CC-244-2 MAC-M-193-2

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

TO:	CTSS users.
FROM:	J. H. Saltzer
SUBJECT:	TYPSET and RUNOFF, Memorandum editor and type-out commands.
DATE:	January 11, 1965

The command TYPSET is used to create and edit 12-bit BCD line-marked files. This command permits editing and revising by context, rather than by line number. The command RUNOFF will print out a 12-bit BCD line-marked file in manuscript format. RUNOFF contains several special features not available with the DITTO command, including type-justification.

These two commands provide an alternative to the MEMO, MODIFY, and DITTO commands, and are intended to provide experience with a different approach to editing symbolic files. This work represents one more iteration in the arduous task of creating an "ultimate" editing scheme. As such, it is primarily a synthesis of techniques which have been proven valuable in several separate problem areas. It is felt that this particular synthesis brings to bear on the editing problem an easy to use package of techniques, and might provide a model for an editor on a "next generation" time-sharing system. Here is a list of some of the sources of ideas for these commands:

J. McCarthy	(Colossal typewriter)
S. Piner	(Expensive typewriter)
P. Samson	(Justify)
Comp. Center staff	(Input, edit, and file)
M. L. Lowry	(Memo, Modify, and Ditto)
M. P. Barnett	(Photon)
V. H. Yngve	(Comit, Vedit)
R. S. Fabry	(Madbug)
R. L. Samuels	(Edits)
F. J. Corbato	(Revise)

An Edit-by-Context Program

Program Name: TYPSET

Description:

TYPSET is a command program used to type in and edit memorandum files of English text. TYPSET, along with the command RUNOFF, is an alternative to the commands MEMO, MODIFY, and DITTO. Editing is specified by context, rather than line number, and input is accomplished at high speed since the program does not respond between lines.

Usage:

TYPSET name

"name" specifies the name of a file to be edited, or a file to be created, or it may be absent, in which case a file is to be created and named later by the "FILE" request.

When TYPSET is ready for typing to begin, the word "Input" or "Edit" is typed, and the user may begin. If he is creating a file, he begins in high-speed input mode; if he is editing a file, he begins in edit mode.

High-Speed Input Mode.

In high speed input mode, the user may type lines of up to 360 characters in length (e.g., 120 underlined characters) separated by carriage returns. He does not wait for response from the program or the supervisor between lines, but may type as rapidly as desired. The full character set of his keyboard may be used.

The user leaves high-speed input mode, and enters edit mode by typing an extra carriage return. When switching modes, the program acknowledges the switch by typing the new mode, "input" or "edit".

Edit Mode.

In edit mode, the program recognizes "requests" of the form given below. All requests take effect immediately on a copy of the file being edited. Except where a request is expected to cause a response, such as "PRINT," successive requests may be entered immediately on successive lines without waiting for a response from the program. Each separate request must begin on a separate line. Responses from the program are generally typed in red.

Requests.

Editing is done line by line. Conceptually, we may envision a pointer which at the beginning of editing is above the first line of the file. This pointer is moved down to different lines by some requests, while other requests specify some action to be done to the line next to the pointer. All requests except FILE may be abbreviated by giving only the first letter. Illegal or misspelled requests will be commented upon and ignored.

For purposes of description, the requests have been divided into two categories, those necessary for effective use of the command, and special-purpose requests which are not so generally useful. The first category includes eight requests:

LOCATE character string

This request moves the pointer down to the first line which contains the given character string. Only enough of the line need be specified to uniquely identify it. Since the pointer only moves down through the file the second occurrance of a line containing a given character string may be located by giving the LOCATE request twice. The line which has been found is printed in its entirety.

It is not necessary to count blank characters exactly. If one blank character appears at some point in the request string, any number of blank characters or tabs at the corresponding point in the file will be deemed to satisfy the request. If 2 blank characters appear together in the request string, there must be at least two blank characters or tabs at the corresponding point in the file, etc.

If the LOCATE request fails to find a line containing the given character string, a message is printed, and the pointer is set to point after the last line in the file. Any requests which were typed in between the LOCATE which failed and the message from the program about the failure are ignored. Another LOCATE request will move the pointer back to the top of the file to begin another scan down through the file.

PRINT n

Starting at the pointer, n lines are printed on the typewriter console. The pointer is left at the last line printed. If n is absent, 1 line is printed and the pointer is not moved. If the pointer is not at a line (e.g., above or below the file, or at a line just deleted) only a carriage return is typed.

NEXT n

This request moves the pointer down "n" lines. If "n" is absent, the pointer is moved to the next line.

DELETE n

This request deletes "n" lines, starting with the line currently being pointed at. The pointer is left at the last deleted line. If "n" is absent, the current line is deleted and the pointer not moved.

INSERT new line

The line "new line" will be inserted after the line by the pointer. The first blank following the request word is part of the request word, and not part of the new line. The pointer is set to the new line. To insert more than one line, give several INSERT requests, or just type a carriage return to switch to high-speed input mode. All lines typed are inserted after the line being pointed at. When the user returns to edit mode by typing an extra return, the pointer is set to the last inserted line. If the very first edit request given is an INSERT, the inserted lines are placed at the beginning of the file. If an INSERT is given after the pointer has run off the bottom of the file, the inserted lines are placed at the end of the file.

CHANGE "string 1"string 2" n G

In the line being pointed at, the string of characters "string 1" is replaced by the string of characters "string 2". Any character not appearing within either character string may be used in place of the double quote character. If a number, n, is present, the change request will affect n lines, starting with the one being pointed at. All lines in which a change was made are printed. The last line scanned is printed whether a change was made or not. The pointer is left at the last line scanned. If the letter "G" is absent, only the first occurrance of "string 1" within a line will be changed. If "G" is present, all occurrances of "string 1" within a line will be changed. Blanks in CHANGE-request strings must be counted exactly.

```
Example:
line:        It is a nice day in Boston.
request:     CHANGE "is"was"
new line:    It was a nice day in Boston.
request:     CHANGE xwasxisx
new line:    It is a nice day in Boston.
request:     CHANGE ' '.' g
new line:    It.is.a.nice.day.in.Boston.
request:     CHANGE '.''
```

```
new line:    Itis.a.nice.day.in.Boston.
request:     CHANGE "tis"t is"
request:     CHANGE '.' ' G
request:     CHANGE 'on 'on.'
new line:    It is a nice day in Boston.
```

FILE name

This request is used to terminate the editing process and write the edited file on the disk. The edited file is filed as "name (MEMO)". If "name" is absent, the original name will be used, and the older file deleted. If no name was originally given, the request is ignored and a comment made. When this request is finished, the user returns to command level, and the supervisor will respond by typing "R" and the time used.

TOP

This request moves the pointer back to above the first line in a file.

The following seven requests are handy for special purposes, but will probably not be used as often as the ones previously described.

BOTTOM

This request moves the pointer to the end of the file and switches to input mode. All lines which are typed are placed at the end of the file.

ERASE c

The character "c" becomes the erase character. Normally, the character "#" is the erase character. (The erase character is used to delete the previously typed character or characters.)

KILL c

The character "c" becomes the kill character. Normally, the character "±" is the kill character. (The kill character is used to delete the entire line currently being typed.)

VERIFY p

If the parameter, "p" is "OFF", the following program responses are not automatically typed:

"INPUT" or "EDIT" when the mode is changed.

Lines found by the FIND or LOCATE requests.

Lines changed by a CHANGE request.

If the parameter "p" is "ON", the responses are restored. The command begins in "ON" mode.

RETYPE new line

The line "new line" replaces the line being pointed at. The first blank following the request word is part of the request word and therefore is not part of the new line.

FIND character string

This request moves the pointer down to the first line which <u>starts with</u> the given character string.

SPLIT name

All the lines above the pointer are split into a file named "name (MEMO)". Any old copy of "name (MEMO)" is deleted. The remainder of the file may still be edited, and filed under another name. The SPLIT request may be used several times during a single edit, if desired. Unless at least one "TOP" request has been given, "name" must be different from the original name of the file being split.

Backspacing.

The backspace key may be used to create overstuck or underlined characters. All overstruck characters are stored in a standard format, independent of the way they were typed in. CHANGE-, LOCATE- and FIND-request strings are also converted to this standard format, so it is not necessary to remember the order in which an overstruck character was typed in order to identify it. For example, suppose the line:

```
     The NØRMAL MØDE statement of MAD
```

had been typed in by typing the letters NORMAL, five backspaces, a slash, and four forward spaces. The slashed Ø can be changed to a normal O by typing

```
     CHANGE 'Ø'O'
```

Restricted Names and Recovery Procedures.

Two special names are used for intermediate files by TYPSET. They are

(INPUT (MEMO)

(INPT1 (MEMO)

Following a quit sequence (or a CTSS system breakdown) one of these files may be found. (Whenever a quit sequence has been given, a SAVE command should be issued to save the status of all files.) Since the intermediate file generally contains a complete copy of the file at the end of the last pass, it may be renamed and used as a source file, and may permit recovery of lost requests. The original file is never deleted until the new, edited file has been successfully written and closed.

The user's disk status and file directory are updated at the end of each pass through the file, thereby providing additional insurance against accidental loss. If an automatic logout occurs during use of TYPSET, the 12-bit indicator is lost. After resuming the logout saved file, type a carriage return. The program will then switch the input-edit mode and, incidentally, reset the 12-bit indicator as it responds.

The intermediate files are normally written in permanent mode. If the user's track quota becomes exhausted while editing, TYPSET will switch to temporary mode intermediate files. If it is necessary to leave the edited file in temporary mode, a comment will be made.

Summary of TYPSET requests.

abbreviation	request	response
Basic requests:		
L	LOCATE string	line found * end-of-file
D	DELETE n	end-of-file
N	NEXT n	end-of-file
I	INSERT line	none
P	PRINT n	printed lines, end of file
C	CHANGE QxxQyyQ n G	changed lines *
T	TOP	none
	FILE name	Ready message
Special purpose requests:		
B	BOTTOM	"Input" *
V	VERIFY ON (or OFF)	none
S	SPLIT name	no name given
R	RETYPE new line	none
E	ERASE x	none
K	KILL x	none
F	FIND string	line found * end-of-file

* These responses will not occur if VERIFY mode is off.

A Right-Justifying Type Out Program

Program Name: RUNOFF

Program Description:

RUNOFF is a command used to type out memorandum files of English text in manuscript format. Control words scattered in the text provide detailed control over the format, if desired. Input files may be prepared by the context editor, TYPSET.

Usage:

```
                RUNOFF name
             or RUNOFF name n
             or RUNOFF name NOSTOP n
```

where "name" is the primary name of a file "name (MEMO)" to be typed out. If a number, "n" is present, typing starts with the page numbered "n". The optional parameter "NOSTOP" is explained below, under "Non-Stop Typing."

Control Words:

Input generally consists of English text, 360 or fewer characters to a line. Control words must begin a new line, and begin with a period so that they may be distinguished from other text. RUNOFF does not print the control words.

.append A

Take as the next input line the first line of A (MEMO).

.line length n

Set the line length to "n". The line length is preset to 60.

.indent n

Set the number of spaces to be inserted at the beginning of each line to "n". Indent is preset to 0.

.undent n

In an indented region, this control word causes a break, and the next line only will be indented n spaces fewer than usual. This control word is useful for typing indented numbered paragraphs.

.paper length n

This control word is used for running off a memorandum file on a non-standard paper. The number "n" is a line count, figured at 6 lines per inch. If this control word is not given, "n" is assumed to be 66, for 11-inch paper.

.single space

Copy is to be single spaced. This mode takes effect after the next line. (The normal mode is single space.)

.double space

Copy is to be double spaced. This mode takes effect after the next line.

.begin page

Print out this page, start next line on a new page.

.adjust

Right adjust lines to the right margin by inserting blanks in the line. The next line is the first one affected. (This is the normal mode.)

.nojust

Do not right-adjust lines.

.fill

Lengthen short lines by moving words from the following line; trim long lines by moving words to the following line. (This is the normal mode.) A line beginning with one or more blanks is taken to be a new paragraph, and is not run into the previous line.

.nofill

Print all lines exactly as they appear without right adjustment or filling out.

.page n

Print page numbers. (The first page is not given a page number. It has instead a two-inch top margin.) If "n" is present, print out the present page immediately, and number the next page "n".

.space n

Insert "n" vertical spaces (carriage returns) in the copy. If "n" carries spacing to the bottom of a page, spacing is stopped. If "n" is absent or 0, one space is inserted.

.header xxxxxxxxxxxxxxxx

All of the line after the first blank is used as a header line, and appears at the top of each page along with the page number, if specified.

.break

The lines before and after the .break control word will not be run together by the "fill" mode of operation.

.center

The following line is to be centered between the left and right margins.

.literal

The following line is not a control word, despite the fact that it begins with a period.

All control words may be typed in either upper case or lower case. Illegal control words are ignored by the RUNOFF command. A comment may appear to the right of a control word, as long as it is on the same line.

Manuscript Conventions.

The RUNOFF program assumes a page length of 11 inches, with 6 vertical lines per inch. The top and bottom margins are 1 inch, except for the first page which has a 2-inch top margin. If page numbers are used, they appear flush with the right margin, 1/2 inch from the top of the page. If a header is used, it will be on the same line as the page number. The first page is not numbered, or given the header line unless the control words ".header" and ".page 1" appear before the first line of text.

Customary margins are 1 1/2 inches on the left and 1 inch on the right, implying a 60-character line. This is the standard line length in the absence of margin control words.

The program stops between pages and before the first page for loading of paper. The paper should be loaded so that after the first carriage return typing would take place on line 1 of the paper. The left margin stop of the typewriter should be placed at the point typing will begin, and the right margin moved as far right as possible. If you now type the first carriage return, the program will continue typing.

Tabs.

Tabs included in the text are faithfully reproduced in the final copy. Since there will be interactions between inserted tabs and the right-adjust and fillout procedure, the control word ".nofill" should precede any use of tabs. If you wish to indent, say, a whole paragraph, with right-adj.stment and filling, change "indent" and "line length" rather than using a tab at the beginning of each line. In order to type out a memo which uses tabs, the typewriter tab stops and left margin must be set up properly.

Backspacing.

Underlining or overtyping may be accomplished with the aid of the backspace key, even in a line that is subject to right adjustment.

Non-stop Typing.

If continuous form paper is used, RUNOFF can be instructed to not stop between pages by inserting the parameter "NOSTOP" after the file name (and before any initial page number) when the command is typed, e.g.

```
RUNOFF ALPHA NOSTOP
```

Abbreviations.

All control words may be abbreviated if desired. A list of abbreviations is given in the summary. In most cases, a single word is abbreviated by giving its first two letters; two words are abbreviated by giving the first letter of each word.

Summary of RUNOFF Control Words.

abbreviation	control word	automatic break
.ap	.append A	no
.ll	.line length n	no
.pl	.paper length n	no
.in	.indent n	no
.un	.undent n	yes
.ss	.single space	yes
.ds	.double space	yes
.bp	.begin page	yes
.ad	.adjust	yes
.fi	.fill	yes
.nf	.nofill	yes
.nj	.nojust	yes
.pa	.page (n)	yes, if n
.sp	.space (n)	yes
.he	.header xxxx	no
.br	.break	yes
.ce	.center	yes
.li	.literal	no

If "automatic break" is yes, the lines before and after the control word will never be run together, and the previous line will be printed out in its entirety before the control word takes effect.