MRUNOFF

MRUNOFF is a program which will produce a formatted manuscript from a source file. With MRUNOFF it is quite simple to achieve presentable results. However, MRUNOFF contains a number of sophisticated capabilities which allow the seasoned user a great deal of flexibility and control over the resulting manuscript.

The original MRUNOFF was called ROFF and was written for Multics by Doug McIlroy of Bell Labs in March, 1969. Art Evans made extensive modifications to it in May and June, 1969. Dennis Capps added footnoting in 1970. Harwell Thrasher maintained the program during 1971. Bob Mabee added many new features (during 1971-1972) and brought the program to the 6180 Multics (from the 645) in 1973. Bernard Cosell converted the program to TENEX BCPL in February, 1975.

Running MRUNOFF

When you call MRUNOFF, it will first ask you for an input file - the main input file to be formatted - and then will await the specifictions of any desired options. Typing in a null line will start the processing of the input file. The option acceptor uses a standard TENEX command recognition routine.

Output to file filename
Manuscript output will be directed to the indicated file. Any underscoring will be done on a line-by-line basis rather than the default per-character basis. In the absence of this option, output will appear on the controlling terminal.

Paginate
Page breaks in the output are retained. This is the normal mode. Page breaks in the output can be suppressed by use of the "Don't Paginate" option.

Pause between pages
or
Stop between pages
The program will wait for a carriage-return on the controlling terminal before beginning a new output page. The default mode is "Don't Stop between pages".

Wait before beginning
The program will wait for a carriage-return on the controlling terminal before starting the first page, but will not otherwise pause. The default mode is "Don't Wait before beginning".

Number lines
Source line numbers wil be printed in the left margin. This option forces a minimum indentation of ten spaces. The default mode is "Don't Number lines".

Indent nnn
or
Margin nnn
The output will be indented "nnn" spaces from the left margin. The default mode is "Margin Ø".

From page nnn
Specifies the lowest numbered page of the output that will actually be printed. Default is "From page Ø".

To page **nnn**
Specifies the highest numbered page of the output that will actually be printed. Default is "To page 99999".

Passes to be done **nnn**
Specifies that nnn passes should be made over the input file. Output will be produced only on the last pass. The default mode is "Passes to be done 1".

Parameter to be set to **xxxxxx**
Establishes an initial value for the variable "Parameter". The default is for "Parameter" to be set to the null string.

Hyphenate
Enables the hyphenation package. The operation of the hyphenation package is discussed on page 36. The default mode is "Don't hyphenate".

Chars File
When this option is selected, various key characters in the output file will be flagged. The flags will be written to a file with the same directory and name as the main source file, but with extension .CHARS. The default mode is "No Chars File".

Start numbering with page **nnn**
Output page numbering will begin with the first page of output being assigned number nnn. The default mode is "Start numbering with page 1".

Don't
or
No As indicated in the affected options above, this command may be used with some options to reverse their sense. For example, a "Don't hyphenate" command would undo the effects of any preceding "Hyphenate" command.

The Basics of MRUNOFF

An MRUNOFF source file contains two type of lines: control lines and text lines. A control line begins with a period; all other lines are considered to be text lines. The control lines are, in general, directives to MRUNOFF and will not appear in the output file. Text lines are formatted as directed by preceding control lines and program-initiation options and written to the controlling terminal or the selected output file.

File name defaults. MRunoff attempts to make its defaults for the various fields in the file names it uses as reasonable as possible in order to allow the User as much abbreviation as possible. For its main input file, MRunoff will look in the connected directory for extension **.MRN;** the file name must always be specified. For all other files, MRunoff will always default to the directory used for the main input file. If hyphenation is desired, the default file for the dictionary is **HYPHENATION.DICTIONARY.** The output and chars files default to have the same name as the main input file with extensions **.DOC** and **.CHARS,** respectively. Inserted files have default extensions of **.MRN,** and the file name must always be specified.

Filling. When the text is being "filled" all ends-of-lines in the input file are ignored beyond identifying control lines. Material is taken from the input file and added to an output line as long as there is room. A line will be written only when either there is insufficient room for the next input file

word or a "break" is encountered. If the output is being "adjusted" (the default mode), before the filled output line is written it will be padded with sufficient spaces so that it exactly fills the specified output line length. Successive lines are padded alternately from the left end and from the right end.

Breaks.. A "break" insures that the text following it will not be run together with the text that preceded it. Breaks can be introduced into the text explicitly by use of a .br control line. In addition, many other control lines will cause a break. If an input line begins with a space, it will cause a break.

Tabs. MRUNOFF maintains a table of tab stop locations. Tab stops are initialized to occur at locations 11, 21, 31, etc., and can be changed with a .tb control line. As text is prepared for output, tab characters (I, ASCII Ø11) are replaced with the required number of spaces (but always at least one) to fill out to the next tab stop counting locations from the first print position on the line, independent of where that print position may fall on the page due to .in or .un control lines. Tab characters in control lines are not handled any differently than any other character.

Sentences. When an input text line ends with any of the characters ".", "?", "!", ";", or ":", or with ".", "?", or "!" followed by a double quote or ")", two blanks will precede the following word (if it is placed on the same output line), instead of the normal single blank. te that if these characters appear

anywhere else on the input line no special action will be taken; however, within a text line, any multiple spaces are preserved. Thus, if there are three spaces between a pair of words and they get put on the same output line, then they will be separated by at least three spaces, and perhaps more if the line is adjusted.

The layout of the page. Verticlly, an output page contains three areas. The first consists of **headers,** which are printed at the top of each page. There may be up to twenty lines of headers. Header lines are numbered from the top down. Next comes the text body, which contains both text and footnotes. Lastly, come **footers,** which are printed at the bottom of each page. Again, there may be up to twenty lines of footers, but footers are numbered from the bottom line upwards. These three areas are spaced by four margins. The first margin on the page is the number of blank lines above the first header; the second is the number of blank lines between the last header and the first line of text; the third is the number of blank lines between the last line of text and the last footer; the fourth is the number of blank lines below the first footer. These margins are set up, respectively, by **.m1, .m2, .m3,** and **.m4** control lines, and the current values of these may be referenced through the symbols **Mr1, Mr2, Mr3** and **Mr4;** the default margins are four lines at the top and bottom of the page, and two lines around the text body.

Line spacing is always done just before a line is printed. For example, when a "page break" occurs (say, because of a **.bp**

control line), the footnotes and footers for the current page are written immediately, but the headers on the new page are not written until the first line of text is ready to be written on it. In particular, when MRUNOFF begins processing a file, it considers itself just "above" the first page, and if any headers are set up before any text is placed on the first output page, the headers will be printed on the first page.

Character Set. MRUNOFF expects its input to utilize the full ASCII character set. However, it will perform case conversion if that is necessary. The variable **CASECONTROL** should be set to the ASCII code for the character to be used for signalling case information; setting **CASECONTROL** to zero disables the case conversion facility, and it is initialized that way. MRUNOFF performs case conversion by interchanging characters with ASCII codes #100 through #137 with those of codes #140 through #177. Double occurrences of **CASECONTROL** will complement the overall mode of converting either all characters or none, while single occurrences will complement the effect of the overall mode for the immediately following character only.

MRUNOFF is nominally set up to produce output using the full set of ASCII graphics. However, MRUNOFF contains various facilities for dealing with devices with other character sets. The variable **TrTable** is a table whose **n**th entry provides the translation for the character with ASCII code **n**. If the entry is a number, then that number is the ASCII code which will be added to the output file (or sent to the terminal); if the entry is a

string then the entire string will be used as the translation. Note that these translations are used for **all** output including the formfeed (via the translation for ASCII 12.) used to eject pages and the carriage return - line feed sequence (via the translations for ASCII 13. and 1Ø.) used to advance from line to line. **TrTable** is initialized to have all of the ASCII graphics and all of the ASCII format effectors translate to themselves, and all other codes translate to Ø (ASCII NUL). By use of **.tr** control lines or **.st** control lines, **TrTable** can be modified to suit MRUNOFF's output to the desired device. For example, if the User's device has some special graphics the User has the choice of using the appropriate control code directly and declaring the code to translate to itself, or for more convenient source preparation in a standard ASCII environment some little used ASCII graphic could be translated into the appropriate control code. Further, if the output device has some special capability (e.g., subscripting), the User can have any convenient character translate into the appropriate escape sequence.

In order to do filling and justifying correctly, MRUNOFF must know the printing width each ASCII code takes up on the output device. This is done by means of another table, **Widths.** The **n**th entry of this table specifies how much space the character with code **n** will occupy on a line. This width is calculated before any **TrTable** translation is done; that is, the code **n** refers to the ASCII code of a source file character.

MRUNOFF also has a facility to assist in the preparation of manuscripts for devices with an inadequate collection of graphics for the job at hand. For these cases, by requesting the **Chars file** option and using **.ch** control lines, or **.st** control lines directly upon the table **CharsTable**, the User can specify that whenever MRUNOFF encounters a particular character, in addition to translating it into the output, MRUNOFF should make an entry into a second output file to indicate that that particular location in the manuscript will require some attention after it is typed out. **CharsTable** is indexed by the ASCII code of the input file character (i.e., before it has been translated). If the entry for a code contains a SPACE (ASCII 32.), then the character will be passed to the .CHARS file with its **TrTable** translation and will not be flagged. If the entry for a character contains any other number, then that code will be sent to the .CHARS file instead of the normal translation and the character will be flagged. Unless an output line contains at least one flagged character, it will be suppressed from the .CHARS file. MRUNOFF initializes **CharsTable** so that all of the ASCII graphics and format effectors print as themselves, unflagged, the other control characters print as an appropriate lower case letter, flagged, NUL becomes a flagged accent grave, and RUBOUT becomes a majuscule R, flagged.

Page numbering. As the output is being prepared, a page number counter is kept. In addition to being set by the **Start numbering with page** option, the counter can be incremented or set

directly from the source file (e.g., by a **.pa** control line). A page is called odd (even) if the current value of the counter is odd (even). To facilitate preparing manuscripts to be printed "double sided", the headers and footers can be specified independently for odd and even pages.

Symbols. Many of the variables internal to MRUNOFF are avaiable to the user by means of the symbol facility. In addition, the user may define and use his own symbols. This facility combined with the conditional command **(.ts)** and the looping commands **(.gf, .gb** and **.la)** allow the user, in effect, to write a program within MRUNOFF to "build" his output text. Details on the use and definition of symbol appears on page 18. In general, though, there is a special character which is reserved to indicate symbol substitution. This character is nominally **%**, but can be changed by a **.cc** control line or by resetting the value of the symbol **SpecChar.** When a symbol is to be substituted, the name of the symbol is surrounded by **SpecChars.** The **SpecChars** and the name of the symbol are replaced in the line with the value of the symbol before the line is processed. In order to include a **SpecChar** in the text, it must be doubled. Also, if a single **SpecChar** occurs in the text, it will b replaced by the current page number. For example, **.ts %=42** would test whether the current page number were 42, or **.ts "%Parameter%"="%%"** would test whether the symbol **Parameter** was equal to a per-cent sign.

Underscoring and Multiple Printing. MRUNOFF contains a facility for underscoring the output text. When this facility is enabled, a character is usurped for use as an underscoring flag. MRUNOFF maintains a global underscoring mode which indicates whether all output characters are being underscored or not. Underscoring is enabled, and a control char is selected, by means of a **.uc <char>** control line. Double occurrences of the selected character reverse the global underscoring mode, while single occurrences of the selected character reverse the global mode for the following character, only. For example, if the user included a **.uc $** control line at the beginning of his input file, then the text **$$Now is t$he $$time for al$l good men** would appear in the output as Now is the time for all good men.

While MRunoff does not contain any facility for specifying changes of font (and for that matter, most output devices don't have the capability, anyway), MRunoff does have the ability to multiply strike characters which results in a fairly attractive simulation of a boldface font. The facility can be enabled by setting the variable **OverprintCount** to be greater than zero; is disabled by setting **OverprintCount** to be less than or equal to zero. This facility replaces the normal underscoring facility when it is being used. In particular, when enabled any character which would normally be underscored will instead be repetitively struck a number of additional times as given by the value of **OverprintCount.** Setting **OverprintCount** zero or negative will restore normal underscoring; disabling underscoring does not af-

fect **OverprintCount** (and thus multiple printing, and not underscoring, would continue when underscoring is next enabled).

Titles. Several command lines accept a **title line** as an argument. A title line consists of a line with three distinct fields: the first is text to be placed flush with the left margin, the second centered, and the third flush with the right margin. The first non-blank character in the title will be used as the delimiter to separate the three fields. If fewer than four delimiters are present on the line, the missing parts of the title are taken to be blank. The justification and centering is done relative to the line length and margins in effect at the time the title line is processed, and is independent of their values at the time of use. For example, this document used the commands: **.he 1 'MRUNOFF''%FancyDate%'** and **.fo 1 ''- % -'.**

Flagging blocks of text. MRunoff has the capability to print a marginal flag (usually a vertical bar, |) along side the output text. The flagging character may be put to the left of the text or to the right of the text, or even in the middle of the text. If on an even numbered page and the variable **EvenFlagCol** is neither less than nor equal to zero, then its value gives the absolute print position in which a flag is to be printed, otherwise, no flags are added; the variable **OddFlagCol** functions similarly for the odd-numbered pages. The character which would normally be printed in that position is replaced by the character whose ASCII code is given by the value of **FlagChar.**

An MRunoff output line will always have **ExtraMargin** spaces at its beginning which are totally invisible to the formatting processes. An absolute print position is one which takes these "free" spaces into account (as, for example, the tab stop specification does not). Thus, if **ExtraMargin** were set to three, then a flag specified to occur in column 1 would appear three spaces to the left of the output text. If the flagging column is beyond the right end of the line, MRunoff will add spaces to the line as necessary to place the flagging character in the correct column. MRunoff will never flag headers or footers.

To make the use of this facility more convenient, a **.fl** control line will set **FlagChar** and both **EvenFlagCol** and **OddFlagCol.**

Footnotes. The occurrence of a **.ft** control line will cause all of the text and commands until the next occurrence of a **.ft** control to be collected as a footnote. The text of all footnotes defined on a page is accumulated and written together at the bottom of the page. The footnote text is completely set up at the time MRUNOFF encounters the defining **.ft** control line; thus, any commands embedded within the footnote will be executed while the footnote is read, not when the footnote is eventually inserted into the output. In particular, it is not possible to nest footnotes.

In order to make it more convenient to number footnotes, MRUNOFF maintains a special symbol, **Foot.** The value of this symbol is incremented by one whenever a **.ft** control line which ends

a footnote is encountered. Thus, while in the main text, **Foot** will yield the next number to be assigned to a footnote, and while in the body of a footnote %Foot% yields the number of the footnote in progress. The user may select whether he prefers footnotes to be numbered consecutively through the entire document or whether he prefers that the numbering revert to "1" with each new page. Also, the automatic incrementing of **Foot** may be suspended entirely if desired.

The body of a footnote should occur immediately after the word which is to include the footnote reference. MRUNOFF will insure that the output line containing the reference will be kept on the page which receives the footnote. When MRUNOFF encounters a **.ft** control line beginning a footnote, it will append the value of the value of the **TextRef** to the last word on the preceding line **TextRef** is initialized to be **(%Foot%)**. Since a footnote does not cause a break in the main text, the only effect on the output is that the footnote reference has been appended to a word. Similarly, the value of the value of the symbol **FootRef** will be included as the first line of the footnote input text; **FootRef** is also initialized to be **(%Foot%)**.

Pictures. The User may set, and format, blocks of space for pictures. The formatting possible includes top- or bottom-of-the-picture captions, footnotes within the captions, etc; in fact, the full power of MRUNOFF* is available for use in laying

*With the exception of the **.gb** command. Due to the organization of the deferred text processor there is no way to loop back and re-execute any of the deferred text. If this is necessary, it

out a space for a picture. A pair of **.pc** control lines ("Picture with caption") are used to delimit a section of text and commands which should do any desired formatting (including actually leaving any blank space needed). The first **.pc** control contains an expression which specifies how much space the picture (including all captions, etc.) will require. The entire section between the **.pc** lines will be queued and deferred until a point is reached where all previously queued pictures have been processed and the indicated amount of space is available. Then the queued text and commands will be processed before any further text is taken from the input file. Notice that no space is "automatically" set aside; any desired empty space must be explicitly provided for. The space requested in the opening .pc control and the space actually taken up during the eventual processing of the caption need have no relation to one another. Also, regardless of how much space is actually requested, if MRUNOFF reaches the top of a new page at least one caption will be processed as the first thing (after the headers) on the new page; beyond the first caption, though, captions will only be processed if the space left on the page is at least as large as was originally requested for the caption.

---

can be done by putting the loop in a file and using .if to include it in the caption.

Expressions

Several commands require numeric or string arguments. For all such commands, the argument may be the result of evaluating an expression. The basic arithmetic operators, in order of decreasing precedence, are:

```
(, ) grouping
~ (bit-wise negation), - (unary)
*, /, // (remainder)
+, - (binary)
=, <> and >< (not equals), <, >, <=, >=, =<, =>
     (Comparison operators. Yield -1 (if true) or
     Ø (if false))
& (bit-wise AND)
\ (bit-wise OR)
```

Octal number are indicated by prefixing the number with a "#". All other numbers are considered to be decimal.

Strings can also be dealt with. However, the only way that strings are handled is by building up a single string "constant" from other string constants in a strictly left-to-right manner. The basic string constant is a sequence of characters surrounded by double quote characters. Certain special characters are represented by sequences, as follows:

**	asterisk
*"	double quote
*b	backspace
*n	new-line
*t	horizontal tab
*s	space
***cnnn**	character whose ASCII code (in decimal) is **nnn** (one to three digits).

If two string constants are placed next to one another, the result is a new string constant which is the concatenation of the original two strings. Placing **(i)** or **(i,k)**, where **i** and **k** are

arithmetic expressions, after a string constant results in a new string constant consisting of a substring of the original constant, defined as follows: **i** indicates the character position at which the substring is to begin, if **i** is negative, it indicates an offset from the right end of the string; **k** (or **-1** if **k** is not supplied) indicates the length of the substring to be extracted, or if **k** is negative it specifies the distance from the right end of the string at which the substring is to end. There are **no** other string operations; the presence of **any** arithmetic operator in the expression will convert it to being a numeric expression. The comparison operators work on strings: if "=" (or "<>" or "><") is used to compare two strings, the comparison will be done over their entire lengths, in order to result in "true" (or "false") the two strings must **exactly** match - in both content and length; if any other comparison operator is used between two strings, the result will be appropriate to the collating sequence (independent of case) of the two strings. One other numeric operation is defined for strings: if a string constant is immediately followed by a "#", it will evaluate to be the length of the string. In any other context in a numeric expression, a string is converted to a number in such a way that a one-character string results in the ASCII numeric value of the character; the numeric value of multi-character string constants is undefined.

For example, "ab" "cd" = "abcd"; "abcdefg"(3) = "cdefg"; "abc" "defg"(2,5) = "bcdef"; "abcd" "efgh"(3) "ijkl"(2,-2) = "defghijk".

Definition and Substitution of Variables.

Names of variables are composed of upper- and lower-case alphabetic characters, decimal digits, and "_" (underscore). Variables have either "string" or "numeric" values.

Values will be substituted for variables under the following circumstances:

1) In expressions if a **SpecChar** (normally **%**) is found as either the first or second character following the spacing after the control word. Note: any such **%** is not used as a flag and removed. Thus, there will be expressions in which symbol substitution is desired but which cannot be reordered to bring a (desired) symbol reference to the beginning. For example, **.sr Symbol "(%Date%)"**. In such cases, you must specify substitution explicitly with a **.ur** control line.
2) All **.ur** control lines.
3) In all title lines.

To indicate a symbol substitution, you enclose the name of the variable in **%** characters. **%** in any other context will be replaced by the value of the page counter; to include a **%**, you must code it as **%%**. If it is inconvenient to use **%**, you can change the substitution control character by means of a **.cc** control line.

When a substitution takes place, the **%name%** is removed from the line being processed and is replaced with the value of the variable: either the value itself if the variable is a string, or the appropriate string of (decimal) digits if the variable is numeric. Notice, then, that if you wish to preserve the "stringness" of a variable in an expression, you must surround the variable reference in double-quote characters. For example,

to test if **%Parameter%** is equal to the string **"abc"**, you must write **.ts "%Parameter% = "abc".**

If a variable is defined to be a table, references to its entries have a somewhat different syntax. The sequence **%Name|nnn%** will be replaced by the **nnn**th entry of the table **Name.** The index must be a decimal number; it may neither be an expression nor supplied by another substitution. Thus, to retrieve an element whose index has been saved in another variable, some technique like **.ur ... %%Name|%Index%%%...** would have to be used. Each element of a table may be a string or a number independent of the table's other elements.

Many of the variables internal to MRUNOFF are available to the user (a complete list will be found on page 33); these include control argument values (or their defaults), values of switches and counters, and certain special functions. Most are user settable, but for the ones that are not, any attempt to redefine them will merely make their system values inaccessible. This will cause no harm, and the user may freely use the name for a private variable.

A special symbol is provided for use in footnote numbering: **Foot** contains the value of the next footnote number available (or the current footnote if referred to from within the body of a footnote). The value of **Foot** is incremented by one when the closing **.ft** control line of a footnote is encountered. The user may select whether **Foot** should maintain a running count through-

out the manuscript or whether it should be reset at the end of each page.

Also, by use of **.mc** control lines, variables can be set up to be counters. Any reference to such a variable provides its current value, and causes its value to be incremented by one automatically. This facility is useful for numbering equations, numbering references and other applications where the ability to maintain a sequential counter through the manuscript is convenient.

Control Line Formats

This section gives a description of each of the control lines which may be interspersed with the text for format control. Control lines do not cause an automatic break unless so specified. If a control line's command is defined as a string valued variable, the value of the variable will replace the period and the variable name and the line will be reprocessed (and hence might not be a control line when rescanned). This substitution and rescanning takes precedence over any definition of a particular control as described below; thus, this capability would allow one to redefined the built-in commands. Arguments of the control words are in the following form:

#	integer constant
n	integer expression
+n	integer expression preceded by optionial plus sign or minus sign
exp	arbitrary expression (string or integer)
c	single character
cd	sequence of character pairs
f	file name
ttl	a title line
xxx	remainder of the control line
+n,	a sequence of integer expressions (with optional preceding signs), separated by commas
name	a character sequence to be interpreted as a variable name (Note: the name should not be surrounded by %s)
str	string expression

(blank line)
A blank line occurring in the text is treated as if it were a **.sp** 1 control line unless the blank line would be the first line on a page, in which case the line is treated as though it were a **.br** control line.

(form feed)
Any line beginning with a form feed will be treated as though it were a **.bp** control line.

.name xxx
If **name** is string valued, then the **.name** portion of the line will be replaced by the value of **name** and the line will be reprocessed. This line is similar to the control line **.ur %name% xxx** excepp that no symbol substitution in the **xxx** portion of the line will take place.

.ad **A**djust: text is printed right justified. Fill mode must be in effect for right justification to occur. Fill mode and adjust mode are the default conditions. This control line causes a break.

.ap name str
Append: If **name** is a string valued symbol, then **str** will be appended to that string (and left as the value of **name**). Otherwise, this control line has the same effect as **.sr sym str**.

.ar **A**rabic page numbers: when the page number counter is substituted into text or control lines by means of **%** (not by **%Np%**) it will be done in Arabic notation. All other numeric substitutions are always done in Arabic. This is the default condition.

.bp **B**egin page: the next line of text will be put on a new output page. This control line causes a break.

.br **B**reak: the current output line is written as is, and the next input text line will be put on a new output line.

.cc c **C**ontrol **c**harcter: this control line changes the character used to surround the names of variables when they are referenced to be **c**. This changes all uses of the control character: a single **c** will become the current page number, and **c** will have to be doubled to be get a **c** into the output text. The default control character is **%**. If **c** is omitted, the control character will be reset to **%**.

.ce n **C**enter: the next **n** lines of source text are centered between the current margins. If **n** is omitted, **1** is assumed. Filling and adjusting are suspended until the centering is completed. This control line implies **.ne %Ms%***n so that all lines centered will be on the same page. This control line causes a break.

.ch cd **Ch**aracters: each occurrence of the character **c** will be replaced in the .CHARS file by the character **d**, and the character will be flagged. If the **d** character is blank, or an unpaired **c** character appears at the end of the line, the **c** character will not be flagged, and will occur as itself in the .CHARS file, or not at all if no other character in the line is to be flagged.