```
$Id: in0.hello.rf,v 1.1 2011-04-08 17:21:44-07 - - $

Hello, World!
This is a simple test of `jroff'.
```

```
This is a very simple example of some text that should be formatted.
As the lines are fetched form the input,
the output lines are adjusted so that each line is approximately
the same length.

Note that a blank line produces a paragraph break.
So we can easily use this as a primitiver formatter just like
fmt(1),
even if we don't want to do anything complicated.

The present assignment does not require that any line justification
be given.
Neither need any fancy PostScript formatting output be done.
Note that there is one space between words except when a word
ends a sentence.
A sentence ends when the word ends with one of the following
punctuation marks:
?!;:.

Many text formatters also allow for both right justification
rather than ragged right typesetting and also for hyphenation.
We do not require that here.
Also not that while sentence-ending punctuation works,
abbreviations ending with periods are treated as ending sentences,
which is not quite right.
And if we're not careful in formatting things,
very-long-words-which-appear-in-the-wrong place can cause
funny-looking line-breaking.

$Id: in1.simple.rf,v 1.1 2011-04-08 17:21:44-07 - - $
```

```
.ll 20
Note that some paragraphs are very narrow and look funny.
It is even possible to make them very very narrow.
.sp
.ll 10
Now we can't get much more than one word per line.
.sp 5
.ll 60
There should be five blank lines above this line.
And this paragraph should be about 60 characters wide.
If we want to make an indented quotation,
we need to use both the .ll and the .in commands.
.sp
.ll 50
.in 10
This is a paragraph that is indented at both ends.
This is a paragraph that is indented at both ends.
This is a paragraph that is indented at both ends.
This is a paragraph that is indented at both ends.
This is a paragraph that is indented at both ends.
This is a paragraph that is indented at both ends.
.sp
.ll 60
.in 0
This paragraph is not indented and has a line width of 60 characters.
This paragraph is not indented and has a line width of 60 characters.
This paragraph is not indented and has a line width of 60 characters.
This paragraph is not indented and has a line width of 60 characters.
This paragraph is not indented and has a line width of 60 characters.

$Id: in2.paragraph.rf,v 1.1 2011-04-08 17:21:44-07 - - $
```

```
.po 0
.pl 55
This document has a page length of 20 lines and so should produce
several pages.
Note the page eject after the first para.

This is a very simple example of some text that should be formatted.
As the lines are fetched form the input,
the output lines are adjusted so that each line is approximately
the same length.
.bp

.\"-- these comments should not appear in the output
.\"-- no comment
.\"-- Any line beginning with .\" is completely ignored.
Note that a blank line produces a paragraph break.
So we can easily use this as a primitiver formatter just like
fmt(1),
even if we don't want to do anything complicated.

The present assignment does not require that any line justification
be given.
Neither need any fancy PostScript formatting output be done.
Note that there is one space between words except when a word
ends a sentence.
A sentence ends when the word ends with one of the following
punctuation marks:
?!;:.

Many text formatters also allow for both right justification
rather than ragged right typesetting and also for hyphenation.
We do not require that here.
Also not that while sentence-ending punctuation works,
abbreviations ending with periods are treated as ending sentences,
which is not quite right.
And if we're not careful in formatting things,
very-long-words-which-appear-in-the-wrong place can cause
funny-looking line-breaking.

$Id: in4.longer.rf,v 1.1 2011-04-08 17:21:44-07 - - $

This is a very simple example of some text that should be formatted.
As the lines are fetched form the input,
the output lines are adjusted so that each line is approximately
the same length.

Note that a blank line produces a paragraph break.
So we can easily use this as a primitiver formatter just like
fmt(1),
even if we don't want to do anything complicated.

The present assignment does not require that any line justification
be given.
Neither need any fancy PostScript formatting output be done.
Note that there is one space between words except when a word
ends a sentence.
A sentence ends when the word ends with one of the following
punctuation marks:
```

```
?!;:.

Many text formatters also allow for both right justification
rather than ragged right typesetting and also for hyphenation.
We do not require that here.
Also not that while sentence-ending punctuation works,
abbreviations ending with periods are treated as ending sentences,
which is not quite right.
And if we're not careful in formatting things,
very-long-words-which-appear-in-the-wrong place can cause
funny-looking line-breaking.

$Id: in4.longer.rf,v 1.1 2011-04-08 17:21:44-07 - - $

This is a very simple example of some text that should be formatted.
As the lines are fetched form the input,
the output lines are adjusted so that each line is approximately
the same length.

Note that a blank line produces a paragraph break.
So we can easily use this as a primitiver formatter just like
fmt(1),
even if we don't want to do anything complicated.

The present assignment does not require that any line justification
be given.
Neither need any fancy PostScript formatting output be done.
Note that there is one space between words except when a word
ends a sentence.
A sentence ends when the word ends with one of the following
punctuation marks:
?!;:.

Many text formatters also allow for both right justification
rather than ragged right typesetting and also for hyphenation.
We do not require that here.
Also not that while sentence-ending punctuation works,
abbreviations ending with periods are treated as ending sentences,
which is not quite right.
And if we're not careful in formatting things,
very-long-words-which-appear-in-the-wrong place can cause
funny-looking line-breaking.

$Id: in4.longer.rf,v 1.1 2011-04-08 17:21:44-07 - - $

This is a very simple example of some text that should be formatted.
As the lines are fetched form the input,
the output lines are adjusted so that each line is approximately
the same length.

Note that a blank line produces a paragraph break.
So we can easily use this as a primitiver formatter just like
fmt(1),
even if we don't want to do anything complicated.

The present assignment does not require that any line justification
be given.
Neither need any fancy PostScript formatting output be done.
```

```
Note that there is one space between words except when a word
ends a sentence.
A sentence ends when the word ends with one of the following
punctuation marks:
?!;:.

Many text formatters also allow for both right justification
rather than ragged right typesetting and also for hyphenation.
We do not require that here.
Also not that while sentence-ending punctuation works,
abbreviations ending with periods are treated as ending sentences,
which is not quite right.
And if we're not careful in formatting things,
very-long-words-which-appear-in-the-wrong place can cause
funny-looking line-breaking.

$Id: in4.longer.rf,v 1.1 2011-04-08 17:21:44-07 - - $
```

```
.foobar
.ba dcomm
.an d
.this is a bad command
Thisisaverylonglinewhichcannotbebrokenandyouwillgetanerrormessageabout``can'tbr
eakline''.
The message should be something like
ZZ
.sp needs to have a numeric operand.
.sp 3 needs one operand.
$Id: in5.badcmds.rf,v 1.1 2011-04-08 17:21:44-07 - - $
```

```
.po 10
.pl 50
.\" ========================================================
.in 0
NAME
.sp
.in 4
jroff - format documents for display on line printer
.sp
.\" ========================================================
.in 0
SYNOPSIS
.sp
.in 4
jroff [filename...]
.sp
.\" ========================================================
.in 0
DESCRIPTION
.sp
.in 4
`jroff' formats text according to control lines embedded in the text
of the given input files.
Control lines begin with a period (.) in column 1 and all other
lines are text lines to be formatted.
Paragraphs are delimited by empty lines or control lines.
.sp
.\" ========================================================
.in 0
OPTIONS
.sp
.in 4
None.
.sp
.\" ========================================================
.in 0
OPERANDS
.sp
.in 4
All operands are filenames.
Each file is read in turn as input to the formatter.
If a filename is specified as a minus sign (-),
the standard input is read at that point.
If no filenames are given, the standard input is read.
The formatted document is written to the standard output.
.sp
.\" ========================================================
.in 0
EXIT STATUS
.sp
.in 4
0
.in 8
No errors were encountered.
.in 4
1
.in 8
One or more files were not found or invalid commands were found.
```

```
Error messages were written to the standard error.
.sp
.\" ============================================================
.in 0
SEE ALSO
.sp
.in 4
fmt(1),
nroff(1),
groff(1),
latex(1),
tex(1).
.sp
.\" ============================================================
.in 0
INPUT TEXT
.sp
.in 4
Input lines are of three types:
empty lines,
which cause a paragraph to be terminated and are equivalent to
the `.sp' command;
text lines, which are added to the current paragraph,
and commands, which being with a period (.) in column 1.
White space is ignored, except as separators.
Whenever a word of output is printed,
it is followed by one space unless it is at the end of a sentence.
A word is at the end of a sentence if its last character is one
of the characters period (.), question (?), bang (!), colon(:),
or semi-colon (;).
.sp
.\" ============================================================
.in 0
COMMANDS
.sp
.in 4
The following commands are recognized.
Whenever an operand of `N' is specified on a command,
the operand must be numeric, and if missing,
the default value is used.
`C' indicates a single character operand.
.sp
.\" ------------------------------------------------------------
.cc :
.\" anything
:cc
.in 8
`jroff' comments are ignored and are useful for private
documentation purposes.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.bp
:cc
.in 8
Begin page, causing a break.
Write a form feed (^L) character immediately before printing
```

```
the page header for the next page.
When the next paragraph is dumped,
The page title will be printed when the next line after this
is printed.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.br
:cc
.in 8
Break.
The paragraph is dumped and a new paragraph is begun.
There is no empty line between paragraphs.
Same as `.sp 0'.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.cc C
:cc
.in 8
Sets the control character to C.
The control character is that which appears in position 0 of
the input line to signal a command.
Default:
`.'.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.in N
:cc
.in 8
Set the indentation to N characters.
This is the number of spaces printed at the beginning of each
line in addition to the page offset.
Also causes a break (.br).
Default:
0.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.ll N
:cc
.in 8
Set output line length to N characters.
The indentation is included in this, but not the page offset.
Default:
65.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.mt N
:cc
.in 8
```

```
Specifies the margin at the top of the page in lines.
The title is vertically centered in this margin.
N/2 empty lines are printed,
then the title,
then N/2-1 empty lines.
Default:
0.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.tl 'left'mid'right'
:cc
.in 8
A three part page header may be specified for the left,
middle, and right side of the page.
If a percent (%) appears anywhere in the title,
it is replaced by a page number.
The left part is left justified within the output line,
the right part is right justified,
and the mid part is centered one the entire output line.
Default:
'''%'.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.pl N
:cc
.in 8
Set the page length to N lines.
Whever this many lines have been output, a page eject occurs,
and the page header is printed immediately before printing
the next line of text.
Default:
60.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.po N
:cc
.in 8
Page offset is N characters.
This is the number of space characters printed at the beginning
of every output line.
No characters are printed if the line is empty.
Default:
10.
.in 4
.sp
.\" ------------------------------------------------------------
.cc :
.sp N
:cc
.in 8
End a paragraph and print N blank lines.
Each blank input line is the same as `.sp 1'.
```

```
If N is more than the number of output lines left on the current
page, behaves exactly like `.bp'.
Default:
1.
.in 4
.sp
.\" ------------------------------------------------------------
.sp
.\" ============================================================
.in 0
SEE ALSO
.sp
.in 4
nroff(1),
groff(1),
latex(1),
tex(1).
.sp
.\" ============================================================
.in 0
BUGS
.sp
.in 4
Very few commands have been implemented,
the most serious being the lack of a macro preprocessor in order
to make typing in the markup language less tedious.
.sp
.\" ============================================================
.in 0
RCSID
.sp
.in 4
$Id: in6.jroff.rf,v 1.1 2011-04-08 17:21:44-07 - - $
.in 0
.sp
```