YAWP 0.5.1 MANUAL

Yet Another Word Processor

2022-03-19

Carlo Alessandro Verre

carlo.alessandro.verre@gmail.com

I sound my barbaric yawp over the roofs of the world

Walt Whitman

CONTENTS

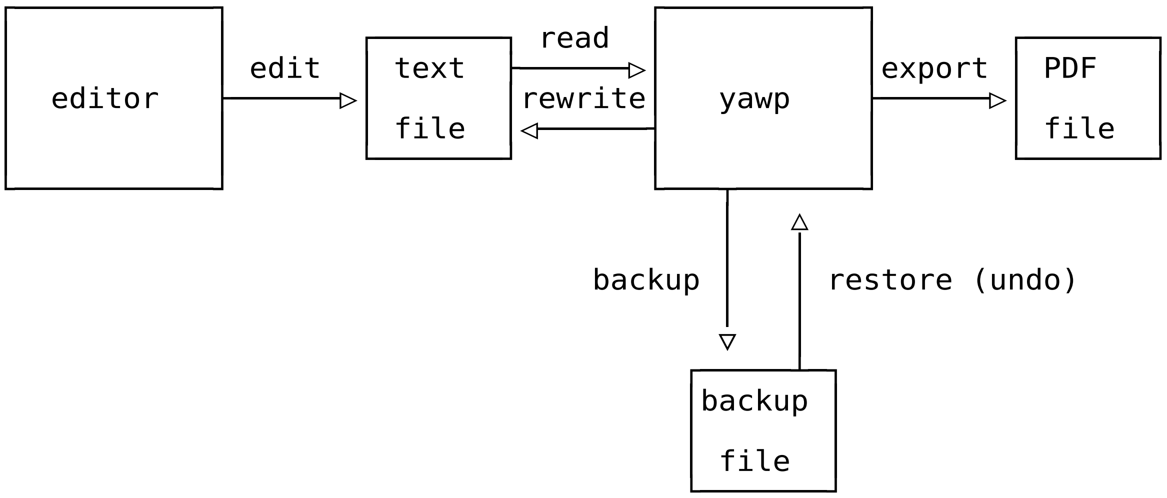
1. INTRODUCTION

1.1. WHAT IS YAWP?

The name "yawp" here means Yet Another Word Processor, and yawp is an automatic word processor for plain text files, with PDF output. If you want to quickly create a no-frills but well-formatted document, you can:

- edit a text file by your favorite editor
- run yawp in order to:
  - backup read format and rewrite the text file
  - export the text file in a PDF file
  - open the PDF file for check or print
- go back to the editor and update the text file or finish

The following flowchart illustrates inputs and outputs:



Main features are:

- yawp processes in place a single text file, hereinafter referred to simply as the "file"
- yawp before processing makes a timestamped backup of the file, allowing undo operation (see 2. Usage Modes)
- yawp processing is driven by the text in the file and by arguments only, not by commands or tags embedded in text
- yawp justifies (see 3. Justification) text at left and right in:
  - unindented paragraphs
  - dot-marked indented paragraphs (as this one)
- yawp accepts unjustified pictures (as schemas, tables and code examples) freely intermixed with text
- you can sketch in pictures segments (by '`') and arrowheads (by '^'), yawp redraws them by proper graphic characters (see 4. Graphics)
- yawp adopts an ad hoc policy for Python files, formatting the docstrings but not the Python code (see 5. Python Files)
- yawp performs multi-level chapter renumbering (see 6.1. Numbered Chapters)
- yawp inserts an automatic contents chapter in the file (see 6.2. Contents Chapter)
- yawp recognizes relevant subjects (quoted by '"') and inserts an automatic index chapter in the file (see 6.3. Index Chapter)
- yawp cuts the file in pages, by automatic insertion of two-lines page headers (see 7. Pages)
- yawp exports the resulting lines in PDF format, with control over character size and page layout, and opens for you the generated PDF file, allowing preview and printing (see 8. Pdf Export)
- yawp corrects errors made by CUPS-PDF about font size and page margins (see 9. Size Correction)
- yawp is "stable", namely if after a yawp execution you run yawp again

  on the same file with the same arguments then the file content doesn't change (except date and time in page headers, see 7. Pages)
- as a beta release, yawp 0.5.1. contains some debug functionalities not aimed at the end user, they will disappear in some future release (see 10. Debug)

Believe or not, anything has been kept as simple as possible.

As an example, this documentation you're reading has been created as yawp.pdf from yawp.txt by typing:

```
| $ yawp -v -w 75 -F -E 'Yawp 0.5.1 Manual' yawp.txt
```

Other examples are scattered below.

1.2. HISTORY

- version 0.5.1
  - completely redefined and rewritten
  - beta release

- version 0.4.2
  - obsolete and deprecated

- version 0.4.1
  - first version on pypi.org
  - obsolete and deprecated

1.3. INSTALLATION

CUPS-PDF provides a PDF Writer backend to CUPS, and yawp needs it to export the file in PDF format. For example, if your Linux belongs to the Debian family, type:

```
| $ sudo apt-get -y update
| $ sudo apt-get -y install printer-driver-cups-pdf
```

If you type at terminal:

```
| $ pip3 install -U yawp
```

this command will:

- install current version of yawp if not present
- upgrade yawp to the current version if already installed

2. USAGE MODES

General behaviour is controlled by the following arguments:

- "-h, --help": show a short help message and exit

- "-H, --manual": open this yawp-generated "yawp Manual" in PDF format and exit

- "-V, --version": show program's version number and exit

- "-v, --verbose": display information messages on stderr

During execution yawp can write two kinds of messages:

- an "information message" says what's going on (if -v option is on)
- an "error message" says what's wrong and stops processing

Any error is a fatal error. If an error message is issued, file backup and rewriting don't take place and yawp execution is terminated. There are no warning messages, which allow the process to continue after an error.

All messages are written on stderr, in order to avoid interference with -p option, which writes the file on stdout.

When applicable, error messages are preceded by the position in the file and the content of the offending line.

Normally the file is backed up into a timestamped copy and formatted, this may be changed by -U or -N arguments:

- "-U, --undo": restore the file from its the previous version, example:

```
$ yawp -v -U -P0 example.txt
Read: yawp <-- '~/example.txt'
    0 header lines, max 0 chars per line, 1 page
    13 body lines, max 14 chars per line
    13 total lines, max 14 chars per line
Compute: -w 14
Compute: -W 34.4207pt = 0.4781in = 12.1429mm = 1.2143cm
Restore: '~/example.txt' <-- '~/example-2022.03.15-14.28.32.txt'
    0 header lines, max 0 chars per line, 1 page
    13 body lines, max 14 chars per line
    13 total lines, max 14 chars per line
```

BEWARE: undo operation can't be undone, the file goes back to the penultimate version and the last one is lost.

- "-N, --no-format": leave the file unchanged, example:

```
$ yawp -v -N -P0 example.txt
Read: yawp <-- '~/example.txt'
    0 header lines, max 0 chars per line, 1 page
    13 body lines, max 14 chars per line
    13 total lines, max 14 chars per line
Compute: -w 14
Compute: -W 34.4207pt = 0.4781in = 12.1429mm = 1.2143cm
```

If -U and -N are both left off, normal operation follows, the file is backed up (in a timestamped copy) and formatted. To turn on both -U and -N is not allowed.

You may see at terminal (or redirect) the file by -p argument:

- "-p, --print-file": at end print file on stdout

3. FORMATTING

Let's distinguish in four categories the input file lines:

- a line is an "empty line" if it contains no characters, in input lines all trailing blanks are stripped away, hence every input line containing only blanks becomes an empty line
- otherwise a line is a "dot line" if the first nonblank character is a "decimal point character" '.' or a "black small circle character" '•' followed by a blank (but on output such a '.' is always replaced by '•')
- otherwise a line is an "indented line" if it starts with a blank
- otherwise a line is an "unindented line"

Header lines inserted on output by yawp (see 7. Paging) are preliminarily canceled from input.

The formatting algorithm, driven by the input lines, oscillates between two states:

- "picture state", where input lines are directly written out as they are
- "text state", where input lines are accumulated into a paragraph buffer for further justification and writing at paragraph end

The picture state is the initial state. In this state, if the input is:

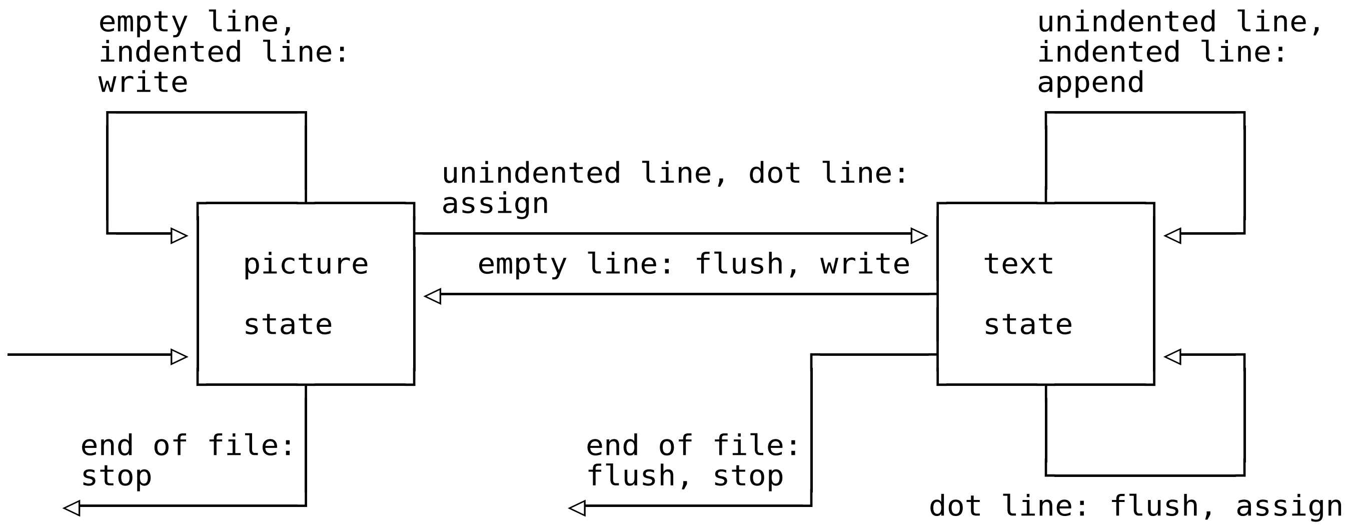
- an empty line or an indented line: the line is written out as is
- an unindented line: text state is entered, an "unindented paragraph" begins, the line is shrunk and assigned to the paragraph buffer, paragraph left indentation is set to zero
- a dot line: text state is entered, an "indented paragraph" begins, the line is shrunk and assigned to the paragraph buffer, paragraph left indentation is set to the position of initial dot character plus two
- end of input file: processing is terminated

Here to "shrink" a string means to strip away all leading, intermediate multiple, and trailing blanks.

When we are in text state, if the input line is:

- an empty line: the paragraph buffer is flushed (justified, written out and emptied), state goes back to picture state, the empty line is written out
- an indented or unindented line: the line is shrunk and appended to the paragraph buffer
- a dot line: paragraph buffer is flushed, a new paragraph is started, the line is shrunk and assigned to the paragraph buffer, paragraph left indentation is set to the position of initial dot plus two
- end of input file: paragraph buffer is flushed, processing is ended

The following state diagram illustrates states and transitions:



Actions associated to transitions are:

- write: the input line is immediately written out unchanged as a "picture line"
- assign: the input line is shrunk and assigned to the paragraph buffer
- append: the input line is shrunk and appended to the paragraph buffer
- flush: the paragraph buffer is flushed, namely is justified, written out as "text lines", and emptied
- stop: formatting is finished

Max length of text lines can be controlled by:

- "-w, --chars-per-line": line width in chars per line (default: 0 = automatic)

-w and -W (character width, see later) can be zero hence "automatic", namely:

- if -w is automatic and -W is not, -w is computed from -W
- if -W is automatic and -w is not, -W is computed from -w
- if both -w and -W are automatic,
  - -w is deduced from max line length in file (page headers are not considered)
  - -W is computed from -w

Text justification is controlled by -l argument:

- "-l, --left-only": justify at left only (default: left and right)

After justification:

- consecutive empty lines between text paragraphs are reduced to a single empty line
- consecutive empty lines in contact with a picture are left unchanged

Now we can define some "best practice" to follow writing the file.

Unindented paragraphs should be:

- preceded by an empty line
- started by an unindented line
- continued by indented or unindented lines
- ended by an empty line (better) or by a dot line

Indented paragraphs should be:

- preceded by a line of any kind
- initiated by a dot line

- continued by indented or unindented lines
- ended by an empty line or by another dot line

Pictures should be:

- preceded by an empty line
- initiated and continued by indented lines only
- ended by an empty line (better) or by an unindented line or by a dot line

4. GRAPHICS

You can sketch pictures with boxes arrows and tables by two "draw characters":

- "back quote character" '`' to draw horizontal and vertical segments
- "circumflex accent character" '^' to mark an arrowhead

If you turn on the argument:

- "-g, --graphics": redraw '`'-segments and '^'-arrowheads

then yawp will redraw the pictures by replacing the draw characters with suitable graphic ones, depending on the other four characters around (left right above and below). Standalone draw characters are not replaced.

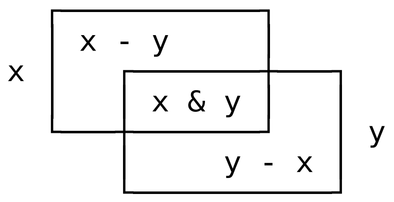
This feature is active in picture mode only, so it does not work in text paragraphs.

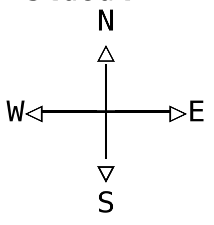
If -f or -F are turned on, and if a figure is too tall to enter into the current page, a page eject is forced. But if the figure is higher than one page, page eject doesn't take place.

An example:

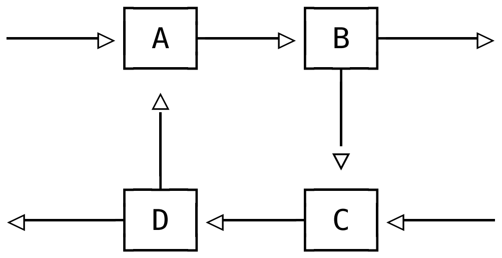
~~~
$ cat graphics.txt
      `````````````             N
      ` x - y     `             ^
    x `   `````````````         `       `````````````
      `   ` x & y `   `    W^```````^E  `N `N2 ` N3 `
      `````````````   ` y       `       `````````````
          `     y - x `         ^       ` 0`  0`   0`
          `````````````         S       ` 1`  1`   1`
                                        ` 2`  4`   8`
          `````     `````               ` 3`  9`  27`
    `````^` A `````^` B ``````^         ` 4` 16`  64`
          `````     `````               ` 5` 25` 125`
            ^         `                 ` 6` 36` 216`
            `         `                 ` 7` 49` 343`
            `         ^                 ` 8` 64` 512`
          `````     `````               ` 9` 81` 729`
    ^`````` D `^````` C `^`````         `10`100`1000`
          `````     `````               `````````````
~~~

```
$ yawp -v -g -p -P0 graphics.txt
Read: yawp <-- '~/graphics.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 53 chars per line
    17 total lines, max 53 chars per line
Compute: -w 53
Compute: -W 9.0923pt = 0.1263in = 3.2075mm = 0.3208cm
Backup: '~/graphics.txt' --> '~/graphics-2022.03.11-17.35.02.txt'
Rewrite: yawp --> '~/graphics.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 53 chars per line
    17 total lines, max 53 chars per line
Print: '~/graphics.txt' --> stdout
```





N	N2	N3
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000



5. PYTHON FILES

Python files deserve a special treatment. If the textfile filename ends with '.py' extension, then we suppose the file is a Python source, hence we are interested to format docstrings and not Python code. So the formatting function is alternatively turned on and off by switch lines. A "switch line" is a line containing a "'''" string.

Note that yawp never formats switch lines, formatting takes place from the line after the start switch line until the line before the next stop switch line.

So your Python file must follow some simple rules:

- docstrings to be formatted must start and ended by "'''" and not '"""'
- long strings not to be formatted must start and end with '"""' and not "'''"
- a "'''" inside a string can be coded for instance as '\'\'\'', so containing line will not be identified as a switch line

An error in switch lines could format and destroy your Python code. A preliminary check prints an error message and stops execution before file formatting if the total number of switch lines is odd. This should intercept 90% of errors, anyway after yawp processing check the result and if needed go back to previous version by -U. An example:

```
$ cat pycode.py
''' Start switch line is not formatted.
This is a one-line unindented paragraph.

This is a first multiline unindented paragraph.
This is a first multiline unindented paragraph.
This is a first multiline unindented paragraph.

    This is a picture.
        This is a picture.
    This is a picture.

    . This is a first multi line indented paragraph.
This is a first multiline indented paragraph.
This is a first multiline indented paragraph.
        . This is another multi line indented paragraph.
This is another multiline indented paragraph.
This is another multiline indented paragraph.
''' # Stop switch line line is not formatted.

def double(x): # Python code is not formatted.
 ''' Start switch line is not formatted.
This is another multiline unindented paragraph.
This is another multiline unindented paragraph.
This is another multiline unindented paragraph.
''' # Stop switch line is not formatted.
    return x + x # Python code is not formatted.
```

```
$ yawp -v -w 45 -p -P0 pycode.py
Read: yawp <-- '~/pycode.py'
    0 header lines, max 0 chars per line, 1 page
    28 body lines, max 56 chars per line
    28 total lines, max 56 chars per line
Compute: -W 10.7087pt = 0.1487in = 3.7778mm = 0.3778cm
Backup: '~/pycode.py' --> '~/pycode-2022.03.17-12.48.12.py'
Rewrite: yawp --> '~/pycode.py'
    0 header lines, max 0 chars per line, 1 page
    33 body lines, max 48 chars per line
    33 total lines, max 48 chars per line
Print: '~/pycode.py' --> stdout
''' Start switch line is not formatted.
This is a one-line unindented paragraph.

This   is   a   first   multiline  unindented
paragraph.   This   is   a   first  multiline
unindented   paragraph.   This   is  a  first
multiline unindented paragraph.

    This is a picture.
        This is a picture.
    This is a picture.

    • This  is  a  first  multi line indented
      paragraph.  This  is  a first multiline
      indented  paragraph.  This  is  a first
      multiline indented paragraph.
        • This is another multi line indented
          paragraph.   This   is   another
          multiline  indented paragraph. This
          is   another   multiline   indented
          paragraph.
''' # Stop switch line line is not formatted.

def double(x): # Python code is not formatted.
 ''' Start switch line is not formatted.
This    is   another   multiline   unindented
paragraph.    This   is   another   multiline
unindented   paragraph.   This   is   another
multiline unindented paragraph.
''' # Stop switch line is not formatted.
    return x + x # Python code is not formatted.
```

6. CHAPTERS

The file can be partitioned in "chapters" by "chapter lines". A chapter can be:

- the "nameless chapter", the first one, from first line until first chapter line
- a "numbered chapter", started by a "numbered line", how many do you want
- the "contents chapter", started by a "contents line", no more than one
- the "index chapter", started by an "index line", no more than one

A chapter line:

- is the first line or is preceded by an empty line
- is the last line or is followed by an empty line
- is a non-empty unindented (not starting with blank) line

So, as stated above, a chapter line must be a one-line unindented paragraph. But not all one-line unindented paragraphs are chapter lines, as follows.

6.1. NUMBERED CHAPTERS

Start of a "numbered chapter" is recognized by a numbered line. A line is a "numbered line" if:

- is the first line or is preceded by an empty line
- is the last line or is followed by an empty line
- does not start with blank
- contains:
    - one or more "number-dot couples", each made by
        - one or more decimal digits
        - a "decimal point character" '.'
    - a blank
    - a chapter title

The "level" of a numbered line is the count of number-dot couples in its prefix, examples:

- 12345. a level-1 numbered line
- 1.345. a level-2 numbered line
- 0.0.0. a level-3 numbered line

Numbered lines must follow two quite obvious sequence rules:

- first numbered line in file must be a level-1 numbered line
- each other numbered line can have a level between 1 and the level of the previous chapter line plus 1, but no more

Numbered chapter title is:

- shrunk and uppercased in text
- shrunk and titlecased when inserted:
    - into contents chapter
    - into page headers by '%c'

To "titlecase" a string means to uppercase the first character of each word in it and to lowercase all the others.

Numbers in input don't matter, yawp replaces them by the right ones, only the level matters, example:

```
$ cat chapters.txt
0. AAA aaa

32.33. BBB bbb

0.0. CCC ccc

0. DDD ddd

3.14. EEE eee

0.0.0. FFF fff

0.0. GGG ggg
```

```
$ yawp -v -p -P0 chapters.txt
Read: yawp <-- '~/chapters.txt'
    0 header lines, max 0 chars per line, 1 page
    13 body lines, max 14 chars per line
    13 total lines, max 14 chars per line
Compute: -w 14
Compute: -W 34.4207pt = 0.4781in = 12.1429mm = 1.2143cm
Backup: '~/chapters.txt' --> '~/chapters-2022.03.15-14.05.05.txt'
Rewrite: yawp --> '~/chapters.txt'
    0 header lines, max 0 chars per line, 1 page
    13 body lines, max 14 chars per line
    13 total lines, max 14 chars per line
Print: '~/chapters.txt' --> stdout
1. AAA AAA

1.1. BBB BBB

1.2. CCC CCC

2. DDD DDD

2.1. EEE EEE

2.1.1. FFF FFF

2.2. GGG GGG
```

6.2. CONTENTS CHAPTER

For "contents chapter" we mean the list of chapters, possibly with the number of the page they begin on. Namely, the contents chapter will list all numbered chapters and the index chapter, but will not list the nameless chapter or the contents chapter itself.

Contents line starting the contents chapter is defined by -c argument:

- "-c, --contents-title": title of contents chapter (default: 'contents'), example:

```
$ yawp -c 'list of chapters' ...
```

A line is a "contents line" if:

- is the first line or is preceded by an empty line
- is the last line or is followed by an empty line
- does not start with blank
- its shrunk and uppercased value is equal to the shrunk and uppercased value of the -c argument

If the file contains a contents line then:

- lines until next chapter line (or until end of file) are supposed to be the old contents chapter and are deleted
- a new contents chapter is inserted instead

The file may or may not contain a contents chapter, but no more than one. The contents chapter may appear everywhere, on top, on bottom or in the middle of file, and after or before the index chapter.

Contents chapter title is:

- shrunk and uppercased in text
- shrunk and titlecased when inserted into page headers by '%c'

BEWARE: an error in the next chapter line could erase a piece of your file, so after yawp processing check the result and if needed go back to previous version by -U.

Example:

```
$ cat contents.txt
Contents

0. AAA aaa

32.33. BBB bbb

0.0. CCC ccc

0. DDD ddd

3.14. EEE eee

0.0.0. FFF fff

0.0. GGG ggg
```

```
$ yawp -v -p -P0 contents.txt
Read: yawp <-- '~/contents.txt'
    0 header lines, max 0 chars per line, 1 page
    18 body lines, max 25 chars per line
    18 total lines, max 25 chars per line
Compute: -w 25
Compute: -W 19.2756pt = 0.2677in = 6.8mm = 0.68cm
Backup: '~/contents.txt' --> '~/contents-2022.03.15-14.20.42.txt'
Rewrite: yawp --> '~/contents.txt'
    0 header lines, max 0 chars per line, 1 page
    23 body lines, max 20 chars per line
    23 total lines, max 20 chars per line
Print: '~/contents.txt' --> stdout
CONTENTS

    • 1.     Aaa Aaa
    • 1.1.   Bbb Bbb
    • 1.2.   Ccc Ccc
    • 2.     Ddd Ddd
    • 2.1.   Eee Eee
    • 2.1.1. Fff Fff
    • 2.2.   Ggg Ggg

1. AAA AAA

1.1. BBB BBB

1.2. CCC CCC

2. DDD DDD

2.1. EEE EEE

2.1.1. FFF FFF

2.2. GGG GGG
```

6.3. INDEX CHAPTER

For "index chapter" here we mean an alphabetical list of subjects, possibly showing which pages a subject is on. For "subject" we mean any string preceded and followed by a "double quote character" '"'.

Subject length is controlled by:

- "-m, --max-subject": max subject length in index chapter (default: 36)

this should intercept unpaired double quotes. But double quotes preceded or followed by a "single quote character" "'" or by another double quote are not taken into account, in order to allow things like '"' in text.

Index line starting the index chapter is defined by -i argument:

- "-i, --index-title": title of index chapter (default: 'index'), example:

```
$ yawp -i 'list of subjects' ...
```

A line is an "index line" if:

- is the first line or is preceded by an empty line
- is the last line or is followed by an empty line
- does not start with blank
- its shrunk and uppercased value is equal to the shrunk and uppercased value of the -i argument

If the file contains an index line then:

- lines until next chapter line (or until end of file) are supposed to be the old index chapter and are deleted
- a new index chapter is inserted instead

The file may or may not contain an index chapter, but no more than one. The index chapter may appear everywhere, on top, on bottom or in the middle of file, and after or before the contents chapter.

Index chapter title is:

- shrunk and uppercased in text
- shrunk and titlecased when inserted:
    - into contents chapter
    - into page headers by '%c'

BEWARE: an error in the next chapter line could erase a piece of your file, so after yawp processing check the result and if needed go back to previous version by -U.

For an example, see the index chapter at end of this manual.

7. PAGING

Normally the file is not splitted in pages by page headers. Insertion of page headers is controlled by -f and -F arguments:

- "-f, --formfeed" : insert a page header on full page

If -f is turned on, a page header is inserted into the file when:

- current line doesn't fit into current page
- current picture doesn't fit into current page

- "-F, --formfeed-chapters": insert a page header on full page and before contents index and level-one chapters

If -F is turned on (or both -f anf -F are), a page header is inserted into the file when:

- current line doesn't fit into current page
- current picture doesn't fit into current page
- current line is a contents line or an index line or a level-one numbered line

So each page, except the first one, is prefixed by a page header. Each page header is made up of two lines:

- a line starting with a "formfeed character" '\f' = '\x0c', containing data such as file name or page number
- a dashed separation line of "macron characters" '¯' = '\xaf'

So a line in the output can be considered as:

- a "header line" if it's first or second line of a page header, as above, or
- a "body line" otherwise

When the file is initially read, all header lines are eliminated. So, if you want to eliminate page headers from your file, just run yawp without -f or -F.

If -f and -F are off and contents chapter and/or index chapter are requested, contents chapter and/or index chapter will still appear, but without page numbers.

Putting page headers in a Python file doesn't make sense. So if the file is a Python file then -f and -F are automatically turned off.

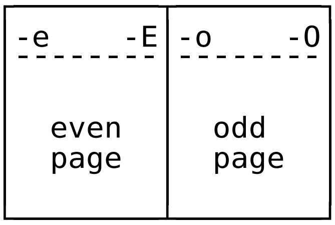
The content of first header line is controlled by -e -E -o -O and -a arguments:

- "-e, --even-left": even page headers, left (default: '%n/%N')

- "-E, --even-right": even page headers, right (default: '%f.%e %Y-%m-%d %H:%M:%S')

- "-o, --odd-left": odd page headers left (default: '%c')

- "-O, --odd-right":odd page headers, right (default: '%n/%N')



Each "%-variable" is evaluated as follows.

VAR	VALUE
'%P'	file long path, with no ending '/'
'%p'	file short path, with no ending '/'
'%f'	file name, with no extension
'%e'	file extension, with no separator '.'
'%Y'	current year, 4 digits
'%m'	current month, 2 digits
'%d'	current day, 2 digits
'%H'	current hour, 2 digits
'%M'	current minute, 2 digits
'%S'	current second, 2 digits
'%n'	current page number
'%N'	total number of pages
'%c'	current contents, index or level-one chapter
'%%'	'%'

No other %-variable is allowed.

If for instance the file is '/home/xxxx/yyy/zzz.www.txt' and the user is 'xxxx' then we get:

VAR	VALUE
'%P'	'/home/xxxx/yyy'
'%p'	'~/yyy'
'%f'	'zzz.www'
'%e'	'txt'

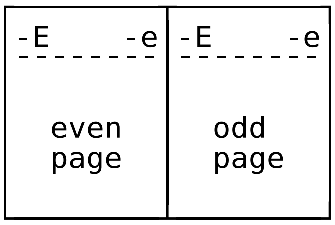
and so:

- '%P/%f.%e' --> '/home/xxxx/yyy/zzz.www.txt'
- '%p/%f.%e' --> '~/yyy/zzz.www.txt'

If you don't need front-back printing, set:

- "-a, --all-headers-E-e": all page headers contain -E at left and -e at right

and so you will get the same header format on even and odd pages. Note the swap between -e and -E.



8. PDF EXPORTING

Normally the file is exported to a PDF file, which is then opened by calling the system default PDF browser (as evince or atril). If you don't want this to happen, type '-P0'.

- "-P, --file-PDF": at end export and open PDF file (0 = no PDF export, default: '%P/%f.pdf')

Each "%-variable" is evaluated as explained in previous chapter, but '%n' '%N' and '%c' are not applicable and not allowed. Resulting file name must end with '.pdf'.

Character size is defined by -W and -A arguments:

- "-W, --char-width": char width (pt/in/mm/cm, default: 0 = automatic)

Value is a float literal followed by a case-insensitive suffix:

- 'pt' for points (1 inch = 72 points)
- 'in' for inches
- 'mm' for millimeters
- 'cm' for centimeters

so for instance these are all equivalent, giving a value of one inch:

- -W 72pt
- -W 1in
- -W 25.4mm
- -W 2.54cm

Only a zero value (as '0' or '0.0') can lack the suffix, because of course 0pt = 0in = 0mm = 0cm, but a zero value for -W means "automatic".

Both -W and -w (line width in chars per line, see before) can be zero hence automatic, namely:

- if -w is automatic and -W is not, -w is computed from -W
- if -W is automatic and -w is not, -W is computed from -w
- if both -w and -W are automatic,
  - -w is deduced from max line length in file (page headers are not considered)
  - -W is computed from -w

Given the character width by -W, the character height is derived from -A:

- "-A, --char-aspect": char aspect ratio = char width / char height (1 = square grid, default: 3/5)

-A is a ratio, so can be:

- an integer or float literal (as 1 or 0.6)
- two integer or float literals, separated by a "slash character" '/' (as 3/5 or 3/5.0)

Dimensions of the paper sheet, expressed in points inches centimeters or millimeters (as explained for -W before) are defined by -S:

- "-S, --paper-size": portrait paper size (width x height, pt/in/mm/cm, default: 'A4' = '210x297mm')

Format is portrait, in other words the width can't be greater than the height.

Values can be names too, see the following table of allowed names.

NAME	VALUE
half letter	5.5x8.5in
letter	8.5x11.0in
legal	8.5x14.0in
junior legal	5.0x8.0in
ledger	11.0x17.0in
tabloid	11.0x17.0in
a0	841x1189mm
a1	594x841mm
a2	420x594mm
a3	297x420mm
a4	210x297mm
a5	148x210mm
a6	105x148mm
a7	74x105mm
a8	52x74mm
a9	37x52mm
a10	26x37mm
b0	1414x1000mm
b1	1000x707mm
b1+	1020x720mm
b2	707x500mm
b2+	720x520mm
b3	500x353mm
b4	353x250mm
b5	250x176mm
b6	176x125mm
b7	125x88mm
b8	88x62mm
b9	62x44mm
b10	44x31mm

These names, the 'x' separator and the suffix are all case-insensitive.

Paper width and height are exchanged with each other by -Z, in order to obtain a landscape orientation:

- "-Z, --landscape": turn page by 90 degrees (default: portrait)

Print quality is defined by -Q, but reducing the print quality doesn't make much sense, it only slightly reduces the length of the exported PDF file:

- "-Q, --print-quality": print quality (0 1 or 2, default: 2)

Unprintable margins around the paper sheet are controlled by:

- "-L, --left-margin": left margin (pt/in/mm/cm, 2cm..8cm, default: 2cm)

- "-R, --right-margin": right margin (pt/in/mm/cm, 2cm..8cm, default: -L)

- "-T, --top-margin": top margin (pt/in/mm/cm, 2cm..8cm, default: 2cm)

- "-B, --bottom-margin": bottom margin (pt/in/mm/cm, 2cm..8cm, default: -T)

Margins are expressed in points inches centimeters or millimeters as explained above for -W.

If -R has the default value '-L' then it's forced to the value of -L.

If -B has the default value '-T' then it's forced to the value of -T.

9. SIZE CORRECTION

Export of the PDF file is performed by CUPS (Common Unix Printing System) via the "lp" Unix command. Geometry is controlled by yawp passing to lp various arguments:

- -o cpi=N (number of characters per inch, default=10)
- -o lpi=N (number of lines per inch, default=6)
- -o page-left=N (left page margin, value in points)
- -o page-right=N (right page margin, value in points)
- -o page-top=N (top page margin, value in points)
- -o page-bottom=N (bottom page margin, value in points)

These options are undocumented in the lp man page, but you can find them for instance in:

    https://www.computerhope.com/Unix/ulp.htm

Unfortunately, by printing the PDF file on paper the above options are not respected, but are affected by not negligible errors. With "lpr" command these results don't change.

A hardwired mechanism in yawp tries to correct such errors. This device has been tuned for the default paper size 'A4' = '210x297mm', both portrait and landscape. In a future release this could be made parametric thru a configuration file.

10. DEBUGGING

This is a beta release, so it contains some functionality aimed to the debug and not to the end user. They could disappear in a future release.

If you turn on:

- "-b, --dump-buffer": debug, dump content of internal buffer

then the internal buffer where file processing happens is dumped on stderr.

If you turn on:

- "-s, --echo-shell": debug, display invoked Unix commands

then Unix commands invoked during execution are displayed on stderr, along with their output.

If you turn on:

- "-k, --calibration": debug, don't adjust char size and page margins

then the correction device described in previous chapter is turned off. This is useful in setting up the correction device of the previous chapter. As a side effect the page margins -L -R -T and -B are no more bounded in the 2cm..8cm interval.

11. SUMMARY OF ARGUMENTS

General arguments:

- "-h, --help": show this help message and exit
- "-H, --manual": open yawp Manual in PDF format and exit
- "-V, --version": show program's version number and exit
- "-v, --verbose": display information messages on stderr
- "-U, --undo": restore the file from its the previous version
- "-N, --no-format": leave the file unchanged
- "-p, --print-file": at end print file on stdout

Formatting arguments:

- "-w, --chars-per-line": line width in chars per line (default: 0 = automatic)
- "-l, --left-only": justify at left only (default: left and right)
- "-g, --graphics": redraw '`'-segments and '^'-arrowheads
- "-c, --contents-title": title of contents chapter (default: 'contents')
- "-i, --index-title": title of index chapter (default: 'index')
- "-m, --max-subject": max index subject length (default: 36)

Paging arguments:

- "-f, --formfeed": insert a page header on full page
- "-F, --formfeed-chapters": insert a page header on full page and before contents index and level-one chapters
- "-e, --even-left": even page headers, left (default: '%n/%N')
- "-E, --even-right": even page headers, right (default: '%f.%e %Y-%m-%d %H:%M:%S')
- "-o, --odd-left": odd page headers, left (default: '%c')
- "-O, --odd-right": odd page headers, right (default: '%n/%N')
- "-a, --all-headers-E-e": all page headers contain -E at left and -e at right

PDF exporting arguments:

- "-P, --file-PDF": file name of exported PDF file (0 = no PDF export, default: '%P/%f.pdf')
- "-W, --char-width": character width (pt/in/mm/cm, default: 0 = automatic)
- "-A, --char-aspect": char aspect ratio = char width / char height (1 = square grid, default: 3/5')
- "-S, --paper-size": portrait paper size (width x height, pt/in/mm/cm, default: 'A4' = '210x297mm'
- "-Z, --landscape": turn page by 90 degrees (default: portrait)
- "-Q, --print-quality": print quality (0 1 or 2, default: 2')
- "-L, --left-margin": left margin (pt/in/mm/cm, 2cm..8cm, default: 2cm)
- "-R, --right-margin": right margin (pt/in/mm/cm, 2cm..8cm, default: -L)
- "-T, --top-margin": top margin (pt/in/mm/cm, 2cm..8cm, default: 2cm)
- "-B, --bottom-margin": bottom margin (pt/in/mm/cm, 2cm..8cm, default: -T)

Debugging arguments:

- "-b, --dump-buffer": debug, dump content of internal buffer
- "-s, --echo-shell": debug, display invoked Unix commands
- "-k, --calibration": debug, don't adjust char size and page margins

Positional argument:

- "file": text file to be processed

INDEX