GH20-0860-0

Type III Class A Program

IBM

# Control Program-67/Cambridge Monitor System (CP-67/CMS) Version 3
# Program Number 360D-05.2.005
# CMS Script User's Manual

This manual introduces you to the Script manuscript facility that operates under the Control Program-67/Cambridge Monitor System (CP-67/CMS). Script enables you to type in a manuscript, correct it, examine your corrections, and print it out — all at a typewriter terminal — with spacing performed automatically by the system. The spacing feature enables you to type in text lines without regard to margins; your text is automatically formatted to fit within the margin size that you specify, as in book and newspaper printing. Other capabilities of Script include page numbering, page heading, tab setting, indentation, centering, margin setting (top and bottom of page, as well as right and left sides), double spacing, page skipping, line breaking, and several special features, such as entering lines from the terminal during manuscript printout. The hard copy, or finished product, can be outputted to either the typewriter terminal or offline printer.

Sample documents are provided for practice in creating, editing, and printing a Script file.

The procedure for logging in from CP-67 (the Control Program) to CMS, and document maintenance within CMS are explained.

The Edit facility in CMS is described in some detail.

## PREFACE

The following document is referenced in this manual:

CP-67/CMS User's Guide GH20-0859-0

The following documents provide further information on the CP-67/CMS system:

CP-67 Operator's Guide GH20-0856-0

CP-67/CMS Installation Guide GH20-0857-0

CP-67/CMS System Description Manual GH20-0802-1

CP-67 Program Logic Manual GY20-0590-0

CMS Program Logic Manual GY20-0591-0

CP-67/CMS Hardware Maintainability Guide GH20-0858-0

First Edition (October 1970)

This manual is a minor revision of the IBM Cambridge Scientific Center Report entitled *Script User's Manual* (320-2053).

This Type III Program performs functions that may be fundamental to the operation and maintenance of a system.

It has not been subjected to formal test by IBM.

Until the program is reclassified, IBM will provide for it: (a) Central Programming Service, including design error correction and automatic distribution of corrections; and (b) FE Programming Service, including design error verification, APAR documentation and submission, and application of Program Temporary Fixes or development of an emergency bypass when required. IBM does not guarantee service results or represent or warrant that all errors will be corrected.

You are expected to make the final evaluation as to the usefulness of this program in your own environment.

THE FOREGOING IS IN LIEU OF ALL WARRANTIES EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This edition applies to Version 3, Modification Level 0, of Control Program-67/Cambridge Monitor System (360D-05.2.005) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (GN20-0360) for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Technical Publications Department, 112 East Post Road, White Plains, New York 10601.

CONTENTS

# INTRODUCTION

## FEATURES OF SCRIPT

"Oh nuts! I have to type this whole page over". That complaint is probably heard many times in the course of preparing manuscripts--reports, letters, minutes, and so on. Then there is the problem of making duplicate copies--frequently duplicate original copies--of a typed manuscript. When we use an ordinary typewriter, text-processing, as this is called, can be a time-consuming and harassing job. The Script facility, which operates under the Cambridge Monitor System (CMS), was written to handle these procedures for you; you merely type in the first version and make any necessary corrections. By using the "commands" that you type in with your lines of text, the computer prints out as many "first" copies as you wish, with margins, spacing, indentation, etc. performed in accordance with your commands.

One of the most useful features of Script is right-margin justification, as in book and newspaper printing. This means that your text is spaced as evenly as possible between the margins of your printed page, filling in with blanks where necessary. The printed line is under control of a "line-length" command; any short line that you type in will "grab" characters from a longer line above or below it, and fill out the line with blanks where necessary to avoid splitting words at the margins. This is the ordinary (or default) mode under Script. You can, however, have lines printed out exactly as you type them in, with no justification performed by Script, by including a command that instructs the computer to turn off the format mode. This is useful for typing figures and charts. Line justification can later be respecified if desired.

Other capabilities of Script include: page numbering; page heading; tab setting; indentation; centering; top, bottom, right, and left margin settings; double spacing; page skipping; line breaking (preventing character-grabbing by short lines in format mode); and several special commands that will be discussed later.

Your two primary tools are the Edit and Script commands. Edit commands are used to create and modify the source data for your document. Script commands operate on this source data to produce a finished document. Some additional CMS commands that may be useful to you will be described later.

## YOUR TYPEWRITER TERMINAL

Your communication with the computer is by means of an IBM 2741 Communications Terminal, similar to an IBM Selectric (R) typewriter. You will find a switch on the left side of the cabinet in which the typewriter is mounted; when the switch is set to LCL you have an ordinary Selectric typewriter; by switching to COM you can be linked to the central computer by telephone line, and your typewriter is now a typewriter "terminal". In this latter mode, you and the computer "converse" through messages that you type via your terminal to the computer, and responses you receive back at your terminal.

## PREPARING TO LOG IN

In order to log in, that is, to establish a connection with the system from your terminal, you need an ID and a password, which you can obtain from the operations group at your installation. To prepare your terminal, set the left margin at 1 and the right margin at 130, in order to provide maximum typing width. (You should have a #963 typing element for an EBCDIC terminal, and a #015 element for a correspondence terminal.) Make sure the switch on the left side of the cabinet is on COM.

## LOGGING IN

Procedure 1. If your terminal has a direct wire connection, simply press the ON button. If there is no response, hit ATTN. The system types

```
#
cp-67 online   xdh65 qsyosu
```

Hit the ATTN key.

Procedure 2. If your terminal has a telephone line connection, proceed as follows:

a) Turn the terminal on.
b) Depress the TALK button on the data set and dial the proper telephone number.
c) When a high-pitched continuous tone is heard on the phone, depress the DATA button on the data set and replace the phone in its holder.

d) After the system types out

```
#
cp-67 online  xdh65 qsyosu
```

e) Hit the ATTN key.

After you have completed either procedure 1 or procedure 2, you are ready to log into the system. Log in by typing

```
login 'ID'
```

For example,

```
login smith
```

Hit RETURN. System responds with

```
ENTER PASSWORD:
```

Enter your password and hit RETURN. (Typeout is suppressed so that no copy of your password appears on the terminal sheet.)

The system may respond with messages giving special instructions for system users. It always respond with

```
READY AT 'time of day' ON 'date'
CP
```

Now that you are logged in, you are ready to access CMS, which contains Script. Type

```
ipl cms
```

and hit RETURN. The system responds with

```
CMS..VERSION n   LEVEL  m
```

The printout at your terminal should look something like

this

```
#
cp-67 online  xdh65 qsyosu

1 smith
ENTER PASSWORD:

CP WILL RUN UNTIL 17:00 AND FROM 18:00 TO 24:00
READY AT 12.07.36 ON 06/20/69
CP

ipl cms
CMS..VERSION 3   LEVEL  0
```

You are now ready to create a file, print an existing file, edit (make changes to) an existing file, etc. You are in the CMS command environment. By entering simple commands (such as edit, printf, etc.) at a console, you can access the full System/360 capability in this environment, as well as create and format Script files.

To distinguish your entries from system responses, it is a good idea to type your commands in lowercase. (Text lines of a Script file should, of course, be typed exactly as you wish them to appear on output.) The system generally types its messages to you in uppercase. For example, you could type

```
login SMITH              (combination of uppercase
                            and lowercase)
```

or

```
LOGIN SMITH              (uppercase only)
```

or simply

```
login smith              (this is recommended)
```

Your command entries appear to the system as uppercase, no matter how you type them in; the practice of using only lowercase for commands is merely a convention to improve the readability of your terminal printout.

Note: Single quotation marks denote that the entry is not a literal entry; it is merely a description of the information to be entered. For instance, 'ID' indicates that some particular userid should be entered--without the quotation marks.

## YOUR SCRIPT FILE

The source data for your document is stored in the computer and is called a Script file. It is composed of records and resides on your permanent disk. Files are identified by a unique combination of filename (of your own choosing), filetype (Script, for all Script files), and filemode (you need not concern yourself with this). Script files consist of text lines and special command words that you type in at your terminal.

## CREATING AND MODIFYING YOUR SCRIPT FILE WITH EDIT

The EDIT facility enables you to create your Script file, make changes to the file, or simply peruse its contents.

There are two modes of operation when using the Edit command: Input and Edit.

When you create a new file, your terminal is placed in Input mode, in which you can type in only text lines and Script command words; you cannot make corrections in Input mode.

To make corrections to a file you have created in Input mode, you must enter Edit mode. Entry to Edit is automatic when you issue the command below for a file that has any entries at all, even only one Script command, or one word of text (or even a blank line, for that matter). In Edit, you can use Edit commands to make revisions to your file; for example, you can locate a word or string of words, change a word string to another word string, go to the next line in your file, delete words from that line, and so on.

You can go back and forth between Edit and Input modes (you will learn how to do this later on), typing in text and Script commands (Script commands all begin with a period in column 1) in Input mode, and correcting your entries in Edit mode.

The same command is used to access both modes:

```
edit 'filename' script
```

If the file is new, you are placed in Input mode. Otherwise, you are placed in Edit mode.

Along with a description of Input and Edit facilities is a sample Script file, which is provided as a practice session (See under "Logging Out"). You should enter this file (Input mode), edit it, using several of the Edit commands, and print it out. (It will be under format control of the Script commands included with your input lines.) It is important to familiarize yourself with the system by means of a practice session at the terminal; in no other way can

you gain the facility that you will need later on for creating and operating on your own files. While at the terminal, you might wish to refer to the appendices, which list and briefly describe all the commands described in greater detail in the body of this manual.

## CORRECTING TYPING ERRORS

Errors must be corrected before hitting the RETURN key. (If you have failed to do this, the EDIT facility can be used to correct errors to a file that has been written and "saved". This will be described later.)

One or several @'s deletes one or several preceding characters--which may be blanks--and effectively backspaces the typing element.

Thus

    This es@xail@@mple

is interpreted as

    This example

since the first @ canceled the s (and backspaced one space), and the next two @'s canceled the i and l (and backspaced twice).

A line is canceled by the ¢ symbol. Thus if, instead of correcting individual mistakes in the above example, you had decided to start over again, you would have typed

    This esail¢

and hit RETURN. Then you would have retyped

    This example

Note: @ cannot cancel ¢. Once a ¢ has been typed, all preceding characters (and blanks) are canceled, and the effective line begins with the next character after the ¢ sign.

## CREATING A SCRIPT FILE

### ENTERING A NEW FILE

First, decide on a filename, sometimes abbreviated to fn in this manual. Maximum length is eight characters. The filename, together with filetype (Script, in this case) and filemode (which we are not concerned with in this manual), uniquely identifies a file for access by CMS commands.

When you are creating a file, make certain it has a new filename, unless you wish to replace an existing file with a new one of the same name. Next, issue the command

```
edit 'filename' script
```

and hit RETURN. The system responds with

```
NEW FILE.
INPUT:
```

You are now in Input mode and ready to type in your file, one line at a time, hitting RETURN at the end of each line.

Your Script file contains textual information plus special Script command words for controlling the format of your output. These commands are typed on separate lines from the text, and always start with a period in column 1.

### FILING IT AWAY

When you have finished typing in your file you must store it. To do this, you must leave Input mode and enter Edit mode. To enter Edit, hit RETURN on a "null" line (a line on which nothing has been typed--not even spaces--and, therefore, the typing element is at the left margin).

The system responds with

```
EDIT:
```

You type

```
file
```

and hit RETURN. The system types

```
R; T= 'CPU times' 'time of day'
```

### SAVING ENTRIES FOR FUTURE FILING

At intermediate points in entering your file, it is generally a good idea to save your entries up to that point

(as a safeguard against system "crashes"), without leaving the Edit program. You can do this by hitting RETURN (on a null line).

System responds with

EDIT:

You type:

save

and hit RETURN. System types

INPUT:

You may now continue typing in your file from the point where you stopped in order to issue the Save command. (Remember that the File command takes you from Edit mode to the general CMS command environment--not back to Input mode.)

## CORRECTING YOUR FILE

For making changes, additions, or deletions to your file, you now simply reissue

edit 'fn' script

which places you in Edit mode. (See "Correcting a Script File" for a complete description of the Edit facility.)

## PRINTING YOUR FILE

Later on, you will learn the commands for printing out your file at your terminal and on the offline printer. These printouts can be done either in the format you have determined with your Script control words or unformatted, that is, with text lines and commands in the same format in which they were entered. (See "Printing a Script File".)

## SUMMARY

To save part of a file hit RETURN on a null line. System responds with

EDIT:

You issue

save

and hit RETURN. System types

INPUT:

You are back in Input mode.

To file the entire document (and make it available for later use) hit RETURN on a null line.

System responds with

EDIT:

You issue

file

and hit RETURN. System types

R;T= 'CPU times' 'time of day'

You are now in the CMS command environment.

Preliminary note: Script commands begin in the first column, as in the following example. (They do not, however, appear in the first column in this manual, which has itself been prepared using Script; if they had been entered at the left margin exactly like legitimate Script commands, they would have controlled printed output.)

```
.tm 4
```

One space separates the command from the command operand (if any). Since there can be only one command on a line, hit RETURN to position the typing element at the beginning of the next line before entering another command or a text line.

The first thing you will want to do when creating a new file is to plan your page layout, including heading and page size parameters.

## HEADING

A heading, specified via

```
.he 'heading line'
```

is printed on each page except the first. (This is the only exception to the rule that commands and text must be entered on separate lines.) If the heading is to appear on page 1, it must be followed by the page advance command (see "Other Page-Planning Commands"), as follows:

```
.he 'heading line'
.pa
```

Changes in the heading line must be specified by

```
.he 'new heading line'
```

before the text for the new pages.

The heading is printed at the left margin, and 'heading line' must be at least ten characters less than line length (.ll, see below), to allow for a page number at the right margin. Spaces can be inserted in front of the heading to center it. For example

```
.he                         'heading line'
```

forces the heading 24 spaces away from the left margin. This effectively centers your heading on the page, if you

have specified a line length of 60 characters (or if you have specified no line length, in which case the 60-character length is automatically in effect).

## PAGE SIZE PARAMETERS

There are several of these to set:

| | |
|---|---|
| top margin | .tm |
| heading margin | .hm |
| bottom margin | .bm |
| line length | .ll |
| page length | .pl |

For each of these, there are default values which take effect if you decide not to set your own values. They are

```
.tm 5   (lines)
.hm 1   (lines)
.bm 3   (lines)
.ll 60  (characters)
.pl 66  (lines)
```

These values are assumed and, therefore, do not have to be entered unless you are resetting default (from nondefault) values. When entered, these page-size commands should be entered first in the file.

Note: .tm must be larger than .hm . Therefore, to print the heading higher on the page, increase .hm, making sure that .tm has a larger value.

| Example | Comments |
|---|---|
| .tm 8<br>.hm 3<br>.he Human Relations<br>'lines of text<br>taking up several<br>pages' | Since the heading command was not followed by the page advance command (.pa), the heading "Human Relations" will first appear at the top of page 2. |
| .tm 5<br>.hm 1<br>.he New Methods<br>'more text' | The next page will have the new heading, "New Methods". Note that default values have been respecified for .hm and .tm. Since no values were set for bottom margin, line length, or page length, default values are in effect. |

## Line Length and Underscoring

Line length can be varied to leave space on a page for figures or drawings (see "Sample Script File").

If deletion symbols (@) are included in the line, up to 132 characters may be entered, so long as the effective length after deletions is not more than 120 characters.

The usual maximum number of characters in a line is 120, but the maximum number of characters in a completely underscored line is 40. This is because each underscored character counts as three characters: the character, the backspace, and the underscore. Underscoring is very simple. Type in the letter(s), backspace (using the physical backspace key), and underscore (using the shift key and the underscore character). Lines of more than 40 characters can be underscored using hyphens in the next line.

## PAGE NUMBERING

```
.pn on    (default)
.pn off
.pn offno
```

If on, each page except the first is numbered.

If off, page numbers are not printed, although the processor keeps track of the numbering.

If offno, no page numbering takes place.

Thus if .pn off is entered, no pages are numbered, but page numbers are incremented internally, and if the page numbering command is later reset to .pn on, page numbering resumes with the number sequence preserved. If you want page numbering to resume with a <u>different</u> number, you can specify .pa 'n', which causes an eject to a new page, numbered 'n'. (See below for a description of .pa 'n'.)

## OTHER PAGE-PLANNING COMMANDS

There are two page eject commands: .pa (unconditional, which always produces a page eject), and .cp 'n' (conditional, which causes a page eject if fewer than 'n' lines remain on the page).

### <u>Unconditional</u>

```
.pa
```

The line following .pa is printed on a new page.

```
.pa 'n'
```

'n' is optional. If specified, it is the page number of the new page. Otherwise, sequential numbering takes place. For example, in page-numbering a report, the text of the report is preceded by introductory pages (title page, table of contents, etc.). Numbering begins with the first page of text. Set .pn off at the beginning of the file; after typing the introductory pages, set .pn on followed by .pa 1, which prints a number 1 on the first page of text. If you just specify .pa after .pn on, the first page is numbered page 2, since page numbering is incremented internally during .pn off. Thus

```
.pn off
'introductory pages'
.pn on
.pa 1
'first page of text'
```

will print a number 1 on the first page of text, while

```
.pn off
'introductory pages'
.pn on
.pa
'first page of text'
```

will print a number 2 on the first page of text.

Conditional

```
.cp 'n'
```

This causes a page eject if less than 'n' lines are left on the page. 'n' must be specified. This command is especially useful before section headings, to avoid printing a section heading as the last line of a page. For example

```
.cp 4
Format control
.sp 2
This section describes.....
```

The .cp 4 causes a page advance unless four lines remain in the current page, that is, sufficient space for the heading (Format control), the two spaces after it (.sp 2), and the first line of the new section.

Note: A .cp command not accompanied by an integer value ('n') is ignored.

SPACING

It is not necessary to issue the .ss (single space) command except to cancel the effect of the .ds (double space) command, which places a blank line between printed lines. Single spacing is the default mode.

To generate blank lines anywhere in a file, issue the .sp 'n' command, as in the example above. (If 'n' is not specified, 1 is assumed.)

Note 1: If .ds is in effect, the .sp command generates twice as many blank lines as are specified. In other words, the .sp command in

```
.ds
  .
  .
.sp 2
```

generates four blank lines.

Note 2: If page end is reached during a spacing operation, remaining blank lines are inserted after the heading on the following page.

CENTERING

Page titles, captions, figures, etc. can be centered by using the .ce command. The information to be centered must be preceded by .ce. Remember that when centering several lines, .ce is required for each line (see example below).

```
.ce
-------------------------
.ce
| Format | Example n     |
.ce
|        | E             |
.ce
-------------------------
```

produces the following printout:

```
-------------------------
| Format | Example n     |
|        | E             |
-------------------------
```

Leading blanks are counted as part of the line length, and therefore offset the line from the center of the page.

As you have probably noticed, the default values for most of the page-planning parameters (.tm 5, .hm 1, .bm 3, .ll 60, .pl 66, etc.) are adequate for a standard type of file.

Therefore, you probably will not have to adjust them, for day-to-day Script work.

## LINE BREAKING

You will recall from the Introduction that text lines typed beginning in column 1 are right-justified to fit neatly within the margins specified in the line length (.ll) command. For paragraph headings and such, it is desirable after certain lines to prevent justification, that is, to force a "break". One, or several, of the following methods may be used:

### No-Fill Mode

Use the .nf (no-fill) command

Lines are printed exactly as they appear in the file. This is useful--often essential--for figure drawing or lists. Some practice is needed to produce attractive text in this mode. Note: Remember to issue .fi (fill) at the conclusion of a no-fill section.

The figure illustrating the .ce command above could have been drawn in .nf mode as follows:

```
.nf
          -------------------------
          | Format | Example n    |
          |        | E            |
          -------------------------
.fi
```

Notice that each line of the figure was padded on the left with blanks to force the figure to the center of the page. The number of blanks has to be determined fairly accurately, as no .ce command is being used to compute center of page.

In .nf mode, no leading blanks or .br's (see "Fill Mode" below) are needed to prevent line justification, even for lines that are typed beginning in column 1.

Note: A simpler method is available for figure drawing, using the indentation command (.in), (see below).

### Fill Mode

The rest of the breaking methods apply to fill mode, which is the default condition and is therefore in effect unless .nf is issued.

Use of leading blanks (one is sufficient) prevents filling of the previous line. Therefore, an automatic break is in

effect for a line that begins in any column but column 1, for example, in the case of indentation of the first line of a paragraph. Use the space bar, or a tab (.tb) setting (see "The Tab Command") to space over to the beginning column.

A blank line with or without leading blanks in the line following it creates a break; for example:

```
'text'.............
..................
..................
'blank line'
    This line begins
with several blanks
..................
```

Use of the break command (.br) between input lines prevents filling of short lines with characters from the line following.

Most of the command words cause a break. Any of the commands listed below has the same breaking effect as .br.

```
.sp   .tm   .ce   .ds   .in   .tb
.ll   .hm   .he   .ss   .un   .fi
.pl   .bm   .pa   .rd   .of   .nf
```

## INDENTATION

Several methods of indenting are available under Script. Generally, it is a matter of preference which one is used, but there are certain features that should be learned before choosing one or the other.

### The Indent Command

The indent command (.in 'n') causes all following lines to be indented n spaces from the left margin.

Thus, .in 5 preceding text lines causes printing of each line to begin in the sixth column. This command can be altered in several ways.

If you want to move one, or a few, lines back toward the left margin, you may use the "undent" command (.un 'n') before each line that is to be undented. (Do not undent with a larger 'n' than the indent 'n', since it is impossible to go farther than the left margin of the page.) Hence

```
.in 10
.un 8
'line 1'
'line 2'
```

prints line 1 beginning in column 2, and line 2 and all following lines beginning in column 11.

Indentation can be changed with .in 'new n'. For example

```
.in 10
'Text'.....
..........
..........
.in 5
..........
..........
```

simply begins printing in column 11, then moves back to column 6 when .in 5 is encountered.

Indentation can be canceled, that is, printing be made to resume at the left margin, with .in followed by no integer, or with .in 0. In the example

```
.in 10
..........
..........
..........
.in
..........
..........
```

the lines after .in begin in column 1.

Indentation can also be canceled by using .un 'n', with the same n as .in 'n' (.un 10 in the example above), but, if many lines are to resume printing at the left margin, .un 'n' would have to be issued before each line, which is obviously an uneconomical procedure.

Note: In the sample figure under "no-fill mode" , leading blanks were used to force the figure to the center of the page. Indent provides a simpler method. For example:

```
.nf
.in 17
-------------------------
| Format | Example N    |
|        | E            |
-------------------------
.in
.fi
```

For line length of 60 characters (default .ll), .in 17 places the above 25-character box in the center of the page (leaving 17 spaces on one side and 18 on the other).

In summary, then, .in 'n' is altered as follows:

It is canceled by .in (or .in 0)

It is changed by .in 'new n'

It is undented by .un 'n'

## The Offset Command

The offset command (.of 'n') causes all but the first line following to be indented n spaces from the left margin. If an .in 'n' setting is already in effect, the .of starts at the indent value. The input file

```
.in 5
.of 10
'Text'.....
..........
..........
```

begins the first line of text in column 6 (remember, .of affects all but the first line following); the offset value of 10 starts printing of the remaining lines in column 16, due to the already existing .in 5.

The offset command can be altered by being either canceled or changed.

Canceling the Offset Command. This is done by issuing either

```
.of
```

or

```
.in
```

with no integer value (or with a zero value). Following either of these two commands, printing resumes at the left margin (unless .of is used for canceling, and a previous indent value is in effect; in that case, printing resumes at the indent level, since .of can cancel only a previous .of 'n' setting).

In the previous example, suppose .in follows the text lines, as shown below.

```
.in 5
.of 10
'Text'....
..........
..........
```

```
.in
..........
..........
```

After the second .in command, the .of 10 and .in 5 are canceled, and printing continues with column 1.

To cancel the offset value (.of 10) but not the .in 5, issue .of rather than .in. In this case, printing resumes with column 6, since the indent setting is still in effect.

In other words, .in is more inclusive for cancellation than .of, since .of can cancel only an .of 'n' setting, whereas .in can cancel both .of 'n' and .in 'n'.

Changing the Offset Command. This is done with .of 'new n'. In the example above, if

```
.in 5
.of 10
'Text'.....
..........
..........
```

were followed by

```
.of 5
'Text'.....
..........
```

the lines after .of 5 would start printing in column 11 (since .in 5 is still in effect).

To summarize, the offset command is altered as follows:

It is canceled by .of (or .of 0), or by .in (or .in 0).

It is changed by .of 'new n'.

## The Tab Command

This is used for paragraph-type indentation, which is the opposite of offset; that is, the first line is indented and succeeding lines start at the left margin. The tab command (.tb 'n' 'n' ...) can be set to any desired spacings (n's). These override default tab stops of 5, 10, 15, 20, ...75. Only the line that is tabbed during input is indented; all others revert to the left margin.

If any previous indent (.in) setting is in effect, the .tb

command, like .of, begins at the indented column. For example

```
.tb 8 15 24
(2 tab presses).....
.....
.in 10
(2 tab presses).....
.....
.....
```

produces

```
        column fifteen.....
  column one.............
  column one.............
            column 26.....
     column eleven.........
     column eleven.........
```

The first line was tabbed twice, to column 15; .in 10 preceded the next tabbed line, so that two tabs brought it to column 26 instead of column 15; succeeding lines of text resume printing from column 11.

In order to return to default settings (tab stops of 5, 10, 15, 20, ...75), it is necessary only to issue .tb with no integers.

The tab effect, alone, can be achieved without using any special control words, merely by leaving the desired number of blanks before the first line of input (for example, the first indented line of a paragraph) and starting the next lines at the left margin. Remember that any input line with a leading blank, or blanks, appears in the text just as it is typed; the blanks act as a break to prevent filling in of each previous line. Consequently, when entering paragraphs into a file, do not leave any blanks before any but the first line of the paragraph.

The tab command is useful for charts where printing begins in specified columns across the page. It is also useful for bibliographies, where line justification usually alters the number of spaces, from entry to entry, between the item number and the first word of the item. An example of the formatted output of a bibliography created without the use of .tb follows:

```
1.  Adair, R.....

2.   Field, M.S.....
```

Although, in both entries, two spaces were typed between the

item number and the author's name, the line-justified printout resulted in one extra space after item 2. This problem can be remedied in the following manner:

```
.tb 5 9
.of 9
(tab)1.(tab)Adair, R......
.of 9
(tab)2.(tab)Field, M.S......
```

Pressing the tab key once before typing the item number and again before the author's name makes the item number always appear in column 5 and the name in column 9, thereby generating two (and only two) spaces between them for every entry.

Be careful to turn off the .of 'n' command (by either .in or .of) and the .in 'n' command (by .in) at the completion of a section under control of either or both of these commands. It is easy to overlook a previous setting, and this can be particularly troublesome when an indentation is unknowingly in effect, as described above (.tb or .of following .in). For instance, if the first example under "The Tab Command" above were not followed by .in, subsequent tab presses would continue to be incremented by 11, resulting in tabs of 19, 26, and 35, rather than the intended 8, 15, and 24.

General Example on Indentation. Watch out for line filling when using .in, .of, .un, etc. for outlines, tables of contents, and such. For example, the short memo

```
A.  CP/CMS User's memo
    1.  Changes to the Script command
    2.  Minor revisions
    3.  Major revisions
        a.  new page size parameters
        b.  deadline changes for Script memos
```

could be entered in a Script file in the following manner (note that the three sections start in columns 6, 10, and 14, respectively):

```
.in 5
.of 4
A.  CP/CMS User's memo
.br
1.  Changes to the Script command
.br
2.  Minor revisions
.of 8
3.  Major revisions
.br
a.  new page size parameters
.br
```

b. deadline changes for Script memos

The .br commands are necessary where they are typed, since no other command that causes an automatic break is typed between successive text lines. (Text lines are supposedly typed starting in column 1.) Thus, if there were no .br between

A. CP/CMS User's memo

and

1. Changes to the Script command

characters from subheading 1 would be used to fill out the heading line A. The same is true for subheadings 3, a, and b. Notice that no .br is required between subheadings 2 and 3, because the offset command acts as a break.

You could also use .in with .un to type in the same memo as follows:

```
.in 13
.un 8
A.  CP/CMS User's memo
.un 4
1.  Changes to the Script command
.un 4
2.  Minor revisions
.un 4
3.  Major revisions
.br
a.  new page size parameters
.br
b.  deadline changes for Script memos
```

You can probably figure out why .br's were used only between subheadings 3 and a, and between a and b. It's for the same reason that .br's were used in the previous example, namely, there are no commands between these subheadings, because the last two subheadings start in column 14 and therefore no undent is needed to alter the initial .in value of 13. Since the subheadings are typed into the file starting in column 1, incomplete lines would be filled if no .br's were placed between them.

By using .tb, you could have created the same memo in the following manner:

```
.tb 6 10 14
(one tab press) A.  CP/CMS User's memo
(two tab presses) 1.  Changes to the Script command
(two tab presses) 2.  Minor revisions
(two tab presses) 3.  Major revisions
```

(three tab presses) a. new page-size parameters
(three tab presses) b. deadline changes for Script memos

This memo could also have been created in No-Fill mode, in which no .br's are required between typed lines.

## IMBEDDING A FILE WITHIN A FILE

A file can be inserted at a specified point in another file by the use of the imbed(.im) command. The correct form is

```
.im 'filename'
```

where 'filename' is the file to be inserted.

Various uses of .im will no doubt occur to you, such as insertion of a standard title page, bibliography, or distribution list in a document. Imbedded files can be added, removed, or their location changed within a file, according to the different uses for a file, at different times.

At the completion of the imbedding process, printing of the first file resumes.

Another use of the imbed command is to insert standard sets of commands at desired locations in a file. For example:

```
'commands and text lines'
.in 4
.im addend
.ds
'continuation of interrupted file'
```

The file called addend could consist solely of commands, which would be executed between the indent command (.in 4) and double-space command (.ds) of the imbedded file.

## ADDING A FILE TO THE END OF A FILE

The imbed command can be used to add a file to the end of a file by making .im 'filename' the last command in the file. At the completion of printout for the first file, the imbedded file will be printed out. Another command that can be used to append one file to another is

```
.ap 'filename'
```

Remember, as the word "append" implies, this command can be used only to attach a file to the end of another file, not to the middle (use the imbed command for that).

ENTERING LINES DURING SCRIPT PRINTOUT

The read (.rd 'n') command allows n lines to be entered from the terminal during Script output. (If n is omitted, 1 is assumed.) No formatting is performed by the program on the lines that are entered during .rd. This feature is useful for adding headings to form letters, etc.

An example of the use of .rd is

```
.he memo to Script users
.sp 2
.rd 3
```

After the heading is printed, the typing head spins and accepts three lines of input before continuing printout of the file. No formatting is performed on these lines. In the above example, a name and address could be typed in on the three "read" lines.

Note: The inclusion of .rd in a Script file does not cause any problems during printing on the offline printer, where lines cannot be typed in when a read command is reached (see next section for the offline printer command). The printer merely leaves n blank lines in response to .rd 'n' and continues printout.

---

EXERCISE 1: PRACTICE SCRIPT FILE

Create the following practice Script file, using any filename you wish. Use deletion symbols (@ and ¢) to correct errors as you go along. Assume in this exercise that the text below begins at the left margin. You may check your file with the two files listed on the next page (illustrating .in with .un, and .in with .of). These are provided only as possible methods--if you have found another that works, fine!

15a. Phrases Specified by Grammatical Structure

Prepositional Phrases: From morning coffee break to afternoon coffee break is too long. (Noun phrase describing coffee breaks is used as subject.)

The girl from Atlanta is pretty. (Adjective phrase, from Atlanta, modifies the noun girl.)

He lives in the valley. (Adverb phrase, in the valley, modifies the verb lives.)

```
edit exper1 script
NEW FILE.
INPUT:
15a.  Phrases Specified by Grammatical Structure
.sp
.in 20
.un 20
Prepositional Phrases: From morning coffee break to afternoon
coffee break is too long.  (Noun phrase describing coffee
breaks is used as subject.)
.un 5
The girl from Atlanta is pretty.  (Adjective phrase, from Atlanta
modifies the noun girl.)
.un 5
He lives in the valley.  (Adverb phrase, in the valley, modifies
the verb lives.)
.in

EDIT:
file
R; T=0.28/1.63 10.18.03



edit exper2 script
NEW FILE.
INPUT:
15a.  Phrases Specified by Grammatical Structure
.sp
.of 20
Prepositional Phrases: From morning coffee break to afternoon
coffee break is too long.  (Noun phrase describing coffee breaks
is used as subject.)
.in 15
.of 5
The girl from Atlanta is pretty.  (Adjective phrase, from
Atlanta, modifies the noun girl.)
.of 5
He lives in the valley.  (Adverb phrase, in the valley, modifies
the verb lives.)
.in

EDIT:
file
R; T=0.28/1.63 10.22.07
```

## PRINTING A SCRIPT FILE

The basic command for printing is

script 'fn' 'options'

Each option refers to a way you can modify some feature of your printing. If you do not specify a certain option, a default value is automatically applied. For example, if you do not specify, among the options in the above command, whether you want to print on your terminal or on an offline printer, the computer assumes that you want to print on your terminal.

Defaults and the corresponding options are described below.

| Default | Option |
|---|---|
| The command script 'fn' prints the specified file at your terminal. | offline (or of) prints on the offline printer instead of at the terminal (recommended for long files.) See "translate" below. |
| When typing is ready to begin at the terminal the message LOAD PAPER;HIT RETURN appears and execution pauses to allow you to position the paper at the top of a page. When ready, hit RETURN. | nowait (or no) starts printing at the terminal without waiting for paper adjustment or RETURN. |
| On terminal or printer there is no further pause during printout until the end of file is reached. | stop (or st) causes printing to pause at the bottom of each page on the terminal. (At this time you may change paper if you are using noncontinuous forms.) The paper should be positioned to the first line to be printed (the heading, if any) rather than to the physical top of the page. Typing resumes when RETURN is hit. |
| Lowercase and uppercase letters are printed on the offline printer just as they appear in your file. This is possible only if the printer chain is equipped with lowercase and uppercase letters ('TN-train'). If it is not (you can find out by | translate (or tr) translates lowercase to uppercase letters on terminal or offline printer. If the offline printer isn't equipped with uppercase and lowercase letters (TN-train), this option is required ("option"?!) |

| | |
|---|---|
| calling the computer room), it prints only the uppercase letters in your file. Therefore, you must specify the tr option unless your entire file is in uppercase (an unlikely situation). | |
| When printed at the terminal or printer your file is formatted by your Script control words. | unformatted (or un) prints on the terminal or printer a copy of your file, unformatted by control words. |
| Printing at the terminal or printer begins with the first page of your file. | page 'nnn' causes printing on the terminal or printer to begin on pagennn. nnn is a 3-digit page number and must include leading zeros (for example, page 4 is page004, page 12 is page012). This works even if you have specified .pn off and no page numbers have been printed on your output sheets. The only time it does not work is if you have issued .pn offno, and no numbering takes place, either printed or internally. |
| On the terminal or printer, no matter what page you begin printout with, printing continues to the end of the file. | single (or si) terminates after one page of printing on the terminal or printer. Usually used in conjunction with the page 'nnn' option to selectively format and print portions of a manuscript. |
| Printer output is aligned at the left margin of the printer paper. | center (or ce) centers printer output on the paper. |

Notes

1. Options are separated by a blank in the command

   script 'fn' 'options'

2. Type in options with your command. They will not be in effect automatically.

3. For additional options, see CP-67/CMS User's Guide.

Below are some practice exercises and Exercise 2.

Log in, using the ID and password supplied by the operations

group at your installation.

Type the commands and text as they appear in the practice exercise in this section. Your entries are indicated in lowercase, and system responses in uppercase.

Enter Edit mode from Input mode by typing a null line followed by RETURN, and file away your entries.

You can now practice obtaining printouts, either formatted (under format control of your Script commands) or unformatted (a line-by-line copy of commands and text just as you typed them into your file).

PRACTICE EXERCISES

Examine the printing commands below and, as practice, try issuing some of them (or make up some of your own).

```
script 'fn' no
```

This will simply print your file at the terminal (since the offline option was not specified), formatted by your Script control words. The nowait option (no) causes printing to begin wherever the paper is positioned at the time the command is issued.

```
script 'fn' of tr
```

Note: If the TN-train is not on the printer, the tr option must be included.

Your file, formatted by Script control words, will be printed on the offline printer (due to the of option), entirely in uppercase letters (due to the tr option).

When you receive the message R;T= 'CPU times' 'time of day', you may enter another CMS command (while your file is being printed offline). Try this: Choose one page of your file for printing (say, page 2). Then issue the command

```
script 'fn' page002 single
```

Only page 2 of your file will be printed (at your terminal) since you have used the page 'nnn' and single options.

You can print a copy of your file, either at the terminal or on the offline printer, unformatted by control words (that is, a copy of the file exactly as you typed it in).

For terminal output, issue

```
script 'fn' un
```

plus any additional options you wish (except, of course, offline).

For the offline printer, issue

```
script 'fn' un of tr
```

with or without additional options. (If the offline printer is not equipped with a TN-train, remember to include the translate option, abbreviated tr, in the command above).

As practice, using the sample practice file, you might want to issue some similar print commands, varying your choice of options.

EXERCISE 2

Print out the practice Script file you created in Exercise 1 in the following ways:

1. Unformatted at your terminal
2. Unformatted on the offline printer
3. Formatted at your terminal
4. Formatted on the offline printer

Examples of correct commands are listed on the next page.

## Answers to Exercise 2

1. script 'fn' un

2. script 'fn' of un tr

3. script 'fn'

4. script 'fn' of tr

If the printer is not equipped with the TN-train, use the tr option for 2 and 4. Otherwise, omit the tr option.

## INTERRUPTING TYPEOUT

If you wish to interrupt typeout (printout at the terminal), which was begun by the command script 'fn' 'options', hit the ATTN key once. Typing will be interrupted, and a pound sign (#) may be printed.

If you want typing to resume, hit ATTN a second time, then hit RETURN. You will notice that the line that was interrupted by the first ATTN is not completed; typing resumes with the following line.

If you want, instead, to "kill" typing (after stopping it by hitting ATTN once), hit ATTN a second time and type

```
kt
```

Then hit RETURN. The system will respond with

```
R; T= 'CPU times' 'time of day'
```

In summary:

To interrupt file printout, hit ATTN

To restart an interrupted printout, hit ATTN again and then hit RETURN.

To terminate typing, hit ATTN again, then type kt and hit RETURN.

## CORRECTING A SCRIPT FILE

Even if you have been exceptionally careful when entering a file in Input mode, you will probably want to make corrections to your file. In Edit mode, you can insert, replace, or delete entire lines, and also change parts of a line. Edit mode is entered in either of two ways:

1. For a file previously created in the Input mode and filed away with the File command, by using the command

```
edit 'fn' script
```

2. After you have entered part of a Script file but have not yet saved it with the File command, by issuing a null line followed by RETURN in Input mode.

Operation in Edit mode is by means of a pointer, which can be moved up and down through the file and can be interrogated about its position in the file at any time. In order to make corrections, you must know where the pointer is located.

If you have entered Edit by method 1 above, the pointer is at a blank line preceding the first line of your file; by method 2, it is at the last line of input.

### POSITIONING THE POINTER

The Top Command (top or t). This command positions the pointer to the blank line preceding the first line of the file. It enables insertions to be made at the top of the file.

Note: The pointer automatically performs a Top if you reach the bottom of the file (you will be notified with the message EOF:, meaning end of file) and then issue a Find (f), Locate (l), or Change (c) request.

The Next Command (next 'n' or n 'n'). If no number is specified, 1 is assumed. Pointer moves down the specified number of lines in the file. If EOF is reached, Next positions the pointer at the last line of the file. This command functions as a Bottom request (see below) if 'n' is greater than the number of lines between the pointer and the bottom of the file.

The Up Command (up 'n' or u 'n'). If no number is specified, 1 is assumed. Moves the pointer up in the file

the specified number of lines. This functions as a Top request if n is greater than the number of lines between pointer and top of file. The line at which the pointer is positioned is printed out.

The Bottom Command (bottom or bo). This command positions the pointer at the last line of the file.

The Print Command (print or p). If at any time you are uncertain about the position of the pointer, issue this command. The system responds by typing the line at which the pointer is directed.

The pointer can be moved either according to its line position (as described above), or by line context (searching for a line by its content). Ordinarily it is easier to do context editing, since the relative position of a file entry is not usually known.

The Locate Command (locate /'string'/ or l /'string'/). This command searches for the first appearance of the specified string of characters. The search begins with the line after which the pointer is positioned, and the line in which the string occurs is printed.

The string is enclosed by any pair of characters (called delimiters) not occurring in it. Generally a slash is used as a delimiter except when the string includes a slash. The rule is that any character can be used as a string delimiter as long as it does not appear in the string, and as long as the same character is used to enclose both sides of the string. For example, the following are legitimate uses of string delimiters:

```
locate 8string8

locate .string.
```

The character string must be unique to the statement being searched, since the first occurrence of the string will be located. However, if it is not unique and is found first in the wrong line, the command can be issued again. Remember to include in the string any blanks or other unique characters that appear in the statement being searched: that is, the string to be searched must be typed exactly as it appears in your file. Delimiters are also used with the Change command (see below).

Note: If you receive an end-of-file message (EOF:) rather than the text line you are searching for, try reissuing the Locate with fewer characters in the string. The reason for

this is that Edit can recognize only a character string that appears in the file entirely within one line. Therefore, if any part of your Locate string carries over to the next line, Edit will search through your entire file without finding it, and the pointer will be positioned at the end of your file.

The Find Command (find 'line' or f 'line'). The Find request, unlike Locate, is column dependent, and no delimiters are used. Hence, to find a line beginning in column 2 with the words "Section one", type

```
f  Section one
```

The first blank after the command is part of the command format. The second blank indicates a blank in column 1, and the search for nonblank characters begins in column 2. As in Locate, the line in which the characters occur is printed.

Note: To find a tabbed line, you must press the tab key before typing the characters you are searching for. For example, to find the subheading "1. Changes to the Script command" from the sample memo (see under "General Example on Indentation"), you would type

```
find (2 tab presses) 1. Changes to the Script command
```

In other words, after typing the command and one blank space, you press the tab key twice and then type in characters from the line you are searching for.

## CHANGING PART OF A LINE

The Change Command (c /'string1'/'string2'/).
Change searches left to right in the current line for the first occurrence of string1 and replaces it with string2. The current line is expanded or compressed as required (string1 and string2 can be of different lengths), and the line is printed in its changed form. If no match is found, the line remains unchanged, the message FIELD NOT FOUND is printed, and the pointer (as above) remains positioned at the current line so that the line can be operated on until the pointer is moved. As for the Locate command, delimiters for the string must not occur in either of the strings.

To delete, without substituting new characters for string1, issue a null string2, as follows:

```
c /'string1'//
```

## Example Using Locate and Change Commands

Suppose the second line of the file below is to be searched for and changed (note that the text begins in column 4).

```
   The quick brown fox jumps
   over the quiet stream.
```

If, by mistake, you issue

```
     l /qui/
```

Edit locates and prints

```
   The quick brown fox jumps
```

since that is the line in which the string "qui" first occurs.

If you know that the desired line is the next line in the file, you issue

```
     n
```

Otherwise, you reissue

```
     l /qui/
```

In either case, the system types

```
   over the quiet stream.
```

With the pointer now positioned at the desired line, you would use the Change command to make your correction.

Say you want the line to read

```
   over the quiet street.
```

Issue

```
     c /stream/street/
```

The system types the altered line

```
   over the quiet street.
```

and the pointer remains positioned at that line.

Suppose you now want to change "street" to "streets". If

you issue

```
c /et/ets/
```

Edit locates the first occurrence of "et" (in "quiet") and prints the altered line:

```
over the quiets street.
```

To correct this error, you have to issue something like this:

```
c /ts/t/
```

The system responds:

```
over the quiet street.
```

You have corrected your error. Now change "street" to "streets" by including more letters in the string than you did before (these letters should be unique). Suppose you issue:

```
c /eet/eets/
```

The first occurrence of "eet" is in "street". The desired altered line would then be printed out:

```
over the quiet streets.
```

To delete, say, the word "quiet", you could use the Change command with no string2 specification:

```
c /quiet//
```

The line now reads:

```
over the  street.
```

If you decide to reinsert "quiet", issue:

```
c /the/the quiet/
```

Using the Find Command. If you had used the Find command (instead of Locate) to search for the first line in the example above, (remember, the line begins in column 4) you could have typed:

```
f    The quick
```

The first blank is required by format rules; the next three blanks indicate the occurrence of blanks in columns 1, 2, and 3 of the line being searched for. Note the absence of string delimiters in the Find command.

Space Accomodations. In the above examples, the changed lines were automatically expanded or contracted to accommodate additions and deletions. Without this space-accommodating feature of Script, deletion of the word "quiet" from "over the quiet street" would have left a gap:

```
over the       street
```

When making changes to charts and diagrams, you often have to do your own space accommodating, especially if you want to retain the space left by a deletion. Take the following example:

```
|  2314  |  RPQ  |  23456  |
|  2311  |  RPQ  |  12345  |
```

(There are two spaces on each side of 2314 and 2311.)

If, to remove the 11 from 2311, you type

```
c /2311/23/
```

the two spaces left by the deleted 11 would be compacted, and the revised line (second line below) would read:

```
|  2314  |  RPQ  |  23456  |
|  23  |  RPQ  |  12345  |
```

If, on the other hand, you include the blank spaces in string2 of the Change command

```
c /2311/23  /
```

the desired spacing is retained (second line below):

```
|  2314  |  RPQ  |  23456  |
|  23    |  RPQ  |  12345  |
```

Repeated Changes

The Change command can actually be issued just once to make a correction in any number of lines--either on just the first, or on every occurrence of the string in those lines. This is done by adding parameters to the Change command. For example:

```
c /string/bring/ 10
```

In the next ten lines (starting with the one pointed to) "string" will be searched for and the first occurrence of "string" in each line will be changed to "bring". If you want every occurrence of "string" in those lines to be changed to "bring" you need a second parameter, an asterisk:

```
c /string/bring/ 10 *
```

The * denotes "in every case".

To make the specified change in every line as well, use an asterisk as the first parameter instead of a number:

```
c /string/bring/ * *
```

Note: You cannot use just the second parameter without specifying a first parameter. For example, to change every occurrence of a string in just one line, you cannot issue

```
c /'string1'/'string2'/   *
```

leaving a blank for the first parameter. In such a case, type the number 1 for the first parameter:

```
c /'string1'/'string2'/ 1 *
```

## CHANGING AN ENTIRE LINE

The Retype Command (r 'line'). The current line is replaced with 'line'. If no 'line' is specified (that is, if only r is typed) the current line is deleted, and the Input mode is entered. This is useful when it is necessary to replace the current line with more than one line, or when many additions must be made to a file that has already been created (it is easier to use Input mode for this purpose). Remember that you must type a null line to reenter Edit mode, and issue the File command to save your insertions.

Retype corresponds in format rules to Find: 'line' is separated from the request by only one blank, and any other blanks are considered part of 'line'; no delimiters are used; and the request is column-dependent.

Example: We shall use the same two lines as in the example above illustrating Locate, Change and Find (remember that the lines start in column 4):

```
The quick brown fox jumps
over the quiet stream.
```

Let's use Retype to change "stream" to "streets". (This is a more laborious procedure, since Change is easier to use for small changes; Retype is used here merely to compare it with Change.) Assuming the pointer is positioned at the desired line, issue:

```
r    over the quiet streets.
```

Note that the entire new line must be typed, since the old line is going to be overlaid by the new one. Also, the same number of blanks separate r from 'line' as were used in the Find command above. (Both commands are column-dependent.)

The Delete Command (d 'n'). Starting with the current line, the specified number of lines are deleted. If no number is specified, only the current line is deleted. The pointer is positioned at the line after the last deleted line.

Thus, to reposition the pointer at the line following the deleted one(s), the next 'n' command is not issued (this would position it one line too far down).

Note: Because of this, before making additional changes it is helpful to issue p (for Print) to ascertain current line position following a Delete command.

## ADDING A LINE IN EDIT MODE

The Insert Command (i 'line'). This command allows a line to be added to a file without transferring to the Input mode. The line is inserted after the line at which the pointer is positioned, and the pointer is advanced to the inserted line. Thus, additional lines can be entered between existing lines. A blank line can be inserted by using one or more spaces for 'line' but, if 'line' is omitted and no spaces are inserted, the Input mode is entered. The same format rules as for Find and Retype are observed.

### Example of Use of Retype Command to Insert Lines

As mentioned in the description of the Retype command, if r is issued with no 'line' specified, the current line is deleted, Input mode is entered (the word INPUT is typed by the system), and many lines can be inserted in place of the deleted line. Edit mode must be reentered later (null line followed by RETURN) in order to file your additions.

For example, suppose the pointer is positioned at the first line of the sample:

```
The quick brown fox jumps
```

If you issue

r

followed immediately by RETURN, the system responds with:

INPUT:

The line previously pointed to has now been deleted, and you can insert any number of lines; in the file the inserted lines precede "over the quiet stream". For example, to insert a line beginning in column 4, issue:

The slow green chicken hops

This first inserted line replaces, and begins in the same column as, the deleted line. If you then issue

around the farmyard
and swims

the altered file will appear as:

The slow green chicken hops
around the farmyard
and swims
over the quiet stream.

In other words, the three inserted lines replace the deleted one, and precede the next one ("over the quiet stream.") in the file.

To review.....with the pointer at the first line of

The quick brown fox jumps
over the quiet stream.

we issued the command

r

System answered:

INPUT:

Then we entered

The slow green chicken hops
around the farmyard
and swims
(null line)

The system now responds with

EDIT:

Then we enter

```
file
```

and the result is the revised file listed above.

Example of Use of Delete Command to Insert Lines

The same result could have been obtained using Delete and Input. In the above example, with the pointer positioned at the first line, we issued r followed immediately by RETURN. This caused Input mode to be entered.

Instead, this time we issue d followed immediately by RETURN. This simply deletes the line (the system responds with a click sound).

Then we issue

```
input
```

and proceed as with Retype (making additions in Input mode).

## LISTING PART OF A FILE

In the process of editing, it is easy to lose track of the changes that have been made. By using the Print command, we can list one line, several lines, or the entire file within Edit mode. (Remember that the command script 'fn' 'options' produced an entire listing, but in CMS command environment, not in Edit mode.) To print only one line, issue

```
p 1
```

or simply

```
p
```

To print part of a file, use Next, Find, or Locate to position the pointer at the first line to be printed, and issue

```
p 'n'
```

where n is the number of lines to be printed.

In order to print the entire file on the terminal, issue

```
t
```

```
p 'n'
```

where n is greater than the number of lines in the file.

Using the practice Script file in Exercise 1 (in Edit mode), issue

```
t

p 5

p 3
```

You will notice that the p 3 command begins typing with the last line typed by p 5. This is because the pointer remains positioned at the last line printed in a print command.

## EXITING FROM EDIT MODE

There are three ways to exit from the Edit environment:

1. To save your edited text and enter CMS command environment, issue

```
file
```

followed by RETURN. System responds with

```
R; T= 'CPU times' 'time of day'
```

2. If you decide <u>not</u> to save the changes made in Edit, and to enter CMS command environment, issue

```
quit
```

followed by RETURN. System will respond as before.

3. If you wish to enter Input mode without saving your edited text, issue

```
input
```

followed by RETURN.

You will then be returned to Input environment, and be able to enter lines after the one pointed to when you issued the input request. Input entries will be inserted into your file at that point.

EXERCISE 3

Make the following changes to your practice Script file:

| | |
|---|---|
| Change | "morning" to "mid-afternoon". |
| Retype | "The girl from Atlanta is pretty" as "The man from Sioux City is handsome". |
| Remove | "lives" from the last line. |

A possible answer is listed on the next page.

EXERCISE 4

The sample Script file, called "SCRIPT example", that is included with this manual employs a wider variety of commands than the practice file you just created, printed, and edited, but you might find that SCRIPT example is easier to create because the commands it requires are simpler to master than indentation commands.

Using any filename you wish, write your own file, print it out, and make any necessary corrections until it looks like SCRIPT example. (Crib sheet following "Sample Script File".)

TYPICAL TERMINAL SESSION IN ANSWER TO EXERCISE 3

```
edit exper1 script
EDIT:
f Prepositional
Prepositional Phrases: From morning coffee break to afternoon
c /morning/mid-afternoon/
Prepositional Phrases: From mid-afternoon coffee break to afternoon
1 /Atlanta/
The girl from Atlanta is pretty.  (Adjective phrase, from Atlanta,
r The man from Sioux City is handsome.  (Adjective phrase,
i from Sioux City
n
modifies the noun girl.)
c /girl/man/
modifies the noun man.)
1 /lives/
He lives in the valley.  (Adverb phrase, in the valley, modifies
1 /lives/
the verb lives.)
c / lives//
the verb.)
file
R; T=0.21/1.27 11.18.51
```

Notes: The Retype'd line (r The man from....) was too long to fit on this page. Therefore, part of the revised sentence, "from Sioux City", was Insert'ed under the Retype'd line.

Locate /lives/ was issued erroneously ("lives" occurs in the line above the line searched, as well as in the line searched). Therefore it was issued a second time. The Next command could have been used just as well.

In order to remove the space after the "b" in "verb" and before the period, a space is included in string1 of the Change command. If the command

```
c /lives//
```

had been issued, the result would have been:

```
the verb .)
```

There are other ways these revisions could have been made; this terminal session merely illustrates one of them.

If printed out using the Script command for printout, the Edit'ed file, exper1 script, would look like this:

15a. Phrases Specified by Grammatical Structure

Prepositional Phrases: From mid-afternoon coffee break to afternoon coffee break is too long. (Noun phrase describing coffee breaks is used as subject.)

The man from Sioux City is handsome. (Adjective phrase, from Sioux City, modifies the noun man.)

He lives in the valley. (Adverb phrase, in the valley, modifies the verb.)

This section describes procedures for listing your files (listf), transferring files to another user (xfer), or from another user to you (disk load), and erasing files (erase).

## FILE TRANSFER

If another user wishes to access one or several of your files, he might ask you to transfer them to him, that is, to his ID. If you agree to do this, proceed as follows: Issue

```
cp xfer d to 'ID'
```

System responds

```
R; T= 'CPU times' 'time of day'
```

For each file that you wish to transfer, issue

```
disk dump 'fn' script .
```

System will respond each time with

```
R; T= 'CPU times' 'time of day'
```

At the completion of your transfer, issue

```
cp xfer d off
```

System response is

```
R; T= 'CPU times' 'time of day'
```

If, instead, files are begin transferred to you, after you receive the message

```
..CARDS XFERD by 'ID' ...
```

issue

```
disk load
```

System responds

```
'FILENAME'   SCRIPT   P1   LOADED
R; T= 'CPU times' 'time of day'
```

Note: Disk load will overlay an already existing file of the same name on your disk.

## FILE ERASING AND LISTING

Note: Read "Listing" below before erasing files.

### Erasing

Since you have a limited amount of disk space, you will not be able to maintain voluminous files--nor, in fact, are you likely to want to, since a certain number of your files will probably become obsolete. Using the Erase command, you can delete specified files from your disk. (Make certain you wish to do this, since no copy of the file will remain under your ID). The command format is:

```
erase 'fn' script
```

### Listing

In preparation for an Erase, it is sometimes a good idea to have a listing of all, or part, of your files on your disk. The command is

```
listf
```

or

```
l
```

followed by RETURN.

You will get a response of this type:

```
FILENAME  FILETYPE  MODE  NO.REC.  DATE
SAMPLE    SCRIPT    P5    018      8/27
PRAC      SCRIPT    P5    033      9/1
R; T= 'CPU times' 'time of day'
```

If all your files are Script filetype (column 2 of the listing above), the only column that need concern you is the first, called FILENAME.

If, on the other hand, you have files other than Script filetype, you will be interested in the complete form of the listf command, which is

```
listf 'filename' 'filetype'
```

where 'filename', or 'filetype', or both, can be denoted "all" by using an asterisk (*). For example, if you want a listing of all files of a specified filename, but any filetype, type

```
listf 'filename' *
```

If, instead, you want a listing of all files of a specified filetype, but of any filename, type

```
lisft * 'filetype'
```

Finally, if you want a listing of all your files, type either

```
listf
```

or

```
listf * *
```

Note: Wait for the R; T= 'CPU times' 'time of day' message at the completion of each command before issuing another command.

## Summary

In preparation for an erase (or for any other reason) you can list your files by filename, filetype, or neither (that is, all your files regardless of name or type) using the listf command, as shown below.

For a specific file:

```
listf 'filename' 'filetype'
```

By filename:

```
listf 'filename' *
```

By filetype:

```
listf * 'filetype'
```

For all files:

```
listf
```

or

```
listf * *
```

You are now ready to erase. The Erase procedure is the same whether or not you have issued listf. For every file you wish to erase, issue

```
erase 'filename' script
```

## OTHER CMS FILE MAINTENANCE COMMANDS

There are additional CMS commands that you might want to learn to enhance your terminal sessions.

The "tape" commands enable you to dump your files, for backup, from your disk to magnetic tape, and to load them back from tape to disk as you need them.

Statistics on your disk use--number of records in use, total available records, percentage of total space in use, etc.--can be ascertained with the "stat" command.

Through another usage of the listf command, you can perform certain operations, such as disk dump, tape dump, or print on the offline printer, on every file on your disk (or every file of a specified filetype, like Script), by issuing just one command.

These commands and others are described in the CP-67/CMS User's Guide (see Bibliography).

## LOGGING OUT

Normally, when you are working on a file (creating, editing, printing, etc.) you are in the CMS command environment. At the completion of your work you will want to transfer out of CMS to CP (Control Program) environment by issuing the Logout command to CMS, and then log out a second time before turning power off.

To log out from CMS, type

```
logout
```

and hit RETURN.

The system responds

```
R; T= 'CPU times' 'time of day'
CP ENTERED, REQUEST, PLEASE
```

Actually, you can bypass this first logout by hitting ATTN while in CMS.

Then, to log out from CP, type

```
logout
```

and hit RETURN.

The system types out three sets of times:

```
CONNECT= 'time'  VIRTCPU= 'time'  TOTCPU= 'time'
```

and

```
LOGOUT AT 'time' ON 'date'
```

Press the OFF button on the terminal.

FORMATTED

SCRIPT Example

This example will demonstrate some of the capabilities of the SCRIPT facility. This file was created by issuing:

edit example script

Since the file did not previously exist, the terminal was placed directly into the Input environment. This paragraph was double-spaced with the .ds control. (Notice the heading on this page--it was created using the .ce command, since the header line in the 'heading' doesn't start printing until the second page.)

No Break command was needed here, since the .ss (Single-Space) control acts as a break. Although this is in Fill mode, tabular information can be included. For example:

| | | |
|---|---|---|
| SPACE | .SP | .sp |
| SINGLE SPACE | .SS | .ss |
| DOUBLE SPACE | .DS | .ds |

The leading blanks caused each line to be handled separately.

Use of the Line Length control allows space to be left within a page for figures or drawings. Naturally, it may take some experimentation to find how many paragraphs will fit next to a figure.

The new line length must take effect at a paragraph, since it acts as a Break. The switch back to standard line length, usually 60, also is a Break, and must end a paragraph. This works only in Fill mode. By switching out of Fill mode and doing some justification by eye, fancier effects can be obtained. This also takes some practice and experimentation.

CAPTION

PARAGRAPHS

If no space follows a paragraph heading, and if the paragraphs are not indented, a Break is necessary in Fill mode, to keep the heading line from being justified.

A few leading blanks are the easiest way to force a Break and separate paragraphs. A line with only a blank will also force a Break and a blank line, if the following line also begins with a blank, as follows:

The Center control is handy for small figures included in the text. A .ce in front of each line of the figure is necessary. Note that leading blanks count for figuring the length to be centered:

| FORMAT | EXAMPLE n |
|---|---|
| | E |

Figure EX.A

```
.tm 10
.he                          SCRIPT Example
.ce
SCRIPT Example
.sp 2
.ds
This example will demonstrate some of the capabilities of the
SCRIPT facility.  This file was created by issuing:
.br
edit example script
.br
Since the file did not previously exist, the terminal was placed
directly into the Input environment.  This paragraph was
double-spaced with the .ds control.  (Notice the heading on this
page--it was created using the .ce command, since the header
line in .he 'heading' doesn't start printing until the second
page.)
.ss
.sp
No Break command was needed here, since the .ss (Single-Space)
control acts as a break.  Although this is in Fill mode, tabular
information can be included.  For example:
.sp
    SPACE           .SP   .sp
    SINGLE SPACE    .SS   .ss
    DOUBLE SPACE    .DS   .ds
.sp
The leading blanks caused each line to be handled
separately.
.sp
Use of the Line Length control allows space to be left within
a page for figures or drawings.  Naturally, it may take some
experimentation to find how many paragraphs will fit next
to a figure.
.ll 30
.sp
The new line length must take effect at a paragraph, since it
acts as a Break.  The switch back to standard line length,
usually 60, also is a Break, and must end a paragraph.  This works
only in Fill mode.
.ll 60
.nf
By switching out of Fill mode                  CAPTION
and doing some justification
by eye, fancier effects can be obtained.  This also
takes some practice and experimentation.
.sp 2
.fi
.cp 5
PARAGRAPHS
.br
If no space follows a paragraph heading, and if the paragraphs
```

are not indented, a Break is necessary in Fill mode, to keep the heading line from being justified.

A few leading blanks are the easiest way to force a Break and separate paragraphs. A line with only a blank will also force a Break, and a blank line, if the following line also begins with a blank, as follows:

The Center control is handy for small figures included in the text. A .ce in front of each line of the figure is necessary. Note that leading blanks count for figuring the length to be centered:

```
.cp 5
.sp
.ce
- - - - - - - - - - - - - - - - -
.ce
|  FORMAT   |  EXAMPLE n    |
.ce
|           |  E            |
.ce
- - - - - - - - - - - - - - - - -
.ce
```

Figure EX.A

## APPENDIX A: COMMANDS IN EDIT MODE
(in alphabetical order)

Edit mode is entered when

```
edit 'filename' script
```

is typed for a file that was previously saved, using the Save and/or File commands. Edit mode can also be entered by hitting RETURN on a null line while in Input mode.

In this listing, the standard format of each command is given at the right.

### Bottom — bo

Positions the pointer at the last line of the file.

### Change — c /'string1'/'string2'/

'string2' replaces the first occurrence of 'string1' in the line pointed to. The pointer is not advanced by this command. The changed line is printed out.

```
c /'string1'/'string2'/ 'n' *
```

'n' denotes the number of lines to be searched for the occurrence of 'string1'.

* means that every occurrence of string1, not just the first, is to be replaced by string2 in 'n' lines. (If * is specified, 'n' must be specified, even if it is only 1.)

### Delete — d 'n'

Deletes n lines starting with the line pointed to. The pointer is positioned at the line following the last deleted line. If 'n' is omitted, 1 is assumed.

### File — file

Saves a copy of your file on your disk. Transfers from the Edit environment to the CMS command environment. (See Appendix C for a summary of commands that can be used in the CMS command environment.)

### Find — f 'characters'

The search begins with the line after the line where the pointer is positioned; a column-dependent match with

'characters' is sought, and if found, the line in which it occurs is printed out. The pointer is positioned at this line. Note: Do not surround 'characters' with delimiters (slashes, etc.).

Input i

Transfers from the Edit to the Input environment. (See Appendix B for a summary of Script commands that can be placed in a file in the Input environment.)

Insert i 'line'

Inserts 'line' immediately following the line at which the pointer is positioned. The pointer is then positioned at the inserted line. A blank line can be inserted by using one or more spaces for 'line'. If nothing is typed for 'line' (not even spaces) Input mode is entered.

Locate l /'string'/

The search begins with the line after the line where the pointer is positioned; the search is from left to right in each line for the first occurrence of 'string'. If found, the entire line containing 'string' is printed out, and the pointer is positioned at that line. Note the use of string delimiters. (Remember that the Find command did not use delimiters.)

Next n 'n'

Moves the pointer ahead n lines. If no 'n' is specified, 1 is assumed. (If 'n' is larger than the number of lines between the pointer and the bottom of the file, an automatic Bottom is performed.)

Print p 'n'

Prints out 'n' lines, beginning with the line pointed to. Pointer will be positioned at the last line printed. If 'n' is omitted, 1 is assumed. (See Appendix C for a summary of the Print command that can be used in the CMS command environment.)

Quit q

Prevents saving of a newly created file or of changes to an existing file. The CMS command environment is entered. (See Appendix C for a summary of commands that can be used in the CMS command environment.)

Retype r 'line'

Replaces the line currently pointed to with 'line'; the pointer remains positioned at 'line'. If 'line' is omitted in this command, the current line is deleted and Input mode entered. In other words, the Retype command on a null line has the same effect as the pair of commands d and i , namely, delete the current line and enter Input environment.

Save save

Saves a copy of your file on your disk and enters Input environment. (Compare this with File, which transfers into CMS command environment.)

Top t

Positions pointer to the blank line preceding the first printed line of your file (whereby lines can be inserted at the very beginning of your file).

Up u 'n'

Positions pointer 'n' lines above line pointed to. If 'n' is omitted, 1 is assumed. If 'n' is larger than the number of lines between the pointer line and the top of the file, an automatic Top is performed.

Input mode is entered when

    edit 'filename' script

is typed for a file that does not exist. Another way to enter Input mode from Edit mode is to type the Edit command i , which is an abbreviation for "input". Entry to Input mode can also be made from several other Edit commands (See Appendix A, "Insert" and "Retype").

The standard format for each command is shown on the right.

## Page Layout

Bottom margin .bm 'n'

Causes n blank lines to be left at the bottom of each page. Default value = 3 lines.

Heading margin .hm 'n'

Causes n blank lines to be left between the heading and the first line of text. (This does not include top margin, .tm) Default value = 1 line.

Line length .ll 'n'

Specifies a line length of n characters. Default value = 60 characters.

Page length .pl 'n'

Specifies n lines of printing per page. Default value = 6[illegible] lines.

Top margin .tm 'n'

Specifies the number of lines to be left for the top margin of each page (includes the heading line). Default value = 5 lines. Check to see that .tm is always larger than .hm.

Note: All of the above commands require an 'n' specification.

## Page Control

### Conditional page eject .cp 'n'

Causes a page eject if fewer than 'n' lines are left on the page. 'n' must be specified.

### Heading line .he 'heading line'

Prints 'heading line' at the top of each subsequent page. This command must be followed by .pa if the heading is to appear on the first page of a printed file. Note: This is the only command that is typed together with the text that it controls, all on the same line.

### Unconditional page eject .pa ('n')

Causes an immediate page eject. The line following .pa ('n') will be printed on a new page. 'n' is optional; if specified, it will be the page number of the new page. Otherwise, sequential page numbering will take place.

### Page number

.pn on (default condition)
.pn off
.pn offno

The default is .pn on and therefore does not have to be specified if sequential page numbering is desired. The command .pn off suppresses printing of page numbers; page numbering will resume if .pn on is specified later on in the file. The command .pn offno suppresses printing and internal registering of page numbers.

## Spacing

### Double space .ds

Causes double spacing of the printed page.

### Space .sp 'n'

Produces n blank lines between printed lines. If 'n' is omitted, 1 is assumed.

Single space .ss

Single-spaces the printed page. This is the default mode and therefore has to be specified only in order to cancel a previous double space (.ds) command.

## Paragraph Layout

Indent .in 'n'

Indents the left side of the printed page 'n' spaces.

Offset .of 'n'

Offsets the lines following this command, except for the first line, 'n' spaces from the printed left margin. This means that if a previous indent (.in 'n') has been set, offset begins at the indented column.

Tab setting .tb 'n1 n2 n3 ...nN'

Creates internal tab settings of n1...nN that take effect when the tab key is depressed. Overrides default settings of 5, 10, 15...75.) Tabbing, like offsetting, is affected by a previously set indent value.

Undent .un 'n'

The line following this command will start 'n' spaces further left than the column set by .in 'n'. Cannot be used if no indent value (.in 'n') has been set. The undent 'n' cannot be larger than the indent 'n' preceding it.

## Formatting Modes

Break .br

Prevents line justification in Fill mode. Therefore, information typed before and after .br will appear on separate lines of the printout. Breaking is not necessary in No-Fill (.nf) mode.

Note that many of the Script commands themselves cause a break between printed lines.

Center .ce

The immediately following line will be centered on the printed page.

Fill mode .fi

Causes line justification. Fill mode is the default mode in Script. It therefore needs to be set only to cancel a previously set no-fill (.nf).

No-fill mode .nf

Prevents line justification. Lines are printed exactly as they were typed in the file. Therefore, no breaking (.br) is required between typed lines.

## Special Feature

Read .rd 'n'

'n' lines can be typed in during terminal printout. If 'n' is omitted, 1 is assumed. No formatting takes place on these lines.

## Manuscript Layout

Append .ap 'filename'

Appends the contents of the named file ('filename') to the end of the file just printed.

Imbed .im 'filename'

Places the contents of the named file ('filename') at a specified location within another file; printout of the original file resumes when printing of the imbedded file has been completed.

## APPENDIX C: COMMANDS IN CMS COMMAND ENVIRONMENT

The CMS command environment is entered from the CP environment by typing

```
ipl cms
```

after logging in (login 'userid'). Entry to CMS can be made from Edit mode by typing

```
file
```

or

```
quit
```

### Erasing — erase 'filename' 'filetype'

Erases from your disk the file or files specified in 'filename' and 'filetype'. (This manual is concerned with 'filetype' Script.) An asterisk (*) used in place of 'filename' or 'filetype' specifies "all".

### "Killing" typeout — (ATTN) (ATTN) kt

Stops terminal typeout begun by a CMS command.

### Listing files — listf 'filename' 'filetype'

Prints a table of the specified files that you have on your disk. Lists filename, filetype, filemode, number of records, and the date last worked on (for example, changed under Edit and filed away) for each file specified in the listf command. An asterisk (*) used in place of 'filename' or 'filetype' specified "all".

### Printing files

The command format is

```
printf 'filename' script '1st line' 'last line'
```

or

```
script 'filename' 'options'
```

The first command above enables you to print a part of your Script file, at the terminal, unformatted by the control words. The options are beginning line number and last line number. (If neither of the options is specified, the entire

file will be printed.)

The second command is used to print the entire file, formatted or unformatted, at the terminal or offline printer. See "Printing a Script File" in the manual for a description of options that can be used with the Script command.

## Transferring files

The command formats are

```
cp xfer d to 'userid'

disk dump 'filename' 'filetype'
```

or

```
disk load
```

Use the first two commands to transfer a copy of your files to a another user; you must issue the second command for each file that you wish to transfer.

If a file has been transferred to you from another user (you are notified with the message ..CARDS XFERD BY 'USERID'..), issue the third command above, until the message ..CARDS XFERD BY 'USERID'.. no longer appears.

Note: Transferring a file does not erase it from your disk.

# BIBLIOGRAPHY

For further information on the CP-67/CMS system and the virtual machine concept, refer to the following publications:


Adair, R. J., R. U. Bayles, L. W. Comeau, and R. J. Creasy, A Virtual Machine System for the 360/40, IBM Cambridge Scientific Center Report 320-2007, Cambridge, Massachusetts, May 1966.

Field, M. S., Multi Access Systems - The Virtual Machine Approach IBM Cambridge Scientific Center Report 320-2033, Cambridge, Massachusetts, September 1968.

Hendricks, E. C., C. I. Johnson, R. D. Seawright, ONLINE/OS User's Guide, IBM Cambridge Scientific Center Report 320-2037, Cambridge, Massachusetts, February 1969.

Hendricks, E. C., C. I. Johnson, R. D. Seawright, Introduction to ONLINE/OS, IBM Cambridge Scientific Center Report 320-2036, Cambridge, Massachusetts, February 1969.

Madnick, S. E., G. A. Moulton, SCRIPT: An Online Manuscript Processing System, IBM Cambridge Scientific Center Report 320-2023, and IEEE Transactions on Engineering Writing Systems, August 1968.

The following documents are available through your local IBM Branch Office:

CP-67/CMS System Description Manual, IBM Form Number GH20-0802-1

CP-67/CMS User's Guide, IBM Form Number GH20-0859-0

CP-67 Operator's Guide, IBM Form Number GH20-0856-0

CP-67 Program Logic Manual, IBM Form Number GY20-0590-0

CMS Program Logic Manual, IBM Form Number GY20-0591-0

CP-67/CMS Installation Guide, IBM Form Number GH20-0857-0

INDEX TO COMMANDS

The symbols placed to the left of index entries indicate the environment(s) in which commands are entered (E=Edit, S=Script).

| | | |
|---|---|---|
| | .of | 18 |
| | .pa | 12 |
| | .pl | 11 |
| | .pn | 12 |
| | .rd | 24 |
| | .sp | 14 |
| | .ss | 14 |
| | .tb | 19 |
| | .tm | 11 |
| | .un | 17 |
| CMS, CP | Xfer | 46 |

(CP-67/CMS) Version 3 CMS Script User's Manual Printed in U.S.A. GH20-0860-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

# READER'S COMMENT FORM

(CP-67/CMS) Version 3
CMS Script User's Manual

GH20-0860-0

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

## COMMENTS

fold fold

fold fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
FOLD ON TWO LINES, STAPLE AND MAIL.

## YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

(CP-67/CMS) Version 3 CMS Script User's Manual Printed in U.S.A. GH20-0860-0