Program Name: FORMAT

Date: June 1972

Language: IMP

Keywords: documentation,paginate,reformat,page size,right justify

Purpose: to segment a text file into numbered pages of arbitrary dimensions

General Description

The basic function of this program is to reformat an input file into lines of a specified length and put a specified number of these lines on a page; this behaviour may be modified locally to produce for example 'new line', 'new paragraph' and so on. The distinction between an acceptable reformatting and an unacceptable one is to a large extent subjective and dependent on the vagaries of individual users; this program is therefore intended to deal with the clear-cut cases and provide the means for the user to deal with the rest as he chooses. To this end, a large number of options are available with which to modify the program's behaviour. Defaults exist for all options to ease the load on the user.

Because of deficiencies in the character set presently guaranteed (Edinburgh 60 Character Standard Subset of ISO) it is necessary to extend it artificially by using shift characters to represent uppercase letters and underlining.

In addition to producing formatted primary output, the program can produce a new source file and a paged, line-numbered listing of it. Plain language files which are repeatedly altered tend to become fragmented because modifications seldom fit conveniently into the space available on existing lines; use of the 'new source file' facility /greatly

greatly reduces the editing effort subsequently required by preventing this fragmentation from occurring. Also, by keeping the source file in a standard format, it simplifies editing directly from a primary output listing.

Primary output can be in ISO or an alternative code (Atlas in the distributed version); the output code is selected as one of the options mentioned above. To provide a different alternative code, only one subroutine need be changed.

To facilitate making minor alterations to existing documents, specified pages may be generated singly.

Various conditions are diagnosed as abnormal and a monitor message printed along with the offending source line; the program recovers from all such errors and continues. Note however that a new source file generated from a faulty one is itself likely to have parts missing rather than contain the original errors.

If, as may be convenient when using a small, non-multiprogrammed computer, the primary output is sent directly to the console typewriter, boundary markers can be printed to simplify the subsequent separation of the output into pages.

Utility programs are available to convert various literally represented input formats into the standard representation required by FORMAT, descriptions of these appearing as appendices to this document. Where input equipment with an extended character-set is available, use of these utilities can ease the initial inputting of a document considerably.

Operating the Program

The program uses two streams for input and four for output; these are allotted as follows:

input

0	console(command information)
1	source input file

output

0	console(diagnostic messages)
1	primary output
2	new source file
3	paged,line-numbered listing of (2)

The source file contains, normally at the beginning, one or more flag lines which contain requests for various options to be set or cause a query to be typed on the console inquiring how the option is to be set on this particular run of FORMAT. The file will also contain control symbols, consisting of a control marker (left angle bracket followed by a right angle bracket) followed by a letter. For example:

'<>P This is the start of a paragraph.....'

marks the start of a new paragraph.

Details of the options and control symbols available are given on following pages.

All letters are taken as lower case unless explicitly marked as uppercase. The convention followed is that a commercial 'at'(@) acts as an uppercase shift on the immediately following character. Its scope may /be

be extended by enclosing the character string in parentheses or square or angle brackets preceded by the shift.

An underline(_) is used similarly to denote underlining.

Where uppercase and underlining scopes overlap, separate bracketting must be used for each, even if their scopes are in fact the same; such bracketting must be strictly nested. Unmatched closing brackets are not recognised as such and do not therefore cause special problems.

Ambiguous situations may arise when the source text already contains bracketting which interferes with that intended to delimit the scope of shift characters as above. Under these circumstances:

(1) '<', '(', and '[' each require their appropriate closing brackets, e.g. ')' will not balance '['.

(2) FORMAT always matches the first suitable closing bracket encountered.

Control Symbols

Control symbols are used to implement those features which will be required frequently; options are used for those facilities which in simple applications would be set only once. Obviously this distinction will break down if the available features are exploited to the utmost; one might for example choose to reset the line length on a particular page to make the page boundary fall in a more convenient place.

A list of control letters and their effects follows.

letter	effect
L n	Explicit new line. Close off the current line if it is non-null and follow it by n-1 blank lines.
N n	Explicit new page. Close off the current page and follow it by n-1 blank(but numbered) pages. If a new page has just been started but is as yet empty, the 'close' is omitted.
P	New paragraph. Close off the current line if it is non-null and indent the following line to the first tab stop. New page if only one free line is left on the current page.
S	New section. Close off the current page, increment section number by one, reset page number to one.
T n	Tab. If n is omitted tab to next stop set; if n is supplied go to the nth stop. Stops are implicitly /numbered

numbered upwards from 1.

I n — Indent subsequent lines to the nth tab stop. This is overridden for any line following an explicit new line(L). Cancelled by P, S, N.

D n — Diagram: the lines up to but excluding the next control marker are to be copied directly without reformatting except to follow the the underlining and uppercase input conventions; n is used to decide whether the diagram should go on the current page or the next one. A diagnostic is printed if the diagram does not in fact contain n lines. A control marker can be used alone, without the usual letter, on a line by itself, and against the left-hand margin to terminate the diagram if need be. Reformatted source files will have this marker inserted anyway to avoid problems connected with double spacing paragraphs so there is usually no need for the user to put it in.

While a switch to diagram mode and back does not cancel indenting, as do P, S, and N, the diagram is not itself indented.

E — Marks end of input file. Close off last page, terminate this run of FORMAT.

V n — Verify that at least n free logical lines are left on the current page. In detail: close current line off if it is /non-null,

non-null, check whether or not n or more logical lines are left and if not, take a new page.

N.B.

Except for tab(T), n takes a default value of one, thus for example, 'L 1' can be abbreviated to 'L' and so on.

Options

The program is parameterised so that by altering various flags and constants its behaviour can be modified on a rather more permanent basis than by using the control symbols just described; these variables all have default values and for any one application it is likely that only a few will need to be altered.

Options are set by a flag line, distinguished by its containing control symbol F. When this is encountered, the part of the line lying to the right of the control symbol is decoded as option-setting information.

The basic options are as follows: (defaults in brackets)

P	primary formatted output is required. (yes)
S	a new source file is required. (yes)
L	a paged,line-numbered listing of S is required. (no)
I	primary output is to be in ISO code. (yes)
J	primary output lines are to be right justified. (no)
O n1,n2;	set origin: initial section no. = n1, initial page no. = n2. (0,0) n2=0 suppresses both page and section number. n1=0 suppresses section number only.
D n1,n2;	set page dimensions: lines/page = n1, characters/line = n2. (60,72)

T n1,n2,n3,.....; set tab stops:

tab1 = n1, tab2 = n2, tab3 = n3,..... etc

where: n1 <= n2 <= n3 <= ..... etc

Note that it is permissible to set more than one stop to the same print position. If this is done, note that tabs without a following number skip to the next print position at which a tab-stop is set in the same way that typewriter tabs behave but that where a particular tab number is specified, this stop is indexed directly.

Up to 20 tab stops may be set. (8,16,24,32, ... etc)

Where an option requires no parameters and presents a simple binary choice, e.g. primary output is or is not required, the letter may be preceded by '\' to unset that option thus:

<>F S \L P \I

requests a new source file, no listing, and formatted primary output in the secondary output code.

<>F O 3,1; D 50,80;

sets the section number to 3 and page to 1, and also sets the page dimensions to 50 lines, each containing 80 characters. Obviously \D for example, is meaningless.

Any option setting letter may be preceded by '?'; if this is done, a query will be printed on the console requesting the appropriate information. These queries may be answered either with Yes/No or with a response specific to that particular question. Thus:

<>F ?I

would cause 'ISO OUTPUT(I)?' to be printed on the console. For this particular query, the reply 'Y' is equivalent to 'I' and 'N' to '\I'.

To facilitate typing ahead where an operating system permits it, that option is set to which the answer received is specific. Y/N is non-specific and is taken as reply to the question asked. The benefit is that if a number of options involve console queries, it is not necessary to answer the questions in the order in which they are asked. Thus if typing ahead one supplied 'P I \S \L' as a response to:

<>F ?S ?P ?I ?L

the options requested would be set correctly.

Irrevelant queries are suppressed; for example, '?I' is redundant unless primary output is required. The number of queries actually typed is thus in general dependent on the order in which they are answered and the replies received.

In an attempt to eliminate a possible hazard, requests for a listing are ignored unless a new source file is also requested.

A simple example follows, after which more information about the available options is given.

Example

This is an example which will produce 7 line pages, each line containing 50 characters. The section is set to number 4. Following these pages is a printout of the source text which was supplied to 'FORMAT' to /generate

generate them.

Note that '7' refers to the number of actual text lines; two more are used by the page numbering scheme to give a total of, in this case, 9 physical lines on each page.

Important points to note are:

(a) That the line structure of the source text is ignored completely.
(b) That in particular, multiple spaces and newlines in the source are not transmitted to the output.
(c) That 'L' over-rides 'I' for one line only.

On the following page are listings of the original input supplied to FORMAT and the reformatted source file available on output 2.

```
<>F ?P ?I ?S ?L
<>F  O 4,1; D 7,50; M 10;
<>F T 6,16,26,36,46; G 1;

_[@EXAMPLE]
<>P  @THIS IS AN EXAMPLE WHICH WILL PRODUCE 7 LINE PAGES, EACH
LINE CONTAINING 50 CHARACTERS. @THE SECTION IS SET TO NUMBER 4.
@FOLLOWING THESE PAGES  IS A PRINTOUT OF THE SOURCE TEXT
WHICH WAS SUPPLIED TO @['FORMAT'] TO GENERATE THEM.
<>P @NOTE THAT '7' REFERS TO THE NUMBER OF ACTUAL TEXT LINES;
TWO MORE ARE USED BY THE PAGE NUMBERING SCHEME TO GIVE A TOTAL OF,
IN THIS CASE, 9 PHYSICAL LINES ON EACH PAGE.
<>P @IMPORTANT POINTS TO NOTE ARE: <>I <>L
(A) <>T @THAT THE LINE STRUCTURE OF THE SOURCE TEXT IS IGNORED
   COMPLETELY. <>L
(B) <>T @THAT IN PARTICULAR, MULTIPLE SPACES AND        NEWLINES

   IN THE SOURCE ARE NOT TRANSMITTED TO THE OUTPUT. <>L
(C) <>T @THAT '@L' OVER-RIDES '@I' FOR ONE LINE ONLY.
<>L2
<>P @ON THE FOLLOWING  PAGE ARE LISTINGS OF THE ORIGINAL
INPUT SUPPLIED TO @[FORMAT] AND THE REFORMATTED SOURCE FILE
AVAILABLE ON OUTPUT 2.
<>N
<>E
```

```
<>F ?P ?I ?S ?L
<>F  O 4,1; D 7,50; M 10;
<>F T 6,16,26,36,46; G 1;
_[@EXAMPLE]

<>P @THIS IS AN EXAMPLE WHICH WILL PRODUCE 7 LINE PAGES, EACH LINE
CONTAINING 50 CHARACTERS.  @THE SECTION IS SET TO NUMBER 4.  @FOLLOWING
THESE PAGES IS A PRINTOUT OF THE SOURCE TEXT WHICH WAS SUPPLIED TO
@['FORMAT'] TO GENERATE THEM.

<>P @NOTE THAT '7' REFERS TO THE NUMBER OF ACTUAL TEXT LINES; TWO MORE
ARE USED BY THE PAGE NUMBERING SCHEME TO GIVE A TOTAL OF, IN THIS CASE,
9 PHYSICAL LINES ON EACH PAGE.

<>P @IMPORTANT POINTS TO NOTE ARE: <>I <>L
  (A) <>T @THAT THE LINE STRUCTURE OF THE SOURCE TEXT IS IGNORED
  COMPLETELY.  <>L
  (B) <>T @THAT IN PARTICULAR, MULTIPLE SPACES AND NEWLINES IN THE
  SOURCE ARE NOT TRANSMITTED TO THE OUTPUT.  <>L
  (C) <>T @THAT '@L' OVER-RIDES '@I' FOR ONE LINE ONLY.  <>L2

  <>P @ON THE FOLLOWING PAGE ARE LISTINGS OF THE ORIGINAL INPUT SUPPLIED
TO @[FORMAT] AND THE REFORMATTED SOURCE FILE AVAILABLE ON OUTPUT 2.
<>N .

<>E
```

More Options

Another nine options are available in addition to those previously listed. They are as follows: (defaults in brackets)

C Print carryover words (a slash followed by the next word) at the bottom of each page; suppressed if this next word is preceded by an explicit tab. (on)

E Print (on the console) the current line number in the input file when the bottom of a primary output page is reached. This information can be used along with a source listing to detect unfortunate paging and help in deciding how best to correct it. (off)

M n: (default is zero)

Move the margin n print positions to the right. The whole page is moved bodily including tab-stop settings, its dimensions being unaffected; this feature can be used with devices such as lineprinters where it is not possible to achieve the same result by moving the paper.

R Tabs are to be relative to the current indentation level, i.e. if indented to tab stop n with this option enabled:

(1) a subsequent 'tab x' goes to stop n+x rather than x.

(2) after explicit newline, the next line starts at tab-stop n-1 rather than the left-hand margin.

(3) indent mode is not cancelled automatically by new paragraph, section and page and diagrams are indented.

/As

As a general guide, '\R' is intended to be simple to use for most jobs; 'R' gives more power when it is needed.

U x,y; (default is @,_)

Reset the uppercase and underline shift characters to x and y respectively. If x,y or both are omitted a non-character value is set, effectively disabling that shift function; the comma and semi-colon are both obligatory.

W Process the whole file(W) rather than generate single pages(\W). If \W is specified, the program requests a section and page number then generates that page only. This sequence is repeated until the user types '*' in response to the query. '*REWOUND*' is typed on the console whenever a page request makes it necessary to rewind the input file.

If ?W or \W is set it must be the very first option marker appearing in the input file; once set, it cannot be unset.

N.B.

Note that single page mode can be entered by answering '\W' to any query and that FORMAT generates primary, and only, primary output when in that state. It is therefore not usually necessary to set '?W' as an option in the source file.

G n; (default is 2)

Set the number of logical newlines between paragraphs.

X n; (default is 2)

/Set

Set the number of spaces to be inserted between sentences when line justification is 'off', i.e. \J is set.

Z n: (default is 1)

Set the number of newline characters which represent one logical newline in the primary output. Note that if the page size requested is not a multiple of n, the carryover word will in general be unevenly spaced at the bottom of the page.

Page Dimensions

The page size set by the 'D' option is specified in terms of physical lines on the output device; the 'Z' option defines a relationship between physical and logical lines but does not directly affect the value set by 'D'.

The page numbering scheme and carryover word each increase the actual page size by one logical line over that set by 'D', the former being a constant overhead, the latter only applying if the appropriate option is set. The actual page size is therefore D+Z physical lines if the carryover option is turned off and D+2*Z if it is on.

It is possible to generate pages containing a number of lines which is not a multiple of 'Z' at the time the page is turned; if this occurs, the page will in fact be the correct size but the carryover word and line number will be unevenly spaced at the bottom.

Further Example

As a further example of how the program can be used, a listing is attached of the main input file from which this document was itself generated. Note that the previous example was produced by overprinting two blank pages specially left for the purpose; the content of these pages is not repeated in the attached listing.

Summary

control symbols

D n	n line diagram
E	end of input
I n	indent n tab stops
L n	n new lines
N n	n new pages
P	new paragraph
S	new section
T (n)	tab to next (nth) stop
V n	verify n lines left on current page.

options (defaults in parentheses)

P	primary output(yes)
S	new source file(yes)
L	listing of S(no)
I	P is in ISO code(yes)
J	P is right-justified(no)
O	set section/page origin(0,0)
D	set page dimensions(60,72)
T	set tab stops(8,16,24,32.....)
C	print carryover words(yes)
E	end-of-page line nos.(no)
M n	set left-hand margin(0)
R	relative(not absolute) tabs(no)

/U

U	reset uppercase/underline shift characters(@,_)
W	process whole file not single pages(W)
G	set inter-paragraph gap(2)
X	set intersentence gap(2)
Z	set physical/logical newline ratio(1)

Restrictions 3.1.72

1) 'end-of-page' lines printed on the console using the 'E' option are in fact the second line on the following page.

2) When in single page mode(\W), consecutive pages cannot be generated without rewinding the input file and doing a separate pass over it to access each page; this can be avoided by outputting alternate pages on two passes.

3) Underlined and/or uppercase words which would in fact just fit on a line may on occasion be put on the next line. If it is imperative in a particular case that this does not happen, insert some redundant spaces to alter the structure of the input line:
e.g. change @[FRED] to @[FRED].

These restrictions stem from basic design faults which, while the cures are conceptually very simple, would in practice require quite extensive changes to be made to the program.

Program Name: UCCONV

Date: April 1972

Language: IMP?

Keywords: FORMAT,punching conventions

Purpose: to convert literally represented upper and lowercase and paragraph blocks into the input representation required by FORMAT.

General Description

This program converts upper and lower case source text to the representation required on input to FORMAT by inserting '@', '[' and ']' as appropriate; no account is taken of cases where these characters are already present in the source text.

A newline character followed by at least three spaces is taken to represent a new paragraph and a paragraph marker substituted for the spaces. Note that literally represented text contains many ambiguous sequences of characters, e.g. newline + spaces may not really be a new paragraph, and this should be borne in mind when using the program. Despite this limitation it can correctly convert the vast bulk of free-form text and so reduce the typing load.

Input and output are to the default-selected streams.

Program Name: FMTIMP

Date: May 1972

Language: IMP7

Keywords: IMP, FORMAT

Purpose: to convert an IMP program text into an input file for FORMAT.

General Description

This program is intended to facilitate the inclusion of sections of IMP program in documentation generated with FORMAT. It converts the program fed to it as follows:

1) The 'percent' convention for keywords is replaced by making all keywords lowercase underlined.

2) All other letters are made lowercase except where a string is quoted when they are uppercase; this applies whether the string is genuinely part of the program or appears in a comment.

3) '@' and '%' are used respectively as uppercase and underline shift characters.

4) The output consists of a series of 'diagrams' each containing (lines on page//10)*10 lines; these lines are numbered sequentially upwards from one throughout the length of the file, the numbers occupying six print positions.

Primary i/o is to/from stream 1; initial console dialogue is via /stream

stream 0.

Note that the output from this program is still in ISO code and that as a consequence, any minor errors introduced by it can be edited out.

Example

```
 1  begin
 2  integer  i,j
 3     spaces(15); printtext 'MULTIPLICATION TABLES'; newlines(2)
 4     cycle  i = 1,1,12
 5        write(i,3) if i # 1; spaces(4*(i-2))
 6        cycle  j = i,1,12
 7           write(i*j,3)
 8        repeat
 9        newlines(2)
10     repeat
11  endofprogram
```