NAME

RUNOFF — the predecessor of the *roff* langugage

DESCRIPTION

RUNOFF is a language for creating documents. This was the first document language at all. It is the ancestor of*roff*.

See section **SEE ALSO** at the end of this document for internet addresses.

HISTORY

The first text processing language was *DITTO* at the *CTSS* computer at *MIT*. But now there isn't any documentation about the corresponding programs nor files written in the *DITTO* language.

So *RUNOFF* can regarded as the oldest text processing language, because there is enough documentation and files written inb this language.

RUNOFF was built by*Jerome H. Saltzer* in 1963 and 1964 at MIT using the operating system CTSS on computers IBM 7090 and 7094 at the *MIT* in Boston.

At the*Unix* operating system, the *RUNOFF* language w as the base for the generation of the text generator language *roff*. Today *roff* is maintained by *GNU troff*, the program **groff**(1) and its language **groff**(7). So *RUNOFF* is the ancestor for *groff*.

Old RUNOFF Documentation

Look at section SEE ALSO for the internet connections to these documents.

All started at the operating system called *CTSS* in the early 1960s at the *MIT*.

There was a first documentation in 1964 by*Saltzer* who created the *RUNOFF* language. He published a documentation titled*Jerome H. Saltzer — TYPSET and RUNOFF, Memorandum editor and type−out commands*.

In december 1966,*Saltzer* published an updated documentation *Jerome H. Saltzer — Manuscript Typing and Editing*. This article can be regarded as the documentation of the original *RUNOFF*.

Moreover *Saltzer* published another document in 1965. It's titled*Jerome H. Saltzer — Experimental Additions to the RUNOFF Command*W e integrate the *control words* in this documentation in a section about *Experimental Additions*.

In later years, many people worked with the operating system *Multics*. There *RUNOFF* was further developed.

There is a good documentation of 1973 titled*Larry Barnes — RUNOFF: A Program for the Preparation of Documents*. The *RUNOFF* was here further developed. We use this document as well.

Another good document comes from 1974 at the DEC RSTS. It is titled *RUNOFF User's Guide*. It contains the best description of the text lines. We are glad to use that.

The latest *RUNOFF* documentation is file **RUNOFF.DOC** from PDP−11 at 1981, see SEE ALSO. The content of this document is also included in this document, although it has some errors.

There is still more documentation by the DEC PDP−10 archive. So far this information is not yet included in this document, but it will be done later on.

Early Environment 1963−66 (CTSS)

Saltzer originally worked on *MIT*'s *CTSS time-sharing operating system*. There he had an editor **TYPSET** that he also documented in the documentation cited above. This editor was an ancestor for **ed**(1).

To use his *RUNOFF* language, he programmed a tool that he called **RUNOFF**.

There is still an emulator and the old source files for **RUNOFF** and **TYPSET** at IBM 7090 CTSS ⟨http://www.cozx.com/~dpitts/ibm7090.html⟩.

The original RUNOFF program 1963-66

The original **RUNOFF** program is also documented in the documentation of 1966 above.

Saltzer uses upper case **RUNOFF** to denote his program. So we will also use **RUNOFF** to refer to the original program of 1963-66.

This program has mainly the task to adjust a printer of that time and then print a *RUNOFF* document with this configuration. Today this does not make much sense, but some parts are still available in the options of **groff**(1), but under different names. So we will not build this ancient program, but we will document its old command line here. A lower case program **runoff** will be something different.

RUNOFF is a command used to type out files of the *RUNOFF* language in manuscript format. *Control words* (command names) scattered in the text may be used to provide detailed control over the format. Input files may be prepared by the context editor **TYPSET** which does not exist today.

Usage of RUNOFF Program

RUNOFF *filespec* [*parameter* . . .]

filespec is the primary name of a file to be typed out.

parameter

arguments are any number of the following parameters, in any order:

STOP Pause between pages.

NOWAIT

Suppress the initial pause to load paper and the pause between pages (not necessary today).

PAGE *n*

Begin printing with the page numbered *n*.

BALL *n*

Typewriter is using printing ball *n*. If this parameter is omitted,**RUNOFF** assumes that the ball in use will properly print all*CTSS c haracters* in the file. The number *n* is engraved on top of the printing ball. *CTSS c haracters* not appearing on the ball being used will be printed as blanks, so that they may be drawn in. This parameter does not make sense in our modern printers.

BASIC RUNOFF LANGUAGE OF ALL TIMES

Files written in the *RUNOFF* language are similar to modern *roff* files.

They are both written in text mode. So they can be manipulated with text editors like **emacs**(1).

In files written in the text languages *RUNOFF* or *roff*, there are command lines and text lines.

Basic Command Lines

All lines beginning with a period (dot) **.** are command lines in both languages *RUNOFF* and *roff*. The period is followed by the name of the command (1 or 2 words of arbitrary length, later one even more words), optionally followed by 1 or more arguments.

The inventor *Saltzer* called the command name *control word*, but we keep using *command name* as was done in later times.

In *roff*, lines starting with a single quote **'** are also command lines, but that's not true for the *RUNOFF* language by default.

But in *RUNOFF*, the period **.** at the beginning of a line can be changed into any character. This can be done by the command **.FLAG CONTROL**. So the starting period is only one possible special character. But in this man−page, we use the period before a command name, because it is the default and helps recognizing commands directly.

In *RUNOFF*, the command names were quite long, sometimes consisting even of several words. This is the same in the*mom* language in *groff*.

Moreover, the *RUNOFF* commands can be appreviated by defined 2 letters words (or 3 later on). Later on,

the *classical roff* languages used only the 2-letter abbreviations as requests; but *groff* expanded these to arbitrary length.

In *RUNOFF*, each*control word* (command name) can be written in upper or lo wer case. That comes from the time where the computers had only upper case input. This was not taken into*roff*, because there were not enough 2-letter requests.

In *RUNOFF*, comments could be appended to a command line, see section **Comments**.

Lines beginning with a period but having unrecognizable format are treated as error.

No lines beginning with a dot are printed unless the preceding line was a *command line* with control name **.LITERAL**. Then the the line is output or printed as is, special characters are output without their special meaning.

Abbreviations for command names are normally based on the first two letters of a one word command or the first letter of the first two words of a multi−word command. Later on 3-letter abbreviations where used for command names of 3 words.

An example of a *control line* (command line) with a single *control word* with 2 arguments is a long name with upper case

.COMMAND *arg1 arg2*

or the same name in lower case

.command *arg1 arg2*

or an abbreviation with upper case

.CO *arg1 arg2*

or the same abbreviation with lower case

.co *arg1 arg2*

Another example of a *control line* (command line) with 2 *control words* with 1 argument is a long name with upper case

.WORD1 WORD2*ar g*

or the same name in lower case

.word1 word2*ar g*

or an abbreviation with upper case

.WW *arg*

or the same abbreviation in lower case

.ww *arg*

These *control words* were renamed to *requests* and *macros* later on in *roff*. In the 1973 document, the words *macros* and *formats* are used, but there isn't any documentation for these terms.

Comments

In *roff*, comments can be included by preceding the special character combination **\"**.

That was unknown in *RUNOFF*. There was only 1 method for including comments. Command lines could be appended by comments.

In the original *RUNOFF* language, a comment could be appended after the command arguments without using separators. For example,

In the newer *RUNOFF* languages (documented in 1974), they preceded the comments in command lines by the exclamation point (bang) **!**. For example,

Basic Text Lines

All lines that are not command lines are text lines in both languages.

There are 2 modes of text lines in *RUNOFF*. The newer mode had upper and lower case in the source file. This was similar to *roff*.

In the *CTSS* computer (early 1960s), there were only upper case input hardw are. So the *RUNOFF* files had only upper case ASCII characters in the original *RUNOFF* language.

As the printers could print in upper and lower case, there were special characters as case-shifters. That was also used in late *RUNOFF* files. This process is very different from *roff*.

In the following sections, only the upper case text lines are documented. ,P In *RUNOFF* and *roff*, there are special character combinations that can change the handling of the text. But these special characters are totally different in both languages.

One or more *blank lines* are not printed, but mean a *line break*. This can also be reached by the **.BREAK** *control word*.

In *groff*, blank lines are printed as lines of their own. This is not a paragraph break, because a line is bigger than a paragraph break.

A text line that starts with one or more space characters means *begin a new paragraph*.

In *groff*, this will start a new line and inserts the space characters at the beginning of the line.

Appending several Lines in RUNOFF

In 1974, it is documented that several text or command lines can be appended into a single line starting with a command.

These lines should be separated by a *semi−colon;*. If the appended line is a command line, then it starts with a period. That's enough for separation. In this case, separating semi−colon can be omitted.

Default Conditions (modes) in RUNOFF

The starting case-mode (for output or printing) is lower case. Each text line starts with that mode.

Usually the text is *filled* and *justified* as it is processed. That is, the program *fills* a line by adding successive words from the source text until one more word would cause the right margin to be exceeded. The line is then *justified* by making the w ord spacings larger until the last word in the line exactly meets the right margin.

The user may occasionally wish to reproduce the source text exactly, which is done by disabling *filling* and *justification* or by use of the .LITERAL command. The program may be set to *fill* but not *justify*, in which case the output will be normal except that lines will not be justified to the right margin. The program may also be set to *justify* b ut not *fill*, although this w ould probably produce peculiar results and is not recommended.

When the *fill mode* is on, spaces and carriage returns occurring in the source te xt are treated only as word separators. Multiple separators are ignored.

Some of the commands cause a BREAK in the output. A *break* means that the current line is output without justification, and the next word goes at the beginning of the next line. This occurs at the end of paragraphs.

The program will advance to new pages as necessary, placing the title (if given) and the page number at the top of each page. The user may call explicitly for a *pa ge advance* where desired, and may inhibit the occurrence of a *pa ge advance* within specified material.

By the documentation of 1974 and 1981, some special characters in text lines are initially disabled: **<** (**CAPITALIZE**), **>** (**INDEX**), **=** (**HYPHENATE**), and **%** (**OVERSTRIKE**).

The other special characters seem to be enabled by default. These should be:**.** (**CONTROL**), **!** (**ENDFOOTNOTE**), **^** (**UPPERCASE**), **** (**LOWERCASE**), **&** (**UNDERLINE**), **#** (**SPACE**), and _ (**QUOTE**).

This can be changed by the commands

.FLAGS <*mode*>

(enabling) and

.NO FLAGS<*mode*>

(disabling). Also, each special character can be set to another character by the

.FLAGS <*mode*> <*some_character*>

command.

The following informs about the<*mode*> ar guments:

CONTROL
default: *period* on first column (start command line): **.**

ENDFOOTNOTE
default: *exclamation character* on first column (end of footnote): **!**

UPPERCASE
upper case (single character) and case lock (2 characters), default: **^**

LOWERCASE
lower case (single character) and case lock (2 characters), default: ****

UNDERLINE
underlining word witout spaces, default: **&**

SPACE quoted space (no filling and justifying), default: **#**

QUOTE quoting character (output special character without specialization), default: _

CAPITALIZE
upper case for next character, default: **<**

INDEX set following word into index, default: **>**

OVERSTRIKE
superimpose the surrounding characters, default: **%**

HYPHENATE
default: **=**

ALL all but the 1st column characters **.** and **!**

null same as **ALL**

The following special characters are only documented in 1981, but it is not clear if they are initially enabled. Also no<*mode*> for **.FLAGS** are documented: **{**, **}**, **|**, and **Ctrl-N**.

RUNOFF's ability to change most special characters is unique. *roff* cannot do that.

UPPER CASE TEXT LINES

The original *RUNOFF* text lines are different from the*roff* language.

As the early*CTSS* computers could only produce upper case characters as input, the te xt lines look very strange today. This wasn't documented in the documentation of the 1960s. But there are good documentations of 1974 and 1981 which contain also the old style.

Case Changing of Text Lines

In this section, the specification of case for files prepared on an upper case terminal is documented. There are special characters that in printing act as case-shifters for ASCII characters into lower (ASCII code 97 to 122 decimal) or upper case (ASCII code 65 to 90 decimal).

The lower case mode seems to be the default mode. Also, according to existing old *RUNOFF* files, each text line starts with this default mode.

single circumflex **^**
The following ASCII character is shifted to*upper case*. The follo wing from the document of 1981 seems to be wrong: (It is also used to lock the *case mode* in *upper case*, and the *underline mode* to . nop underline all text).

single back-slash ****
: The following ASCII character is shifted into lower case. The following from the document of 1981 seems to be wrong: (It is also used to lock the *case mode* in *lower case*, and disable underlining.)

double circumflex **^^**
: The case mode is shifted into upper case.

double back-slash ****
: The case mode is shifted into lower case.

A common example with starting mode in lower case for these 4 special characters is:

ˆHERE IS A ˆSAMPLE ˆSENTENCE IN ˆˆUPPER CASE\ AND LOWER CASE.

is printed as:
: Here is a Sample Sentence in UPPER CASE and lower case.

Further special Characters in Text Lines

ampersand **&**
: This is used for underscoring the next following character. For example:

&s&o&f&t&w&a&r&e

becomes:

. nop software

in the output or printing.

circumflex and ampersand **ˆ&**
: This is used for underscoring all following characters except for blanks. One could say that by this character combination the underline mode is put on.

back-slash and ampersand **\&**
: This is used for stopping the underscoring. One could say that by this character combination the underline mode is put off.

For example:
: ˆ&PLATO\& was a very ˆ&wise \&man.

becomes
: . nop PLATO was a very . nop wise man.

In *groff*, spaces are not underlined as well.

number sign **#**
: *RUNOFF* interprets this character as a *quoted space*. It outputs exactly 1 space character, it is not justified or filled. It cannot end a line. In the text it is not treated as a *word separator*.

less-than **<**
: This character preceding a word capitalizes the entire word up to the first space character. This is the same as preceding the word with **ˆˆ** and ending it with ****. For example, if the current case mode is lower case, the following text line

<DIGITAL OF ˆMAYNARD, <MA

becomes

DIGITAL of Maynard, MA

in the output (printing). This special character is initially disabled. It can be activated by the command **.FLAGS**.

greater-than **>**
: This character enters the immediately following word (up to the first space character) into the index, including all case shifters in the word. This special character is initially disabled. It can be enabled by the command **.FLAGS**.

percent **%**

This is for *superimposition* of the surrounding characters, one character o ver the other. For example, the combination **/%=** overstrikes the 2 characters *slash* and *equal* into the character ≠ (unequal). In underlining, superimposition cannot be done. This special character is initially disabled. So a percent character is output as is. This character can be enabled by the command **.FLAGS**.

Special Characters in Text Lines of 1981

= *equals−sign* *— hypenation disable*

If **.FLAGS HYPHENATE** has been engaged, the *equals* character **=** used to disable *hyphenation* for the word it precedes.

{ *left−brace* *— Reverse half−linefeed*

If the output device type is no **N**, then the *left* and *right braces* are used for *superscripting* and *subscripting*. The *left−br ace* (**{** 173 octal) produces a *reverse half−linefeed*. When combined with the *right brace* (**}** 175 octal) scripting is created; e.g. **{super}** becomes ^{super} and **}sub{** becomes $_{subscript}$

} right−brace *forward half−linefeed*

As described above, the *right brace* (**}** 175 octal) when coupled with the *left brace* will produce scripting. This will only occur when a scripting output device is selected.

| vertical−bar *Engage/disengage alternate character set*

The *vertical bar* (**|**, 174 octal) acts as an on/off switch. It will alternately transmit a *shift−out* and a *shift−in* character to change the selected character set; e.g. **|ABC|** becomes *Ctrl−N***ABC***Ctrl−O*.

Escape Sequences in Text Lines

The *escape character* in *RUNOFF* (also called *quote character*) is the *sub character* _. Using this character as a prefix before a special character, outputs the special character as it is without its speciality, no formatting is done by it.

_^ outputs the special character ^

_\ outputs the special character \

_**&** outputs the special character **&**

_**#** outputs the special character **#**

_**<** outputs the special character **<**

_**>** outputs the special character **>**

_**%** outputs the special character **%**

__ outputs the special character _

COMMAND NAMES (CONTROL WORDS) IN THE ORIGINAL RUNOFF LANGUAGE OF 1966

The documentation for *control words* in this paragraph are taken from the *RUNOFF* documentation of 1966. Often this documentation refers to the **RUNOFF** program that doesn't exist any more. When the *RUNOFF* language will be implemented for **groff**(1) these documentations must be adjusted.

.ADJUST

.AD Enable *fill* mode. The next line is the first one affected. This is the default mode.

.APPEND *file*

.AP *file*

Take as the next input line the first line of *file*. Note that the whole of *file* is appended, and that the appending is an irreversible process — that is, once **RUNOFF** encounters the **.APPEND** *control line* it will switch to the file *file* and continue from the first line of *file*. All lines following the **.APPEND** *control line* will not be processed by **RUNOFF**. The file *file* may, of course, itself call for appending of still another file, and so on.

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

.BREAK
.BR The lines before and after the **.BREAK** *control word* will not be run together by the *fill* mode of operation.

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

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

.FILL
.FI Enable *fill mode*. That means: Lengthen short lines by moving words from the following line; trim long lines by moving words to the following line. This is the default mode. **.NOFILL** disables the *fill* mode.

.HEADER *word1 word2 . . .*
.HE *word1 word2 . . .*
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.

.HEADING MODE*ar g*
.HM *arg*
This *control sequence* alters the mode of the running head to that specified by the parameter *arg*. Any of the following parameters are allowed for *arg:*

CENTER
The header will be centered on the page.

MARGIN
The header will be adjusted against the right margin of the page.

FACING
On even−numbered pages, the header will be adjusted against the left margin, on odd numbered pages against the right.

OPPOSED
The header will be adjusted against the opposite margin from the page number. In the absence of a **.HEADING MODE** *control sequence*, the default option is **OPPOSED**.

.INDENT *n*
.IN *n* The argument *n* is a number. Set the number of spaces to be inserted at the beginning of each line to *n*. Indent is preset to 0.

.LINE LENGTH*n*
.LL *n* The argument *n* is a positive number. Set the line length to*n*. The line length is preset to 60.

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

.NOFILL
.NF Disable *fill mode*. That means: Print all lines exactly as they appear without right adjustment or filling out. In *NOFILL* mode each input line produces one output line; further blank lines are output in this mode. Use the **.FILL** *control word* to restart *filling*.

.NOJUST
.NJ Disable *fill mode*.

.ODD PAGE
.OP This *control word* causes the current page to be printed out, and the next page to be numbered with the next higher odd page number.

.PAGE [*n*]
.PA [*n*]

Print page numbers. (The first page is not given a page number. It has instead a two−inch top margin. See also**Manuscript Conventions**, belo w.) If argument *n* is present, insert a page break and number the next page *n*. Note that**RUNOFF** does not output or print completely empty pages.

.PAGING MODE*ar g1 arg2 . . .*
.PM *arg1 arg2 . . .*

This *control sequence* alters the mode of page numbering to that specified by the arguments. The arguments may be in any order, and must be selected from the following list:

MARGIN
Page numbers will be adjusted against the right margin.

FACING
Odd page numbers are adjusted against the right margin, even page numbers are adjusted against the left margin.

CENTER
Page numbers are centered between the right and left margin.

TOP Page numbers are placed on the fourth line from the top of the page.

BOTTOM
Page numbers are placed on the fourth line from the bottom of the page.

OFF Page numbers are discontinued.

PREFIX *"string"*
The string of characters between quotation marks is prefixed to the page number. The quotation marks may be next to each other, in which case no prefix is used.

ROMANU
Page numbers will be printed in upper case Roman numerals.

ROMANL
Page numbers will be printed in lower case Roman numerals.

ARABIC
Page numbers will be printed in Arabic. (This is the normal mode.)

SET *n* Set the next page number to be the positive number *n*.

SKIP *n*
Skip *n* page numbers.

If in a single use of **.PAGING MODE** several arguments specify competing functions, the last one specified takes precedence. When the **.PAGING MODE** sequence appears in text at point A, all text up to A (and probably some text after A) will appear on a page controlled by the previous paging mode. The new *paging mode* will take effect on the next page. Then there is no danger of getting page numbers both at the top and bottom of the same page.

Use of the **.TOP** parameter may conflict with the *heading mode*. If a heading and a page number should be printed in the same column, the page number will take precedence. In the absence of a **.PAGING MODE***contr ol sequence*, the default options are: **TOP MARGIN PREFIX "PAGE"**.

.PAPER LENGTH*n*
.PL *n* This *control word* is used for running off a documentation file on 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
.SS Copy is to be single spaced. This mode takes effect after the next line. (The normal mode is single space.)

.SPACE [*n*]
.SP [*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.

.UNDENT *n*
.UN *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.

RUNOFF ADDITIONS 1973

Here are described only the additional *control words* that are documented in the 1973 documentation.

Formats

.FORMAT *name*
This command causes subsequent text to be output under the control of tne specified format (see below at **.DEFINE FORMAT**). Each following logical line will be fit into the format until a **.FILL** or **.NOFILL** command 1s encountered.

.DEFINE FORMAT<*name*> <*pos*> <*field_ definition*> . . .
.END FORMAT
These commands define a format for use in producing tables, etc.

<*name*>
identifies the format. It can be activated by the **.FORMAT** *command*.

<*pos*> is the position and may be one of **.LEFT**, **.RIGHT**, or **.CENTER**, and determines the overall positlon of the format with respect to the margins.

<*field_definition*>
There can be several arguments of this type. Each has the form:
<*type*>**(**<*letter*> . . . <*letter*>**)**

where the<*type*> is one of

L for left,

R for riqht,

C for center,

F for *fill*, or **J** for *justify*.

The first three types define fixed fields; the text to be formatted must fit within the allocated space. The latter types define variable fields; the text will be handled as in normal fill mode processing.

A picture showing the manner in which text should be output follows the **.DEFINE FORMAT** command; following the picture should be an **.END FORMAT** command. The following lines give an example:

```
".DEFINE FORMAT SUMMARY L(A) F(C) C(B)"
"AAAA CCCCCCCCCCCCCCCCCCCCCCCC     BBBBBBB"
"     CCCCCCCCCCCCCCCCCCCCCCCC           "
".END FORMAT"
```

The first field of text is left justified; the second is centered; the third is subjected to *fill mode* processing without justification. After the first line of output is generated using this format, all subsequent lines are produced using the last picture 1ine. (Strictly speaking the third line is unnecessary.)

Text for formatted processing consists of a logical line (or paragraph). Each field except the last must be separated by *tab*. The*tab c haracter* is displayed here as backslash character (\).

The first field of text is **A**, the second **B**, etc. Typical input for our example might be:
|A\YES\/THIS IS SOME TEXT

TO BE FILLED.

The characters 1n the picture lines were 1nterpreted as follows. Contiguous sequences of letters determine the fie1d positions; non−alphabetic characters are output literally. (Note: **Q.QQ** wi11 not work, put the period "**.**" in the text. A sequence of characters written between double quotes is considered literal text. The*double quotes* are not output, and there is no way to use *double quote* as a *literal*.

Hyphenation Processing

.HYPHENATE

Enable *hyphenation mode*. This is the default when starting up. The **RUNOFF** program used a small *glossary* for splitting. In *hyphenation mode* **RUNOFF** would try to find a word 1n the glossary which is the same (except for the endings **−S**, **−ES**, **−ED**, and **−E**) as the word at the end of the line of text. When running in**gr off**(1) there are *glossaries* being much more complete than in **RUNOFF**.

.NOHYPHEN

Disable *hyphenation mode*.

.GLOSSARY *word*

This command inserts words into the *glossary* for use in *hyphenation*. Each word should have the form **hy-phen-ate** and be separated by spaces.

.HYPHENATION BREAK*n*

This command set the parameter which determines the allowable number of spaces to te inserted in a line before **RUNOFF** tried to hyphenate the last word. Each space counts ten points. If more than *n* points per word would have to be inserted, then *hyphenation* will be attempted. The initial setting of this parameter is 5 (one−half space per word).

Margin Controls

There are two types of margins involved in *RUNOFF*.

(1) The physical margins. These are determined by the nature of the printing device. The margins outline the area where it is physically possible to print characters.

(2) The logical margins. These can be set bv the user as he wishes. (Limits are imposed by the physical margins.) They are initialized for standard 8.5" by 11" printing.

Commands concerning vertical and horizontal margins are:

.PAGE LAYOUT TM, EM, TOL

This sets the vertical logical margins and vertical tolerence. Parameters are top margin, bottom margin and tolerance. The tolerance is used to determine where to break between pages on page overflows. If there is a line break within*TCL lines* of the bottom, **RUNOFF** will break the page there; otherwise it will fill the page completely.

.LINE LAYOUT LM, RM, NO, CS

This sets the logical left and right margin, the number of columns, and the number of spaces to insert between columns. These margins are used for the page headings. To adjust the relative text position, use the subsequent commands.

.REDUCE MARGIN LM, RM
.EXPAND MARGIN LM, RM
.END REDUCTION

These commands enable the user to indent a certain portion of his text using the first command, or **.UNDENT** his text using the second command. In either case the original margins are restored by the third command. The use of several **.REDUCE MARGIN** commends before the corresponding **.END REDUCTION**commands succesi vely indents the text more, and more. Thus these commands are 1ike brackets (i.e. recursive). **LM** is added to the left logical margin and **RM** is subtracted from the right logical margin in the first command. Just the opposite is done on the second command. Negative numbers are permitted. These commands do not effect the position of page headings.

.LAYOUT PLM, PRM, PTM, PBM, LL, LO

This command defines the physical margins in the followinq complex manner. (It should only be used for non−standard devices, normally this command should not be necessary.) The parameters are the physical left margin (in spaces), the physical right margin, the physical top 1ine, the physical bottom line, the line lenqth, and line origin. The first four parameters define the physical 1imits of the printing device. The final two parameters define the length of the logical line and its origin with respect to the left edge of the paper. Printing starts at column**LO + LM**, and ends at **LO + RM**, where **LM** and **RM** are the logical margins established by **.LINE LAYOUT**. When using the *facing feature*(see **.PAGING MODE**), the logical left mar gin is **LL − RM** on even pages, and the right margin is **LL − LM**. The parameters for the layout command must satisfy:

"min(LO + LL − PLM, PRM − LO) > max(PLM − LO, LO + LL − PRM),"
LL > 25, and **PBM − PTM > 6**

This command sets **LM** to 15, **RM** to **LL − 10**, **TM** to **PTM**, and to **PBM − 6**. (These margin settings produce the standard 1.5 inch left, and 1 inch right, too, and bottom margins.)

Initially **RUNOFF** sets the margins for *teletype* output to:

The printer layout is:

".layout 5, 137, 6, 66, 85, 15"
".page layout 6, 60, 4"

The logical margins must satisfy:

min(LL, PRM − LO, LO − LL − PLM) >= RM >
" LM >= max(0, PLM − LO, LO + LL − PRM)",
"PBM >= BM > TM >= PTM" ", and**"**
BM − TM > TOL.

Paragraph Formatting

.PARAGRAPH SPACING*n*

This specifies how many lines are to be inserted between paragraphs. Initial setting = 1.

.PARAGRAPH INDENTATION*n*

This specifies how many additional spaces to insert at the beginning of a paragraph. Initial setting = 5.

.PARAGRAPH UNDENTATION*n*

This command is the same as **.PARAGRAPH INDENTATION** −*n*. That is, *n* fewer spaces are inserted at the beginning of the paragraph.

Special Line Justification and Control

These commands pertain to the next logical line. The end of the line should be designated with a break.

.CENTER

Center the next line.

.INDENT *n*

Indent the next line *n* spaces. If *n* is not provided, 5 is assumed.

.UNDENT *n*

Start the next line *n* spaces to the left of the normal margin. This command is the same as **.INDENT** −*n*.

.MARGIN

Justify the next 1ine against the right hand margin.

Heading and Paging

.HEADER *XXXXXXX*

RUNOFF accepts a heading to go on the first line of each paqe. The heading strinq is assured to start at the first non−blank character after thd control word and end at carriage return.

.HEADING MODE<*par am*>

<*param*> determines the position of the heading on the 11ne. <*param*> may be any of the following.

CENTER

The header will be centered on the 1ine.

MARGIN

The header will be adjusted against the riqht margin.

PAGING

On even numbered pages the header is adjusted against the right margin. On odd pages it is adjusted against the left mergin.

OPPOSED

The header will be adjusted against the opposite margin from the page numter. This is the initial mode.

.PAGING MODE<*par am*>

This command determines the placing of the page number. All parameters are optional.<*par am*> may be anyone or more of the following oommands. In case of conflict the latest command wins.

CENTER

The page numbers are centered between the logical margins.

MARGIN

The page number is adjusted against the right margin.

FACING

On even numbered pages the number will be adjusted against the right margin. On odd numbered pages tne number will be adjusted against the left margin.

TOP Page numbers are placed on the first line.

BOTTOM

Page numbers are placed on the last line.

OFF Printing page numbers is discontinued.

.PREFIX <*string*>
.SECTION <*string*>
.SUFFIX <*string*>

The strings of characters between quotation marks are used to form the page string, which has the form:
<*prefix*><*section*><*page number*><*suffix*>

Any or all of these strings mav be null. The section string is considered to be part of the page number for purposes of indexing.

Initial mode is:

```
".PAGING MODE TOP MARGIN PREFIX "Page"" ".PAGING MODE
SECTION "" SUFFIX """
```

If neither page number nor heading is used, the text will start on the first logical line. Otherwise it will start on the fourth logical line. If the page number is at the bottom, text will end on the fourth line from the bottom. If the paging and headinq mode conflict, the page string overwrites the heading.

.ODD PAGE

This *control word* causes the ourrent page to be printed out and the next page to be started with the next higher odd number.

.PAGE *n*

If *n* is present, insert a page break and start numbering the next pege with *n*. Otherwise, turn the *paging mode* on and do not insert a page break.

.EJECT *n*

Insert a page break if either there are fewer than *n* lines left on the page or *n* is not present.

Lines and Spacing

.SINGLE SPACE

Sing1e space all 1ines within paragraphs. This is the initial state.

.DOUBLE SPACE

Double space all lines within paragraphs.

.SPACE *n*

Output *n* 1ine spaces. If *n* is not provided, 1 is assumed. In case of page overflow all remaining blank 1ines to be output are deleted.

.FIGURE SPACING*n*

This command is equivalent to **.EJECT** *n* followed by **.SPACE** *n*. These commands provide the only means of creatinq blank 11nes.

.BREAK

The lines before and after this command will not be run together in *fill mode*. A simpler w ay to get a line break is to insert one or more blank lines in the text.

.BEGIN GROUP
.END GROUP

The output lines enclosed between these two commands are forced to lie on a page. Thus this command acts in a manner similar to **.EJECT** *n*, where *n* has the 'right' value.

Miscellanous

.UNDERLINE

The following 1ine is underlined.

.LITERAL

The next line is taken as part of text whether or not it begins with dot.

.ESCAPE<*char*>
.SHIFT<*char*>
.TAB CHARACTER<*char*>

The given character becomes the *escape*, *shift*, or *tab* character. The parameter for the **.SHIFT** and **.TAB CHARACTER** commands may be null, if no *shift* or *tab* character is desired.

.DEFINE COMMAND<*name*>
.END COMMAND
.CALL <*name*>

These commands give the user the opportunity to combine text and control lines to form his own commands. All text and command lines between the first and second commands is stored away under *name*. When the third command is executed, the stored string is read and the commands within the string are executed. Recursion is not permitted.

.INDEX <*phrase*>, <*phrase*>

RUNOfF saves the first phrase in the main index table and the second phrase (if any) in a sub−index table associated with the first phrase.

The index is formatted and output after the last page of text. Two built−in but redefinable formats, *RINDEX* and *SINDEX*, are used to format the index as shown in the following example.

```
Algorithms, 40, 78,      \" uses RINDEX
  analysis of, 27,       \" uses SINDEX
```

The following lines give the initial definitions for the indexing formats.

```
".define format RINDEX f(A)"
"AAAAAAAAAAAAAAAAAAAAAAAAAA"
"    AAAAAAAAAAIAAAAAAAAAAAA"
".end format"
".define format SINDEX f(A)"
```

```
"  AAAAAAAAAAAAAAAAAAAAAAAA"
"    AAAAAAAAAAIAAAAAAAAAAAA"
".end format"
```

In order to get an index output in two columns,

```
.LINE LAYOUT 15, 75, 2, 4
```

should be the last line of the input.

RUNOFF DOCUMENTATION 1974

This document is the best documentation about text lines. Parts of that is used in the document of 1981. The documentation of text lines is the best at all. This is reguarded above.

The documentation of commands will be including later on.

RUNOFF COMMAND ADDITIONS 1981

The following *commands* will be recognized if they are at the beginning of a line started with a period. Any line in the source file beginning with a period is assumed to be one of these *commands*. If it is not, an *error diagnostic* will be typed and the line will be ignored. Some *commands* take one or more decimal numeric arguments. These are separated from the*command* by a space. More than one *command* may be entered on a single line by separating the *commands* with a *semicolon* ';' or a *period* '.'.

Multi−word commands may appear in any form. Thus, **.NO HEADER** and **.NOHEADER** are both legal.

Many *commands* may be abbreviated. Standard*abbr eviations* are given below each *command*.

Text Formatting Commands

.BREAK

.BR causes a *break*, i.e. the current line will be output with no *justification*, and the ne xt word of the source text will be placed at the beginning of the next line.

.SKIP *n*

.SK *n*

.S *n* causes a BREAK after which is multiplied by the number of *spaces* between lines. The result is the number of lines *skipped*. Output is advanced to the top of the next page if there is no room on the current page. If the current page is empty, **.SKIP** does nothing.

.BLANK *n*

.B *n* causes the current line to be output with no *justification*, skips *n* line spaces, and then starts output of the current source text. **.BLANK** is like **.SKIP**, except that the *space* to be left is independent of line spacing. If the page is empty, **.BLANK** does nothing.

.FIGURE *n*

.FG *n* leaves *n* lines blank to make room for a figure or diagram. If fewer than *n* lines remain on the current page, text continues to *fill* this page, then the page is adv anced and *n* blank lines are left at the top of the next page.

.INDENT *n*

.I *n* causes a and sets the next line to begin *n* spaces to the right of the left margin. The*n* can be ne gative to allow beginning a line to the left of the left margin. However, a line cannot begin to the left of column 0. If *n* is not supplied, the current paragraph indent is used.

.PARAGRAPH *n, v, t*

.P *n, v, t* causes a *break* and formats the output paragraphs. The *n* is optional and, if present, sets the number of spaces the paragraph is to be indented. The default value for *n* is **5** (*n* can also have a negative value). *v* is the vertical spacing between paragraphs. *v* can range from **0** to **5**. (**1** is *single spacing*, **2** is *double spacing*, etc.) *t* causes an automatic *.TEST PAGE* (see the **.TEST PAGE** *command*).

.CENTER *n*;*text*

.CENTRE *n*;*text*

.C *n*;*text*

causes a *break* and centers the following text in the source file. The centering is over column **(***n* **+** *left margin***)/2**. If *n* is not given, it is assumed to be the *right margin*.

NOTE

.CENTER, **.RIGHT MARGIN**, **.LEFT MARGIN**, **.PAGE SIZE**, and **.STANDARD** take both relative and absolute values. Relative values are expressed as **+***n* or **−***n*, while absolute values of *n* are unsigned.

.FOOTNOTE *n*

.FN *n* saves *n* lines at the bottom of the current page for a *footnote*. The*n* is multiplied by the number of spaces set with the **.SPACING** command. If insufficient room remains on the current page, space is allocated at the bottom of the following page. The text of the *footnote* should be gin on the line following the **.FOOTNOTE** command. *Indentation*, *case lock*, *justify*, *margins*, *spacing*, and *fill* are preserved around footnotes. However, *commands* that affect page formatting are illegal in a *footnote*. *Tab stops* are illegal because they are not preserved. A footnote within a footnote is also illegal.

The actual space taken by a footnote can be more or less than specified by *n*. If necessary`adjust` *n* after examining a draft printout.

The *footnote* is terminated with a line beginning with an exclamation point (the remainder of which is ignored).

.NOTE *text*

.NT *text*

starts an *indented note*. This command*blanks* **2**, reduces both *mar gins*, *centers* the text (if no text is given, it centers the word **NOTE**), and then *blanks* **1**. At this point you enter the text of the *note*. If the left margin is at **0**, the *margin reduction* is **15**, otherwise it is **5**.

.END NOTE

.EN terminates the **.NOTE** command, *blanks* and reverts the margins and spacing modes to their settings before the last **.NOTE** command.

.LIST *n*

.LS *n* starts an indented list with *n* spacing, moves the left margin **9** spaces to the right for the first **.LIST** command, and **4** more spaces for each subsequent nested **.LIST**. The normal *fill* and *justify modes* remain in effect. Therefore, you must disengage them just after the **.LS** command if you want a ragged right.

.LIST ELEMENT;*te xt*

.LE; *text*

starts an item in the list, used in conjunction with the **.LIST** comand. The elements are numbered sequentially and the number is given a negative indent so that the list lines up. The number is followed by a *period* and tw o *spaces* so that the indent will be by **−4**. The *list elements* are separated by the standard paragraph spacing and*TEST PAGE*. If you w ant to type the text on the same line as the command, you must separate the text from the command with any number of intervening *spaces* or *tabs*, or (optionally) one *semicolon*.

.END LIST

.ELS terminates the **.LIST** command and returns to settings before the last **.LIST** command.

.COMMENT *text*

.; *text* causes the line to be ignored. The text is not printed in the output file, but rather is used as a *comment* line in the source text.

Page Formatting Commands

.PAGE **.PG** causes a *break* and an *advance* to a new page. If the current page is empty, this *command* does not *advance* the page. Just like an *automatic page advance*, this *command* prints the *title* (if given) and *pa ge numbers* on every page.

.TEST PAGE*n*

.TP *n* causes a *break* followed by a *conditional page advance*. It skips to the next page if fewer than *n* lines are left on the page. This capability is to ensure that the following *n* lines are all output on the same page. This *command* has the form *t* as an optional argument to the **.PARAGRAPH** command.

.NUMBER *n*

.NM *n* starts page numbering. This is the default so there is no reason to issue this command unless page numbering is disengaged. If*r esumption* of page numbering is desired at a certain page, specify *n*.

.NONUMBER

.NNM disengages page numbering. However, pages continue to be counted, so that the normal page number can appear if page numbering is re−entered with the **.NUMBER** command.

.CHAPTER *text*

.CH *text*

starts a new chapter using the text as the title of the chapter. This*command* acts as if the follo w-ing *command string* were entered:

```
".BREAK;.PAGE;.BLANK 12;.CENTER;CHAPTER n"
```

The *n* is incremented by **1** automatically. After the CHAPTER*n* is typed on the page,

occurs. This*command* then resets the *case*, *mar gins*, *spacing*, and *justify/fill modes*. It also clears an y *subtitles* and sets the *chapter name* as the *title*.

.NUMBER CHAPTER *n*

supplies a number *n* to be used in a subsequent **.CHAPTER** command. **.NUMBER CHAPTER** would be used when a *chapter* of a document occupies a source file of its own. In such a case, **.NUMBER CHAPTER**w ould be the first command of the source file.

.HEADER LEVEL*n te xt*

.HL *n text*

starts a section at the level specified and takes the following text as the header. *n* can range from **1** to **5**. The sections are incremented by **1** automatically, and the number is output in the form *i***.***j***.***k***.***l***.***m*. If this is a chapter oriented document, the *i* is the chapter number. Otherwise, it is the number of the **.HL 1** level. This command acts as a

.BREAK;.TEST PAGE 9;.BLANK 3

followed by the *section number*, two *spaces*, and the *section name*. *HEADER LEVELS* **1** and **2** end with a *break*. *HEADER LEVELS* **3**, **4**, and **5** end with a space−dash−space combination (#−#).

.TITLE *text*

.T *text* takes the remaining text as the title and outputs it on every page at line **0**. The default is no title. If a *title* is desired, this *command* must be entered in the source file.

.FIRST TITLE*te xt*

.FT *text*

Same as **.TITLE**, but used to specify the title to be printed on the first page of the document. This command must precede all text in the source file. Use of the **.FIRST TITLE** command is the only way to print a title line on the first page of the document.

.SUBTITLE *text*

.SUBTTL *text*

.ST *text*

takes the remaining text as the *subtitle* and outputs it on every page. It appears directly under the title. The*subtitle* is not *indented*, b ut *indentation* can be achieved by typing leading spaces.

.INDEX *text*

.X *text* takes the remaining text on the line as a keyword and adds it, along with the current *pa ge number*, to the internal index buffer. The command does not cause a*br eak*. It should appear immediately before the item to be *indexed*. A keyword may be *indexed* more than once.

.DO INDEX*te xt*
.DX *text*

forces a new page, centers the text, if given, otherwise it centers the word *INDEX*. This command prints the entire contents of the index buffer. Entries are printed in alphabetic order and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The page number of each entry is placed on the same line as the entry and in the middle of the page. Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

.PRINT INDEX
.PX forces a new page after which it prints the entire contents of the index buffer. Entries are printed in alphabetical order and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The number of the first page on which each entry appeared is put on the same line as the entry, beginning at the middle of the line (midway betweeen the left and right margins). Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

.PRINT INDEXand **.DO INDEX** perform the same task. The only dif ference is that **.PRINT INDEX** does not interrupt the normal chapter and page sequencing.

.SUBPAGE

executes a **.PAGE** with page numbering suspended. The page number is unchanged, but letters are appended to the page number. This permits insertion of additional pages within an existing document without changing the existing page numbering.

.END SUBPAGE

disengages the **.SUBPAGE** command by executing a **.PAGE** command with page numbering resumed.

.APPENDIX *text*

.AX *text*

starts a new appendix using the text as the title of the appendix. This command acts as if the following command string were entered:

```
".BREAK;.PAGE;.BLANK 12;.CENTER;APPENDIX a"
```

The a is a letter that is incremented alphabetically automatically. After the**APPENDIX A** is typed on the page,

occurs. This command then resets the*case*, *mar gins*, *spacing*, and *justify/fill* modes. It also clears an y subtitles and sets the appendix name as the title.

.NUMBER APPENDIX*a*

supplies a letter *a* to be used as the letter for a subsequent **.APPENDIX** command.

.HEADER *arg*
.HD *arg*

causes the page header (*title*, *subtitle*, and *pa ge number*) to be printed. *ar g should be* **UPPER** to specify *upper case characters* for the title text, **LOWER** to specify *lower case*, or **MIXED**. The initial setting is **.HEADER UPPER**.

.NOHEADER

.NHD causes the page header (*title*, *subtitle*, and *pa ge number*) to be omitted. The header lines are completely omitted, so that text begins at the top of the page with no *top margin*.

Mode Setting Commands

.JUSTIFY
.J causes a break and sets subsequent output lines to be justified (initial setting). The *command* increases the spaces between words until the last word exactly meets the right margin.

.NOJUSTIFY

.NJ causes a *break* and prevents *justification* of subsequent output lines to make a ragged right margin.

.FILL

.F causes a break and specifies that subsequent output lines be filled (inital setting). Sets the justification mode to be that specified by the last appearance of **.JUSTIFY** or **.NOJUSTIFY**. **.FILL** adds successive words from the source text until the adding of one more word will exceed the right margin. It stops before putting the last word in. (If *hyphenation* has not been disabled, **RNO** will attempt to *break* words which cause line overflow into syllables.)

.NOFILL

.NF disengages the *fill* and *justify modes*. This*command* is used to permit typing a table.

NOTE

1. The *nofill−nojustify mode* need be used only where there are several lines of material to be copied exactly. A single line example will not require using these commands if there are breaks before and after.

2. Normally **.FILL** and **.NOFILL** are used to turn both *filling* and *justification* on and off. It is usually desirable to do both. A subsequent appearance of a *justification command* will override the *fill command* however.

3. Because of the action of **.FILL**, a single occurrance of **.NOJUSTIFY** will cause the remainder of the file to be *unjustified*, with *filling* as specified. In order to *justify* but *not fill* (not recommended), a **.JUSTIFY** command must follow every **.NOFILL** command.

.UPPER CASE

.UC sets the output mode to *upper case*. This comand acts the same as typing two ^^. This is the default mode. There is no need to type this command unless the mode was previously altered to *lower case*.

.LOWER CASE

.LC sets the typeout mode to *lower case*. This command acts the same as typing two *back−slashes* \\.

.FLAGS CAPITALIZE

.FL CAPITALIZE

enables the **<** character to *capitalize* the entire word it precedes. It then returns the file to the current case mode. This *special character* is usually `off` and must be typed at the very beginning of the source text to enable this character. Typing a space or another *less−than* **<** returns the file to the current *case lock*.

.NO FLAGS CAPITALIZE

.NFC disengages the **FLAG CAPITALIZE** command (inital setting).

.HYPHENATION

.HY engages *hyphenization* (initial setting).

.NO HYPHENATION

.NHY disengages IR hyphenization .

.FLAGS HYPHENATE

.FL HYPHENATE

enables the *equals character* **=** to disengage *hyphenization* for the word it precedes. This *special character* is initially `off` and must be typed at the beginning of the source file to enable this character. The*FLAGS HYPHENATE c haracter* is used to disengage *hyphenization* for words improperly *hyphenated* by the *hyphenization algorithm*.

.NO FLAGS HYPHENATE
.NFH disengages the **.FLAGS HYPHENATE** command (initial setting).

.NFL disengages the **.FLAGS CAPITALIZE** and the **.FLAGS HYPHENATE** commands (initial setting).

.PERIOD
.PR enables printing of two *spaces* after every *period* **.** that is followed by at least one *separator character* (initial setting).

.NOPERIOD
.NPR disengages conversion of *period/separator* to *period/two spaces*.

.LITERAL
,FONT CB .LIT
disengages *fill/justify* to permit printing of text exactly as entered in source file.

.END LITERAL
.ELI used after **.LITERAL** command to re−engage *fill/justify*.

Parameter Setting Commands

.LEFT MARGIN*n*
.LM *n* sets the left margin to *n*. The*n* must be less than the right margin but not less than **0**. The initial setting is **0**. If *n* is not supplied, **0** is used.

.RIGHT MARGIN*n*
.RM *n* sets the right margin *n*. The*n* must be greater than the left margin. The initial setting is**60**. If *n* is not supplied, the current page width (set with the **.PAGE SIZE** command) is used.

.PAPER SIZE*n*,*m*

.PAGE SIZE*n*,*m*

.PS *n*,*m*
sets the size of the page *n* lines by *m* columns and sets the right margin to *m*. The default setting is **58,60**.

.PITCH *n*,*m*
.PIT *n*,*m*
sets the horizontal and vertical pitch on a supporting output device. The horizontal pitch is*n* and is specified as characters per inch. The value must divide evenly into **60** for *Anderson−Jacobson devices* and into **120** for *Diablo devices*. The default is **12**.

The vertical pitch is *m* and is specified as lines per inch. *m* must divide evenly into **48**. The default is the hardware setting. *n* and *m* may be set independently.

.SPACING *n*
.SP *n* sets the number of spaces between lines. The *n* can range from **1** to **5**. The default setting is **1**. **.SPACING 1** is like *single spacing* on a typewriter and **.SPACING 2** is like *double spacing*. **.SPACING 2** puts one *blank line* between lines of text.

.STANDARD *n*
.SD *n* returns all parameters, except the pitch settings, to their initial settings and sets *n* as the page width. If **.STANDARD 60** is specified, margins are reset **.LM 0**, **.RM 60**, **.PAGE SIZE 58,60**, **.SPACING 1**, **PARAGRAPH INDENT 5**, and *fill* and *justify* are enabled. **.STANDARD 70** sets right margin to **70** and *page size* to **58,70**.

.TAB STOPS*n*,*n*,...
.TS *n*,*n*,...
sets tabs. The *n* must be greater than **0** and listed in ascending order. If tabs already exist, the issuing of another **.TAB STOPS** command clears all previous *tabs* before setting new ones. The *default tabs* are set at eight−column intervals to match the `Digital` hardware standard. These *tabs* are at columns **8**, **16**, **24**, **32**, **40**, **48**, **56**, **64**, **72**, and **80**. The tabs are converted to the ap-

propriate number of non−expandable spaces. If there are no regular spaces to the left of the *tabs*, they will print out at the appropriate position, even if *fill* is on. If *literal* is on, the *tabs* are not converted to *spaces*, but are output as *tabs*.

.AUTOPARAGRAPH

.AP causes any *blank line* or any line starting with a *space* or *tab* to be considered as the start of a new paragraph. This command allows normally typed text to be *justified* without special commands. It does not cause a paragraph if *blank lines* are followed by a command.

.NOAUTOPARAGRAPH

.NAP disengages the *AUTOPARAGRAPH* mode.

List of Commands (Alphabetical)

This list of commands is a table over 5 pages.

Command or Abbreviation		Related Commands
.AP	(=**.AUTOPARAGRAPH**)	
.APPENDIX *text*	(=**.AX**)	**.NUMBER APPENDIX** *a*
.AX	(=**.APPENDIX**)	
.AUTOPARAGRAPH	(=**.AP**)	**.NOAUTOPARAGRAPH** (**.NAP**)
.B	(=**.BLANK**)	
.BLANK *n*	(=**.B**)	**.SKIP** *n* (**.S**)
.BR	(=**.BREAK**)	
.BREAK	(**.BR**)	
.C	(=**.CENTRE**)	
.CENTER	(=**.CENTRE**)	
.CENTRE *n*;*text*	(=**.C**)	
.CH	(=**.CHAPTER**)	
.CHAPTER *text*	(=**.CH**)	**.NUMBER CHAPTER***n*
.COMMENT *text*		
.DO INDEX*te xt*	(=**.DX**)	**.PRINT INDEX** (**.PX**)
.DX	(=**.DO INDEX**)	
.ELS	(=**.END LIST**)	
.EN	(=**.END NOTE**)	
.END LIST	(=**.ELS**)	**.LIST** *n* (**.LS**)
.END LITERAL	(=**.ELI**)	**.LITERAL** (**.LIT**)
.END NOTE	(=**.EN**)	**.NOTE** *text* (**.NT**)
.END SUBPAGE		**.SUBPAGE**
.F	(=**.FILL**)	
.FG	(=**.FIGURE**)	

Command or Abbreviation		Related Commands
.FIGURE *n*	(=**.FG**)	
.FILL	(=**.F**)	**.NOFILL** (**.NF**)
.FIRST TITLE*te xt*	(=**.FT**)	**.TITLE** *text*
.FLAGS CAPITALIZE	(=**.FL CAPITALIZE**)	**.NO FLAGS CAPITALIZE** (**.NFL**)
.FLAGS HYPHENATE	(=**.FL HYPHENATE**)	**.NO FLAGS HYPHENATE** (**.NFH**)
.FN	(=**.FOOTNOTE**)	
.FOOTNOTE *n*	(=**.FN**)	
.FT	(=**.FIRST TITLE**)	
.HD	(=**.HEADER**)	
.HEADER *arg* [*arg*=**UPPER**, **LOWER**, or **MIXED**]	(=**.HD**)	**.NOHEADER** (**.NHD**)
.HEADER LEVEL*n te xt*	(=**.HL**)	
.HL	(=**.HEADER LEVEL**)	
.HY	(=**.HYPHENATION**)	
.HYPHENATION	(=**.HY**)	**.NO HYPHENATION** (**.NHY**)
.I	(=**.INDENT**)	
.INDENT *n*	(=**.I**)	
.INDEX *text*	(=**.X**)	
.J	(=**.JUSTIFY**)	
.JUSTIFY	(=**.J**)	**.NOJUSTIFY** (**.NJ**)
.LC	(=**.LOWER CASE**)	
.LE	(=**.LIST ELEMENT**)	
.LEFT MARGIN*n*	(=**.LM**)	**.RIGHT MARGIN***n* (**.RM**)
.LIST *n*	(=**.LS**)	**.END LIST** (**.ELS**)

Command or Abbreviation		Related Commands
.LIST ELEMENT;*text*	(=**.LE**)	**.END LIST** (**.ELS**)
.LIT	(=**.LITERAL**)	
.LITERAL	(=**.LIT**)	**.END LITERAL** (**.ELI**)
.LM	(=**.LEFT MARGIN**)	
.LOWER CASE	(=**.LC**)	**.UPPER CASE** (**.UC**)
.LS	(=**.LIST**)	
.NAP	(=**.NOAUTOPARAGRAPH**)	
.NF	(=**.NOFILL**)	
.NFC	(=**.NO FLAGS CAPITALIZE**)	
.NFH	(=**.NO FLAGS HYPHENATE**)	
.NFL	(=**.NO FLAGS CAPITALIZE** and **.NO FLAGS HYPHENATE**)	
.NHD	(=**.NO HEADER**)	
.NHY	(=**.NO HYPHENATION**)	
.NJ	(=**.NO JUSTIFY**)	
.NM	(=**.NUMBER**)	
.NNM	(=**.NO NUMBER**)	
.NOAUTOPARAGRAPH	(=**.NAP**)	**.AUTOPARAGRAPH** (**.AP**) **.PARAGRAPH** *h*,*v*,*t* (**.P**)
.NOFILL	(=**.NF**)	**.FILL** (**.F**)
.NO FLAGS CAPITALIZE	(=**.NFL**)	**.FLAGS CAPITALIZE** (**.FL CAPITALIZE**)
.NO FLAGS HYPHENATE	(=**.NFH**)	**.FLAGS HYPHENATE**
.NOHEADER	(=**.NHD**)	**.HEADER** *arg* (**.HD**)

Command or Abbreviation		Related Commands
.NO HYPHENATION	(=**.NHY**)	**.HYPHENATION** (**.HY**)
.NOJUSTIFY	(=**.NJ**)	**.JUSTIFY** (**.J**)
.NONUMBER	(=**.NNM**)	**.NUMBER** *n* (**.NM**)
.NOPERIOD	(=**.NPR**)	**.PERIOD** (**.PR**)
.NOTE *text*	(=**.NT**)	**.END NOTE** (**.EN**)
.NPR	(=**.NO PERIOD**)	
.NT	(=**.NOTE**)	
.NUMBER APPENDIX*a*		**.APPENDIX** *text*
.NUMBER CHAPTER *n*		**.CHAPTER text**
.P	(=**.PARAGRAPH**)	
.PAGE	(=**.PG**)	
.PAGE SIZE	(=**.PAPER SIZE**)	
.PAPER SIZE*v*,*h* (=**.PS**, **.PAGE SIZE**)		
.PERIOD	(=**.PR**)	**.NOPERIOD** (**.NPR**)
.PG	(=**.PAGE**)	
.PITCH	(=**.PIT**)	
.PRINT INDEX	(=**.PX**)	**.DO INDEX***te xt* (**.DX**)
.PS	(=**.PAPER SIZE**)	
.PX	(=**.PRINT INDEX**)	
.RIGHT MARGIN *n*	(=**.RM**)	
.RM	(=**.RIGHT MARGIN**)	
.S	(=**.SKIP**)	

Command or Abbreviation		Related Commands
.SD	(=**.STANDARD**)	
.SKIP *n*	(=**.S**)	**.BLANK** *n* (**.B**)
.SP	(=**.SPACING**)	
.SPACING *n*	(=**.SP**)	
.ST	(=**.SUBTITLE**)	
.SUBPAGE		**.END SUBPAGE**
.SUBTITLE *text*	(=**.ST**)	
.T	(=**.TITLE**)	
.TAB STOPS*n,n, . . . ,n*	(=**.TS**)	
.TEST PAGE*n*	(=**.TP**)	
.TITLE *text*	(=**.T**)	
.TP	(=**.TEST PAGE**)	
.TS	(=**.TAB STOPS**)	
.UC	(=**.UPPER CASE**)	
.UPPER CASE	(=**.UC**)	**.LOWER CASE** (**.LC**)
.X	(=**.INDEX**)	

EXPERIMENTAL ADDITIONS 1965

These *control words* are documented in *Saltzer*'s documentation of 1965. It is unsure whether they were really implemented.

In this documentation, all *control words* are written in *lower case*. The writing in*upper case* is not mentioned, the same is true for *abbreviations*. So this documentation uses only*lower case*.

.FIGURE

This *control word* turns control over to a *figure processor*, which creates in *core memory* a representation of a *flow diagram* under the control of a few special *control words*. When the*control word* **.END FIGURE** is encountered, the completed picture is printed immediately on the page being generated if there is room on that page; otherwise the *figure* will appear at the top of the next page.

Text following the **.END FIGURE** *control word* will be smoothly attached to text before the **.FIGURE**. No break is generated. (Restriction: If a *figure* is being held for placement at the top of the next page, another *figure* may not be encountered before the first one is printed.)

The only *control words* which are recognized when in the *figure processor* are the following three: **.FRAME**, **.BOX**, and **.END FIGURE**.

.FRAME *m n*

This *control word* intitalizes the *figure processor* by giving the height and width of the figure to be produced. *m* is the height, in lines; and *n* is the width, in characters. (Note that a 1050 types 6 lines per inch, and 10 characters per inch.)

Any attempt to place items in the picture which extend beyond the boundaries will cause an error comment to be generated. *m* and *n* must both be less than 100 and their product must be smaller than 5400. We may now think of the *figure* to be produced as an array of *m* times *n* elements.

.BOX *i j*

The text on the lines following this *control word* will be placed in the *figure* such that the first character on the first line following the **.BOX** will appear in row *i*, character position -IR j . The end of the text is indicated by a **.BOX** *control word* for another piece of text or the **.END FIGURE** *control word*. Temporarily, the text should not include underlined or overtyped characters.

.END FIGURE

This *control word* causes control to return to the regular *control processor* of the **.RUNOFF** command, for the decision to print the picture. Note that another **.FIGURE** *control word* may not appear until after this *figure* has been printed.

One further *control word* has been added which is intended to facilitate bringing out revised editions of a memorandum.

.FLAG The next line to be printed after this *control word* is encountered will have an asterisk placed two spaces to the right of the right margin, as illustrated.

.DEFINE *symbol*

This *control word* defines the value of the symbol*symbol* to be the number of the page currently being printed. The symbol may be used later with the **.USE** *control word* to cause printing of the page number in text. The characters in the symbol must be mappable into the six−bit character set, and all symbols must be six of fewer characters.

.USE *symbol*

The value of the symbol *symbol* is inserted into the text with a single blank preceding and no blank following. If the symbol has not been previously defined, its value is **0**. Text may continue following a blank typed after the symbol.

Here is an example of the use of these *control words*.

In one area of text:

> We now discuss the operation of the typewriter **.DEFINE REF1** coordinator module, which . . .
>
> In a later area of text:
>
> > As we saw in the discussion of the typewriter coordinator on page **.USE REF1**, the rest of . . .
>
> if the first area of text were on page 14, the later line would read:
>
> > As we saw in the discussion of the typewriter coordinator on page 14, the rest of . . .

-

Further Study of Experimental Additions

A number of suggestions have been made for extending the *control word* language of *RUNOFF*, and its capabilities. These are listed here, primarily to elicit comment and discussion, both on the language which describes these operations and the less important problem of their implementation. -

1. Word division. This is a whole are of study in itself.
2. Automatic footnote insertion. This was handled somewhat awkwardly in the **.DITTO** *command*, although the basic approach was probably reasonable.
3. Automatic page references, perhaps via some symbolic reference scheme. This would enable the page number in "as was described on page 32" to be inserted by the program. The analogy with an assembly program should be hotly pursued for ideas.
4. Special provision for printing facing pages. This would require alternate running heads, placing page numbers alternately at right and left, and matching line counts on facing pages.
5. Improved page−division rules, to prevent the last line of a paragraph appearing alone at the top of a page, for example. At present, copy must be run off to check by hand that awkward page divisions have not been made.
6. Automatic generation of page numbers for a table of contents. Again, the analogy of an assembly program symbol table appears fruitful.
7. Automatic generations of an index. The problem here is obtaining too many references to a given word, many irrelevant.
8. Arrangement of tabulated data. This problem may have already been partly approached with the above−described figure generator, or the facilities already available in *RUNOFF*, but automatic setup of column widths and positions would be desirable. One could include in this category the ability to call on other programs to computer numbers to place in tables, although this is going pretty far afield.
9. Placing figures in a *cut* or *inset*. The control language is the most difficult problem here.
10. Equation typing and numbering. Again, the control language appears formidable.

SEE ALSO

groff(1), **groff**(7), **roff**(7), **groff_filenames**(7)

1964 Jerome H. Saltzer:
Jerome H. Saltzer — TYPSET and RUNOFF, Memorandum editor and type−out commands a vailable at ⟨http://mit.edu/Saltzer/www/publications/CC−244.html⟩

1965 Jerome H. Saltzer:
Jerome H. Saltzer — Experimental Additions to the RUNOFF Command available at ⟨http://web.mit.edu/afs/athena.mit.edu/user/other/a/Saltzer/www/publications/PSN−40.html⟩

1966 Jerome H. Saltzer:
Jerome H. Saltzer — Manuscript Typing and Editing which is a vailable in the internet at *MIT html* ⟨http://mit.edu/Saltzer/www/publications/AH.9.01.html⟩ or *CTSS html* ⟨http://

web.mit.edu/Saltzer/www/publications/ctssAH.9.01.html⟩ or *CTSS pdf* ⟨http://web.mit.edu/Saltzer/www/publications/ctss/AH.9.01.pdf⟩.

1973 Larry Barnes:
Larry Barnes — RUNOFF: A Program for the Preparation of Documents
available as *pdf* ⟨http://www.textfiles.com/bitsavers/pdf/sds/9xx/940/ucbProjectGenie/mcjones/R−37_RUNOFF.pdf⟩.

There is still more documentation by the
DEC PDP−10 archive. So far this information is not yet included, but it will be done later on.

The latest *RUNOFF* documentation is file **RUNOFF.DOC** from PDP−11 at 1981, see SEE ALSO. The content of this document is also included in this document.

Look at section SEE ALSO for the internet connections to the documents.

Early Environment 1963−66

Saltzer originally worked on *MIT*'s *CTSS time-sharing operating system*. There he had an editor **TYPSET** that he also documented in the documentation cited above. This editor was an ancestor for **ed**(1).

To use his *RUNOFF* language, he programmed a tool that he called **RUNOFF**.

There is still an emulator and the old source files for **RUNOFF** and **TYPSET** at IBM 7090 CTSS ⟨http://www.cozx.com/~dpitts/ibm7090.html⟩.

The original RUNOFF program 1963-66

The original **RUNOFF** program is also documented in the documentation of 1966 above.

Saltzer uses upper case **RUNOFF** to denote his program. So we will also use **RUNOFF** to refer to the original program of 1963-66.

This program has mainly the task to adjust a printer of that time and then print a *RUNOFF* document with this configuration. Today this does not make much sense, but some parts are still available in the options of **groff**(1), but under different names. So we will not build this ancient program, but we will document its old command line here. A lower case program **runoff** will be something different.

RUNOFF is a command used to type out files of the *RUNOFF* language in manuscript format. *Control words* scattered in the text may be used to provide detailed control over the format. Input files may be prepared by the context editor **TYPSET** which does not exist today.

Usage of RUNOFF Program

RUNOFF *filespec* [*parameter* . . .]

filespec is the primary name of a file to be typed out.

parameter

arguments are any number of the following parameters, in any order:

STOP Pause between pages.

NOWAIT

Suppress the initial pause to load paper and the pause between pages (not necessary today).

PAGE *n*

Begin printing with the page numbered *n*.

BALL *n*

Typewriter is using printing ball *n*. If this parameter is omitted,**RUNOFF** assumes that the ball in use will properly print all*CTSS c haracters* in the file. The number *n* is engraved on top of the printing ball. *CTSS c haracters* not appearing on the ball being used will be printed as blanks, so that they may be drawn in. This parameter does not make sense in our modern printers.

THE ORIGINAL RUNOFF LANGUAGE OF 1966

A *RUNOFF* file consists of *command lines* and *text lines*. The*command lines* start with a period (dot) "**.**", all other lines are IR "text lines" .

Command lines are also called *command lines* by *Saltzer*.

Text Lines and Conditions

RUNOFF text lines are different from the *groff* language.

As the early*CTSS* computers could only produce upper case characters, the te xt lines look very strange today. This wasn't documented in the documentation of the 1960s. But there is a good documentation of 1981 which contains also the old style. Have a look at chapter *RUNOFF ADDITIONS 1981* section *Case Information*in this document.

One or more *blank lines* are not printed, but mean a *line break*. This can also be reached by the **.BREAK** *control word*.

In *groff*, blank lines are printed as lines of their own. This is not a paragraph break, because a line is bigger than a paragraph break.

A text line that starts with one or more space characters means *begin a new paragraph*.

In *groff*, this will start a new line and inserts the space characters at the beginning of the line.

Command Lines

A command line begins with a period (dot) "**.**". Following the dot **RUNOFF** expects a *command name*. This is called *control word* by *Saltzer*

These *command names* or *control words* were defined by *Saltzer* as 1 or 2 words of arbitrary length. or an *abbreviation* of defined 2 characters. The later *roff* language uses only 2−character requests; but *groff* expanded these to arbitrary length. Each *control word* (1 or two words) can be written in upper or lower case as you like.

Some *control words* are followed by a *space* and the *parameters* for that *command*, followed optionally by a comment (Comments are not documented further).

Lines beginning with a dot but having an unrecognizable format are treated as errors.

No lines beginning with a dot are printed unless the preceding line was a *control line* **.LITERAL**. All commands are described below. Abbreviations for command names are normally based on the first two letters of a one word command or the first letter of the first two words of a multi−word command. Commands which should close a logical line break do. Information on abbreviations and whether commands cause line breaks will be found in the summary at the end of the manual. In a command line **RUNOFF** will consider multiple blanks as a single blank (space), if a blank character 1s legal. A*RUNOFF* document contains te xt separated by so−called *control words*. As these are full lines, a better name would be *control lines* as is used in the documentation for **groff**(7),

These are lines starting with a period (**.**) and directly followed by a command with or without arguments. The command names are arbitrarily long, they can even consist of several words, possibly followed by arguments.

The *control words* can be written in lower or upper case, just as wanted. Moreover, each command name can be shortened to an abbreviation of 2 characters. When the command name has only 1 word, the first 2 characters are taken. *Command names* with 2 words abbreviate to the first character of each word. These abbreviations led later to the 2 character *requests* of *roff*.

An example of a *control line* with a single *control word* with 2 arguments is a long name with lower case

.command *arg1 arg2*

or a long name with upper case

.COMMAND *arg1 arg2*

or an abbreviation with lower case

.co *arg1 arg2*

or an abbreviation with upper case

.CO *arg1 arg2*

Another example of a *control line* with 2 *control words* with 1 argument is a long name with lower case

.word1 word2*ar g*

or a long name with upper case

.WORD1 WORD2*ar g*

or an abbreviation with lower case

.ww *arg*

or an abbreviation with upper case

.WW *arg*

These *control words* were renamed to *requests* later on in *roff*. In the 1973 document, the words *macros* and *formats* are used without any documentation.

Control Words (Command Names, Requests)

The documentation for *control words* in this paragraph are taken from the *RUNOFF* documentation of 1966. Often this documentation refers to the **RUNOFF** program that doesn't exist any more. When the *RUNOFF* language will be implemented for **groff**(1) these documentations must be adjusted.

.ADJUST
.AD Enable *fill* mode. The next line is the first one affected. This is the default mode.

.APPEND *file*
.AP *file*
Take as the next input line the first line of *file*. Note that the whole of*file* is appended, and that the appending is an irreversible process — that is, once **RUNOFF** encounters the **.APPEND** *control line* it will switch to the file *file* and continue from the first line of *file*. All lines following the **.APPEND** *control line* will not be processed by **RUNOFF**. The file *file* may, of course, itself call for appending of still another file, and so on.

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

.BREAK
.BR The lines before and after the **.BREAK** *control word* will not be run together by the *fill* mode of operation.

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

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

.FILL
.FI Enable *fill mode*. That means: Lengthen short lines by moving words from the following line; trim long lines by moving words to the following line. This is the default mode. **.NOFILL** disables the *fill* mode.

.HEADER *word1 word2 . . .*
.HE *word1 word2 . . .*
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.

.HEADING MODE*ar g*

.HM *arg*

This *control sequence* alters the mode of the running head to that specified by the parameter *arg*. Any of the following parameters are allowed for *arg:*

CENTER

The header will be centered on the page.

MARGIN

The header will be adjusted against the right margin of the page.

FACING

On even−numbered pages, the header will be adjusted against the left margin, on odd numbered pages against the right.

OPPOSED

The header will be adjusted against the opposite margin from the page number. In the absence of a **.HEADING MODE** *control sequence*, the default option is **.OPPOSED**.

.INDENT *n*

.IN *n* The argument *n* is a number. Set the number of spaces to be inserted at the beginning of each line to *n*. Indent is preset to 0.

.LINE LENGTH*n*

.LL *n* The argument *n* is a positive number. Set the line length to*n*. The line length is preset to 60.

.LITERAL

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

.NOFILL

.NF Disable *fill mode*. That means: Print all lines exactly as they appear without right adjustment or filling out. In *NOFILL* mode each input line produces one output line; further blank lines are output in this mode. Use the **.FILL** *control word* to restart *filling*.

.NOJUST

.NJ Disable *fill mode*.

.ODD PAGE

.OP This *control word* causes the current page to be printed out, and the next page to be numbered with the next higher odd page number.

.PAGE [*n*]

.PA [*n*]

Print page numbers. (The first page is not given a page number. It has instead a two−inch top margin. See also**Manuscript Conventions**, belo w.) If argument *n* is present, insert a page break and number the next page *n*. Note that**RUNOFF** does not print completely empty pages.

.PAGING MODE*ar g1 arg2 . . .*

.PM *arg1 arg2 . . .*

This *control sequence* alters the mode of page numbering to that specified by the arguments. The arguments may be in any order, and must be selected from the following list:

MARGIN

Page numbers will be adjusted against the right margin.

FACING

Odd page numbers are adjusted against the right margin, even page numbers are adjusted against the left margin.

CENTER

Page numbers are centered between the right and left margin.

TOP Page numbers are placed on the fourth line from the top of the page.

BOTTOM
Page numbers are placed on the fourth line from the bottom of the page.

OFF Page numbers are discontinued.

PREFIX *"string"*
The string of characters between quotation marks is prefixed to the page number. The quotation marks may be next to each other, in which case no prefix is used.

ROMANU
Page numbers will be printed in upper case Roman numerals.

ROMANL
Page numbers will be printed in lower case Roman numerals.

ARABIC
Page numbers will be printed in Arabic. (This is the normal mode.)

SET *n* Set the next page number to be the positive number *n*.

SKIP *n*
Skip *n* page numbers.

If in a single use of **.PAGING MODE** several arguments specify competing functions, the last one specified takes precedence. When the **.PAGING MODE** sequence appears in text at point A, all text up to A (and probably some text after A) will appear on a page controlled by the previous paging mode. The new *paging mode* will take effect on the next page. Then there is no danger of getting page numbers both at the top and bottom of the same page.

Use of the **TOP** parameter may conflict with the *heading mode*. If a heading and a page number should be printed in the same column, the page number will take precedence. In the absence of a **.PAGING MODE***contr ol sequence*, the default options are: **TOP MARGIN PREFIX "PAGE"**.

.PAPER LENGTH*n*
.PL *n* This *control word* is used for running off a documentation file on 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
.SS Copy is to be single spaced. This mode takes effect after the next line. (The normal mode is single space.)

.SPACE [*n*]
.SP [*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.

.UNDENT *n*
.UN *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.

RUNOFF ADDITIONS 1973

Here are described only the additional *control words* that are documented in the 1973 documentation.

Formats

.FORMAT *name*
This command causes subsequent text to be output under the control of tne specified format (see below at **.DEFINE FORMAT**). Each following logical line will be fit into the format until a **.FILL** or **.NOFILL** command 1s encountered.

.DEFINE FORMAT<*name*> <*pos*> <*field_ definition*> . . .
.END FORMAT
These commands define a format for use in producing tables, etc.

<*name*>
identifies the format. It can be activated by the **.FORMAT** *command*.

<*pos*> is the position and may be one of **LEFT**, **RIGHT**, or **CENTER**, and determines the overall positlon of the format with respect to the margins.

<*field_definition*>
There can be several arguments of this type. Each has the form:
<*type***>(<***letter***> ... <***letter***>)**

where the**<***type***>** is one of

L for left,

R for riqht,

C for center,

F fer fill, or **J** for justify.

The first three types define fixed fields; the text to be formatted must fit within the allocated space. The latter types define variable fields; the text will be handled as in normal fill mode processing.

A picture showing the manner in which text should be output follows the **.DEFINE FORMAT** command; following the picture should be an **.END FORMAT** command. The following lines give an example:

```
".DEFINE FORMAT SUMMARY L(A) F(C) C(B)"
"AAAA CCCCCCCCCCCCCCCCCCCCCCCC     BBBBBBB"
"     CCCCCCCCCCCCCCCCCCCCCCCC           "
".END FORMAT"
```

The first field of text is left justified; the second is centered; the third is subjected to *fill mode* processing without justification. After the first line of output is generated using this format, all subsequent lines are produced using the last picture 1ine. (Strictly speaking the third line is unnecessary.)

Text for formatted processing consists of a logical line (or paragraph). Each field except the last must be separated by *tab*. The*tab c haracter* is displayed here as backslash character (\).

The first field of text is **A**, the second **B**, etc. Typical input for our example might be:

```
|A\YES\/THIS IS SOME TEXT
TO BE FILLED.
```

The characters 1n the picture lines were 1nterpreted as follows. Contiguous sequences of letters determine the fie1d positions; non−alphabetic characters are output literally. (Note: **Q.QQ** wi11 not work, put the period "**.**" in the text. A sequence of characters written between double quotes is considered literal text. The*double quotes* are not output, and there is no way to use *double quote* as a *literal*.

Hyphenation Processing

.HYPHENATE

Enable *hyphenation mode*. This is the default when starting up. The **RUNOFF** program used a small *glossary* for splitting. In *hyphenation mode* **RUNOFF** would try to find a word 1n the glossary which is the same (except for the endings **−S**, **−ES**, **−ED**, and **−E**) as the word at the end of the line of text. When running in**gr off**(1) there are *glossaries* being much more complete than in **RUNOFF**.

.NOHYPHEN

Disable *hyphenation mode*.

.GLOSSARY *word*

This command inserts words into the *glossary* for use in *hyphenation*. Each word should have the form **hy-phen-ate** and be separated by spaces.

.HYPHENATION BREAK*n*

This command set the parameter which determines the allowable number of spaces to te inserted in a line before **RUNOFF** tried to hyphenate the last word. Each space counts ten points. If more than *n* points per word would have to be inserted, then *hyphenation* will be attempted. The initial setting of this parameter is 5 (one−half space per word).

Margin Controls

There are two types of margins involved in *RUNOFF*.

(1) The physical margins. These are determined by the nature of the printing device. The margins outline the area where it is physically possible to print characters.

(2) The logical margins. These can be set bv the user as he wishes. (Limits are imposed by the physical margins.) They are initialized for standard 8.5" by 11" printing.

Commands concerning vertical and horizontal margins are:

.PAGE LAYOUT TM, EM, TOL

This sets the vertical logical margins and vertical tolerence. Parameters are top margin, bottom margin and tolerance. The tolerance is used to determine where to break between pages on page overflows. If there is a line break within*TCL lines* of the bottom, **RUNOFF** will break the page there; otherwise it will fill the page completely.

.LINE LAYOUT LM, RM, NO, CS

This sets the logical left and right margin, the number of columns, and the number of spaces to insert between columns. These margins are used for the page headings. To adjust the relative text position, use the subsequent commands.

.REDUCE MARGIN LM, RM
.EXPAND MARGIN LM, RM
.END REDUCTION

These commands enable the user to indent a certain portion of his text using the first command, or **.UNDENT** his text using the second command. In either case the original margins are restored by the third command. The use of several **.REDUCE MARGIN** commends before the corresponding **.END REDUCTION**commands succesi vely indents the text more, and more. Thus these commands are 1ike brackets (i.e. recursive). **LM** is added to the left logical margin and **RM** is subtracted from the right logical margin in the first command. Just the opposite is done on the second command. Negative numbers are permitted. These commands do not effect the position of page headings.

.LAYOUT PLM, PRM, PTM, PBM, LL, LO

This command defines the physical margins in the followinq complex manner. (It should only be used for non−standard devices, normally this command should not be necessary.) The parameters are the physical left margin (in spaces), the physical right margin, the physical top 1ine, the physical bottom line, the line lenqth, and line origin. The first four parameters define the physical 1imits of the printing device. The final two parameters define the length of the logical line and its origin with respect to the left edge of the paper. Printing starts at column**LO + LM**, and ends at **LO + RM**, where **LM** and **RM** are the logical margins established by **.LINE LAYOUT**. When using the *facing feature*(see **.PAGING MODE**), the logical left mar gin is **LL − RM** on even pages, and the right margin is **LL − LM**. The parameters for the layout command must satisfy:

"min(LO + LL − PLM, PRM − LO) > max(PLM − LO, LO + LL − PRM),"
LL > 25, and **PBM − PTM > 6**

This command sets **LM** to 15, **RM** to **LL − 10**, **TM** to **PTM**, and to **PBM − 6**. (These margin settings produce the standard 1.5 inch left, and 1 inch right, too, and bottom margins.)

Initially **RUNOFF** sets the margins for *teletype* output to:

The printer layout is:

```
".layout 5, 137, 6, 66, 85, 15"
".page layout 6, 60, 4"
```

The logical margins must satisfy:

```
min(LL, PRM – LO, LO – LL – PLM) >= RM >
"     LM >= max(0, PLM – LO, LO + LL – PRM)",
"PBM >= BM > TM >= PTM" ", and"
BM – TM > TOL.
```

Paragraph Formatting

.PARAGRAPH SPACING*n*

This specifies how many lines are to be inserted between paragraphs. Initial setting = 1.

.PARAGRAPH INDENTATION*n*

This specifies how many additional spaces to insert at the beginning of a paragraph. Initial setting = 5.

.PARAGRAPH UNDENTATION*n*

This command is the same as **.PARAGRAPH INDENTATION** −*n*. That is, *n* fewer spaces are inserted at the beginning of the paragraph.

Special Line Justification and Control

These commands pertain to the next logical line. The end of the line should be designated with a break.

.CENTER

Center the next line.

.INDENT *n*

Indent the next line *n* spaces. If *n* is not provided, 5 is assumed.

.UNDENT *n*

Start the next line *n* spaces to the left of the normal margin. This command is the same as **.INDENT** −*n*.

.MARGIN

Justify the next 1ine against the right hand margin.

Heading and Paging

.HEADER *XXXXXXX*

RUNOFF accepts a heading to go on the first line of each paqe. The heading strinq is assured to start at the first non−blank character after thd control word and end at carriage return.

.HEADING MODE<*par am*>

<*param*> determines the position of the heading on the 11ne. <*param*> may be any of the following.

CENTER

The header will be centered on the 1ine.

MARGIN

The header will be adjusted against the riqht margin.

PAGING

On even numbered pages tne header is adjusted against the right margin. On odd pages it is adjusted against the left mergin.

OPPOSED

The header will be adjusted against the opposite margin from the page numter. This is the initial mode.

.PAGING MODE<*par am*>

This command determines the placing of the page number. All parameters are optional.<*par am*> may be anyone or more of the following oommands. In case of conflict the latest command wins.

CENTER
The page numbers are centered between the logical margins.

MARGIN
The page number is adjusted against the right margin.

FACING
On even numbered pages the number will be adjusted against the right margin. On odd numbered pages tne number will be adjusted against the left margin.

TOP Page numbers are placed on the first line.

BOTTOM
Page numbers are placed on the last line.

OFF Printing page numbers is discontinued.

PREFIX <*string*>
SECTION <*string*>
SUFFIX <*string*>
The strings of characters between quotation marks are used to form the page. string, which has the form:
<*prefix*><*section*><*page number*><*suffix*>

Any or all of these strings mav be null. The section string is considered to be part of the page number for purposes of indexing.

Initial mode is:

```
".PAGING MODE TOP MARGIN PREFIX "Page"" ".PAGING MODE
SECTION "" SUFFIX """
```

If neither page number nor heading is used, the text will start on the first logical line. Otherwise it will start on the fourth logical line. If the page number is at the bottom, text will end on the fourth line from the bottom. If the paging and headinq mode conflict, the page string overwrites the heading.

.ODD PAGE
This *control word* causes the ourrent page to be printed out and the next page to be started with the next higher odd number.

.PAGE *n*
If *n* is present, insert a page break and start numbering the next pege with *n*. Otherwise, turn the *paging mode* on and do not insert a page break.

.EJECT *n*
Insert a page break if either there are fewer than *n* lines left on the page or *n* is not present.

Lines and Spacing

.SINGLE SPACE
Sing1e space all 1ines within paragraphs. This is the initial state.

.DOUBLE SPACE
Double space all lines within paragraphs.

.SPACE *n*
Output *n* 1ine spaces. If *n* is not provided, 1 is assumed. In case of page overflow all remaining blank 1ines to be output are deleted.

.FIGURE SPACING*n*
This command is equivalent to **.EJECT** *n* followed by **.SPACE** *n*. These commands provide the only means of creatinq blank 11nes.

.BREAK
The lines before and after this command will not be run together in *fill mode*. A simpler w ay to get a line break is to insert one or more blank lines in the text.

.BEGIN GROUP
.END GROUP

The output lines enclosed between these two commands are forced to lie on a page. Thus this command acts in a manner similar to **.EJECT** *n*, where *n* has the 'right' value.

Miscellanous

.UNDERLINE

The following 1ine is underlined.

.LITERAL

The next line is taken as part of text whether or not it begins with dot.

.ESCAPE<*char*>
.SHIFT<*char*>
.TAB CHARACTER<*char*>

The given character becomes the *escape*, *shift*, or *tab* character. The parameter for the **.SHIFT** and **.TAB CHARACTER** commands may be null, if no *shift* or *tab* character is desired.

.DEFINE COMMAND<*name*>
.END COMMAND
.CALL <*name*>

These commands give the user the opportunity to combine text and control lines to form his own commands. All text and command lines between the first and second commands is stored away under *name*. When the third command is executed, the stored string is read and the commands within the string are executed. Recursion is not permitted.

.INDEX <*phrase*>, <*phrase*>

RUNOfF saves the first phrase in the main index table and the second phrase (if any) in a sub−index table associated with the first phrase.

The index is formatted and output after the last page of text. Two built−in but redefinable formats, *RINDEX* and *SINDEX*, are used to format the index as shown in the following example.

```
Algorithms, 40, 78,      \" uses RINDEX
  analysis of, 27,       \" uses SINDEX
```

The following lines give the initial definitions for the indexing formats.

```
".define format RINDEX f(A)"
"AAAAAAAAAAAAAAAAAAAAAAAAAA"
"    AAAAAAAAAAIAAAAAAAAAAA"
".end format"
".define format SINDEX f(A)"
"  AAAAAAAAAAAAAAAAAAAAAAAAA"
"    AAAAAAAAAAIAAAAAAAAAAA"
".end format"
```

In order to get an index output in two columns,

```
.LINE LAYOUT 15, 75, 2, 4
```

should be the last line of the input.

RUNOFF DOCUMENTATION 1974

This document is the best documentation about text lines. Parts of that is used in the document of 1981.

Command Lines

All lines beginning with a period (dot) are *RUNOFF* command lines. All other lines are text lines.

A command line consists of a period, following by a command, which can consist of one or more words, or a 2- or 3-letter abbreviation, followed by 0 or 1 or more arguments. This can be followed by a comment, which is preceeded by an exclamation point (bang character) **!**. In ancient *RUNOFF*, The comment didn't need to be preceeded.

In this document, several command or text lines can be appended into a multiple line if these parts are separated by a semi-colon **;**. If 2 commands are appended, the semi-colon may be omitted, because the period

is a sufficient separator.

Text Lines

There are 2 modes of text line structures:

ancient style
: This is the original style. Due to very old hardware, there were only input methods for upper case characters. But typewriters and printers were able to use both upper and lower case. So the text lines are all in upper case with special characters that are case-shifters for the printing.

newer style
: By better hardware, it was possible to use input methods with both upper and lower case. Here the text lines are like those in later *roff* and *groff* mode.

This paragraph describes only the *ancient style* of text lines.

This text is filled and justified such as with the later *roff* language. Just as in *roff*, the filling, justification, and line break can be turned on or off by commands.

Case Changing of Text Lines

In this section, the specification of case for files prepared on an upper case terminal is documented. There are special characters that in printing act as case-shifters for ASCII characters into lower (ASCII code 97 to 122 decimal) or upper case (ASCII code 65 to 90 decimal).

The lower case mode seems to be the default mode. Also, according to existing old *RUNOFF* files, each text line starts with this default mode.

single circumflex `^`
: The following ASCII character is shifted into upper case.

single back-slash `\`
: The following ASCII character is shifted into lower case.

double circumflex `^^`
: The case mode is shifted into upper case.

double back-slash `\\`
: The case mode is shifted into lower case.

A common example with starting mode in lower case for these 4 special characters is:

^HERE IS A ^SAMPLE ^SENTENCE IN ^^UPPER CASE\ AND LOWER CASE.

is printed as:

Here is a Sample Sentence in UPPER CASE and lower case.

Further special Characters in Text Lines

ampersand `&`
: This is used for underscoring the next following character. For example:

&s&o&f&t&w&a&r&e

becomes:

. nop software

in the output or printing.

circumflex and ampersand `^&`
: This is used for underscoring all following characters except for blanks.

RUNOFF ADDITIONS 1981

Source File Format

The source file contains the textual material which will appear on the final copy, plus information to specify formatting. Most importantly, upper and lower case information also may be supplied so that copy can be prepared on the terminal or other such device which can input only upper case letters. All command information consists of regular *ASCII* printing characters so that a listing of the source file may be examined if

the final copy is not exactly as desired.

All material in the source file is taken to be source text except those lines beginning with a period. A line beginning with a period is assumed to be a command, and must match one of those listed below. The commands provide the formatting information, and control various optional modes of operation.

Usually the text is *filled* and *justified* as it is processed. That is, the program *fills* a line by adding successive words from the source text until one more word would cause the right margin to be exceeded. The line is then *justified* by making the word spacings larger until the last word in the line exactly meets the right margin.

The user may occasionally wish to reproduce the source text exactly, which is done by disabling *filling* and *justification* or by use of the `.LITERAL` command. The program may be set to *fill* but not *justify*, in which case the output will be normal except that lines will not be justified to the right margin. The program may also be set to *justify* but not *fill*, although this would probably produce peculiar results and is not recommended.

When the *fill mode* is on, spaces and carriage returns occurring in the source text are treated only as word separators. Multiple separators are ignored.

Some of the commands cause a BREAK in the output. A *break* means that the current line is output without justification, and the next word goes at the beginning of the next line. This occurs at the end of paragraphs.

The program will advance to new pages as necessary, placing the title (if given) and the page number at the top of each page. The user may call explicitly for a *page advance* where desired, and may inhibit the occurrence of a *page advance* within specified material.

Case Information of Text Lines

The information in this section documents the style of the text lines that could only be upper case in the ancient *CTSS* computers of the early 1960s.. It seems as if this section documents the old *RUNOFF* style.

Specification of *case* for files prepared on the terminal is done with two characters, circumflex (`^`, *136 octal*), and `back-slash` (\, *134 octal*). The appearance of a circumflex causes the letter immediately following to be transmitted in upper case. The appearance of a `back-slash` causes the letter immediately following to be converted to lower case. Any letter not preceded by one of these characters is transmitted in the current mode.

The mode is initially *upper case* **(wrong!)**. (The initial mode seems to be *lower case*. That's the style used by the available *RUNOFF* files in the PDP-10 archive and by the example below. And each newline seems to switch back to the initial mode, well: *lower case*).

The mode is changed by the occurrance of two successive *case* control characters. Two `circumflexes` `^^` cause the mode to be set to *upper case*, and two `back-slashes` `\\` cause the mode to be set to *lower case*.

The use of the above corresponds to the use of the `shift` and `shift-lock` keys on a typewriter. Usually, typing appears in *lower case*. To type one letter in *upper case*, the `shift` *key* is used. The `shift-lock` is set to type a series of *upper case letters*, after which it is released.

The following shows the uses of the case control characters:

```
^HERE IS A ^SAMPLE ^SENTENCE IN ^^UPPER CASE\\ AND LOWER CASE.
```

becomes:

```
Here is a Sample Sentence in UPPER CASE and lower case.
```

NOTE

Case conversion takes place only on *ASCII codes* `101` to `132 octal`, that is, the *upper case* letters. Any actual *lower case* letters (codes `141` to `172 octal`) appearing in the source will be transmitted unchanged. If the source is prepared on a device such as a *DECwriter* or model *37 Teletype* which produce letters of the proper *case*, the mode should be set to *upper case* at the beginning of the file and left unchanged for the remainder.

An additional character, less−than (<, 074 octal), capitalizes the entire word it precedes. It then returns the file to the current *case*. This character is not engaged unless preceded by the **.FLAGS CAPITALIZE** *command*. Similarly, the **.FLAGS HYPHENATE** *command* engages the special character equals (=,075 octal), which causes *hyphenization* to be suspended for the word it precedes.

Special Characters

& *Ampersand* Underscoring

The character*ampersand* (**&**, 046 octal) is used to specify . nop underscoring . The *ampersand* will cause the character following it to be *underscored*, e.g. **&f&o&o** becomes . nop **foo**.

Underlining of a string of characters can also be specified in a manner similar to that of the operations described above. ; An appearance of*ampersand* preceded by *circumflex* **^&** will cause underlining of all following characters except space. An appearance of preceded by **\&** will disable this mode.

Number Sign *Explicit space*

It is occasionally necessary to include *spaces* in the text which should not be treated as *word separators*. For this purpose, *RUNOFF* treats the *number−sign* character (**#**, 043 octal) as a *quoted space*; i.e. it will print as exactly one *space* in the output, will never be expanded nor changed to a *carriage return*.

_ Underline *Quote next character*

To allow the appearance of the *special characters* (*ampersand* used as a *quote character*. The character immediately following an underscore will be transmitted to the output with no formatting effect. Theunderscore itself is thus another *case r equiring quoting*. The following five cases occur: _**&**, _**^**, _****, __, _**#**, _**{**, _**}**, and _**|**.

^ *Circumflex* *Upper−case shift or mode lock*

As described above, the*circumflex* character **^** is used to con vert the letter following to *upper−case*. It is also used to lock the*case mode* in *upper case*, and the *underline mode* to . nop underline all text. If it is to appear in the printed text, it must be preceded by the *quote character* _**^**.

**** *back−slash* *lower−case shift or mode unlock*

As described above, the *back−slash* character **** is used to output the letter following in *lower−case*. It is also used to lock the*case mode* in *lower−case*, and to disable *underlining*. If it is to appear in the printed text, it must be preceded by the *quote* character _****.

< *less−than* *Capitalize next word*

If **.FLAGS CAPITALIZE** has been engaged, the*less−than* character **<** is a special character used to capitalize the entire word it precedes. If it is to appear in the printed text, it must be preceded by the *quote* character _**<**.

= equals−sign *hypenation disable*

If **.FLAGS HYPHENATE** has been engaged, the *equals* character **=** used to disable *hyphenation* for the word it preceds. If it is to appear in the printed text, it must be preceded by the *quote* character _**=**.

{ left−brace *Reverse half−linefeed*

If the output device type is no **N**, then the *left* and *right braces* are used for *superscripting* and *subscripting*. The*left−br ace* (**{** 173 octal) produces a *reverse half−linefeed*. When combined with the *right brace* (**}** 175 octal) scripting is created; e.g. **{super}** becomes **{super}**, and **}sub{** becomes **}sub{**.

} right−brace *forward half−linefeed*

As described above, the *right brace* (**}** 175 octal) when coupled with the *left brace* will produce scripting. This will only occur when a scripting output device is selected.

| vertical−bar *Engage/disengage alternate character set*

The *vertical bar* (**|**, 174 octal) acts as an on/off switch. It will alternatly transmit a *shift−out* and a *shift−in* character to change the selected character set; e.g. **|ABC|** becomes

*Ctrl−N***ABC***Ctrl−O*.

Ctrl−N Control-N *enchacement on/off; red/black ribbon*
If an *Anderson-Jacobson output device* is selected, a *Control−N* (*Ctrl−N*, 016 octal) will alternatly engage and disengage the print enchancement; e.g. **NFOO***Ctrl−N* becomes **FOO**.

If a *Diablo output device* is selected, a *Control−N* will alternatly select the red and black ribbon color; e.g. **NFOO***Ctrl−N* becomes **FOO**.

Special Characters Overview

Below is a list of *RUNOFF*'s special characters. To appear in the text, each must be preceded by the character (itself a special character).

^ shift character for upper case

**** shift character for lower case

< flag character for upper case. Only becomes a special character if **.FLAGS CAPITALIZE** is engaged.

quoted space character

= flag character for disabling hyphenation. Only becomes a*special c haracter* if **.FLAGS HYPHENATE** is engaged.

& underscore

_ *quote special character*

{ *reverse half−linefeed* for scripting

} *forward half−linefeed* for scripting

| switch on/off the *alternate character set*

^n switch on/off the print enchancement or switch to the red/black ribbon color

RUNOFF Commands

The following *commands* will be recognized if they are at the beginning of a line started with a period. Any line in the source file beginning with a period is assumed to be one of these *commands*. If it is not, an *error diagnostic* will be typed and the line will be ignored. Some *commands* take one or more decimal numeric arguments. These are separated from the*command* by a space. More than one *command* may be entered on a single line by separating the *commands* with a *semicolon* '**;**' or a *period* '**.**'.

Multi−word commands may appear in any form. Thus, **.NO HEADER** and **.NOHEADER** are both legal.

Many *commands* may be abbreviated. Standard*abbr eviations* are given below each *command*.

Text Formatting Commands

.BREAK
.BR causes a *break*, i.e. the current line will be output with no *justification*, and the ne xt word of the source text will be placed at the beginning of the next line.

.SKIP *n*
.SK *n*
.S *n* causes a BREAK after which is multiplied by the number of *spaces* between lines. The result is the number of lines *skipped*. Output is advanced to the top of the next page if there is no room on the current page. If the current page is empty, **.SKIP** does nothing.

.BLANK *n*
.B *n* causes the current line to be output with no *justification*, skips *n* line spaces, and then starts output of the current source text. **.BLANK** is like **.SKIP**, except that the *space* to be left is independent of line spacing. If the page is empty, **.BLANK** does nothing.

.FIGURE *n*
.FG *n* leaves *n* lines blank to make room for a figure or diagram. If fewer than *n* lines remain on the current page, text continues to *fill* this page, then the page is adv anced and *n* blank lines are left at the

top of the next page.

.INDENT *n*

.I *n* causes a and sets the next line to begin *n* spaces to the right of the left margin. The*n* can be ne gative to allow beginning a line to the left of the left margin. However, a line cannot begin to the left of column 0. If *n* is not supplied, the current paragraph indent is used.

.PARAGRAPH *n, v, t*

.P *n, v, t* causes a *break* and formats the output paragraphs. The *n* is optional and, if present, sets the number of spaces the paragraph is to be indented. The default value for *n* is **5** (*n* can also have a negative value). *v* is the vertical spacing between paragraphs. *v* can range from **0** to **5**. (**1** is *single spacing*, **2** is *double spacing*, etc.) *t* causes an automatic *.TEST PAGE* (see the **.TEST PAGE** *command*).

.CENTER *n;text*

.CENTRE *n;text*

.C *n;text*

causes a *break* and centers the following text in the source file. The centering is over column **(***n* **+** *left margin***)/2**. If *n* is not given, it is assumed to be the *right margin*.

NOTE

CENTER, **RIGHT MARGIN**, **LEFT MARGIN**, **PAGE SIZE**, and **STANDARD** take both relative and absolute values. Relative values are expressed as **+***n* or **−***n*, while absolute values of *n* are unsigned.

.FOOTNOTE *n*

.FN *n* saves *n* lines at the bottom of the current page for a *footnote*. The*n* is multiplied by the number of spaces set with the **.SPACING** command. If insufficient room remains on the current page, space is allocated at the bottom of the following page. The text of the *footnote* should be gin on the line following the **.FOOTNOTE** command. *Indentation*, *case lock*, *justify*, *margins*, *spacing*, and *fill* are preserved around footnotes. However, *commands* that affect page formatting are illegal in a *footnote*. *Tab stops* are illegal because they are not preserved. A footnote within a footnote is also illegal.

The actual space taken by a footnote can be more or less than specified by *n*. If necessary`adjust` *n* after examining a draft printout.

The *footnote* is terminated with a line beginning with an exclamation point (the remainder of which is ignored).

.NOTE *text*

.NT *text*

starts an *indented note*. This command*blanks* **2**, reduces both *mar gins*, *centers* the text (if no text is given, it centers the word **NOTE**), and then *blanks* **1**. At this point you enter the text of the *note*. If the left margin is at **0**, the *margin reduction* is **15**, otherwise it is **5**.

.END NOTE

.EN terminates the **.NOTE** command, *blanks* and reverts the margins and spacing modes to their settings before the last **.NOTE** command.

.LIST *n*

.LS *n* starts an indented list with *n* spacing, moves the left margin **9** spaces to the right for the first **.LIST** command, and **4** more spaces for each subsequent nested **.LIST**. The normal *fill* and *justify modes* remain in effect. Therefore, you must disengage them just after the **.LS** command if you want a ragged right.

.LIST ELEMENT;*te xt*

.LE; *text*

starts an item in the list, used in conjunction with the **LIST** comand. The elements are numbered sequentially and the number is given a negative indent so that the list lines up. The number is followed by a *period* and tw o *spaces* so that the indent will be by **−4**. The *list elements* are separated

by the standard paragraph spacing and*TEST PAGE*. If you w ant to type the text on the same line as the command, you must separate the text from the command with any number of intervening *spaces* or *tabs*, or (optionally) one *semicolon*.

.END LIST
.ELS terminates the **.LIST** command and returns to settings before the last **.LIST** command.

.COMMENT *text*
.; *text* causes the line to be ignored. The text is not printed in the output file, but rather is used as a *comment* line in the source text.

Page Formatting Commands

.PAGE .PG causes a *break* and an *advance* to a new page. If the current page is empty, this *command* does not *advance* the page. Just like an *automatic page advance*, this *command* prints the *title* (if given) and *pa ge numbers* on every page.

.TEST PAGE*n*
.TP *n* causes a *break* followed by a *conditional page advance*. It skips to the next page if fewer than *n* lines are left on the page. This capability is to ensure that the following *n* lines are all output on the same page. This *command* has the form *t* as an optional argument to the **.PARAGRAPH** command.

.NUMBER *n*
.NM *n* starts page numbering. This is the default so there is no reason to issue this command unless page numbering is disengaged. If*r esumption* of page numbering is desired at a certain page, specify *n*.

.NONUMBER
.NNM disengages page numbering. However, pages continue to be counted, so that the normal page number can appear if page numbering is re−entered with the **.NUMBER** command.

.CHAPTER *text*
.CH *text*
starts a new chapter using the text as the title of the chapter. This*command* acts as if the follo w-ing *command string* were entered:

```
".BREAK;.PAGE;.BLANK 12;.CENTER;CHAPTER n"
```

The *n* is incremented by **1** automatically. After the CHAPTER*n* is typed on the page,

occurs. This*command* then resets the *case*, *mar gins*, *spacing*, and *justify/fill modes*. It also clears an y *subtitles* and sets the *chapter name* as the *title*.

.NUMBER CHAPTER *n*
supplies a number *n* to be used in a subsequent **.CHAPTER** command. **.NUMBER CHAPTER** would be used when a *chapter* of a document occupies a source file of its own. In such a case, **.NUMBER CHAPTER**w ould be the first command of the source file.

.HEADER LEVEL*n te xt*
.HL *n text*
starts a section at the level specified and takes the following text as the header. *n* can range from **1** to **5**. The sections are incremented by **1** automatically, and the number is output in the form *i***.***j***.***k***.***l***.***m*. If this is a chapter oriented document, the *i* is the chapter number. Otherwise, it is the number of the **.HL 1** level. This command acts as a

.BREAK;.TEST PAGE 9;.BLANK 3

followed by the *section number*, two *spaces*, and the *section name*. *HEADER LEVELS* **1** and **2** end with a *break*. *HEADER LEVELS* **3**, **4**, and **5** end with a space−dash−space combination (#−#).

.TITLE *text*
.T *text* takes the remaining text as the title and outputs it on every page at line **0**. The default is no title. If a *title* is desired, this *command* must be entered in the source file.

.FIRST TITLE*te xt*
.FT *text*

Same as **.TITLE**, but used to specify the title to be printed on the first page of the document. This command must precede all text in the source file. Use of the **.FIRST TITLE** command is the only way to print a title line on the first page of the document.

.SUBTITLE *text*
.SUBTTL *text*
.ST *text*

takes the remaining text as the *subtitle* and outputs it on every page. It appears directly under the title. The*subtitle* is not *indented*, b ut *indentation* can be achieved by typing leading spaces.

.INDEX *text*
.X *text* takes the remaining text on the line as a keyword and adds it, along with the current *pa ge number*, to the internal index buffer. The command does not cause a*br eak*. It should appear immediately before the item to be *indexed*. A keyword may be *indexed* more than once.

.DO INDEX*te xt*
.DX *text*

forces a new page, centers the text, if given, otherwise it centers the word *INDEX*. This command prints the entire contents of the index buffer. Entries are printed in alphabetic order and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The page number of each entry is placed on the same line as the entry and in the middle of the page. Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

.PRINT INDEX
.PX forces a new page after which it prints the entire contents of the index buffer. Entries are printed in alphabetical order and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The number of the first page on which each entry appeared is put on the same line as the entry, beginning at the middle of the line (midway betweeen the left and right margins). Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

.PRINT INDEXand **.DO INDEX** perform the same task. The only dif ference is that **.PRINT INDEX** does not interrupt the normal chapter and page sequencing.

.SUBPAGE

executes a **.PAGE** with page numbering suspended. The page number is unchanged, but letters are appended to the page number. This permits insertion of additional pages within an existing document without changing the existing page numbering.

.END SUBPAGE

disengages the **.SUBPAGE** command by executing a **.PAGE** command with page numbering resumed.

.APPENDIX *text*

.AX *text*

starts a new appendix using the text as the title of the appendix. This command acts as if the following command string were entered:

```
".BREAK;.PAGE;.BLANK 12;.CENTER;APPENDIX a"
```

The a is a letter that is incremented alphabetically automatically. After the**APPENDIX A** is typed on the page,

occurs. This command then resets the*case*, *mar gins*, *spacing*, and *justify/fill* modes. It also clears an y subtitles and sets the appendix name as the title.

.NUMBER APPENDIX*a*

supplies a letter *a* to be used as the letter for a subsequent **.APPENDIX** command.

.HEADER *arg*

.HD *arg*

causes the page header (*title*, *subtitle*, and *page number*) to be printed. *arg should be* **UPPER** to specify *upper case characters* for the title text, **LOWER** to specify *lower case*, or **MIXED**. The initial setting is **.HEADER UPPER**.

.NOHEADER

.NHD causes the page header (*title*, *subtitle*, and *page number*) to be omitted. The header lines are completely omitted, so that text begins at the top of the page with no *top margin*.

Mode Setting Commands

.JUSTIFY

.J causes a break and sets subsequent output lines to be justified (initial setting). The *command* increases the spaces between words until the last word exactly meets the right margin.

.NOJUSTIFY

.NJ causes a *break* and prevents *justification* of subsequent output lines to make a ragged right margin.

.FILL

.F causes a break and specifies that subsequent output lines be filled (inital setting). Sets the justification mode to be that specified by the last appearance of **.JUSTIFY** or **NOJUSTIFY**. **.FILL** adds successive words from the source text until the adding of one more word will exceed the right margin. It stops before putting the last word in. (If *hyphenation* has not been disabled, **RNO** will attempt to *break* words which cause line overflow into syllables.)

.NOFILL

.NF disengages the *fill* and *justify modes*. This*command* is used to permit typing a table.

NOTE

1. The *nofill−nojustify mode* need be used only where there are several lines of material to be copied exactly. A single line example will not require using these commands if there are breaks before and after.

2. Normally **.FILL** and **NOFILL** are used to turn both *filling* and *justification* on and off. It is usually desirable to do both. A subsequent appearance of a *justification command* will override the *fill command* however.

3. Because of the action of **.FILL**, a single occurrance of **NOJUSTIFY** will cause the remainder of the file to be *unjustified*, with *filling* as specified. In order to *justify* but *not fill* (not recommended), a **.JUSTIFY** command must follow every **.NOFILL** command.

.UPPER CASE

.UC sets the output mode to *upper case*. This comand acts the same as typing two **^^**. This is the default mode. There is no need to type this command unless the mode was previously altered to *lower case*.

.LOWER CASE

.LC sets the typeout mode to *lower case*. This command acts the same as typing two *back−slashes* ****.

.FLAGS CAPITALIZE

.FL CAPITALIZE

enables the **<** character to *capitalize* the entire word it precedes. It then returns the file to the current case mode. This *special character* is usually `off` and must be typed at the very beginning of the source text to enable this character. Typing a space or another *less−than* **<** returns the file to the current *case lock*.

.NO FLAGS CAPITALIZE
.NFC disengages the **FLAG CAPITALIZE** command (inital setting).

.HYPHENATION
.HY engages *hyphenization* (initial setting).

.NO HYPHENATION
.NHY disengages IR hyphenization .

.FLAGS HYPHENATE
.FL HYPHENATE
enables the *equals character* **=** to disengage *hyphenization* for the word it precedes. This *special character* is initially `off` and must be typed at the beginning of the source file to enable this character. The*FLAGS HYPHENATE c haracter* is used to disengage *hyphenization* for words improperly *hyphenated* by the *hyphenization algorithm*.

.NO FLAGS HYPHENATE
.NFH disengages the **.FLAGS HYPHENATE** command (initial setting).

.NFL disengages the **.FLAGS CAPITALIZE** and the **.FLAGS HYPHENATE** commands (initial setting).

.PERIOD
.PR enables printing of two *spaces* after every *period* **.** that is followed by at least one *separator character* (initial setting).

.NOPERIOD
.NPR disengages conversion of *period/separator* to *period/two spaces*.

.LITERAL
,FONT CB .LIT
disengages *fill/justify* to permit printing of text exactly as entered in source file.

.END LITERAL
.ELI used after **.LITERAL** command to re−engage *fill/justify*.

Parameter Setting Commands

.LEFT MARGIN*n*
.LM *n* sets the left margin to *n*. The*n* must be less than the right mar gin but not less than **0**. The initial setting is **0**. If *n* is not supplied, **0** is used.

.RIGHT MARGIN*n*
.RM *n* sets the right margin *n*. The*n* must be greater than the left mar gin. The initial setting is**60**. If *n* is not supplied, the current page width (set with the **.PAGE SIZE** command) is used.

.PAPER SIZE*n*,*m*

.PAGE SIZE*n*,*m*

.PS *n*,*m*
sets the size of the page *n* lines by *m* columns and sets the right margin to *m*. The default setting is **58,60**.

.PITCH *n*,*m*
.PIT *n*,*m*
sets the horizontal and vertical pitch on a supporting output device. The horizontal pitch is*n* and is specified as characters per inch. The value must divide evenly into **60** for *Anderson−Jacobson devices*and into **120** for *Diablo devices*. The def ault is **12**.

The vertical pitch is *m* and is specified as lines per inch. *m* must divide evenly into **48**. The default is the hardware setting. *n* and *m* may be set independently.

.SPACING *n*
.SP *n* sets the number of spaces between lines. The *n* can range from **1** to **5**. The default setting is **1**. **.SPACING 1**is lik e *single spacing* on a typewriter and **.SPACING 2** is like *double spacing*.

.SPACING 2puts one *blank line* between lines of te xt.

.STANDARD *n*

.SD *n* returns all parameters, except the pitch settings, to their initial settings and sets *n* as the page width. If **.STANDARD 60** is specified, mar gins are reset **.LM 0**, **.RM 60**, **.PAGE SIZE 58,60**, **.SPACING 1**, **PARAGRAPH INDENT 5**, and *fill* and *justify* are enabled. **.STANDARD 70**sets right mar gin to **70** and *page size* to **58,70**.

.TAB STOPS*n*,*n*,...

.TS *n*,*n*,...

sets tabs. The *n* must be greater than **0** and listed in ascending order. If tabs already exist, the issuing of another **.TAB STOPS** command clears all previous *tabs* before setting new ones. The *default tabs* are set at eight−column intervals to match the Digital hardware standard. These *tabs* are at columns **8**, **16**, **24**, **32**, **40**, **48**, **56**, **64**, **72**, and **80**. The tabs are converted to the appropriate number of non−expandable spaces. If there are no regular spaces to the left of the *tabs*, they will print out at the appropriate position, even if *fill* is on. If *liter al* is on, the *tabs* are not converted to *spaces*, but are output as *tabs*.

.AUTOPARAGRAPH

.AP causes any *blank line* or any line starting with a *space* or *tab* to be considered as the start of a new paragraph. This command allows normally typed text to be *justified* without special commands. It does not cause a paragraph if *blank lines* are followed by a command.

.NOAUTOPARAGRAPH

.NAP disengages the *AUTOPARAGRAPH* mode.

List of Commands (Alphabetical)

This list of commands is a table over 5 pages.

Command or Abbreviation		Related Commands
.AP	(=**.AUTOPARAGRAPH**)	
.APPENDIX *text*	(=**.AX**)	**.NUMBER APPENDIX** *a*
.AX	(=**.APPENDIX**)	
.AUTOPARAGRAPH	(=**.AP**)	**.NOAUTOPARAGRAPH** (**.NAP**)
.B	(=**.BLANK**)	
.BLANK *n*	(=**.B**)	**.SKIP** *n* (**.S**)
.BR	(=**.BREAK**)	
.BREAK	(**.BR**)	
.C	(=**.CENTRE**)	
.CENTER	(=**.CENTRE**)	
.CENTRE *n*;*text*	(=**.C**)	
.CH	(=**.CHAPTER**)	
.CHAPTER *text*	(=**.CH**)	**.NUMBER CHAPTER***n*
.COMMENT *text*		
.DO INDEX*te xt*	(=**.DX**)	**.PRINT INDEX** (**.PX**)
.DX	(=**.DO INDEX**)	
.ELS	(=**.END LIST**)	
.EN	(=**.END NOTE**)	
.END LIST	(=**.ELS**)	**.LIST** *n* (**.LS**)
.END LITERAL	(=**.ELI**)	**.LITERAL** (**.LIT**)
.END NOTE	(=**.EN**)	**.NOTE** *text* (**.NT**)
.END SUBPAGE		**.SUBPAGE**
.F	(=**.FILL**)	
.FG	(=**.FIGURE**)	

Command or Abbreviation		Related Commands
.FIGURE *n*	(=**.FG**)	
.FILL	(=**.F**)	**.NOFILL** (**.NF**)
.FIRST TITLE*te xt*	(=**.FT**)	**.TITLE** *text*
.FLAGS CAPITALIZE	(=**.FL CAPITALIZE**)	**.NO FLAGS CAPITALIZE** (**.NFL**)
.FLAGS HYPHENATE	(=**.FL HYPHENATE**)	**.NO FLAGS HYPHENATE** (**.NFH**)
.FN	(=**.FOOTNOTE**)	
.FOOTNOTE *n*	(=**.FN**)	
.FT	(=**.FIRST TITLE**)	
.HD	(=**.HEADER**)	
.HEADER *arg* [*arg*=**UPPER**, **LOWER**, or **MIXED**]	(=**.HD**)	**.NOHEADER** (**.NHD**)
.HEADER LEVEL*n te xt*	(=**.HL**)	
.HL	(=**.HEADER LEVEL**)	
.HY	(=**.HYPHENATION**)	
.HYPHENATION	(=**.HY**)	**.NO HYPHENATION** (**.NHY**)
.I	(=**.INDENT**)	
.INDENT *n*	(=**.I**)	
.INDEX *text*	(=**.X**)	
.J	(=**.JUSTIFY**)	
.JUSTIFY	(=**.J**)	**.NOJUSTIFY** (**.NJ**)
.LC	(=**.LOWER CASE**)	
.LE	(=**.LIST ELEMENT**)	
.LEFT MARGIN*n*	(=**.LM**)	**.RIGHT MARGIN***n* (**.RM**)
.LIST *n*	(=**.LS**)	**.END LIST** (**.ELS**)

Command or Abbreviation		Related Commands
.LIST ELEMENT;*text*	(=**.LE**)	**.END LIST** (**.ELS**)
.LIT	(=**.LITERAL**)	
.LITERAL	(=**.LIT**)	**.END LITERAL** (**.ELI**)
.LM	(=**.LEFT MARGIN**)	
.LOWER CASE	(=**.LC**)	**.UPPER CASE** (**.UC**)
.LS	(=**.LIST**)	
.NAP	(=**.NOAUTOPARAGRAPH**)	
.NF	(=**.NOFILL**)	
.NFC	(=**.NO FLAGS CAPITALIZE**)	
.NFH	(=**.NO FLAGS HYPHENATE**)	
.NFL	(=**.NO FLAGS CAPITALIZE** and **.NO FLAGS HYPHENATE**)	
.NHD	(=**.NO HEADER**)	
.NHY	(=**.NO HYPHENATION**)	
.NJ	(=**.NO JUSTIFY**)	
.NM	(=**.NUMBER**)	
.NNM	(=**.NO NUMBER**)	
.NOAUTOPARAGRAPH	(=**.NAP**)	**.AUTOPARAGRAPH** (**.AP**) **.PARAGRAPH** *h*,*v*,*t* (**.P**)
.NOFILL	(=**.NF**)	**.FILL** (**.F**)
.NO FLAGS CAPITALIZE	(=**.NFL**)	**.FLAGS CAPITALIZE** (**.FL CAPITALIZE**)
.NO FLAGS HYPHENATE	(=**.NFH**)	**.FLAGS HYPHENATE**
.NOHEADER	(=**.NHD**)	**.HEADER** *arg* (**.HD**)

Command or Abbreviation		Related Commands
.NO HYPHENATION	(=**.NHY**)	**.HYPHENATION** (**.HY**)
.NOJUSTIFY	(=**.NJ**)	**.JUSTIFY** (**.J**)
.NONUMBER	(=**.NNM**)	**.NUMBER** *n* (**.NM**)
.NOPERIOD	(=**.NPR**)	**.PERIOD** (**.PR**)
.NOTE *text*	(=**.NT**)	**.END NOTE** (**.EN**)
.NPR	(=**.NO PERIOD**)	
.NT	(=**.NOTE**)	
.NUMBER APPENDIX*a*		**.APPENDIX** *text*
.NUMBER CHAPTER *n*		**.CHAPTER text**
.P	(=**.PARAGRAPH**)	
.PAGE	(=**.PG**)	
.PAGE SIZE	(=**.PAPER SIZE**)	
.PAPER SIZE*v*,*h* (=**.PS**, **.PAGE SIZE**)		
.PERIOD	(=**.PR**)	**.NOPERIOD** (**.NPR**)
.PG	(=**.PAGE**)	
.PITCH	(=**.PIT**)	
.PRINT INDEX	(=**.PX**)	**.DO INDEX***te xt* (**.DX**)
.PS	(=**.PAPER SIZE**)	
.PX	(=**.PRINT INDEX**)	
.RIGHT MARGIN *n*	(=**.RM**)	
.RM	(=**.RIGHT MARGIN**)	
.S	(=**.SKIP**)	

Command or Abbreviation		Related Commands
.SD	(=**.STANDARD**)	
.SKIP *n*	(=**.S**)	**.BLANK** *n* (**.B**)
.SP	(=**.SPACING**)	
.SPACING *n*	(=**.SP**)	
.ST	(=**.SUBTITLE**)	
.SUBPAGE		**.END SUBPAGE**
.SUBTITLE *text*	(=**.ST**)	
.T	(=**.TITLE**)	
.TAB STOPS*n,n, . . . ,n*	(=**.TS**)	
.TEST PAGE*n*	(=**.TP**)	
.TITLE *text*	(=**.T**)	
.TP	(=**.TEST PAGE**)	
.TS	(=**.TAB STOPS**)	
.UC	(=**.UPPER CASE**)	
.UPPER CASE	(=**.UC**)	**.LOWER CASE** (**.LC**)
.X	(=**.INDEX**)	

EXPERIMENTAL ADDITIONS 1965

These *control words* are documented in *Saltzer*'s documentation of 1965. It is unsure whether they were really implemented.

In this documentation, all *control words* are written in *lower case*. The writing in*upper case* is not mentioned, the same is true for *abbreviations*. So this documentation uses only*lower case*.

.FIGURE

This *control word* turns control over to a *figure processor*, which creates in *core memory* a representation of a *flow diagram* under the control of a few special *control words*. When the*control word* **.END FIGURE** is encountered, the completed picture is printed immediately on the page being generated if there is room on that page; otherwise the *figure* will appear at the top of the next page.

Text following the **.END FIGURE** *control word* will be smoothly attached to text before the **.FIGURE**. No break is generated. (Restriction: If a *figure* is being held for placement at the top of the next page, another *figure* may not be encountered before the first one is printed.)

The only *control words* which are recognized when in the *figure processor* are the following three: **.FRAME**, **.BOX**, and **.END FIGURE**.

.FRAME *m n*

This *control word* intitalizes the *figure processor* by giving the height and width of the figure to be produced. *m* is the height, in lines; and *n* is the width, in characters. (Note that a 1050 types 6 lines per inch, and 10 characters per inch.)

Any attempt to place items in the picture which extend beyond the boundaries will cause an error comment to be generated. *m* and *n* must both be less than 100 and their product must be smaller than 5400. We may now think of the *figure* to be produced as an array of *m* times *n* elements.

.BOX *i j*

The text on the lines following this *control word* will be placed in the *figure* such that the first character on the first line following the **.BOX** will appear in row *i*, character position -IR j . The end of the text is indicated by a **.BOX** *control word* for another piece of text or the **.END FIGURE** *control word*. Temporarily, the text should not include underlined or overtyped characters.

.END FIGURE

This *control word* causes control to return to the regular *control processor* of the **RUNOFF** command, for the decision to print the picture. Note that another **.FIGURE** *control word* may not appear until after this *figure* has been printed.

One further *control word* has been added which is intended to facilitate bringing out revised editions of a memorandum.

.FLAG The next line to be printed after this *control word* is encountered will have an asterisk placed two spaces to the right of the right margin, as illustrated.

.DEFINE *symbol*

This *control word* defines the value of the symbol*symbol* to be the number of the page currently being printed. The symbol may be used later with the **.USE** *control word* to cause printing of the page number in text. The characters in the symbol must be mappable into the six−bit character set, and all symbols must be six of fewer characters.

.USE *symbol*

The value of the symbol *symbol* is inserted into the text with a single blank preceding and no blank following. If the symbol has not been previously defined, its value is **0**. Text may continue following a blank typed after the symbol.

Here is an example of the use of these *control words*.

In one area of text:

> We now discuss the operation of the typewriter **.DEFINE REF1** coordinator module, which . . .
>
> In a later area of text:
>
> > As we saw in the discussion of the typewriter coordinator on page **.USE REF1**, the rest of . . .
>
> if the first area of text were on page 14, the later line would read:
>
> > As we saw in the discussion of the typewriter coordinator on page 14, the rest of . . .

-

Further Study of Experimental Additions

A number of suggestions have been made for extending the *control word* language of *RUNOFF*, and its capabilities. These are listed here, primarily to elicit comment and discussion, both on the language which describes these operations and the less important problem of their implementation. -

1. Word division. This is a whole are of study in itself.
2. Automatic footnote insertion. This was handled somewhat awkwardly in the **DITTO** *command*, although the basic approach was probably reasonable.
3. Automatic page references, perhaps via some symbolic reference scheme. This would enable the page number in "as was described on page 32" to be inserted by the program. The analogy with an assembly program should be hotly pursued for ideas.
4. Special provision for printing facing pages. This would require alternate running heads, placing page numbers alternately at right and left, and matching line counts on facing pages.
5. Improved page−division rules, to prevent the last line of a paragraph appearing alone at the top of a page, for example. At present, copy must be run off to check by hand that awkward page divisions have not been made.
6. Automatic generation of page numbers for a table of contents. Again, the analogy of an assembly program symbol table appears fruitful.
7. Automatic generations of an index. The problem here is obtaining too many references to a given word, many irrelevant.
8. Arrangement of tabulated data. This problem may have already been partly approached with the above−described figure generator, or the facilities already available in *RUNOFF*, but automatic setup of column widths and positions would be desirable. One could include in this category the ability to call on other programs to computer numbers to place in tables, although this is going pretty far afield.
9. Placing figures in a *cut* or *inset*. The control language is the most difficult problem here.
10. Equation typing and numbering. Again, the control language appears formidable.

MANUSCRIPT CONVENTIONS

Initially, *RUNOFF* is set to *FILL* mode, such as by using **.FILL**. The filling is identical to *groff*'s filling mode: Text lines will normally be adjuted by inserting extra spaces in mid−line so that the end of the line is on the right margin.

AVAILABLE RUNOFF FILES

You can still find text files in the *RUNOFF* language.

In the Kermit website ⟨http://www.columbia.edu/kermit/pdp10.html⟩ you find 3 files in *RUNOFF* language maybe of the `Multics` era or later:

- `k10133.rno` ⟨ftp://kermit.columbia.edu/kermit/d/k10133.rno⟩
- `k10mit.rnh` ⟨ftp://kermit.columbia.edu/kermit/d/k10mit.rnh⟩

- `k10v3.rno` ⟨ftp://kermit.columbia.edu/kermit/d/k10v3.rno⟩

At `Saltzer`'s publication website ⟨http://web.mit.edu/Saltzer/www/publications/pubs.html⟩ you find files in *RUNOFF* of the `Multics` era. Search there for `runoff` and you will find the following 5 files:

- `whyring.run` ⟨http://web.mit.edu/Saltzer/www/publications/whyring/whyring.run⟩
- `starring.run` ⟨http://web.mit.edu/Saltzer/www/publications/starring/starring.run⟩
- `tmring.run` ⟨http://web.mit.edu/Saltzer/www/publications/tmring.run⟩
- `RFC1498 florence.run`⟨http://web .mit.edu/Saltzer/www/publications/florence.run⟩
- `zurich.run` ⟨http://web.mit.edu/Saltzer/www/publications/sourcerouting/zurich.run⟩

Most *RUNOFF* files are found in the `DEC PDP-10 archive` ⟨http://pdp-10.trailing-edge.com/cgi-bin/⟩. Many files are very old as they are written in *upper case* only:

- normal documents **`*.rno`** in *RUNOFF* ⟨http://pdp-10.trailing-edge.com/cgi-bin/searchbyname?name=*.rno⟩
- **`*.rnh`** help files in *RUNOFF* ⟨http://pdp-10.trailing-edge.com/cgi-bin/searchbyname?name=*.rnh⟩
- **`runoff.*`** documents about *RUNOFF* ⟨http://pdp-10.trailing-edge.com/cgi-bin/searchbyname?name=runoff.*⟩

SEE ALSO

groff(1), **groff**(7), **roff**(7), **groff_filenames**(7)

1964 `Jerome H. Saltzer`:
*`Jerome H. Saltzer — TYPSET and RUNOFF, Memorandum editor and type-out commands`*a vailable at ⟨http://mit.edu/Saltzer/www/publications/CC−244.html⟩

1965 `Jerome H. Saltzer`:
`Jerome H. Saltzer — Experimental Additions to the RUNOFF Command` available at ⟨http://web.mit.edu/afs/athena.mit.edu/user/other/a/Saltzer/www/publications/PSN−40.html⟩

1966 `Jerome H. Saltzer`:
*`Jerome H. Saltzer — Manuscript Typing and Editing`*which is a vailable in the internet at *MIT html* ⟨http://mit.edu/Saltzer/www/publications/AH.9.01.html⟩ or *CTSS html* ⟨http://web.mit.edu/Saltzer/www/publications/ctssAH.9.01.html⟩ or *CTSS pdf* ⟨http://web.mit.edu/Saltzer/www/publications/ctss/AH.9.01.pdf⟩.

1973 `Larry Barnes`:
`Larry Barnes — RUNOFF: A Program for the Preparation of Documents` available as *pdf* ⟨http://www.textfiles.com/bitsavers/pdf/sds/9xx/940/ucbProjectGenie/mcjones/R−37_RUNOFF.pdf⟩.

1974 `DEC RSTS`:
`RUNOFF User's Guide`: `v8.0-v4-d-rsts_e_runoff_users_guide.pdf` at *DEC RSTS* ⟨http://elvira.stacken.kth.se/rstsdoc/rsts-doc-v80/v8.0-v4-d-rsts_e_runoff_users_guide.pdf⟩.

1981 `PDP-11`:
This is the latest documentation on *RUNOFF* available as text file. ⟨http://malark y.udel.edu/˜dmills/data/du0/RUNOFF.DOC⟩. More exactly, this **`.DOC`** file is an output file produced by the **RUNOFF** program a long time ago. This extension doesn't work on actual systems who expect a Microsoft office file. You have to rename this file by appending the **`.txt`** extension. Then the file can be viewed by **`more`** or **`less`**.

Emulator for `IBM 7090 CTSS` ⟨http://www.cozx.com/˜dpitts/ibm7090.html⟩.

The home page of*`Jerome H. Saltzer`* is ⟨http://web .mit.edu/Saltzer/⟩.

AUTHORS

This file was written by Bernd Warken ⟨groff−bernd.warken−72@web.de⟩.

COPYING