YAWP 0.6.1 MANUAL

Yet Another Word Processor

2022-05-05

Carlo Alessandro Verre

carlo.alessandro.verre@gmail.com

I sound my barbaric yawp over the roofs of the world

Walt Whitman

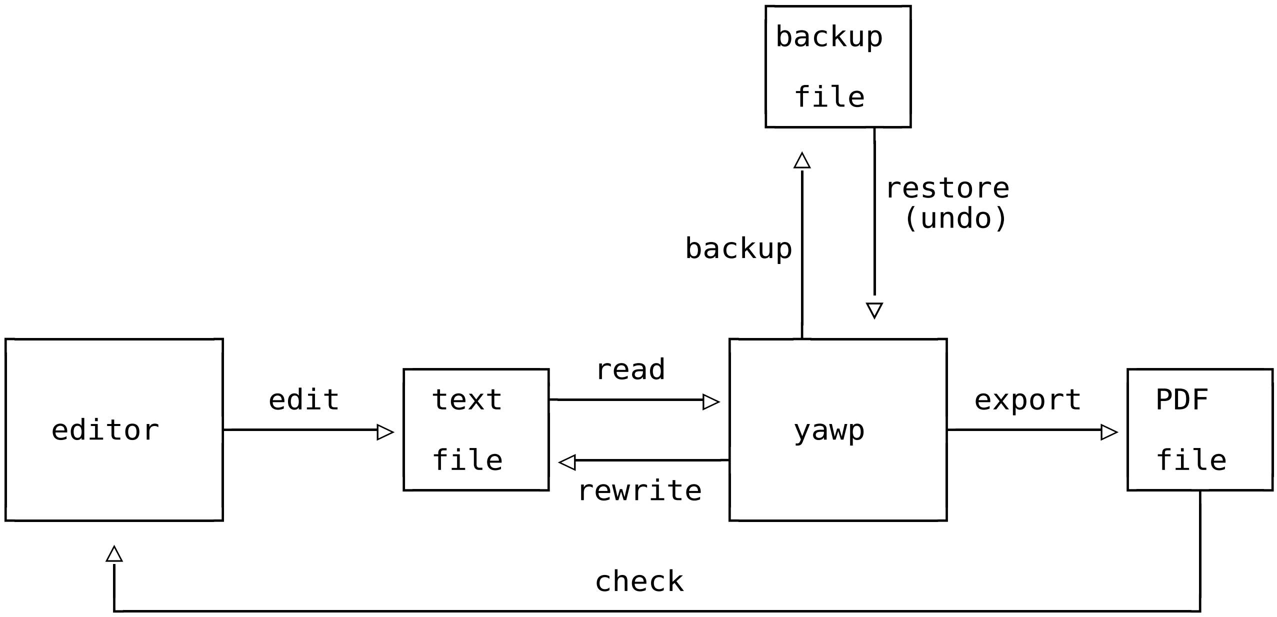
CONTENTS

1. FOREWORDS

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 really need all the features of a full-fledged WYSIWYG word processor as LibreOffice Writer, yawp is not for you. But if you just want to create a simple quick-and-dirty no-frills document, with yawp 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
- possibly go back to the editor and update the text file, or finish



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
- yawp adopts an ad hoc policy for Python files, formatting the docstrings but not the Python code (see '4. Python Files')
- yawp performs multi-level chapter renumbering (see '5.1. Numbered Chapters')
- yawp inserts an automatic contents chapter in the file (see '5.2. Contents Chapter')
- yawp recognizes relevant subjects (quoted by '"') and inserts an automatic index chapter in the file (see 5.3. Index Chapter)
- yawp cuts the file in pages, by automatic insertion of two-lines page headers (see '6. Paging')
- yawp also has some limited graphic features, you can sketch pictures with segments (by '`') and arrowheads (by '^'), yawp redraws them by proper graphic characters (as in the picture above, see '7. Graphics')

- 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 Exporting')
- yawp corrects errors made by CUPS-PDF about font size and page margins, you can use default corrections or redefine them by yawp.cfg (see '9. Corrections')
- yawp is "stable", namely if after a yawp execution you run yawp again on the same file with the same arguments, the file content doesn't change (except date and time in page headers, see '6. Paging')

Everything has been kept as simple as possible.

As an example, this "yawp manual" has been created as yawp.pdf from yawp.txt by typing:

```
| $ yawp -v -w 75 -F -E 'Yawp 0.6.1 Manual' yawp.txt
```

Other examples are scattered below.

1.2. INSTALLATION

CUPS-PDF provides a PDF writer backend to CUPS, and yawp needs it to export the file in PDF format by the lp command. 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 don't have pip, type:

```
| $ sudo apt-get -y install python3-pip
```

If you type at terminal:

```
| $ pip3 install --upgrade yawp
```

this command will:

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

If you see a message like this:

```
| WARNING: The script yawp is installed in ... which is not on PATH.
```

don't worry, a reboot should fix the problem.

1.3. MESSAGES

During execution yawp can write three kinds of messages:

- an "information message" says what's going on (but only if -v is on)
- a "warning message" (starting with 'WARNING:') says what may be wrong, and processing continues
- an "error message" (starting with 'ERROR:') says what's wrong, file backup and rewriting don't take place and yawp execution is terminated

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

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

2. USAGE MODES

General behaviour is controlled by the following arguments:

- "-h, --help": show a short help message and exit
- "-H, --manual": open this PDF "yawp manual" and exit
- "-V, --version": show program's version number and exit
- "-v, --verbose": display information messages on stderr
- "-s, --echo-shell": display invoked Unix commands on stderr

Two arguments, -U and -N, define three usage modes. To turn on both -U and -N is not allowed. If neither -U nor -N is on, yawp runs in "standard mode":

- the file is read, eliminating:
  - page header lines
  - trailing blanks in lines
  - "horizontal tab" '\t' in lines, each replaced by four blanks
- the file content is formatted, see chapters 3. 4. 5. and 6.
- if -g is on, pictures are redrawn, see '7. Graphics' chapter
- the file is backed up in a timestamped copy
- the file is rewritten

If -N is on, yawp runs in "no-format mode":

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

- the file is read, eliminating:
  - trailing blanks in lines
  - "horizontal tab" '\t' in lines, each replaced by four blanks
- if -g is on, pictures are redrawn, see '7. Graphics' chapter
- the file is backed up in a timestamped copy
- the file is rewritten

Note that the input page headers are eliminated (and possibly reinserted) in standard mode, while they are kept in no-format mode.

A text file is a sequence of lines, separated by line terminators. Both in standard and in no-format mode, file reading is in "universal newlines" mode, so three line terminators are accepted:

- "line feed" = '\n' (Unix-Linux-Apple)
- "carriage return" + "line feed" = '\r\n' (Microsoft-Windows)
- "carriage return" = '\r' (Apple old MacOS pre-OSX)

When the file is rewritten, the line terminator is always '\n'.

If -U is on, yawp runs in "undo mode":

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

- the file is recovered from the most recent backup
- -g argument has no effect

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

In all three usage modes, the file is finally written and/or exported, depending on -p and -P arguments:

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

- "-P, --file-pdf": at end export file to this PDF file and open it ('0' = no export, default: '%P/%f.pdf'), see '8. Pdf Exporting'

3. JUSTIFICATION

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

- a line is an "empty line" if it contains no characters (all trailing blanks in input lines 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" '.', or a "black small circle" '•', 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:

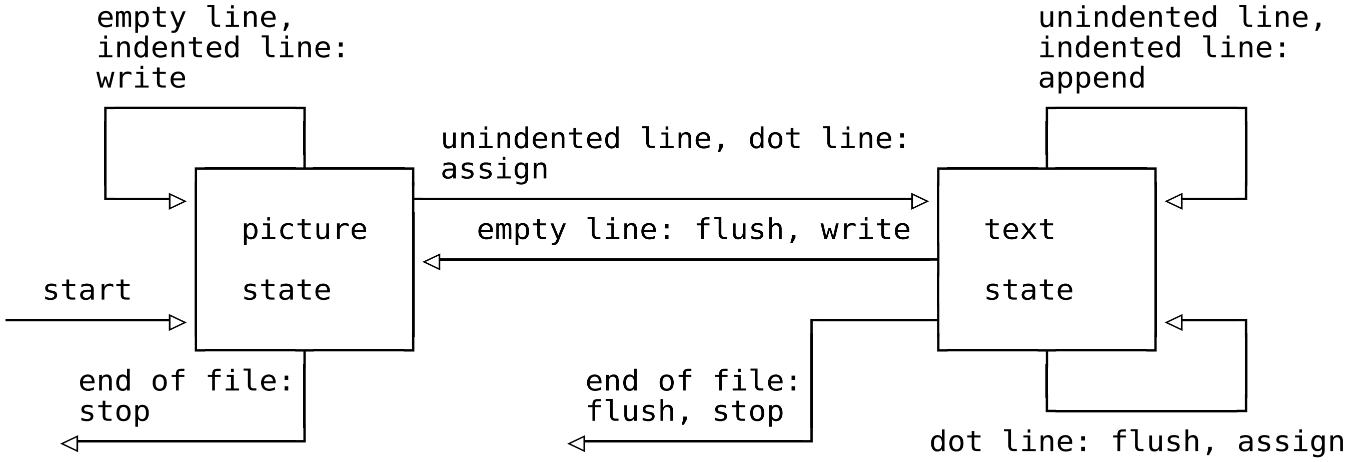
- an empty line or an indented line: the line is written out as is
- an unindented line: text state is entered, an "unindented paragraph" begins, the line is shrunk and assigned to the paragraph buffer, paragraph left indentation is set to zero
- a dot line: text state is entered, an "indented paragraph" 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 and trailing blanks
- and reduce each group of consecutive blanks to a single blank

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



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 characters per line (default: 0 = automatic)

-w and -W (character width, see '8. PDF exporting') 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

As an example, we type:

```
| $ echo BANNER >banner.txt
```

and we impose only one character per line:

```
$ yawp -v -N -w 1 -p banner.txt
Correct: '~/.yawp/yawp.cfg' not found, default corrections
Correct: -L 43.325pt = 0.602in = 15.284mm = 1.528cm
Correct: -R 40.57pt = 0.563in = 14.312mm = 1.431cm
Correct: -T 53.814pt = 0.747in = 18.985mm = 1.898cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
Read: yawp <-- '~/banner.txt'
    0 header lines, max 0 chars per line, 1 page
    1 body line, max 6 chars per line
    1 total line, max 6 chars per line
Compute: -W 481.89pt = 6.693in = 170.0mm = 17.0cm
Correct: -W 507.252pt = 7.045in = 178.947mm = 17.895cm
Compute: char height 803.15pt = 11.155in = 283.333mm = 28.333cm
Correct: char height 854.414pt = 11.867in = 301.418mm = 30.142cm
Backup: '~/banner.txt' --> '~/banner-2022.05.02-09.47.52.txt'
Rewrite: yawp --> '~/banner.txt'
    0 header lines, max 0 chars per line, 1 page
    1 body line, max 6 chars per line
    1 total line, max 6 chars per line
Print: '~/banner.txt' --> stdout
BANNER
Export: '~/banner.txt' --> '~/banner.pdf'
```

Note that banner.txt is not altered because the -N argument, but the exported banner.pdf contains the string 'BANNER', a big character per page.

Line justification in text is controlled by -l argument:

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

After justification:

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

Now we can define some "best practices" to follow when 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. 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
#!/usr/bin/python3

''' Text in start switch line is not formatted.
This is a one-line unindented paragraph.

This is a multi-line unindented paragraph.
This is a multi-line unindented paragraph.
This is a multi-line unindented paragraph.

    This is a picture, it remains as is.
            This is a picture, it remains as is.
        This is a picture, it remains as is.

    • This  is  a  multi-line indented paragraph.
This is a multi-line indented paragraph.
    This is a multi-line indented paragraph.
        • This  is  another  multi  line  indented  paragraph.
    This is another  multi-line indented paragraph.
This is another multi-line  indented paragraph.
''' # Text in stop switch line is not formatted.

def double(x): # Python code is not formatted.
    ''' Text in start switch line is not formatted.
This is another multi-line unindented paragraph.
This is another multi-line unindented paragraph.
This is another multi-line unindented paragraph.
''' # Text in stop switch line is not formatted.
    return x + x # Python code is not formatted.
```

```
$ yawp -v -w 32 -p -P 0 pycode.py
Correct: '~/.yawp/yawp.cfg' not found, default corrections
Correct: -L 43.325pt = 0.602in = 15.284mm = 1.528cm
Correct: -R 40.57pt = 0.563in = 14.312mm = 1.431cm
Correct: -T 53.814pt = 0.747in = 18.985mm = 1.898cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
Read: yawp <-- '~/pycode.py'
    0 header lines, max 0 chars per line, 1 page
    28 body lines, max 62 chars per line
    28 total lines, max 62 chars per line
Compute: -W 15.059pt = 0.209in = 5.312mm = 0.531cm
Correct: -W 15.852pt = 0.22in = 5.592mm = 0.559cm
Compute: char height 25.098pt = 0.349in = 8.854mm = 0.885cm
Correct: char height 26.7pt = 0.371in = 9.419mm = 0.942cm
Backup: '~/pycode.py' --> '~/pycode-2022.05.02-09.47.55.py'
Rewrite: yawp --> '~/pycode.py'
    0 header lines, max 0 chars per line, 1 page
    40 body lines, max 51 chars per line
    40 total lines, max 51 chars per line
Print: '~/pycode.py' --> stdout
#!/usr/bin/python3

''' Text in start switch line is not formatted.
This  is  a  one-line unindented
paragraph.

This  is a multi-line unindented
paragraph.  This is a multi-line
unindented  paragraph. This is a
multi-line unindented paragraph.

    This is a picture,
            it remains
        as is.

    • This   is   a   multi-line
      indented  paragraph.  This
      is  a  multi-line indented
      paragraph.   This   is   a
      multi-line        indented
      paragraph.
        • This  is another multi
          line          indented
          paragraph.   This   is
          another     multi-line
          indented    paragraph.
          This     is    another
          multi-line    indented
          paragraph.
''' # Text in stop switch line is not formatted.

def double(x): # Python code is not formatted.
    ''' Text in start switch line is not formatted.
This   is   another   multi-line
unindented  paragraph.  This  is
another   multi-line  unindented
paragraph.   This   is   another
multi-line unindented paragraph.
''' # Text in stop switch line is not formatted.
    return x + x # Python code is not formatted.
```

Inserting page headers in a Python file doesn't make sense, so if the file is a Python file then -f and -F (see 6. Paging) are automatically turned off.

5. 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, it could be the cover of the document
- a "numbered chapter", started by a "numbered chapter 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

Numbered chapters, contents chapter and index chapter may appear in any order.

5.1. NUMBERED CHAPTERS

Start of a "numbered chapter" is recognized by a numbered chapter line. A line is a "numbered chapter 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 "int-dot couples", each made by
    - one or more decimal digits, between '0' and '9'
    - a "decimal point" '.'
  - a "blank" ' '
  - a chapter title

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

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

Numbered chapter lines must follow two quite obvious sequence rules:

- first numbered chapter line in file must be a level-1 numbered chapter line
- each other numbered chapter line can have a level between 1 and the level of the previous numbered 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' (see '6. Paging')

Length of shrunk title can't be longer than half of '-w, --chars-per-line' argument.

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. So you are free to create destroy or swap chapters as you like, they will be renumbered accordingly. Example:

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

32.33. BBB bbb

0.0. CCC ccc

0. DDD ddd

0.0. EEE eee

9.9.9. FFF fff

0.0. GGG ggg
```

```
$ yawp -v -w 32 -p -P 0 chapters.txt
Correct: '~/.yawp/yawp.cfg' not found, default corrections
Correct: -L 43.325pt = 0.602in = 15.284mm = 1.528cm
Correct: -R 40.57pt = 0.563in = 14.312mm = 1.431cm
Correct: -T 53.814pt = 0.747in = 18.985mm = 1.898cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
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 15.059pt = 0.209in = 5.312mm = 0.531cm
Correct: -W 15.852pt = 0.22in = 5.592mm = 0.559cm
Compute: char height 25.098pt = 0.349in = 8.854mm = 0.885cm
Correct: char height 26.7pt = 0.371in = 9.419mm = 0.942cm
Backup: '~/chapters.txt' --> '~/chapters-2022.05.02-09.47.55.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
```

5.2. 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:

- 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' (see '6. Paging')

Length of shrunk title can't be longer than half of '-w, --chars-per-line' argument.

If -f and -F are off, pages are not numbered. So, if the contents chapter is requested, it will still appear, but without page numbers.

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

    (any
    previous
    content
    of
    the
    contents
    chapter
    will
    be
    deleted)

0. AAA aaa

32.33. BBB bbb

0.0. CCC ccc

0. DDD ddd

0.0. EEE eee

9.9.9. FFF fff

0.0. GGG ggg
```

```
$ yawp -v -w 32 -p -P 0 contents.txt
Correct: '~/.yawp/yawp.cfg' not found, default corrections
Correct: -L 43.325pt = 0.602in = 15.284mm = 1.528cm
Correct: -R 40.57pt = 0.563in = 14.312mm = 1.431cm
Correct: -T 53.814pt = 0.747in = 18.985mm = 1.898cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
Read: yawp <-- '~/contents.txt'
    0 header lines, max 0 chars per line, 1 page
    26 body lines, max 14 chars per line
    26 total lines, max 14 chars per line
Compute: -W 15.059pt = 0.209in = 5.312mm = 0.531cm
Correct: -W 15.852pt = 0.22in = 5.592mm = 0.559cm
Compute: char height 25.098pt = 0.349in = 8.854mm = 0.885cm
Correct: char height 26.7pt = 0.371in = 9.419mm = 0.942cm
Backup: '~/contents.txt' --> '~/contents-2022.05.02-09.47.55.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
```

5.3. INDEX CHAPTER

For "index chapter" we mean an alphabetical list of subjects, possibly showing which pages a "subject" is on. A subject can appear in the file:

- as a "quoted subject" if it's preceded and followed by a "double quote" '"' (but double quotes preceded or followed by a "single quote" "'" or by another double quote '"' are not taken into account, in order to allow things like '"' in text)
- an "unquoted subject" otherwise

Unquoted subjects are recognized everywhere if the file contains at least one corresponding quoted subject somewhere. Hence in index chapter the line relating to a given subject will contain:

- quoted (by '"') page numbers for pages containing one or more quoted instances of the subject (and zero or more unquoted instances)
- unquoted page numbers for pages containing one or more unquoted instance of the subject and zero quoted instances

Subject length can't be greater than half of '-w, --chars-per-line' argument, this should intercept unpaired double quotes.

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:

- 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' (see '6. Paging')

Length of shrunk title can't be longer than half of '-w, --chars-per-line' argument.

For technical reasons, a subject cannot occupy more than one line in the index chapter. Hence if a subject is referenced in many pages and the resulting line is longer than '-w, --chars-per-line' argument, then it's truncated and terminated by '...' with a warning message.

If -f and -F are off, pages are not numbered. So, if the index chapter is requested, it will still appear, but without page numbers.

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.

6. PAGING

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

- "-f, --form-feed" : 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

If a figure is too long 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, see for example the configuration file in '7. Corrections'.

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

If -F is turned on (or both -f and -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 chapter 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 "form feed" '\f', containing data such as file name, chapter title, date, time or page number
- a dashed separation line of "macron" '¯' characters

The form feed character causes a page break but is a non-printable character, therefore it's not considered when measuring the length of lines. Depending on your editor, it can appear as a quarter musical note, or as a small empty rectangle.

So a line in the output can be considered as:

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

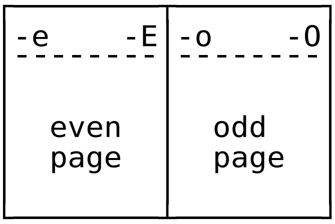
When the file is initially read, if -U and -N are turned off, all header lines are eliminated. So, if you want to eliminate page headers from your file, just run yawp without -f or -F.

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.

Inserting page headers in a Python file doesn't make sense, so if the file name ens with '.py', -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 "percent 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 percent variable is allowed.

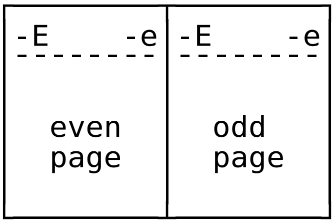
If for instance the file is '/home/xxxx/yyy/zzz.www.txt' and the user is 'xxxx' then we get:

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

7. GRAPHICS

Yawp also has some limited graphic features. You can sketch pictures with boxes arrows and tables by two "draw characters":

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

If you turn on the argument:

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

and -U is off 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.

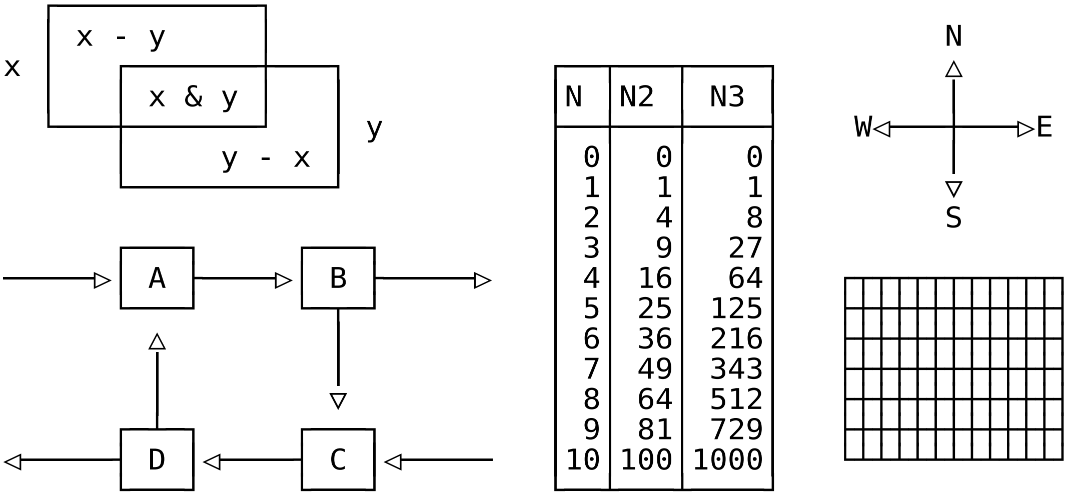
This feature works:

- in "standard mode" (-N and -U off): in picture lines only, but not in text lines
- in "no-format mode" (-N on): in all lines of the file
- in "undo mode" (-U on): nowhere

An example:

~~~
$ cat graphics.txt
      `````````````
      ` x - y     `                                     N
    x `   `````````````           `````````````         ^
      `   ` x & y `   `           `N `N2 ` N3 `         `
      `````````````   ` y         `````````````    W^```````^E
          `     y - x `           ` 0`  0`   0`         `
          `````````````           ` 1`  1`   1`         ^
                                  ` 2`  4`   8`         S
          `````     `````         ` 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 -P 0 graphics.txt
Correct: '~/.yawp/yawp.cfg' not found, default corrections
Correct: -L 43.325pt = 0.602in = 15.284mm = 1.528cm
Correct: -R 40.57pt = 0.563in = 14.312mm = 1.431cm
Correct: -T 53.814pt = 0.747in = 18.985mm = 1.898cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
Read: yawp <-- '~/graphics.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 63 chars per line
    17 total lines, max 63 chars per line
Compute: -w 63
Compute: -W 7.649pt = 0.106in = 2.698mm = 0.27cm
Correct: -W 8.052pt = 0.112in = 2.84mm = 0.284cm
Compute: char height 12.748pt = 0.177in = 4.497mm = 0.45cm
Correct: char height 13.562pt = 0.188in = 4.784mm = 0.478cm
Backup: '~/graphics.txt' --> '~/graphics-2022.05.02-09.47.55.txt'
Rewrite: yawp --> '~/graphics.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 63 chars per line
    17 total lines, max 63 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

Argument -g works with -N too, so you can redraw picture files without the need for a blank at beginning of the lines, and without yawp trying to format the file. We can say that when -N is turned on, the whole file is a single picture. As another example, we want to print a chessboard.

```
$ cat chessboard.txt
`````````````````
` `X` `X` `X` `X`
`````````````````
`X` `X` `X` `X` `
`````````````````
` `X` `X` `X` `X`
`````````````````
`X` `X` `X` `X` `
`````````````````
` `X` `X` `X` `X`
`````````````````
`X` `X` `X` `X` `
`````````````````
` `X` `X` `X` `X`
`````````````````
`X` `X` `X` `X` `
`````````````````
```
```

```
$ yawp -v -N -g -p -A 1 -L 1cm -T 5cm -B 2cm chessboard.txt
Correct: '~/.yawp/yawp.cfg' not found, default corrections
WARNING: -L 1cm < 2cm, you may get unexpected results
Correct: -L 13.737pt = 0.191in = 4.846mm = 0.485cm
WARNING: -R 1cm < 2cm, you may get unexpected results
Correct: -R 10.456pt = 0.145in = 3.689mm = 0.369cm
Correct: -T 144.161pt = 2.002in = 50.857mm = 5.086cm
Correct: -B 15.991pt = 0.222in = 5.641mm = 0.564cm
Read: yawp <-- '~/chessboard.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 17 chars per line
    17 total lines, max 17 chars per line
Compute: -w 17
Compute: -W 31.681pt = 0.44in = 11.176mm = 1.118cm
Correct: -W 33.349pt = 0.463in = 11.765mm = 1.176cm
Compute: char height 31.681pt = 0.44in = 11.176mm = 1.118cm
Correct: char height 33.704pt = 0.468in = 11.89mm = 1.189cm
Backup: '~/chessboard.txt' --> '~/chessboard-2022.05.02-09.47.55.txt'
Rewrite: yawp --> '~/chessboard.txt'
    0 header lines, max 0 chars per line, 1 page
    17 body lines, max 17 chars per line
    17 total lines, max 17 chars per line
Print: '~/chessboard.txt' --> stdout
┌─┬─┬─┬─┬─┬─┬─┬─┐
│ │X│ │X│ │X│ │X│
├─┼─┼─┼─┼─┼─┼─┼─┤
│X│ │X│ │X│ │X│ │
├─┼─┼─┼─┼─┼─┼─┼─┤
│ │X│ │X│ │X│ │X│
├─┼─┼─┼─┼─┼─┼─┼─┤
│X│ │X│ │X│ │X│ │
├─┼─┼─┼─┼─┼─┼─┼─┤
│ │X│ │X│ │X│ │X│
├─┼─┼─┼─┼─┼─┼─┼─┤
│X│ │X│ │X│ │X│ │
├─┼─┼─┼─┼─┼─┼─┼─┤
│ │X│ │X│ │X│ │X│
├─┼─┼─┼─┼─┼─┼─┼─┤
│X│ │X│ │X│ │X│ │
└─┴─┴─┴─┴─┴─┴─┴─┘
Export: '~/chessboard.txt' --> '~/chessboard.pdf'
```

Some remarks:

- '-N' prevents formatting the file as text, but the backup is still performed
- defaults '-w 0' and '-W 0' make the picture take up all available space between left and right margins
- '-A 1' transforms rectangles into squares in PDF file, so by printing it on an A4 sheet you get a square chessboard of about 20x20cm, 8x8in.

8. PDF EXPORTING

Normally the file is exported to a PDF file, which is then opened by calling the system default PDF browser (as atril or evince). If you don't want this to happen, set '-P 0'.

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

Each "percent variable" is evaluated as explained in '6. Paging', but '%n' '%N' and '%c' are not applicable and not allowed. '%P' and '%p' are both allowed but give the same result. The resulting file name must end with '.pdf'.

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

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

Value is an integer or float literal followed by a lowercase 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 1.0in
- -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 '3. Justification') 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": character aspect ratio = char width / char height ('1' = square grid, default: '3/5')

-A is a ratio, so it can be:

- an integer or float literal (as '1' or '0.6')
- two integer or float literals, separated by a "slash" '/' (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. These names are case-insensitive ('A4' and 'a4' are equivalent), while the 'x' and the suffix in the 'width x height' value must be lowercase.

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	1000x1414mm
B1	707x1000mm
B1+	720x1020mm
B2	500x707mm
B2+	520x720mm
B3	353x500mm
B4	250x353mm
B5	176x250mm
B6	125x176mm
B7	88x125mm
B8	62x88mm
B9	44x62mm
B10	31x44mm

Paper width and height are exchanged with each other by -Z:

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

Unprintable margins around the paper sheet are controlled by:

- "-L, --left-margin": left margin (pt/in/mm/cm, default: '2cm')
- "-R, --right-margin": right margin (pt/in/mm/cm, default: '-L')
- "-T, --top-margin": top margin (pt/in/mm/cm, default: '2cm')
- "-B, --bottom-margin": bottom margin (pt/in/mm/cm, 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.

BEWARE: margins of less than 2cm are allowed but not guaranteed, and may give unexpected results.

9. CORRECTIONS

For a quick start, you can skip this chapter.

Export of the PDF file is performed by CUPS 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" it's the same. Namely:

- page margins are enlarged, so they need to be reduced
- character width and height are reduced, so they need to be enlarged
- in portrait and landscape orientation the errors are different, and so must be the corrections

A hardwired mechanism in yawp tries to correct such errors. If -v is on, such corrections are signaled by information messages starting with 'Correct:'. This device has been tuned for the default paper size 'A4' = '210x297mm', both portrait and landscape, on a Hewlett-Packard Officejet 2620 printer. With another format or another printer you may get different results, and so you may need different correctors.

Corrections are controlled by -C argument:

- "-C, --calibration": don't correct character size and page margins

and by the configuration file '~/.yawp/yawp.cfg', if exists:

- if -C argument is turned on, no correction takes place, and we can make experiments in order to define content of '~/.yawp/yawp.cfg'
- otherwise if '~/.yawp/yawp.cfg' doesn't exist, the default corrections are performed
- otherwise corrector coefficients are read from '~/.yawp/yawp.cfg'

An example for '~/.yawp/yawp.cfg' follows, corresponding to the default corrections:

```
#----- yawp configuration file -----

plm 10mm 16mm # portrait left margin
plm 20mm 24mm
plm 30mm 35mm
plm 40mm 43mm
plm 50mm 52mm
plm 60mm 62mm
plm 70mm 72.5mm
plm 80mm 83mm
plm 90mm 92mm
plm 100mm 101mm

llm 10mm 20.5mm # landscape left margin
llm 20mm 29.5mm
llm 30mm 39mm
llm 40mm 48mm
llm 50mm 57mm
llm 60mm 66.5mm
llm 70mm 75.5mm
llm 80mm 84.5mm
llm 90mm 94mm
llm 100mm 104mm

prm 10mm 15mm # portrait right margin
prm 20mm 25.5mm
prm 30mm 34.5mm
prm 40mm 44.5mm
prm 50mm 54.5mm
prm 60mm 64.5mm
prm 70mm 72mm
prm 80mm 81mm
prm 90mm 91.5mm
prm 100mm 100mm

lrm 10mm 22mm # landscape right margin
lrm 20mm 30mm
lrm 30mm 40mm
lrm 40mm 49.5mm
lrm 50mm 58mm
lrm 60mm 68mm
lrm 70mm 77.5mm
lrm 80mm 86mm
lrm 90mm 96mm
lrm 100mm 104mm

ptm 10mm 11.5mm # portrait top margin
ptm 20mm 21mm
ptm 30mm 30.5mm
ptm 40mm 39.5mm
ptm 50mm 49mm
ptm 60mm 59mm
ptm 70mm 68mm
ptm 80mm 77.5mm
ptm 90mm 87mm
ptm 100mm 96mm

ltm 10mm 11mm # landscape top margin
ltm 20mm 20.5mm
ltm 30mm 30mm
ltm 40mm 39mm
ltm 50mm 48.5mm
ltm 60mm 57.5mm
ltm 70mm 67mm
ltm 80mm 76mm
ltm 90mm 85mm
ltm 100mm 95mm

pbm 10mm 24mm # portrait bottom margin
pbm 20mm 34mm
pbm 30mm 43mm
pbm 40mm 52.5mm
pbm 50mm 62mm
pbm 60mm 71mm
pbm 70mm 81mm
pbm 80mm 90mm
ppm 90mm 100mm
pbm 100mm 109.5mm

lbm 10mm 24mm # landscape bottom margin
lbm 20mm 32mm
lbm 30mm 42mm
lbm 40mm 52mm
lbm 50mm 60mm
lbm 60mm 70mm
lbm 70mm 80mm
lbm 80mm 88mm
lbm 90mm 99mm
lbm 100mm 107mm

pcw 100mm 94.674mm # portrait character width

lcw 100mm 92.200mm # landscape character width

pch 100mm 94.358mm # portrait character height

lch 100mm 92.647mm # landscape character height
```

As you see, standard Unix '#'-comments are accepted. The file contains a set of correctors. Each "corrector" has the form 'k y x', where:

- k is a three-letters "key" among 12 allowed values:
  - 'plm' portrait left margin
  - 'prm' portrait right margin
  - 'ptm' portrait top margin
  - 'pbm' portrait bottom margin
  - 'pcw' portrait character width
  - 'pch' portrait character height
  - 'llm' landscape left margin
  - 'lrm' landscape right margin
  - 'ltm' landscape top margin
  - 'lbm' landscape bottom margin
  - 'lcw' landscape character width
  - 'lch' landscape character height
- y is a "wanted value" in points/inches/millimeters/centimeters
- x is an "obtained value" in points/inches/millimeters/centimeters

For instance, take the first corrector, 'plm 10mm 16mm'. This means that, by trying to print a page with -Z off, -C on, and -L 10mm (y = 10mm), we have empirically obtained an actual measured left margin of 16mm on paper (x = 16mm). Therefore if we wish an actual measured left margin of 16mm on paper (x = 16mm), we must give to lp command an argument '-o page-left' equivalent to 10mm (y = 10mm). So, given a wanted measure x and a set of corresponding correctors, which is the function y = f(x) telling us what argument y provide to lp?

For each key we can have zero, one, two or more correctors, namely:

- zero correctors: no correction, y = f(x) = x, the straight line passing by (0, 0) and (1, 1)
- one corrector, say (y0, x0): y = f(x) = (y0 / x0) * x, the straight line passing by (0, 0) and (x0, y0)
- two correctors, say [(y0, x0), (y1, x1)], y = f(x) = y0 + (x - x0) * (y1 - y0) / (x1 - x0), the straight line passing by (x0, y0) and (x1, y1)
- three or more correctors, say [(y0, x0), (y1, x1), (y2, x2), ...], y = f(x) is approximated by the least-squares straight line defined by (x0, y0), (x1, y1), (x2, y2), ...

Finally, if the result is less than zero, it's forced to zero.

10. AFTERWORDS

10.1. ACRONYMS

ACRONYM	MEANING
CUPS	Common Unix Printing System
IDE	Integrated Development Environment
IDLE	Integrated Development and Learning Environment
PDF	Page Description Format
PyPI	Python Package Index
WYSIWYG	What You See Is What You Get
YAWP	Yet Another Word Processor

10.2. CHARACTERS

CHARACTER NAME	CHAR	HEX	DEC
back quote	'`'	0060	96
black small circle	'•'	2022	8226
blank	' '	0020	32
box drawings light down and horizontal	'┬'	252c	9516
box drawings light down and left	'┐'	2510	9488
box drawings light down and right	'┌'	250c	9484
box drawings light horizontal	'─'	2500	9472
box drawings light up and horizontal	'┴'	2534	9524
box drawings light up and left	'┘'	2518	9496
box drawings light up and right	'└'	2514	9492
box drawings light vertical	'│'	2502	9474
box drawings light vertical and horizontal	'┼'	253c	9532
box drawings light vertical and left	'┤'	2524	9508
box drawings light vertical and right	'├'	251c	9500
carriage return	'\r'	000d	13
circumflex accent	'^'	005e	94
decimal point	'.'	002e	46
digit nine	'9'	0039	57
digit zero	'0'	0030	48
double quote	'"'	0022	34
form feed	'\f'	000c	12
horizontal tab	'\t'	0009	9
latin small letter x	'x'	0078	120
line feed	'\n'	000a	10
macron	'¯'	00af	175
percent sign	'%'	0025	37
single quote	"'"	0027	39
slash	'/'	002f	47
white down-pointing triangle	'▽'	25bd	9661
white left-pointing triangle	'◁'	25c1	9665
white right-pointing triangle	'▷'	25b7	9655
white up-pointing triangle	'△'	25b3	9651

10.3. CREDITS

A somewhat analogous (but very different) program is the Unix command "fmt". We are not aware of any other programs of this kind. For details, type:

```
$ man fmt
```

Yawp has been developed under Xubuntu 21.10:

http://www.xubuntu.org

by Python 3.9.7 and IDLE, the Python’s IDE:

https://www.python.org

and published on Pypi by flit 3.7.1:

https://pypi.org/project/flit

10.4. HISTORY

- version 0.6.1
  - '-k, --calibration', renamed '-C, --calibration'
  - '-m, --max-subject', suppressed
  - max subject lenght and max length of chapter titles are -w / 2
  - corrections, made parametric by configuration file yawp.cfg
  - '-Q, --print-quality', suppressed, print quality is always max
  - in index chapter:
    - page numbers for quoted subjects, quoted
    - unquoted page numbers for unquoted subjects, added
  - Development Status :: 4 - Beta

- version 0.5.4
  - just another fix to the yawp manual
  - Development Status :: 4 - Beta

- version 0.5.3
  - just a fix to the installation instructions in yawp manual
  - Development Status :: 4 - Beta

- version 0.5.2
  - correction of page margins, new algorithm
  - page margins, no more bounded by 2cm..8cm interval (but below 2cm you may get unexpected results)
  - '-g, --graphics', now works with -N too (but not with -U)
  - information messages about corrected values, added
  - index subjects, taken not only from text but from pictures too
  - '-b, --dump-buffer', useful for debug only, suppressed
  - yawp manual, updated
  - Development Status :: 4 - Beta

- version 0.5.1
  - completely redefined and rewritten
  - Development Status :: 4 - Beta

- version 0.4.2
  - obsolete and deprecated

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

10.5. LICENSE

11. SUMMARY OF ARGUMENTS

General arguments:

- -h, --help: show a short help message and exit
- -H, --manual: open this PDF yawp manual and exit
- -V, --version: show program's version number and exit
- -v, --verbose: display information messages on stderr
- -s, --echo-shell: display invoked Unix commands on stderr

Usage arguments:

- -N, --no-format: don't format the file
- -U, --undo: restore the file from its previous version
- -g, --graphics: redraw '`'-segments and '^'-arrowheads
- -p, --print-file: at end print file on stdout

Formatting arguments:

- -w, --chars-per-line: line width in characters per line (default: '0' = automatic)
- -l, --left-only: justify text lines at left only (default: at left and right)
- -c, --contents-title: title of contents chapter (default: 'contents')
- -i, --index-title: title of index chapter (default: 'index')

Paging arguments:

- -f, --form-feed: insert a page header on full page
- -F, --form-feed-chap: 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-pages-E-e: all page headers contain -E at left and -e at right

PDF arguments:

- -P, --file-pdf: at end export file to this PDF file and open it ('0' = no export, default: '%P/%f.pdf')
- -W, --char-width: character width (pt/in/mm/cm, default: '0' = automatic)
- -A, --char-aspect: character 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)
- -L, --left-margin: left margin (pt/in/mm/cm, default: '2cm')
- -R, --right-margin: right margin (pt/in/mm/cm, default: '-L')
- -T, --top-margin: top margin (pt/in/mm/cm, default: '2cm')
- -B, --bottom-margin: bottom margin (pt/in/mm/cm, default: '-T')
- -C, --calibration: don't correct character size and page margins

Positional argument:

- file: text file to be processed

INDEX