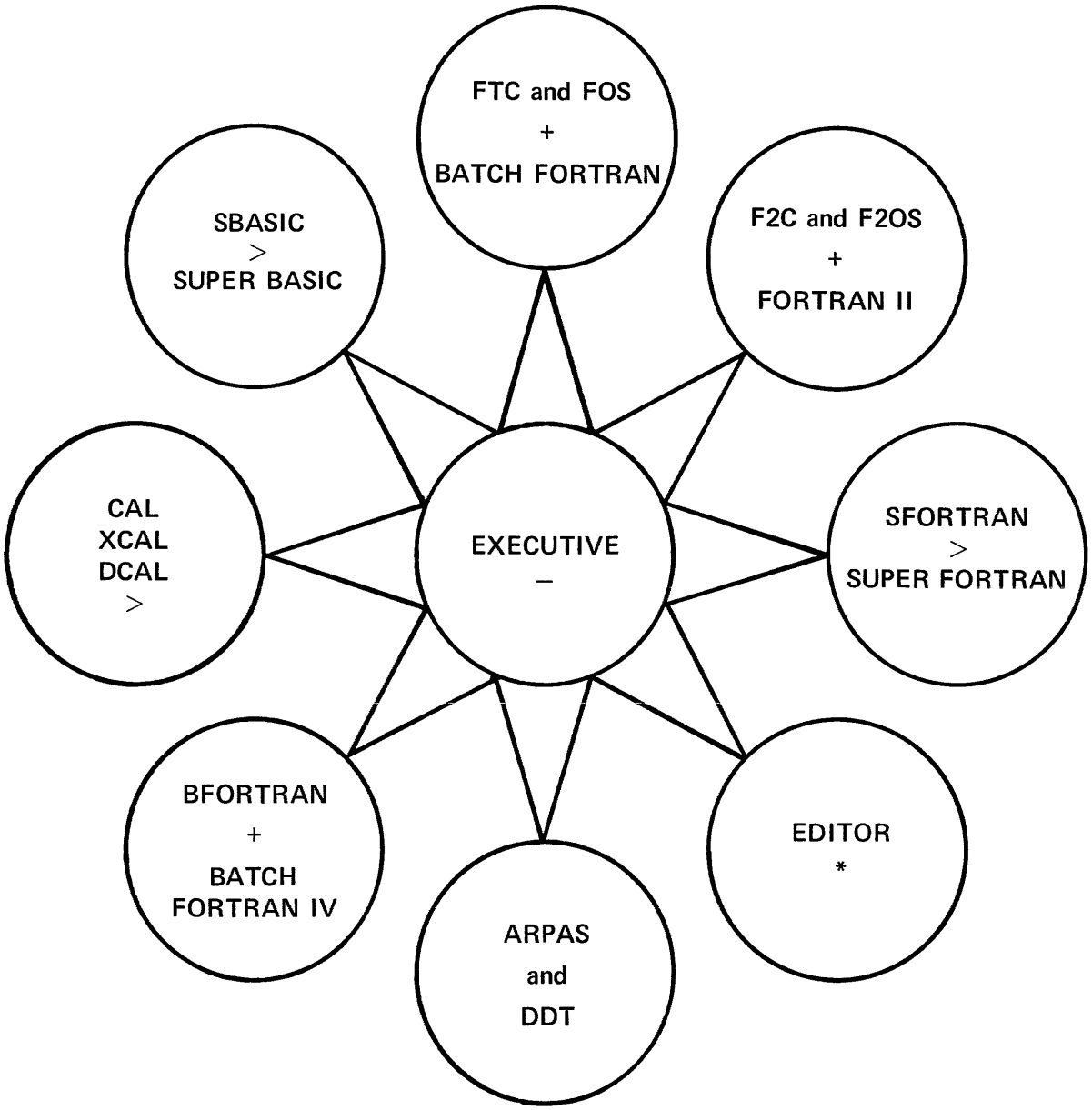
The Tymshare applications programs are fully documented in their own manuals.

Several library programs have been documented in this manual. They are:

CHECKSUM	OFFLC
CIPHER	ONLC
COMPARE	POSTMAN
CUC	SETFAILSAFE
FAILSAFE	SUMMARY
HELP	TAPE
LIST	VERIF

Tymshare Library programs function like the EXECUTIVE commands with the following two exceptions:

1. The commands can be shortened to the first three characters; whereas program names must be typed in full.
2. The REENTER and CONTINUE commands cannot be used after programs are run from the EXECUTIVE. This is discussed under *Returning To A Language* on Page 9.



Calling The Tymshare Languages

User Program Library (UPL)

The UPL contains programs that have been donated by our users and accepted by Tymshare. Tymshare examines the programs for general usefulness, accuracy, and adequacy of documentation before accepting them for the library. Tymshare does not provide the same level of maintenance as it does for the Tymshare Library programs.

The UPL programs are called by placing a crosshatch in front of the name. For example,

– #MULREG ↵

Refer to the Program Availability List for a complete listing of the UPL and Tymshare Library programs.

Manuals describing each language and applications program are available from your Tymshare representative. He also has manuals describing the applications programs and UPL programs applicable to different fields: business and management, statistics, electronics, mathematics, utility functions, and so forth.

RETURNING TO A LANGUAGE

Occasionally a user is working in a language and needs to use an EXECUTIVE command. For example, he finds that he needs to rename a file or to remove a prohibition against writing on a file. He can return to the EXECUTIVE with QUIT, use the commands, and then return to the language with the REENTER command. REENTER will restore not only the language but also the entire working space (programs, variables, etc.).

Example

– SBA ↵ *The user calls SUPER BASIC.*

> LOAD DMPROG ↵ *He loads and runs a program.*
> RUN ↵

ERROR IN STEP 1885:
FILE NAME NOT IN FILE DIRECTORY
The program tried to open a file which does not exist.

> LIST 1885 ↵
1885 OPEN "DATA",INPUT,4

> QUIT ↵ *The user transfers to the EXECUTIVE.*

– RENAME DATA2 AS DATA ↵
He renames his file to correspond to the name used in the program, and reenters SUPER BASIC.

– REENTER ↵
SBASIC
> GOTO 1885 ↵ *All of his variable values have been restored.*

In this example, if the user had called SUPER BASIC by typing

– SBA ↵

instead of REENTER, he would have had to reload his program and rerun it.

When the user leaves a language, any files he had open are automatically closed. After REENTER, the files are still closed. Therefore, occasionally, it is necessary to reopen the data files before continuing at the point of interruption.

Calling another language or library program, or giving certain commands will erase the system's memory and it is impossible to use REENTER after the memory has been erased. The following commands **do not** erase memory of previous work.

COMMAND	FILES	REENTER
CONTINUE	INIT	REMOVE
COPY	LAST	RENAME
DATE	LIMIT	SEND
DECLARE	MAIL	STATUS
DELETE	MEMORY	SYSNO
DIRECTORY	PFDC	TIME
DUMP	PMT	WHO
FDC	PROJECT	WHY

No other commands permit using REENTER.

Tymshare has implemented the Emergency Exit Key. If the user types this key, he is immediately returned to the EXECUTIVE. In contrast, if a user is executing his program in SUPER BASIC and types an ALT MODE, he is returned to the SUPER BASIC command level. If he types the Emergency Exit Key, he will be in the EXECUTIVE. It is impossible to use REENTER after an emergency exit.

If the user has transferred directly from executing his program to the EXECUTIVE, he may be able to resume execution with the CONTINUE command.

Example

– SBA ↵ *The user calls SUPER BASIC and loads a program which prints the integers 1 to 10.*

> LOAD BOBM ↵
> RUN ↵

1 2 ☆

He types an Emergency Exit and returns directly to the EXECUTIVE.

– CONTINUE ↵ *CONTINUE resumes execution.*

SBASIC 3 4 5
6 7 8 9 10

>

It is impossible to use REENTER or CONTINUE for an applications program.

RUNNING PROGRAMS FROM THE EXECUTIVE

Programs can be stored on a special kind of file called a GO file. Programs on GO type files can be run directly from the EXECUTIVE by giving the GO command and the file name.

– GO name of GO file ↵

Example

– GO ANALYSIS ↵ *Analysis is a GO file.*

GO files can be created in several Tymshare languages as shown in the table below.

CREATING GO FILES

Language	Command
SUPER BASIC	SAVE BINARY file name↵
BATCH FORTRAN	;W file name ↵
FORTRAN II	W file name ↵
BATCH FORTRAN IV	WRITE file name ↵ (saves single program or overlay) DUMP file name ↵ (saves all overlays)

GO files can be identified with the DIRECTORY command. GO will appear in the TYP column. Please refer to the discussion of the DIRECTORY command on Page 19.

SECTION 3
CREATING FILES

A file is an area on a large storage disk where programs, data, or text is stored from one session at the terminal to another. Each Tymshare language has commands to create files. The EXECUTIVE, in addition, has commands to list information about files, to set file protection controls, and to remove files from a directory.

RULES FOR NAMING FILES

Files are designated by the name assigned by the user when he creates them. A file name can contain any combination of digits, letters, and @. In addition, a file name can contain a series of characters including any characters except Line Feed or Carriage Return if the series is enclosed in single quote marks or slashes. A series of characters which obeys the following rules is an acceptable file name.

Rule 1 A file name may contain any combination of the characters

0 through 9
A through Z
@

Rule 2 A file name can contain protected strings, that is, a series of characters in slashes or single quotes. Protected strings can contain any characters except Line Feed (J^C), Carriage Return (M^C), and the delimiting character itself (/ or '). *NOTE: To include a Control A in a file name in the EXECUTIVE, the A^C must be preceded by V^C.*

Rule 3 Certain reserved file names cannot be used:

TELETYPE	NOTHING
TELETYP	NOTHIN
TELETY	NOTHI
TELET	NOTH
TELE	NOT
TEL	NO
TE	N
T	
These names always designate the terminal.	These names designate a null file. See Page 12.

Rule 4 A file name may contain a maximum of 45 characters.

Example: Acceptable File Names

TEST	*Unprotected string.*
/DO NO^{C}T^C K^{C}NO^{C}W^C F^{C}IL^{C}E/	*Protected string.*
THIS/ IS A/WILD' FILE!# ?'	*Combination of protected and unprotected strings.*
N@IL	

Example: Unacceptable File Names

TEST#1	*# must be protected.*
STUDENT SURVEY	*Space must be protected.*
N	*Reserved file name.*
/JJ^{C}LP/	*J^C illegal.*

Comments In File Names

Comments can be appended to file names. The comments feature is especially useful because it allows the user to have brief, easy-to-use file names and still have the contents of the file documented. The file name and the comment are separated by a hyphen. The rules for comments are the same as the rules for naming files.

For example, the files above might have the following comments.

```
TEST-TUBE
/DO NOCTC KCNOCWC FCILCE/-'CONTAINS
                          SPECIAL INFO'
```

The comment does not change the name of the file and is not needed to call the file, but will appear in the directory listings. The comments must be appended when the file is first created. Files can be created by the COPY command, which copies the contents of one file to another; or the RENAME command, which changes the name of a file (refer to Pages 13 and 14, respectively). To append a comment (TUBE) to an existing file name (TEST), use a procedure similar to the following:

–RENAME TEST AS 1↵	*Change the name of the file to 1.*

– RENAME 1 AS TEST-TUBE ↵

Change the name of 1 back to the old name with the comment appended.

NOTE: The command

– RENAME TEST AS TEST-TUBE ↵

is invalid since it tries to rename a file as itself.

Reserved File Names

Any of the reserved file names which are a subset of TELETYPE can be used with the EXECUTIVE commands to indicate the terminal (see Rule 3 for naming files above). For example, to enter text from the terminal to be stored on the file RHDA, type

– COPY T TO RHDA-/HIGGIN'S DATA/ ↵
NEW FILE ↵

Then enter information. When finished, type a D^C. Please see the discussion of the COPY command on Page 13 for a full description of this procedure.

Any of the reserved file names which are a subset of NOTHING can be used with the EXECUTIVE commands to indicate a null file. To create a completely empty file, use the command

– COPY NOTHING TO file name ↵

The command

– COPY file name TO NOTHING ↵

causes no action.

When a command is given in a language to write on NOTHING (or its subset), the WRITE command is executed but there is no file output. For example, if a SUPER BASIC program allows the user to specify an output file, the user can specify NOTHING to suppress file output. Likewise, he can specify TELETYPE (or its subset) to print the output on the terminal instead of a file if the output device is specified for sequential rather than random output.

Calling Files

The user has complete control over the files in his directory. In addition, he can access another user's files if the other user permits. Therefore, files can be called from the user's own directory, from another directory in the same account, or from a directory in another account.

A file in the user's own directory can be used in a command by typing just the name of the file with or without any comment. For example, if the file is

MTDB-AUGUST

it can be designated in an EXECUTIVE command as either MTDB or MTDB-AUGUST. In fact, any comment can be given after the file name. The comments are ignored. For example,

MTDB-'RANDOM COMMENT'

would also designate the file MTDB.

In other words, a file is designated by its file name. Therefore, in any directory there cannot be two files with the same name. For example, MTDB-AUGUST and MTDB-JULY cannot exist in the same directory.

The contents of a file in another user name in the same account can be accessed if it has been declared public (see the DECLARE command on Page 17). The contents of a file in another account can be accessed if the file name contains the character @ or a legal control character. The file name is designated by including the user name in parentheses before the file name. For example,

(MILLER)@NEWS

(LEPAGE)/J.L./

(ARMISTEAD)RA

If the user named SMITH has a public file named NORTH, the other users in his account can obtain the contents of the file by typing

– COPY (SMITH)NORTH TO NORTH ↵

Copies the contents of the file named NORTH in user name SMITH to a file named NORTH in this user name.

NOTE: A user can change the contents of another user's file only if it has been declared PUBLIC and WRITE ACCESS (see the DECLARE command on Page 17). Therefore, a file can be shared, but its integrity will be maintained.

UNLIMITED FILE DIRECTORY

There is no limit to the number of files in a user's directory. However, for the user's protection, there is a limit to the number of new file names that may be created after each log in. This limit has been imposed because, in most Tymshare languages, it is possible to write programs which create files. Through a programming error, a user could create many, many files and not even know it. If a file directory becomes full, simply log out and log in again; more room on the file directory is automatically created.

When the message that the directory is full appears, the file the user is attempting to write must be saved on an old file. It is advantageous to have a scratch file that can be used as temporary storage. If this scratch file is given the name /$/, it can double as the fail-safe file in case of premature disconnect. (See Page 5 for a discussion of a fail-safe file.) Likewise, if the fail-safe controls have been set, the file /$/ is available as a scratch file.

NOTE: Deleting files will not create more room in the file directory.

The number of new files that can be created after each log in depends on the length of the file names. If the file names are short (three characters or less), the maximum number of new files is more than forty.

THE COPY COMMAND

The COPY command is probably the most frequently used EXECUTIVE command. It can be used to create files, to list the contents of files, and to copy the contents of one file to another. The general form of the command is

COPY source TO destination

NOTE: All the words must be separated by spaces.

The source and destination may be file names or the reserved names T, TE, etc., and N, NO, etc. The COPY command causes the contents of the source to be duplicated in the destination but does not affect the contents of the source.

If the destination is a file, a check is made to see if a file by that name already exists. The system types either

OLD FILE

if the file name already exists, or

NEW FILE

if a new file name is being created, and waits for the user either to confirm or to abort the command. The command is confirmed simply by typing a Carriage Return. If the command is confirmed, the contents of an old file are completely erased and replaced by the contents of the source. To save the contents of the old file, the command can be aborted by typing the ALT MODE.

In the following example, the user wants a copy of a file in user name ASHBY. The EXECUTIVE warns that this command will erase the present contents of ACCT, so the user types an ALT MODE to abort the command and copies ACCT3 to a different file (ACC).

–COPY (ASHBY)ACCT3 TO ACCT ↵
OLD FILE ⊕

–COPY (ASHBY)ACCT3 TO ACC ↵
NEW FILE ↵

If the user types an N and a Carriage Return after OLD FILE/NEW FILE, the system will ask for another file. For example,

–COPY D1 TO MOD1 ↵
OLD FILEN ↵
ERROR, TYPE: TO MOD2 ↵
NEW FILE ↵

–

COPY can be shortened to the first three letters. Furthermore, the TO can be replaced by a comma. Any of the following forms of COPY will perform the same function.

COPY JEAN TO JN

COP JEAN TO JN

COPY JEAN,JN

COP JEAN,JN

Creating A File

To enter information from the terminal to a file, the form of the command is

COPY T TO file name ↵

Again, the EXECUTIVE responds with OLD FILE or NEW FILE. Type a Carriage Return and then the text or data. The text or data will replace any previous contents of the file. After the entire text is entered, type a Control D.

Example

–COPY T TO MOON ↵
NEW FILE ↵

.02, 3.8
.04, 5.2
.
.
.
D^C

–COP T,PELTZER ↵
NEW FILE ↵

THE MOST RECENT...

.
.
.
D^{C}

–

As discussed under *File Security Controls* on Page 17, it is possible to protect a file from being erased. If a protected file is given as a destination in a COPY command, the error message

ERROR, TYPE: TO

will print. Simply enter a new destination.

A user can create a file in the programming languages or in Tymshare's EDITOR. In EDITOR, more than twenty different control characters are available to make creating files easier. Any Tymshare representative has copies of the EDITOR Reference Manual.

Listing A File

The form of the command to list the contents of a file on the terminal is:

– COPY file name TO T ↵

Some files are written in machine code (binary files) and cannot be listed on the terminal.

Examples

– COPY (RUBIN)RP TO T ↵
– COPY MTEST,T ↵

NOTE: The printing of a file can be interrupted at any time by typing an ALT MODE.

THE DELETE COMMAND

A file can be deleted from the file directory with the DELETE command. It is economical to delete files when they are no longer needed. The form of the command is

– DELETE file name ↵

There must be a space after the word DELETE. Several files may be deleted with one command; the file names must be separated by commas or spaces, such as

– DELETE TEXT,NEWS ↵

The list of files to be deleted can be extended to the next line by typing only a Line Feed or both a space and a Carriage Return.

The files are not actually deleted until a Carriage Return or Line Feed is typed. Therefore, the command can be aborted with an ALT MODE any time before the Carriage Return or Line Feed. However, if the command uses several lines, the files on each line will be deleted when the Line Feed or Carriage Return ending that line is typed.

If several names are listed and there is one which cannot be deleted, all the files up to that one will be deleted, but none of the files after it. For example,

– DELETE A,B,1,C,2 ↵
ERROR ON NAME: 1 ?

Files A and B are deleted but 1, C, and 2 are not.

When deleting many files, it is advantageous to type a Line Feed after each file name. A message will print immediately if a file cannot be deleted.

THE REMOVE COMMAND

REMOVE is most frequently used to delete file names containing forgotten non-printing control characters. The REMOVE command deletes files by number rather than by name. The number is the file's position in the file directory. The numbers are obtained with the DIRECTORY command (see Page 19 for a description of the DIRECTORY command). For example,

```
– DIRECTORY ↵
  #  PVT  PUB  TYP  DATE   USE  SIZE  NAME
  1  R/W  NO   SYM  10-1    15  3586  LMM1
  2  R/W  NO   SYM  10-6     1  1536  TEMP
  3  R/W  NO   SYM  10-19   21  4768  LMM2
– REMOVE 2 ↵

– DIR ↵
  #  PVT  PUB  TYP  DATE   USE  SIZE  NAME
  1  R/W  NO   SYM  10-1    15  3586  LMM1
  2  R/W  NO   SYM  10-19   21  4768  LMM2
–
```

NOTE: When file 2 was removed, file 3 became file 2.

The REMOVE command is more restricted than the DELETE command. Only one file can be removed at a time. If a user has used GFD[1] to access another file directory, he cannot use REMOVE in that directory.

THE RENAME COMMAND

To change the name of a file, use the RENAME command as follows:

– RENAME oldfilename AS newfilename ↵

NOTE: There must be at least one space between each word.

1 - GFD is described on Page 22.

A common use of this command is to rename files so they may be shared with other users.

Example

```
–RENAME FOR2 AS @FOR2↵
–
```

RENAME cannot change a file name to a name that is already in use. For example, suppose a file named JUNEDATA is in the file directory.

```
–RENAME JDATE AS JUNEDATA↵
ERROR, TYPE: NEW NAME: JULYDATA↵
```

The name JUNEDATA was already taken.

The RENAME command has a shortened form: the AS can be replaced by a comma, and RENAME can be shortened to REN.

RENAME is useful to create file names with comments attached. For example, to attach a comment to a file named XB, type

```
–RENAME XB AS 1↵
–RENAME 1 AS XB-'UPDATE PROGRAM'↵
```

EMERGENCY TERMINATION

Occasionally, it is necessary to leave the terminal in the middle of a job. The DUMP and RUN commands save all the data and the work done so that the user may continue at a later time. The DUMP command saves the necessary information on a file and the RUN command recovers it.

The DUMP Command

The DUMP command takes the form

```
–DUMP file name↵
  NEW FILE↵
```

or

```
  OLD FILE↵
```

–

To save work done in a language, the user uses QUIT to return to the EXECUTIVE and then uses the DUMP command. With this command, the user stores on a file the language he was working in, his program and data, and whatever work he had done thus far. Any of the commands that do not erase user memory can be used between the exit from the language and the DUMP command. These commands are listed on Page 9.

The DUMP command cannot be used to save work in Tymshare Library programs or GO files since the memory for these is erased when the user returns to the EXECUTIVE.

CAUTION: Due to continual development of the Tymshare languages, improved versions are frequently released. DUMP files may not be compatible from one version of a language to another. Therefore, DUMP files should not be used for permanent storage.

The RUN Command

When the user returns to the terminal, he gives the command

```
–RUN file name↵
```

to load the contents of the DUMP file back into the system. He is returned automatically to the language he was in before he gave the DUMP command, and may continue from where he left off, as shown in the following example.

```
–SBA↵                          Call SUPER BASIC.
>LOAD MAT↵
>RUN↵
⊕                              While calculations are in
                               process, hit ALT MODE.
INTERRUPTED IN 1845
>QUIT↵

–DUMP MATSBA↵                  Saves calculations on
  NEW FILE↵                    MATSBA.

–LOG↵
```

Some time later, log in again

```
–RUN MATSBA↵
SBASIC
>GO TO 1845↵
.
.
.                              Calculation continues.
```

The RECOVER Command

The RECOVER command restores a DUMP file. A RECOVER followed by a REENTER or CONTINUE has the same effect as RUN.

```
–REC /$/↵          –REC /$/↵
–REENTER↵          –CONTINUE↵
SBASIC             SBASIC
>                  >
```

SECTION 4
THE FILE DIRECTORY

The EXECUTIVE has commands to set the security controls on individual files and on the entire file directory. These controls determine who can access the files and the directory, who can read them, and who can write on them. These controls provide a maximum of security.

This section discusses how to list information about the files and about the directory, and how to set the controls.

FILE SECURITY CONTROLS

The DECLARE command is used to set the file security controls. DECLARE asks two sets of questions: PRIVATE and PUBLIC. These questions and the effect of the user's responses are listed in the chart below. The private controls refer to what the user himself can or cannot do to his file. The other file access controls refer to sharing files with other users in the same account.[1]

–DECLARE ↵
FILE(S): file names ↵

or

–DECLARE file names ↵

The user specifies the file or group of files which he wishes to declare. After the last file name he types a Line Feed or Carriage Return. EXECUTIVE then asks questions which can be answered Y or N **followed by a Line Feed.** (The WRITE ACCESS? question can also be answered with an A. See the third example below.)

Example

SQU and K4 are GO files belonging to user Jones (user name JONES). He declares these files to be proprietary as follows:

–DECLARE ↵
FILE(S): SQU,K4 ↵
PRIVATE:
WRITE ACCESS? Y ↵
READ ACCESS? Y ↵
PUBLIC? Y ↵
PUBLIC:
PROPRIETARY? Y ↵
WRITE ACCESS? N ↵

–

EFFECT OF RESPONSES			
Question	**Y (for YES)**	**N (for NO)**	**A (for APPEND)**
PRIVATE: WRITE ACCESS?	The user may write on the file or delete it.	The file can only be read or opened for input; the user may neither write on it nor delete it. The next question is bypassed.	The user may add to the file, but cannot write over existing information. Append-only files can be deleted.
READ ACCESS?	The user may read the file.	The user may not read the file. The file may not be opened for input or loaded into a language.	
PUBLIC?	Other users in the same account may access the file subject to the public controls.	Other users may not copy or use the file unless the file name contains a control character or @. All subsequent questions are bypassed.	

1 - Files can be shared with users in other accounts by including @ or a legal control character in the file name.

EFFECT OF RESPONSES (Continued)		
Question	**Y (for YES)**	**N (for NO)**
PUBLIC: PROPRIETARY?	The file can be executed by other users but cannot be listed or copied. It can be accessed only by the GO command if it is a GO file, or the RUN command if it is a DUMP file[1]. Memory is cleared whenever a return to the EXECUTIVE is made.	No further limitation is placed on copying or using.
WRITE ACCESS?	In addition to being able to copy and use the file, other users in the same account may write on it.	Other users in the account can copy or load this file, but may not write on it or delete it.

The files SQU and K4 may still be read, written on, or deleted by Jones. In addition, however, all the users in the account can use either file by typing:

–GO (JONES)file name↵

No user can copy either file to his own directory.

NOTE: A file with @ or any control character in its name **is always public.** *It can be accessed by any user on the system if he knows the full name of the file. Therefore, to share a file without risking its exposure to everyone on the account, include some non-printing control characters in the file name and inform only those users who are allowed to access the file. Since the control characters do not print, there will never be any written record of the complete file name to jeopardize its security.*

If the Y or N is followed by a Carriage Return rather than a Line Feed, DECLARE will skip the remaining questions and make no further changes in the condition of the file or files. For example, Jones can control his file SEEB so that even he cannot erase it; he can append data to the end of the file only.

–DECLARE SEEB↵
PRIVATE:
WRITE ACCESS? A↵

–

Append-only data files avoid complicated programming when information is repeatedly added to a file.

The inclusive ALL may be used to refer to every file in the user's directory. This allows a company to set aside one user name as an "account library" by declaring all files under that name to be public to the account.

Example

–DECLARE ALL↵
PRIVATE:
WRITE ACCESS? N↵ *The N reply causes EXECUTIVE to bypass the READ ACCESS question. The READ ACCESS status remains unchanged.*
PUBLIC? Y↵
–

NOTE: The user typed a Carriage Return after his last reply to indicate that the remaining questions need not be asked; that is, he did not wish to make all the files proprietary.

Further file security can be obtained with the CIPHER program described on Page 25. CIPHER encodes files so that they can be reclaimed only by someone who knows the key word.

LISTING FILE INFORMATION

The EXECUTIVE has several commands to list file information. LIST is a Tymshare program which gives minimum information about the files (names only); DIRECTORY is a single command to give the maximum information about the files; and FILES can be used to list selected information. In addition, LAST gives the number of files in the directory.

1 - Only GO files and DUMP files may be made proprietary.

The LIST Program

Typing

– LIST ↵

will list all the file names in the file directory beginning with the file most recently created.

The DIRECTORY Command

All the characteristics of the user's files may be listed with the DIRECTORY command. *NOTE: Most recently created files are listed last. The listing can be stopped at any time by typing ALT MODE/ESCAPE.*

Directory lists the following information

1. File number
2. Private controls
3. Public controls
4. Type
5. Date of last write
6. Number of times accessed since created
7. Size in characters
8. Name

Example

– DIRECTORY ↵

#	PVT	PUB	TYP	DATE	USE	SIZE	NAME
1	R/W	YES	SYM	2-16	2	6912	XYZ
2	R	NO	BIN	3-14	1	10240	PGM
3	R	PRP	GO	4-1	5	2304	BL
4	R/W	NO	DUM	4-31	4	1536	/$/
5	R/W	INT	SYM	5-28	11	1536	POHL

The **file number** is simply the position of the file in the file directory. As files are deleted from the directory, the number of files farther down the list will be adjusted. The file number is used with the REMOVE command.

The **private controls** and **public controls** are set by DECLARE. In the PVT column:

R = Read access only.
W = Write access only.
R/W = Read and write access.
AP = Appendable only.
R/A = Appendable with read access.

In the PUB column:

NO = Not public.
YES = Public.
YES,WT = Public with public write access.
PRP = Proprietary.
PRP,WT = Proprietary with public write access.
INT = Initialized (see Page 24).

The file **type** is one of the following:

SYM — Symbolic files contain information in the standard alphanumeric character representation. They may be used as program files or data files.

BIN — Binary files are written in internal machine code. Compiled programs and data may be stored on binary files for security and economy since they usually consume less storage space and are faster to load. The UPL program #BINDUMP can list the contents of binary files.

GO — A GO file is a binary program file which may be executed directly from the EXECUTIVE by typing GO and the file name. See *Running Programs From The EXECUTIVE* on Page 10. A GO file automatically executes the program. GO files may not be copied to the terminal since they are written in machine code.

DUM — A DUMP file is a machine code file created by the DUMP command as discussed on Page 15. A DUMP file can be executed with the RUN command.

BAD — If a file is ever listed as a bad file, some unusual error has rendered the file unusable. Call the Tymshare representative and he will have a good version of the file restored from the backup tapes.

The **date** is the last time the file was written on. If the date is more than a year old, the year will appear in parentheses after the month-day.

The **use** is the number of times the file has been accessed by either reading or writing since it was created.

The **size** is given in characters.

The full **name** of the file is listed with any appended comments.

By typing a Line Feed instead of a Carriage Return after the DIRECTORY command, the user can specify one file or a group of files for which he wants all the characteristics to be listed. For example,

– DIRECTORY ↴
FILE(S): PGM,BL ↵

#	PVT	PUB	TYP	DATE	USE	SIZE	NAME
2	R	NO	BIN	3-14	1	10240	PGM
3	R	PRP	GO	4-1	5	2304	BL

Alternatively, the DIRECTORY command and the file list can be given on one line.

– DIRECTORY PGM,BL ↵

or

– DIR PGM,BL ↵

NOTE: There must be a space after the command.

Another form of DIRECTORY allows the user to list part of the directory.

– DIR,file number ↵

will list the files beginning with the number specified. For example,

– DIR,24 ↵

lists all the files from number 24 to the end.

The FILES Command

FILES can be used to list particular information.

For a quick listing composed only of file names and file types, the FILES command is given as shown below.

```
– FILES ↵
SYM    XYZ
BIN    PGM
GO     BL
DUM    /$/
SYM    POHL
–
```

By typing a Line Feed after the FILES command, the user can specify the quick listing of one file or a group of files. In addition, by typing a Line Feed after the specified file names, he may select which other characteristics besides the name and type that he wishes to be listed. For example,

```
– FILES ↴
FILE(S): PGM,BL ↴
NUMBER? N ↴
SIZE? N ↴
DATE? N ↴
CONTROLS? Y ↴
PVT  PUB  TYP    NAME
 R   NO   BIN    PGM
 R   PRP  GO     BL
–
```

Alternatively, the file list can be typed on the same line as the FILES command.

As shown in this example, the user types Y (for YES) if he wants to list the information in question; otherwise, he replies with N (for NO).

Either a Line Feed or a Carriage Return may be typed after a Y or N reply. A Line Feed causes FILES to ask the next question; a Carriage Return causes it to skip the remaining questions and assume that the user does not want any other information.

Example

```
– FILES XYZ ↴
NUMBER? N ↴
SIZE? N ↴
DATE? Y ↵        The Y reply causes both DATE
                 and USE to be listed.
TYP  DATE  USE  NAME
SYM  2-16    2  XYZ
–
```

The name ALL will list all of the files in the user's directory. Therefore, the user can select information to be given about all of the files by answering

FILE(S): ALL ↴

FILES is the only EXECUTIVE command which can be shortened to only two characters.

FILES has another feature: if the FILES command is followed by a comma and a file number, the information will be listed starting with that file number and continuing to the end.

```
– FI,4 ↴
NUMBER? Y ↵
  #  TYP    NAME
  4  DUM    /$/
  5  SYM    POHL
```

This feature is especially useful with large file directories.

The LAST Command

Typing LAST prints the number of files in the file directory. For example,

```
– LAST ↵          The user determines the number of
                  files in his directory.
  43
– FI,39 ↵
                  The user requests information about
or                the last five files.
– DIR,39 ↵
```

FILE DIRECTORY SECURITY CONTROLS

In addition to the file sharing option, the Tymshare system has powerful file directory sharing options. These options are exercised with the three commands FDC (File Directory Controls), PFDC (Print File Directory Controls), and GFD (Get File Directory). FDC and PFDC set the directory security status and print the present status. GFD accesses another user's file directory.

The file directory controls allow several user names to share the same directory. These controls also allow several user names to add files to a directory such as the account library without having access to the directory.

The File Directory Controls Command (FDC)

Directory access controls are set with the FDC (File Directory Controls) command.

NOTE: The Account Supervisor automatically has access to all directories in his account.

The following options are available with the FDC command:

- SHARABLE? Other users can access the directory with GFD.
- LISTABLE? Other users can list the file directory.
- CONTROLS? Other users can use DECLARE to change the security status of files.
- NEW FILES? Other users can create files in the directory.

FDC asks each of the above options which must be answered Y or N followed by a Line Feed or Carriage Return. A Line Feed asks the next question; a Carriage Return bypasses the remaining questions and makes no further changes in the file directory controls. These questions and the effect of the responses are listed in the table below.

Example

```
– FDC↵
SHARABLE? Y↓
  LISTABLE? Y↓
  CONTROLS? N↓
NEW FILES? N↓
```

Other users can run the programs and list the files and file directory, but they cannot change the public and private controls or create new files.

```
–
```

```
– FDC↵
SHARABLE? N↓
NEW FILES? Y↵
```

Other users can add new files but cannot use the directory. This is a convenient status for an account library.

```
–
```

When a user name is first created, its directory is LISTABLE only; no other user except the Account Supervisor can access it.

EFFECT OF RESPONSES		
Question	**Y (for YES)**	**N (for NO)**
SHARABLE?	All users in the account may access the files in this directory (run programs, list files, etc.). Access is subject to the private controls on the files set by the owner using the DECLARE command.	Only the owner and the Account Supervisor have access to the files in this directory. *NOTE: If the answer is N, the LISTABLE and CONTROLS options are bypassed.*
LISTABLE?	All users who can access this file directory can use the FILES and DIRECTORY commands.	Other users cannot use the FILE or DIRECTORY command to list the directory contents.
CONTROLS?	All users who can access this file directory can reset file controls on files in this directory with the DECLARE command.	Other users cannot use the DECLARE command on any of the files in this directory.
NEW FILES?	Files can be put into this directory by any user in the account.	Files can be put into the directory only by the owner of this directory.

The Print File Directory Controls Command (PFDC)

If a user wishes to know what controls are in effect for his directory, he types PFDC (Print File Directory Controls). All options that were answered with a Y in the FDC command will be listed. If the option is not listed, it was answered N and is not in effect.

Example

```
– FDC ↵
SHARABLE? Y ↵
   LISTABLE? Y ↵
   CONTROLS? N ↵
NEW FILES? Y ↵

– PFDC ↵
SHARABLE LISTABLE
NEW FILES
–
```

Since CONTROLS? was answered by N, it is not shown in the PFDC command.

The Get File Directory Command (GFD)

The GFD command can be used to access other user directories in the same account if the SHARABLE option in FDC has been answered Y by the user whose directory is to be accessed. The form of the command is

– GFD user name ↵

The user now has access to all files in the directory of that user name. To re-access his own files he must GFD back to his own user name.

If a GFD is made to a directory that is sharable but not listable or has protected control, LIST or DECLARE will result in a question mark.

Example

```
– GFD DMM ↵

– PFDC ↵
SHARABLE

– LIST ↵
  ?
– DECLARE ↵
  ?
–
```

GFD allows more than one user to access the same file directory simultaneously. This can avoid excess storage due to maintaining duplicate file directories.

SECTION 5
AUTOMATIC FILE FEATURES

COMMAND FILES

Under normal operation, commands are typed into the system from the terminal. Occasionally, however, it may be advantageous to store a particular sequence of commands on a file using the Command File feature.

The command

– COMMAND file name ↵

causes the system to accept commands from the file as if they had been entered from the terminal.

The Command File feature is convenient whenever

- An elaborate system of programs needs to be linked into one easy-to-use package.
- Clerical functions are performed on the system by employees with no need or desire to learn the Tymshare commands.
- A sequence of commands must be performed repeatedly, such as each time a program is run.

For example, a company keeps a catalog of its publications on a file. The SUPER BASIC program UPDATE adds new entries to the file, but for safety, a copy is made of the data file before each time the program is run. Therefore, the command sequence entered from the terminal is

– COPY CATALOG TO CATDUP ↵
OLD FILE ↵

– SBA ↵

> LOAD UPDATE ↵
> RUN ↵

To perform the same operations, a Command File named ADDITIONS contains

COPY CATALOG TO CATDUP ↵
↵ *Carriage Return to confirm OLD FILE.*
SBA ↵
LOAD UPDATE ↵
RUN ↵
COMMAND T ↵

Then the program can be run by typing

– COMMAND ADDITIONS ↵

The commands in the file will not print on the terminal but any messages from the system will print such as

OLD FILE

The Command File may be created either in the EXECUTIVE (with the **COPY T TO file name ↵** command) or in EDITOR (see the *Tymshare EDITOR Manual, Reference Series*).

The commands in a Command File may be from any Tymshare language and many library programs. Users can also write programs which can be run from Command Files.

The system will take its commands from the file specified in the COMMAND command until one of the following is reached:

1. A COMMAND T command, which causes the system to return to taking commands from the terminal.
2. The end of the Command File, which has the same effect as 1 except that a question mark will print.
3. Another COMMAND command, which enables the user to link as many Command Files as he wishes. *NOTE: Command Files can link to themselves; that is, the last command in the file can be a command to take commands from itself.*
4. An error.

Example

A user wants to delete all but his first five files. He creates a Command File which removes file number 6 and calls itself to remove file number 6 again. All the files are deleted until only the first five remain. Control returns to the terminal when the REMOVE 6 command can no longer be executed.

– COPY T,FILDEL ↵
NEW FILE ↵

The user uses the COPY command to create the file. There are only two statements in the Command File.

REMOVE 6 ↵
COMMAND FILDEL ↵
D^C

D^C terminates the COPY command.

```
-COMMAND FILDEL↵          Control is transferred to
 ?                        FILDEL. Only 5 files re-
                          main.
-
```

INITIALIZED FILES

A Command File may be initialized. This means that control will be transferred to the file immediately after the user logs in. The command to initialize a Command File is

– INIT file name↵

Example

A company has a user name containing its inventory and inventory control programs. This user name is designed so that the control programs are loaded automatically by a Command File called CONTROL.

```
-COPY T TO CONTROL↵
 NEW FILE↵                The user uses the COPY
                          command to create the file.
SBA↵
5 PRINT↵
10 PRINT "UPDATE OR RETRIEVE":↵
20 INPUT A↵               The file calls SUPER BA-
30 IF LEFT(A,1)="U"↴      SIC and types a program
   THEN A="UPDT"↴         that asks what the user
   ELSE A="RETR"↵         wants to do, and then
40 OPEN "CON2",↴          writes a second Command
   OUTPUT,1↵              File which loads the appro-
50 PRINT ON 1:↴           priate program.
   "GO "+A↵
55 PRINT ON 1:"COM T"↵
60 CLOSE 1↵
RUN↵                      The file runs the short pro-
QUIT↵                     gram and transfers control
COMMAND CON2↵             to a second Command File.
D^C
-INIT CONTROL↵            The user initializes the file.
```

Then, when a user logs in, the following occurs

```
PLEASE LOG IN: INVENTORY;;ENYEDY↵
TYMSHARE  8/22  9:18
UPDATE OR RETRIEVE? R↵
INVENTORY INFORMATION RETRIEVAL
                                PROGRAM
```

An initialized file will continue to execute automatically each time the user logs in or until the file is deinitialized by giving the INIT command again. If an INIT file is deleted without deinitializing it, and another INIT file is not created, a question mark will occur every time the user logs in. To avoid this question mark, initialize and deinitialize any file by giving the same INIT command twice.

Example

```
-INIT A↵
-INIT A↵
-
```

If the INIT command is given to a file that is already initialized, the file will once again be an ordinary file. If the INIT command is given to another file name (after one file has already been initialized), both files will remain initialized; the file which was last given the INIT command will execute first after the user logs in.

Initialized Command Files are listed as INT under the PUB heading in the file directory listing.

When a Tymshare Library Command File or a file from another user name is initialized, a dummy file is set up in the user's directory with the same name as the file, enclosed in quote marks, and containing no words.

A Command File from another user's directory may be initialized to execute after one's own LOG IN as follows:

– INIT (user name)file name↵

GO and DUMP files as well as Command Files can be initialized. If a GO file is initialized, the program will start running immediately after logging in. If a DUMP file is initialized, the file is automatically recovered and transfer is made to the language, but the program does not execute automatically.

Normally, it is possible to stop initialized files by hitting an ALT MODE/ESCAPE. SUPER BASIC and SAVE type[1] GO files can be initialized so that it is impossible to interrupt them. These files are initialized by giving the INIT command followed by a Line Feed and then the file name.

```
-INIT↴
PROTEX↵
-
```

NOTE: A file initialized with a Line Feed must be deinitialized with a Line Feed.

1 - See the SAVE command on Page 32.

SECTION 6
PROGRAMS FOR FILE MANIPULATION

This section describes several Tymshare Library programs. CUC reduces the size of files to save storage charges; CIPHER encodes files so they cannot be interpreted by anyone else; and CHECKSUM, VERIF, and COMPARE can be used to determine if a file has been changed.

CRUNCH AND UNCRUNCH

The crunch and uncrunch program, CUC, reduces storage charges for symbolic files that are not used often. CUC crunches symbolic files into smaller binary files. These binary files cannot be understood by the system. **The binary files must be uncrunched before they can be used.** The program is called by typing

```
– CUC ↵
CUC READY
!
```

Any of the commands listed below are acceptable after the exclamation mark. Type only the first character of the command; CUC will type the rest. See the example below.

Command	Full Text	Action
C	CRUNCH	Reads symbolic file to be crunched.
P	PRINT	Lists uncrunched version of file.
R	READ	Reads crunched file to be uncrunched.
U	UNCRUNCH	Writes uncrunched symbolic file.
W	WRITE	Writes crunched binary file.

To crunch a file, give the CRUNCH command and then the WRITE command. For example, to crunch a file named TEXT and create the binary file SMALTEXT, proceed as follows:

```
– CUC ↵
CUC READY
! CRUNCH FROM FILE: TEXT ↵
FILE COMPRESSED TO 74%
! WRITE ONTO FILE: SMALTEXT ↵
  NEW FILE ↵
832 WORDS WRITTEN.
!
```

NOTE: The crunched and uncrunched files cannot have the same name. After each CUC run, the old version of the file can be deleted with the DELETE command.

To uncrunch a crunched binary file, use the READ command followed by the UNCRUNCH command.

```
– CUC ↵
CUC READY
! READ FROM FILE: SMALTEXT ↵
 (LAST FILE)
 RETURN                    CUC prints the last line
                           of the file; in this case,
                           RETURN.
! UNCRUNCH ONTO FILE: TEXT ↵
 NEW FILE ↵
3372 CHARACTERS WRITTEN
!
```

CIPHER

With the CIPHER program, a user can encode a symbolic file into a form which no one can decode without the key word. An encoded file can be stored on magnetic tape or paper tape[1], or in a public file with complete security. When the user wishes to use the file again, he simply calls CIPHER and gives the key word.

Example

```
– CIPHER ↵                 Call CIPHER.

ENTER KEY: (user types non-printing key) ↵
                           The key does not print so
                           there is no written record.
TYPE C TO CIPHER, U TO UNCIPHER,
                                    C.R.: C ↵
                           The user wants to encode.
INPUT FILE: SECRET ↵       The symbolic file SECRET
                           is to be encoded.
OUTPUT FILE: TERCES ↵
```

1 - The library program BINTAPE can be used to punch a binary tape.

```
 NEW FILE ↵

END OF JOB

– DELETE SECRET ↵        The user deletes the sym-
                         bolic file.
```

To decode the file, proceed as follows:

```
– CIPHER ↵

ENTER KEY ↵              Enter the same key.
TYPE C TO CIPHER, U TO UNCIPHER,
                                   C.R.: U ↵
INPUT FILE:
TERCES ↵
OUTPUT FILE:
SECRET ↵
  NEW FILE ↵

END OF JOB

–
```

CHECKSUM AND VERIF

A checksum is a unique number associated with a particular file content. The checksum is calculated by adding the binary equivalent of all the words on the file. If a file is copied, renamed, or transferred from one user to another, the checksum should remain the same. If the contents of a file are changed in any way, the checksum will change. Therefore, checksums are useful to keep track of various versions of files and to ensure that a file has not been altered in any way.

File checksums can be obtained with CHECKSUM or VERIF. In CHECKSUM, the checksums for individual files can be listed. VERIF gives the checksums for a range of files.

CHECKSUM has the following form:

```
– CHECKSUM ↵          Call CHECKSUM.

INPUT: INVTRK ↵       Give file name.
11325422              CHECKSUM prints the
                      checksum.
INPUT: . ↵            A period returns the user
                      to the EXECUTIVE.
–
```

VERIF has the form:

```
– VERIF ↵

BEG.NO.: 8 ↵          Give the number of the
END NO.: 10 ↵         first file in the range fol-
                      lowed by Line Feed. Give
                      the number of the last file
                      in the range.
10/06  10:33          VERIF prints the date and
                      time.
 8  BIN  SMALTEXT-75766264
 9  BIN  TERCES-46216273
10  SYM  SECRET-57473105

–
```

If the beginning number is followed by a Carriage Return, the ending number is assumed to be the last file number. *NOTE: The checksum is not related to the size of the file.*

COMPARE

The COMPARE program compares two files word for word and lists the content and location of any words which are not exactly the same. This command is especially useful in finding changes made in an ARPAS program.

Example

```
– COMPARE ↵
AAPPLE ↵              The user gives the first file
+0 ↵                  name, APPLE, and starting
                      location 0; then the second
BPEAR ↵               file, PEAR, and starting loca-
+0 ↵                  tion 0.

D 2 10622155 10621555
E 10                  D2 means there is a discrep-
–                     ancy in the second word proc-
                      essed. Then COMPARE lists
                      the contents of the word in
                      each file. E10 means that the
                      end of the file was reached
                      after eight words were com-
                      pared.
```

SECTION 7
UTILITY COMMANDS

DOCUMENTING A SESSION AT THE TERMINAL

Six commands are available to document a session at the terminal: DATE, TIME, SUMMARY, SYSNO, WHO, and " (double quote marks).

The DATE Command

The DATE command prints the date and the time of day.

```
-DATE ↵
3/7 10:32
-
```

The TIME Command

The TIME command prints the computer time used since the user logged in.

```
-TIME ↵
CPU TIME: 17 SECS.
TERMINAL TIME: 0:20:13

-
```

The SUMMARY Program

This program prints the user's total file storage in characters. For example,

```
-SUMMARY ↵
TOTAL STORAGE: 42400
-
```

The SYSNO Command

The SYSNO command prints the computer location and number, the disk number, the system monitor number, and the EXECUTIVE version. For example,

```
-SYSNO ↵

  L3 , DISC 2, SYS. J 28.08-31.01
-
```

location number — *disk no.* — *monitor number* — *EXEC version*

L = Los Angeles
C = Cupertino
N = New Jersey

The Tymshare analyst may use this information to help users with problems.

The WHO Command

The WHO command prints the user name and line connect number.

Example

```
-WHO ↵
16* MACKAL

-
```

If the user uses GFD to access another directory, both user names are listed.

Example

```
-GFD BRET ↵

-WHO ↵
16* MACKAL
 BRET

-
```

Comments (")

This command lets the user type comments on the work he is doing. The form of the command is:

```
- "commentsD^C
```

or

```
- "comments" ↵
```

Example

```
-"WE WILL NOW CALL SBA AND LOAD ↵
A PROGRAM WHICH CALCULATES THE ↵
AREA OF A TRIANGLE. D^C
-
```

The comments are particularly useful in Command Files to notify the user of the progress of the Command File.

THE TYMSHARE MAILBOX

The Mailbox feature lets users communicate with one another through the Tymshare terminal. Mail is

sent to other users with the SEND command and received by the MAIL command. The status of mail is checked by the POSTMAN program.

Sending A Letter

To send a letter to another user, type the SEND command as follows:

– SEND user name ↵

TYPE LETTER: *As many as 240 characters may be typed in one letter.*
.
.
.
D^C
MAIL I.D. NO. n *The system assigns an I.D. number to the letter.*
–

Any character typed in error in a letter may be deleted with A^C. When A^C is typed, a backarrow (←) prints on the terminal and the preceding character is deleted. Repeated use of A^C deletes several characters.

In the following example, Smith sends a letter to Jones and uses A^C twice to delete two erroneous characters.

– SEND JONES ↵

TYPE LETTER:

3/15/69-RECIEA^C←A^C←EIVED YOUR ↵
SUGGESTIONS. WILL DISCUSS ON ↵
MONDAY. ↵
A.SMITH ↵
D^C
MAIL I.D. NO.: 5

–

A letter may be sent to more than one user using the Line Feed as follows:

– SEND JONES ↴
BROWN ↴
MIDDLEMAN ↵

TYPE LETTER:
.
.
.
D^C

Receiving A Letter

The user is notified of a letter in his mailbox by the message MAIL WAITING. To read his mail, the user must give the MAIL command.

Example

User Jones logs in and immediately receives the message:

MAIL WAITING

He reads his mail as follows:

– MAIL ↵

FROM SMITH
3/15/69-RECEIVED YOUR
SUGGESTIONS. WILL DISCUSS ON
MONDAY.
A. SMITH

–

The letter is no longer in Jones' mailbox. If he types MAIL again, the message NO MAIL will print.

Checking The Status Of Letters

To assure the reception of important messages, a program has been written that allows the user to determine if his messages have been received.

The POSTMAN program has four options:

1. COUNT

   Provides the letter I.D. number, the number of addressees who have not received the letters, and the creation date and time.

   Example

   – POSTMAN ↵

   OPTION? COUNT ↵

I.D. NO.:	COUNT	CREATION DATE
15	1	10/6 10:37

   END JOB

   OPTION?

   The user has requested a count of the letters he has sent but which are still waiting to be delivered. This allows him to determine if his messages have been received.

2. COPY

   Copies to the terminal any letter created under that user name.

   Example

   OPTION? COPY ↵

   ENTER MAIL I.D. NO.: 15 ↵

   THERE WILL BE A MEETING OF ALL PERSONNEL AT 16:30- -BRIAN CHOATE

   OPTION?

3. CANCEL

   Cancels any letter. The letter must have originated under your user name.

   Example

   OPTION? CANCEL ↵

   ENTER MAIL I.D. NO.: 15 ↵

   CANCELLED.

   OPTION?

   The user now has cancelled letter number 15.

4. QUIT

   Returns to the EXECUTIVE.

   Example

   OPTION? QUIT ↵

   –

TIME LIMIT

The user can request the system to notify him after a given number of CPU seconds has been used. The time limit is set by the LIMIT command

– LIMIT number of seconds ↵

If the time limit is exceeded while in the EXECUTIVE, the message

CPU TIME LIMIT EXCEEDED WHILE IN EXEC

is printed. If the time limit is exceeded while in program control, the following message is printed:

YOU HAVE EXCEEDED YOUR CPU TIME LIMIT. n CHARS. WERE REMOVED FROM YOUR INPUT BUFFER. DO YOU WISH ANOTHER INCREMENT OF CPU TIME?

Answer NO ↵ or N ↵ if another limit is not desired. Answer YES ↵, Y ↵, or the new limit in seconds to set a new limit. An answer of Y or YES will cause the system to request the new time limit.

Example

– LIMIT 1 ↵
– SBA ↵
> 10 FOR J=1 TO N ↵
.
.
.
> 40 NEXT I,

YOU HAVE EXCEEDED YOUR CPU TIME LIMIT. 10 CHARS. WERE REMOVED FROM YOUR INPUT BUFFER. DO YOU WISH ANOTHER INCREMENT OF CPU TIME?YES ↵

WHAT IS THE ADDITIONAL CPU TIME DESIRED? 4 ↵

In this example, the user set a limit of 1 second. This limit was reached while he was typing a line into SUPER BASIC. The system notified him that ten of the characters in the line were accepted before the limit was reached. He set a new limit of four CPU seconds and continued.

The limit feature never permanently interrupts processing. After the message has been printed and the questions answered, processing continues from the point at which the limit was reached whether or not a new limit is set.

In the example below, a program was running when the time limit was reached. The user did not want a new limit and the program continued.

Example

– SBA ↵
> LOAD COUNT ↵
> RUN ↵
1

YOU HAVE EXCEEDED YOUR CPU TIME LIMIT. DO YOU WISH ANOTHER INCREMENT OF CPU TIME?NO ↵

2 3 4

USER ASSISTANCE

HELP is a library program which lists on the terminal all the EXECUTIVE commands and programs

documented in this manual together with a brief description of each. If you forget the exact name of a command or program, this command can help refresh your memory.

The WHY command gives explanations of errors made in the EXECUTIVE. Whenever the EXECUTIVE responds with an error message, the user can type WHY. The EXECUTIVE will explain the error. For example, the user tries to copy a binary file to the terminal:

```
-COPY LINUM TO T↵
ERROR, TYPE: TO ⊕
-WHY↵

FILE TYPE WRONG (USUALLY FOR
                 TELETYPE)
-
```

The user tries to write on a write protected file:

```
-COPY MOD1 TO D1↵
  OLD FILE↵
ERROR, TYPE: TO ⊕
-WHY↵

FILE NOT PRIVATE WRITABLE

-
```

PAPER TAPE

The Tymshare system has outstanding paper tape handling features. The system has programs to read and punch both binary and symbolic tapes, and to convert character codes for either input or output.

Symbolic paper tape is read and punched with the TAPE program. TAPE has the punching options of punching a title and including a terminating D^C at the end of the tape. TAPE has the reading option of suppressing the echo so that the contents of the tape do not print on the terminal.

To punch a paper tape, call and execute the TAPE program. Then give the file name as the input and T (for terminal) as the output.

Example

-TAPE↵	*Call the paper tape program by typing TAPE in the EXECUTIVE.*
:RUN↵	*Execute the program.*
INPUT FROM RATES↵	*Name the data file to be punched.*
OUTPUT TO T↵	*T indicates the paper tape punch.*
TITLE: INTEREST RATES↵	*This will cause the words INTEREST RATES to be punched at the beginning of the tape as identification.*
CONTROL D AT END? YES↵	*A D^C will be punched at the end of the tape as a terminating character.*

```
TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.
```

Punching begins as soon as the Carriage Return is typed.

The procedure for reading a paper tape is very similar except that the input is from T and the output is to a file. In the following example, a paper tape is read and its contents stored in a file named AFILE. Before reading the tape, position it in the paper tape reader on the leader, or after the title if there is a title.

-TAPE↵	*Call the paper tape program by typing TAPE in the EXECUTIVE.*
:RUN↵	*Execute the Program.*
INPUT FROM T↵	*T indicates the paper tape reader.*
OUTPUT TO AFILE↵ **NEW FILE↵**	*The data file read from the paper tape will be stored in the file named AFILE.*
ECHO? NO↵	*The contents of the tape will not be typed on the terminal as the tape is read.*
TURN ON READER.	*The paper tape is read. Reading terminates when the D^C at the end of the tape is encountered.*

```
NUMBER OF CHARACTERS WRITTEN IS 1728

:
```

NOTE: In the examples above, the storage files are assumed to be symbolic.

Other useful tape programs are BINTAPE, CONFILE, and CONTAPE. BINTAPE punches and reads binary tapes; CONFILE will read a file, convert it to another code, and punch the new code; and CONTAPE will read a tape, convert the code, and write it on a file. Both CONFILE and CONTAPE require a code conversion table which can be created with the TABLEMAKER program.

All of the tape programs are documented in the Tymshare Paper Tape Package Manual.

SECTION 8

FEATURES FOR ARPAS AND DDT USERS

RUNNING A DEBUGGED PROGRAM

The SAVE command creates a GO file. The form of the command is

– SAVE first loc TO last loc ON file name ↵

SAVE will respond with the OLD FILE or NEW FILE message.

If either the command or the OLD FILE/NEW FILE is terminated by a Line Feed rather than a Carriage Return, SAVE will ask

STARTING LOCATION

Enter here the starting location of the program when it is placed again in core. The starting location must be followed by a Carriage Return.

GO files can be saved so that they remain open when they are loaded. This allows the user to append data to the end of the program file.

To save a GO file so it will remain open, set the sign bit when you enter the starting address. This is done by putting 4000 in front of the starting address.

– SAVE 240 TO 307 ON PRESNI ↵
NEW FILE ↵
STARTING LOCATION 4000240 ↵

To restore a program that has been saved on a file but without calling DDT, type

– PLACE file name ↵

If a starting location was specified when the program was saved, the program can be run without calling DDT by using the BRANCH command.

– BRANCH starting location ↵

If a program is not yet debugged, you will probably want to call DDT and take advantage of its debugging capabilities.

COMMANDS TO DETERMINE MEMORY ALLOCATION

Command	Information
MEMORY	Number of words of unused memory.
STATUS	Status of used bytes.
PMT	Status of all bytes allocated to user.

COMMANDS TO RELEASE MEMORY

Command	Action
RELEASE	Releases the subsystem.
KILL	Kills program relabeling.
RESET	Returns all of user's memory.

INDEX

NOTE: Page numbers which appear in **bold** *face type refer to those pages where the listed item receives the most detailed discussion.*