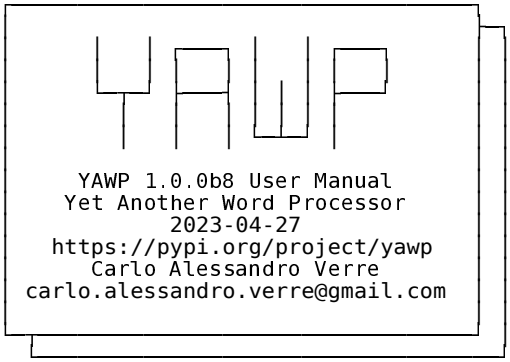


YAWP 1.0.0b8 User Manual
Yet Another Word Processor
2023-04-27
https://pypi.org/project/yawp
Carlo Alessandro Verre
carlo.alessandro.verre@gmail.com

I sound my barbaric yawp over the roofs of the world

Walt Whitman

CONTENTS

1. FOREWORDS

1.1. INTRODUCTION

"YAWP" here means Yet Another Word Processor, and YAWP is a pure-Python Linux-only word processor for plain text files, with PDF export. 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 draft or a simple quick-and-dirty no-frills document as this one, give YAWP a try.

YAWP's main features are:

- YAWP has a GUI interface, but can be used as CLI command too
- YAWP processes in place a single plain text file, hereinafter referred to simply as the "text file"
- YAWP before rewriting the text file creates a timestamped backup, allowing Undo operation
- YAWP justifies text at left and right in:
  - unindented paragraphs
  - dot-marked indented paragraphs (as this one)
- YAWP justification is driven by the text in the text file and by arguments only, not by commands or tags embedded in text
- YAWP accepts unjustified pictures (as schemas, tables and code examples) freely intermixed with text
- YAWP performs automatic multi-level renumbering of chapters and inserts an automatic Contents chapter in the text file
- YAWP recognizes relevant subjects (quoted by '"') and inserts an automatic Index chapter in the text file
- YAWP performs automatic multi-level renumbering of figure captions and inserts an automatic Figures chapter in the text file
- YAWP cuts the text file in pages, by inserting two-lines page headers, allowing page numbering control and insertion of initial Roman-Numbered pages
- YAWP also has some graphic features, you can sketch pictures with horizontal and vertical segments (by '`') and arrowheads (by '^'), YAWP redraws them by suitable Unicode graphic characters
- YAWP exports the text file in PDF format, with control over character size and page layout, and lets you browse the generated PDF file, allowing preview and printing
- YAWP in GUI mode saves the last processed text file and restores it at next invocation
- YAWP keeps a distinct argument set for each text file
- YAWP in GUI mode saves a list of the 20 most recent processed text files and allows to select one at next invocation
- YAWP tries to correct errors made by CUPS-PDF about font size and page margins, you can use default corrections or redefine them by a correction file
- YAWP writes messages about processing on terminal and into a timestamped session log file
- YAWP locks text files to be processed in order to avoid interference by other concurrent YAWP executions
- yawp is stable, if after a yawp execution you run yawp again on the same file with the same arguments, the text file content does not change
- YAWP has been kept as simple as possible, about 3000 lines of Python code plus 3000 lines of this YAWP User Manual

As an example of CLI usage, this YAWP User Manual has been created as yawp.pdf from yawp.txt by typing:

```
$ yawp -M f -v -w 90 -p c -n -5 -E 'yawp 1.0.0b7 User Manual' -X -L 2.5cm yawp.txt
```

where arguments mean:

- -M f: run YAWP in CLI Format mode
- -v: write information messages not only into log file but also on terminal
- -w 90: set 90 characters per line
- -p c: insert a page header on page full, on picture break and before level-1 chapters
- -n -5: first five pages are Numbered by Roman numbers
- -E 'yawp 1.0.0b7 User Manual': set the title to the right of even page headers
- -X: export and view in PDF format
- -L 2.5cm: set left margins on odd pages and right margins on even pages to 2.5 cm
- yawp.txt: this is the text file to be processed

Other examples are scattered below.

1.2. INSTALLATION

In the following we assume that your Linux belongs to the Debian family. Type at terminal:

```
$ sudo apt-get update
$ sudo apt-get purge printer-driver-cups-pdf
$ sudo apt-get install printer-driver-cups-pdf IDLE python3-pip
$ pip3 install yawp
$ echo 'PATH=$PATH:~/.local/bin' >> ~/.bashrc
```

Imagine your user name is 'xxxx'. If you see a message like this:

```
WARNING: The script YAWP is installed in '/home/xxxx/.local/bin' which is not on PATH
```

you can fix the problem by typing:

```
$ echo 'PATH=$PATH:/home/xxxx/.local/bin' >> /home/xxxx/.bashrc
```

The 'sudo apt-get purge printer-driver-cups-pdf' command is recommended because on some Linux systems (as MX) printer-driver-cups-pdf is pre-installed but it doesn't seem to work out of the box, while it works if it is uninstalled and reinstalled.

Now you can close the terminal, open another one, and call YAWP, syntax is:

```
$ yawp -H # show a help message and exit
$ yawp -V # show program's version number and exit
$ yawp -H [-Y pdf_browser] # browse the PDF YAWP User Manual and exit
$ yawp text_file # run YAWP in GUI mode, explicit text file, no arguments
$ yawp # run YAWP in GUI mode, text file from previous session, no arguments
$ yawp -M e [-y text_editor] text_file  # run YAWP in CLI Edit mode
$ yawp -M f [...arguments...] text_file # run YAWP in CLI Format mode
$ yawp -M n [...arguments...] text_file # run YAWP in CLI Noformat mode
$ yawp -M u [...arguments...] text_file # run YAWP in CLI Undo mode
```

Later you can type:

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

in order to upgrade YAWP to a later version.

1.3. GLOSSARY

Some terms here have a different meaning from that commonly in use.

To "shrink" a character string means:

- to strip away all leading and trailing blanks, and
- to reduce each internal group of consecutive blanks to a single blank

For instance, the string:

```
'   aAa   bBb   "cCc"  '
```

if shrunk, becomes:

```
'aAa bBb "cCc"'
```

To "uppercase" a character string means to uppercase all characters in it, except characters quoted by double quote '"', which remain unchanged.

For instance, the string:

```
'aAa bBb "cCc"'
```

if uppercased, becomes:

```
'AAA BBB "cCc"'
```

To "titlecase" a character string means to uppercase the first character of each word in it, and to lowercase all the others, except the characters quoted by double quote '"', which remain unchanged.

For instance, the string:

```
'aAa bBb "cCc"'
```

if titlecased, becomes:

```
'Aaa Bbb "cCc"'
```

Two strings are said to be "equivalent" if, shrunk and uppercased, they become equal. For instance the strings:

```
x = '   aAa   bBb   "cCc"  '
y = '         aaa   BBB   "cCc"'
```

are equivalent because:

```
uppercase(shrink(x)) == 'AAA BBB "cCc"' == uppercase(shrink(y))
```

Other terms will be explained when they will be used.

2. USAGE MODES

General behaviour is controlled by the following arguments:

- "-h, --help": show a help message and exit
- "-H, --view-manual": view the YAWP-generated PDF YAWP User Manual and exit
- "-V, --version": show program's version number and exit
- "-v, --verbose": write all messages on stderr (default: write errors only)
- "-M, --usage-mode": run YAWP in this usage mode (default: 'g' = GUI, 'e' = CLI Edit, 'f' = CLI Format, 'n' = CLI Noformat, 'U' = CLI Undo) The -y argument defines the text editor to be used:

The PDF browser used by -H is defined by -Y (see 7.).

Usage modes defined by -M are five:

- one GUI mode, the default one (see 2.2.)
- four CLI modes:
  - Edit mode (-M e, see 2.1.1.)
  - Format mode (-M f, see 2.1.2.)
  - Noformat mode (-M n, see 2.1.3.)
  - Undo mode (-M u, see 2.1.4.)

In all usage modes, after processing the text file can be exported and viewed in PDF format by the -X argument (see 7.).

Last is the positional argument text_file, which indicates the text file to be processed. A leading path (absolute or relative to current directory) is optional. A '.txt' extension is recommended but not mandatory. The file can contain ASCII characters, or any other Unicode UTF8-encoded character.

A text file is a sequence of lines, separated by line terminators. The text file is read 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 text file is rewritten, the line terminator is always '\n'.

YAWP uses a text editor and a PDF browser, you can choose your preferred ones. To define the text editor, set the argument:

- "-y, --text-editor": editor for text files (default: 'idle')

Such an editor is used:

- to edit the text file by the Edit button
- to edit the correction file by Correct button
- to browse the session log file by Log button

If you use IDLE, the default text editor, you can view the line numbers at left of the text by clicking:

Options ⟶ Configure IDLE ⟶ Shell/Ed ⟶ Show line numbers in new windows

To define the PDF browser, set the argument:

- "-Y, --pdf-browser": browser for PDF files (default: 'xdg-open')

Such a PDF browser is used:

- to browse the PDF file exported by Format Noformat and Undo buttons
- to browse this YAWP User Manual by Help button

The default value 'xdg-open' causes you view the PDF file by the system default PDF browser. For further details, type:

```
$ man xdg-open
```

2.1. "GUI" MODE

YAWP running in GUI mode is a frontend between user and YAWP's CLI modes. In GUI mode the text_file argument is optional and the other arguments are not allowed.

```
$ yawp example.txt # start YAWP in GUI mode with explicit text_file argument
```

If text_file is not given, path and name of the text file to be processed are read from a reserved file, written on finish of last previous YAWP's execution.

```
$ yawp # start YAWP in GUI mode with text_file inherited from the previous session
```

Then the other arguments are read from another reserved file, associated with the text file (if such a file is not found, arguments silently get their default values).

For further details about reserved files, see chapter 8.

Then the Main window appears.

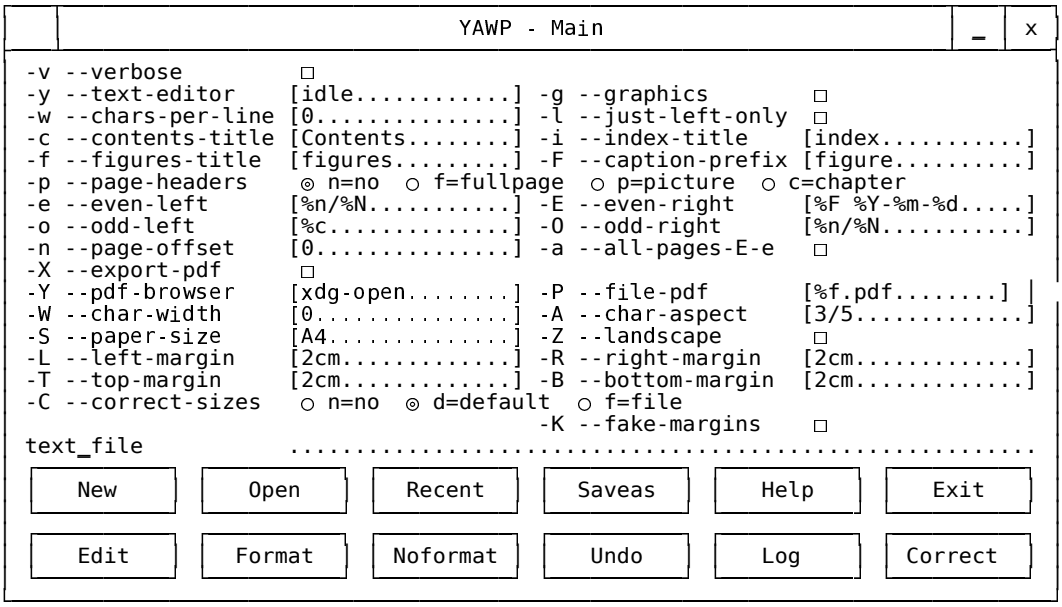


Figure 2.1.a. Main Window

The Main window contains:

- 16 lines, containing a subset of CLI arguments, of three types:
  - "boolean arguments" (-v -g -l -a -X and -Z), all turned off by default
  - "radio button arguments" (-p and -C)
  - "field arguments" (all the others), they can never be null, if they are set to null string by the user, they are automatically reset to their default value
- 1 line, containing the current text_file argument as a read-only field, it can be changed by New Open Recent and Saveas buttons
- 2 lines, containing 12 "action buttons", each corresponding to an action:
  - 4 buttons dedicated to the selection of the text file:
    - New button: create a new empty text file
    - Open button: browse the file system to select the text file
    - Recent button: browse the list of recent files to select the text file
    - Saveas button: clone current text file into a new target text file
  - Help button: browse the YAWP Manual through the PDF browser defined by -Y
  - Exit button: save current text file with its arguments and finish
  - 4 buttons dedicated to the processing of the text file:
    - Edit button: edit the text file through the text editor defined by -y
    - Format button: process the text file by formatting it
    - Noformat button: process the text file without formatting it
    - Undo button: restore the text file to its previous content
  - Log button: browse the log file through the text editor defined by -y
  - Correct button: manage the correction file (Edit Reset or Undo)

By touching a field or a button with the mouse cursor, the associated help tooltip will appear.

You can also use YAWP in GUI mode by keyboard only, without mouse:

- move window's focus, forward by Tab key, backward by Shift+Tab key
- when focus is on a boolean argument, press Space bar to toggle it on and off
- when focus is on a radio button, press Space bar to turn it on and turn all its siblings off
- when focus is on an action button, press Space bar to trigger the associated action

In case of error an error window will appear, click the 'OK' button to close and go back.

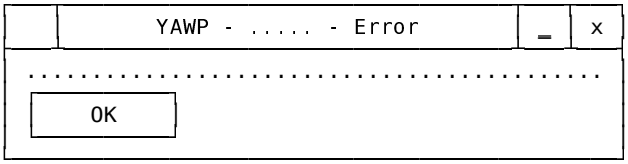


Figure 2.1.b. Error Window

2.1.1. TEXT FILE SELECTION

2.1.1.1. NEW BUTTON

- create a new empty text file, with default argument set
  - if new text file already exists, ask for confirmation
- the new text file becomes the current text file

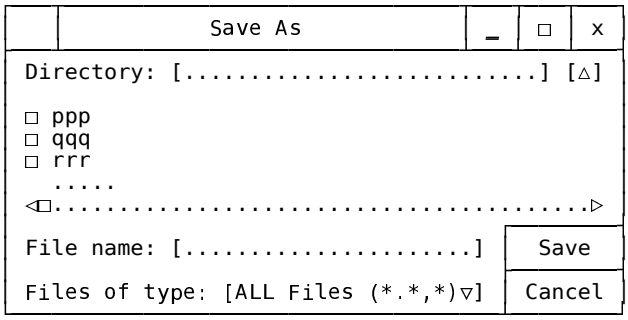


Figure 2.1.1.1.a. New Button, New File Definition Window

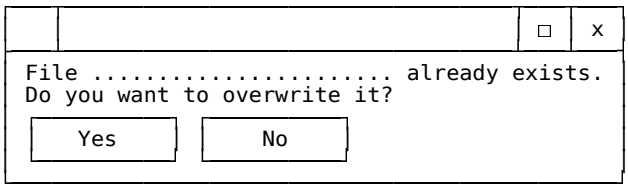


Figure 2.1.1.1.b. New Button, Overwriting Confirmation Window

2.1.1.2. OPEN BUTTON

- browse the file system, starting from the current directory
- click the Open button (of the below window) to select the text file to be processed
  - if not found, open an error window
- click the "Cancel" button to exit with no selection

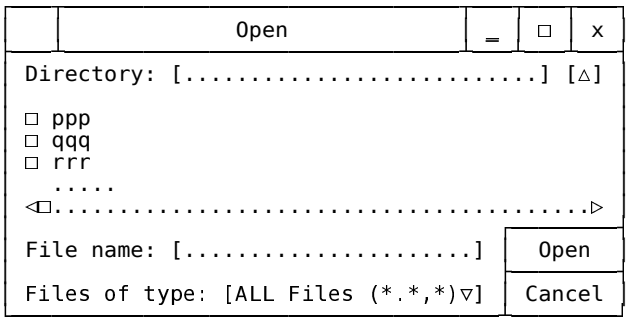


Figure 2.1.1.2.a. Open Button, File Selection Window

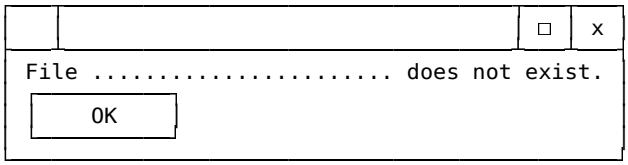


Figure 2.1.1.2.b. Open Button, File Not Found Error Window

2.1.1.3. RECENT BUTTON

- view the list of most recently processed text files (max 20, the most recent one on top)
- click a file button to select the text file to be processed
- click the "Clear" button to clear (after confirmation) the list of recent files
- click the "Cancel" button to exit with no selection

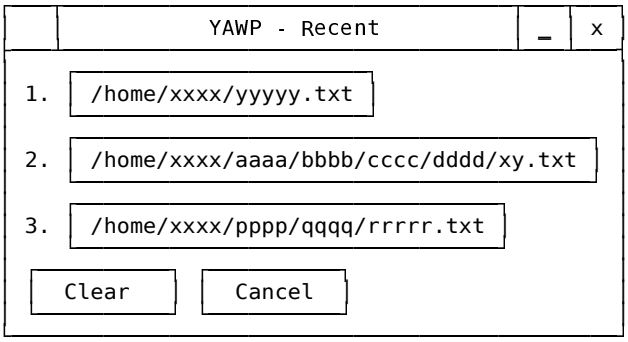


Figure 2.1.1.3.a. Recent Button, File Selection Window

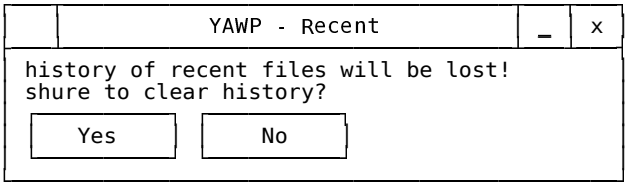


Figure 2.1.1.3.b. Recent Button, Clear Confirmation Window

For further details about backup files, see 8.8.

2.1.1.4. SAVEAS BUTTON

- browse the file system, starting from the current directory
- save current text file into target text file, together with its argument set
  - if target text file already exists, ask for confirmation
- target text file becomes the current text file

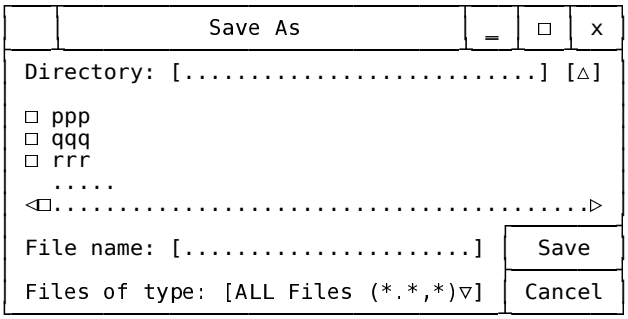


Figure 2.1.1.4.a. Saveas Button, Target File Definition Window

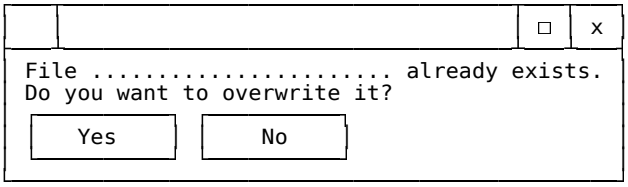


Figure 2.1.1.4.b. Saveas Button, Overwriting Confirmation Window

2.1.2. CORRECT BUTTON

- click the Edit button to edit the correction file, by the text editor defined by -y
- click the Reset button to reset the correction file to default values
- click the Undo button to restore the correction file to its previous version
- click the Cancel button to exit with no action

For further details see 8.3.

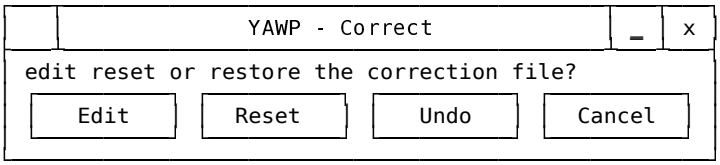


Figure 2.1.2.a. Correct Button, Action Choice Window

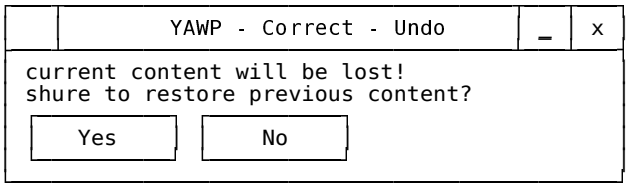


Figure 2.1.2.b. Correct Button, Restore Confirmation Window

2.1.3. HELP BUTTON

- after confirmation, browse this YAWP-generated YAWP User Manual through the PDF browser defined by -Y (like 'yawp -H')

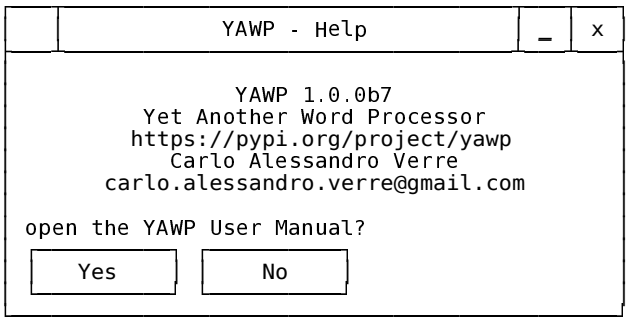


Figure 2.1.3.a. Help Button, Manual Confirmation Window

2.1.4. EXIT BUTTON

- save argument file and session file
- close main window
- close log file
- finish YAWP execution

WARNING: in order to terminate YAWP do not click by mouse the 'x' button in main window's top right corner or press Alt-F4 on keyboard, some argument may lose its value, use always the 'Exit' button instead.

2.1.5. TEXT FILE PROCESSING

2.1.5.1. EDIT BUTTON

- edit the text file through the text editor defined by -y
- if the text file has been altered, backup its original content into a timestamped copy

The text file to be edited must exist. This behavior is different from that of CLI Edit mode, where if the text file does not exist it is automatically created as an empty file.

2.1.5.2. FORMAT BUTTON

- read the text file, eliminating:
  - "horizontal tab" '\t' in lines, each replaced by four blanks
  - trailing blanks in lines
- format the text file content:
  - remove page headers and content of automatic chapters
  - justify the text (see 3.)
  - perform automatic multi-level renumbering of chapters and refill (if needed) the automatic Contents chapter in the text file (see 4.3.1.)
  - recognize relevant subjects and refill (if needed) the automatic Index chapter in the text file (see 4.3.2.)
  - perform automatic multi-level renumbering of figure captions and refill (if needed) the automatic Figures chapter in the text file (see 4.3.3.)
  - if -p ≠ 'n', cut the text file in pages, by inserting two-lines page headers, allowing page numbering control and insertion of initial Roman-Numbered pages (see 5.)
- if -g is on, redraw horizontal and vertical segments and arrowheads by suitable Unicode graphic characters (see 6.)
- if text has been altered:
  - backup the text file into a timestamped copy
  - rewrite the text file
- if -X is on:
  - export the text file into the PDF file (see 7.)
  - browse the PDF file by the PDF browser defined by -Y

The text file to be processed must exist.

2.1.5.3. NOFORMAT BUTTON

- read the text file, eliminating:
  - "horizontal tab" '\t' in lines, each replaced by four blanks
  - trailing blanks in lines
- do not format the text file content
- if -g is on, redraw horizontal and vertical segments and arrowheads by suitable Unicode graphic characters (see 6.)

- if text has been altered:
  - backup the text file into a timestamped copy
  - rewrite the text file
- if -X is on:
  - export the text file into the PDF file
  - browse the PDF file by the PDF browser defined by -Y

The text file to be processed must exist.

2.1.5.4. UNDO BUTTON

- restore the text file to its previous version
- if -X is on:
  - export the text file into the PDF file
  - browse the PDF file by the PDF browser defined by -Y

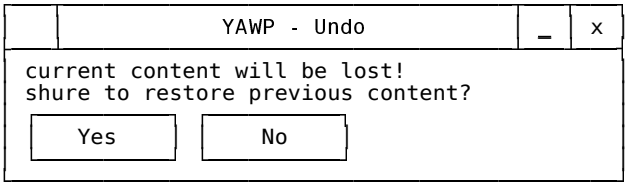


Figure 2.1.5.4.a. Undo Button, Restore Confirmation Window

WARNING: Undo operation cannot be undone, the text file goes back to the penultimate version and the last one is lost forever.

The text file to be restored may or may not exist, while of course a suitable backup file must exist.

2.1.6. LOG BUTTON

- browse the current session log file (see 8.4.) through the text editor defined by -y

2.2. "CLI" MODES

In CLI modes the text_file argument is mandatory and the other arguments are optional. Arguments not set in command line get their default values.

The CLI modes correspond one-to-one to the four GUI buttons dedicated to the processing of the text file.

2.2.1. EDIT MODE

```
$ yawp -M e [...arguments...] text_file
```

Processing is like that of the Edit button in GUI mode (see 2.1.5.1.) with an exception: if the text file does not exists, it is silently created as an empty file, while in GUI mode this is an error.

2.2.2. FORMAT MODE

```
$ yawp -M f [...arguments...] text_file
```

Processing is like that of the Format button in GUI mode (see 2.1.5.2.).

2.2.3. NOFORMAT MODE

```
$ yawp -M n [...arguments...] text_file
```

Processing is like that of the Noformat button in GUI mode (see 2.1.5.3.).

2.2.4. UNDO MODE

```
$ yawp -M u [...arguments...] text_file
```

Processing is like that of the Undo button in GUI mode (see 2.1.5.4.).

3. JUSTIFICATION

3.1. ALGORITHM

As said before, in Format mode each page "header line" inserted by YAWP (see 5.) is removed from the input file. Each remaining "body line" belongs to one of these four types:

- 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 it's not empty and starts with:
  - zero or more blanks ' '
  - a "dot character", which can be:
    - a "black small circle" '•', or
    - a "decimal point" '.' (always replaced by '•' on output)
  - a blank ' '
- otherwise a line is an "indented line" if it's not empty and starts with a blank
- otherwise a line is an "unindented line" if starts with a non-blank character

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

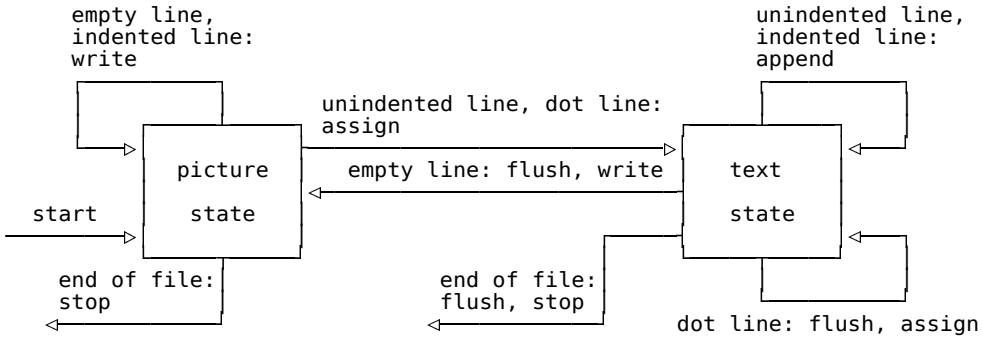


Figure 3.1.a. Formatting State Diagram

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
- at end of input file: processing is ended

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
- at end of input file: paragraph buffer is flushed, processing is ended

Max length of text lines can be controlled by:

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

Both -w and -W (character width) 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 body line length in file (page headers are not considered)
  - -W is computed from -w

Indentation of indented paragraphs cannot be greater than half of -w argument, whether set by user or automatically computed.

As an example of -W computed from -w, we impose one character per line only:

```
$ echo BANNER >banner.txt

$ yawp -M n -w 1 -X banner.txt

$ cat banner.txt
BANNER
```

Figure 3.1.b. Example With "-M n -w 1"

Now banner.txt is not altered because the '-M n', but the exported file 'banner.pdf' contains the string 'BANNER', one huge character per page.

Line justification in text is controlled by -l argument:

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

After formatting:

- 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

3.2. BEST PRACTICES

Now we can define some "best practices" to follow when editing the text 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 type
- initiated by a dot line (better if starting with 4-8-12-... blanks)
- 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

Numbered intented paragraphs are not implemented, but they can be created by hand:

- 1. first...
- 2. second...
- 3. third...

4. CHAPTERS

The text file can be partitioned in chapters by chapter lines. There are five types of chapter:

- the Nameless chapter, the first one, from first line until first chapter line
- the Numbered chapters, as this one
- the automatic chapters, automatically emptied and refilled:
  - the Contents chapter, the list of chapters
  - the Index chapter, the list of subjects
  - the Figures chapter, the list of figure captions

CHAPTER	AUTOMATIC	HOW MANY	WHERE	CHAPTER LINE	IN Contents	EJECT IF -p c	SEE
Nameless	no	max one	file top	no	no	no	4.1.
Numbered	no	no limit	everywhere	yes	yes	level-1 only	4.2.
Contents	yes	max one	everywhere	yes	no	yes	4.3.1.
Index	yes	max one	everywhere	yes	yes	yes	4.3.2.
Figures	yes	max one	everywhere	yes	yes	yes	4.3.3.

Figure 4.a. Chapter Types

All chapters, except the Numbered ones, can appear no more than once in the file.

All chapters, except the Nameless one, may appear everywhere and in any order, and are started by a chapter line. Chapter lines share some properties, each chapter line:

- is the first line in text file or is preceded by an empty line
- is the last line in text file or is followed by an empty line
- is a not empty line and starts with a non-blank character

Therefore we can say that a chapter line must be a one-line unindented text paragraph.

All chapters, except the Nameless one and the Contents one itself, are listed in the Contents chapter.

All chapters, except the Nameless one and the not-level-1 Numbered ones, trigger a page break if argument -p is 'c' (see 5.).

4.1. NAMELESS CHAPTER

As said before, the "Nameless chapter" is the first of the text file, and goes from the first line (included) until the first chapter line (excluded). It's not listed in the Contents chapter. As in this User Manual, it can be the cover of the document.

4.2. NUMBERED CHAPTERS

Start of a "Numbered chapter" is marked by a Numbered chapter line. A chapter line is a "Numbered chapter line" if contains one or more blank-separated words, of which:

- the first one is a "chapter label", made by one or more "int-dot couples", each made by:
  - one or more decimal digits, between '0' and '9'
  - a "decimal point" '.'
- the following words, if any, are the chapter title

The "level" of a Numbered chapter line is the count of int-dot couples in its label, 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 a couple of quite obvious sequence rules:

- first Numbered chapter in the text file and any Numbered chapter next to an automatic chapter must be a level-1 chapter
- any Numbered chapter next to another one can have any level between 1 and the level of the previous one plus one, but no more

Numbered chapter titles are:

- shrunk and uppercased where they are
- shrunk and titlecased when inserted:
  - into Contents chapter
  - into page headers by '%c' (see 5.)

Numbers in input do not 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 it, they will be automatically reNumbered. Example:

```
$ cat chapters.txt
0. aAa aAa aAa

32.33. bBb bBb 'bBb'

0.0. cCc "cCc" cCc

0. 'dDd dDd' dDd

0.0. eEe 'eEe' "eEe"

9.9.9. 'fFf fFf fFf'

0.0. "gGg gGg gGg"
```

```
$ yawp -M f -w 32 chapters.txt
```

```
$ cat chapters.txt
1. AAA AAA AAA

1.1. BBB BBB 'BBB'

1.2. CCC "cCc" CCC

2. 'DDD DDD' DDD

2.1. EEE 'EEE' "eEe"

2.1.1. 'FFF FFF FFF'

2.2. "gGg gGg gGg"
```

Figure 4.2.a. Example Of Chapter Renumbering

4.3. AUTOMATIC CHAPTERS

Contents, Index and Figures chapters are "automatic chapters", this means:

- the file cannot contain more than one automatic chapter for each of the three types
- all input lines after the automatic chapter line until the next chapter line (or until end of file) are supposed to be the old chapter content and are deleted
- the new chapter content is automatically inserted after the chapter line

With '-p n' (see 5.), page headers are not inserted and pages are not Numbered. So, if an automatic chapter is requested, it will still appear, but without page numbers aside, as in the following example.

WARNING: 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 Undo mode.

Automatic chapters are controlled by -c -i -f and -F arguments, which must all be non-null strings. Furthermore -c -f and -i must have, if uppercased, all distinct values, and -F cannot contain blanks.

4.3.1. CONTENTS CHAPTER

For "Contents chapter" we mean an automatic chapter containing the list of chapters, in order of appearance, possibly with the number of the page they begin on. Namely, the Contents chapter will list all Numbered chapters, the Index chapter and the Figures chapter, but it won't list either the Nameless chapter or the Contents chapter itself.

The Contents chapter 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 chapter line is a Contents chapter line if it's equivalent to -c. Such a line is:

- shrunk and uppercased where it is
- shrunk and titlecased when inserted into page headers by '%c' (see 5.)

Example:

```
$ cat Contents.txt
Contents

    (any previous content
    of Contents chapter
    will be deleted)

0. aAa aAa aAa

32.33. bBb bBb 'bBb'

0.0. cCc "cCc" cCc

0. 'dDd dDd' dDd

0.0. eEe 'eEe' "eEe"

9.9.9. 'fFf fFf fFf'

0.0. "gGg gGg gGg"

$ yawp -M f -w 32 Contents.txt

$ cat Contents.txt
Contents

    • 1.     Aaa Aaa Aaa
    • 1.1.   Bbb Bbb 'Bbb'
    • 1.2.   Ccc "cCc" Ccc
    • 2.     'Ddd Ddd' Ddd
    • 2.1.   Eee 'Eee' "eEe"
    • 2.1.1. 'Fff Fff Fff'
    • 2.2.   "gGg gGg gGg"

1. AAA AAA AAA

1.1. BBB BBB 'BBB'

1.2. CCC "cCc" CCC

2. 'DDD DDD' DDD

2.1. EEE 'EEE' "eEe"

2.1.1. 'FFF FFF FFF'

2.2. "gGg gGg gGg"
```

Figure 4.3.1.a. Example Of Contents Chapter

Another example of Contents chapter is on the top of this Manual.

4.3.2. INDEX CHAPTER

For "Index chapter" we mean an automatic chapter containing a list of subjects, in alphabetical order, possibly showing which pages each "subject" is on. Subjects are sought in text lines only, therefore neither in pictures nor in captions nor in chapter lines. A subject can appear:

- 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 text 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 not be greater than half of -w argument, whether set by user or automatically computed.

The Index chapter 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 chapter line is an Index chapter line if it's equivalent to -i. Such a line is:

- shrunk and uppercased where it is
- shrunk and titlecased when inserted:
  - into the Contents chapter
  - into page headers by '%c' (see 5.)

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 argument, then it's truncated and terminated by '...' with a warning message.

An example of Index chapter is on the bottom of this Manual.

4.3.3. FIGURES CHAPTER

For "Figures chapter" we mean an automatic chapter containing a list of figure captions, in order of appearance, possibly showing which page a "caption" is on. Not all pictures need to be followed by a caption, and a caption without a picture does not make sense, so we can say a "figure" is a picture followed by a caption.

Figures chapter is controlled by -f and -F arguments. The Figures chapter line starting the Figures chapter is defined by -f:

- "-f, --figures-title": title of Figures chapter (default: 'figures'), example:

```
$ yawp -f 'list of figures' ...
```

A chapter line is a Figures chapter line if it's equivalent to -f. Such a line is:

- shrunk and uppercased where it is
- shrunk and titlecased when inserted:
  - into the Contents chapter
  - into page headers by '%c' (see 5.)

YAWP recognizes a figure caption by -F:

- "-F, --caption-prefix": first word of figure captions (default: 'figure')

-F must be a single word, in other words it cannot contain blanks.

A line is a "figure caption line" if:

- is a one-line picture, in other words:
  - is the first line in file or it's preceded by an empty line
  - is the last line in file or it's followed by an empty line
  - is a not empty line and starts with blank
- contains two or more blank-separated words, of which:
  - the first one is equivalent to -F
  - the second one is a "caption label" containing:
    - the label of the containing chapter (label of Nameless chapter is null string '')
    - a progressive lowercase letter between 'a' and 'z', meaning caption position in the containing chapter
    - a decimal point '.'
  - the following words, if any, are the figure title

Examples of caption labels are:

- 'a.' (the first caption in the Nameless chapter)
- '7.c.' (the third caption in the '7.' chapter)
- '7.9.b.' (the second caption in the '7.9.' chapter)
- '7.9.8.z.' (the 26th caption in the '7.9.8.' chapter)

A single chapter cannot contain more than 26 caption lines, one for each letter of english alphabet.

The input caption label can be any, of any level, it's automatically updated on output. Caption lines are:

- shrunk titlecased and centered where they are
- shrunk and titlecased when listed in the Figures chapter

The label letter always remains lowercase. Notice the difference between:

- the Figures chapter line, one, defined by -f, starting the Figures chapter
- the figure caption lines, many, defined by -F, listed in the Figures chapter

We suppose a figure caption is logically connected with previous picture, so, when applicable, figure caption lines are logically pasted with previous picture, in order to avoid figure and caption split across two pages.

An example of Figures chapter is on the bottom of this Manual.

5. PAGING

In format mode page headers in input are automatically deleted, then they can be reinserted (or not) under control of -p argument:

- -p, --page-headers: insert page headers (default: 'n' = no, 'f' = on full page, 'p' = and on broken pictures, 'c' = and on level-1 Numbered Contents index or Figures chapters)

With default '-p n', page headers are not inserted. So, if you want to eliminate page headers from your file, just run YAWP in Format mode with '-p n'. Page footers are not implemented.

With '-p f', page headers are inserted when current line does not fit into current page

With '-p p', page headers are inserted when:

- current line does not fit into current page
- current picture does not fit into current page

With '-p c', page headers are inserted when:

- current line does not fit into current page
- current picture does not fit into current page
- current line is a level-1 Numbered or Contents or figures or Index chapter line

With '-p p' and '-p c', if a picture is too long to fit into the current page, a page eject is forced. But if the picture is higher than one page, such a page eject is useless and does not take place, see for example the figure 8.3.a.

Each page, except the first one, is prefixed by a page header. Therefore the total number of pages is equal to the number of page headers plus one. Each page header is made up of two header lines:

- first header line, starting with a "form feed" '\f' character, containing data such as file name, chapter title, date, time or page number
- second header line, a separation line made of "macron" '¯' characters, which form a kind of dashed underline under the first header line

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

WARNING: never start a text line with a form feed or a macron character, after a YAWP run such a line would disappear.

Inserting page headers in a Python file does not make sense, so if the text file name ends with '.py', -p is silently set to 'n'.

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

- "-e, --even-left": headers of even pages, left (default: '%n/%N')
- "-E, --even-right": headers of even pages, right (default: '%F %Y-%m-%d')
- "-o, --odd-left": headers of odd pages, left (default: '%c')
- "-O, --odd-right": headers of odd pages, right (default: '%n/%N')

Each "percent variable" is evaluated as follows, no other percent variable is allowed.

VAR	VALUE
'%P'	text file "long path", without final '/'
'%p'	text file "short path", without final '/'
'%f'	text file name, with no extension
'%F'	text file name, with no extension, altered
'%e'	text file extension, if any, with initial '.'
'%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 level-1/Contents/Figures/Index chapter
'%%'	'%'

Figure 5.a. Percent Variables

What do "long path" ('%P') and "short path" ('%p') mean? If for instance the current text file is '~/yyyy/zzz.www.txt' and the user is 'xxxx' then we get:

- '%P' --> '/home/xxxx/yyyy'
- '%p' --> '~/yyyy'
- '%f' --> 'zzz.www'
- '%F' --> 'Zzz Www'
- '%e' --> '.txt'
- '%P/%f%e' --> '/home/xxxx/yyyy/zzz.www.txt'
- '%p/%f%e' --> '~/yyyy/zzz.www.txt'

[2] What does "altered file name" ('%F') mean? File name is altered by '%F' as follows:

- any special character is replaced by a blank

- resulting words are titlecased

Actual value of '%n' is affected by:

- "-n, --page-offset": offset of page numbers (default: '0', min: '-3999', if negative it is the count of initial Roman numbers)

	-n, --page-offset									
PAGE	-5	-4	-3	-2	-1	0	1	2	3	4	5
1st	i	i	i	i	i	1	2	3	4	5	6
2nd	ii	ii	ii	ii	1	2	3	4	5	6	7
3rd	iii	iii	iii	1	2	3	4	5	6	7	8
4th	iv	iv	1	2	3	4	5	6	7	8	9
5th	v	1	2	3	4	5	6	7	8	9	10
6th	1	2	3	4	5	6	7	8	9	10	11
7th	2	3	4	5	6	7	8	9	10	11	12
8th	3	4	5	6	7	8	9	10	11	12	13
9th	4	5	6	7	8	9	10	11	12	13	14
10th	5	6	7	8	9	10	11	12	13	14	15

Figure 5.b. "%n" Value On Pages Depending On "-n" Argument

As you can see:

- if -n argument is zero (the default), the numbering is the natural one, the first page is '1', the second is '2', and so on
- if -n is more than zero, page numbers are incremented by -n, this can be useful if the text file is only a chapter of a larger document
- if -n is less than zero, a number of lowercase-"Roman"-Numbered pages equal to the absolute value of -n precedes the "Arabic"-Numbered pages

-n is the only numeric YAWP's argument that can take negative values. -n cannot be less than -3999 because 3999 = 'mmmcmxcix' is the maximum standard Roman number.

'%N' is still the total number of pages (Roman plus Arabic) and is not affected by the -n argument.

If you do not need front-back printing, set:

- "-a, --all-pages-E-e": put in all page headers -E at left and -e at right

and so you'll get the same header layout on even and odd pages. Notice the swap between -e and -E.

Argument -a has a side effect on left margin -L and right margin -R:

- if -a is off, left and right margins are swapped with each other on even pages
- if -a is on, left and right margins stay in place on odd and even pages

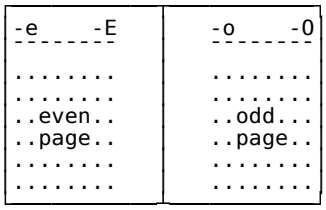


Figure 5.c. Page Layout With "-L" > "-R" And "-a" Off

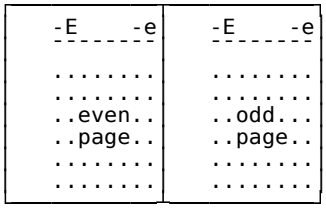


Figure 5.d. Page Layout With "-L" > "-R" And "-a" On

6. 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

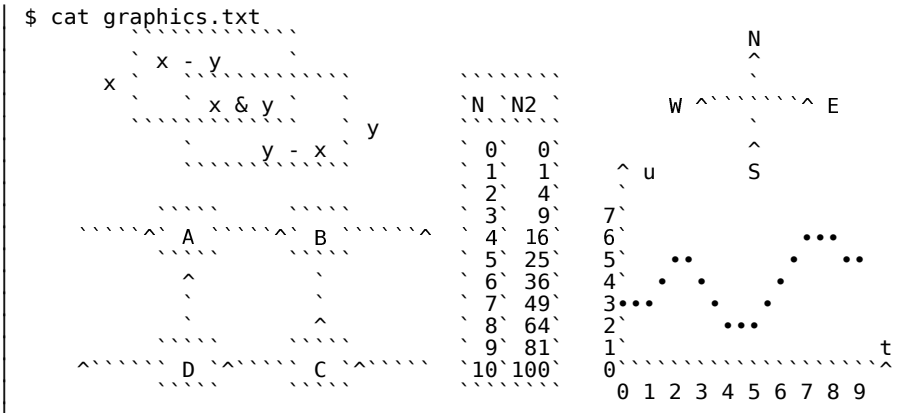
and usage mode is Format or Noformat, then YAWP will redraw the pictures by replacing the draw characters with suitable graphic ones, depending on the other four characters around (left right above and below). Standalone draw characters are not replaced.

This feature works:

- in Format mode: in picture lines only, but not in text lines
- in Noformat mode: in all lines of the text file

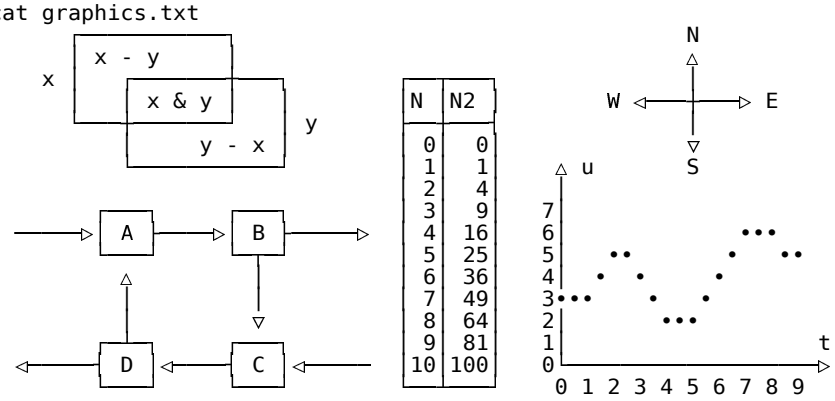
An example:

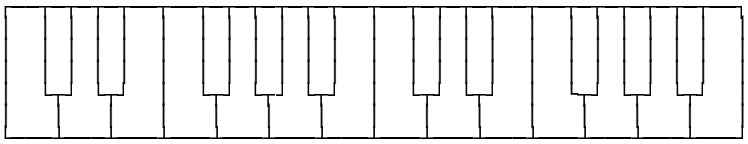
```
$ cat graphics.txt
```

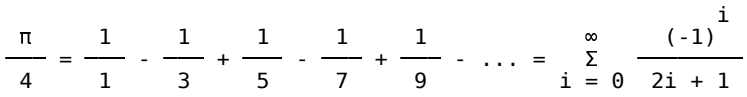


```
$ yawp -M f -g graphics.txt
```

```
$ cat graphics.txt
```







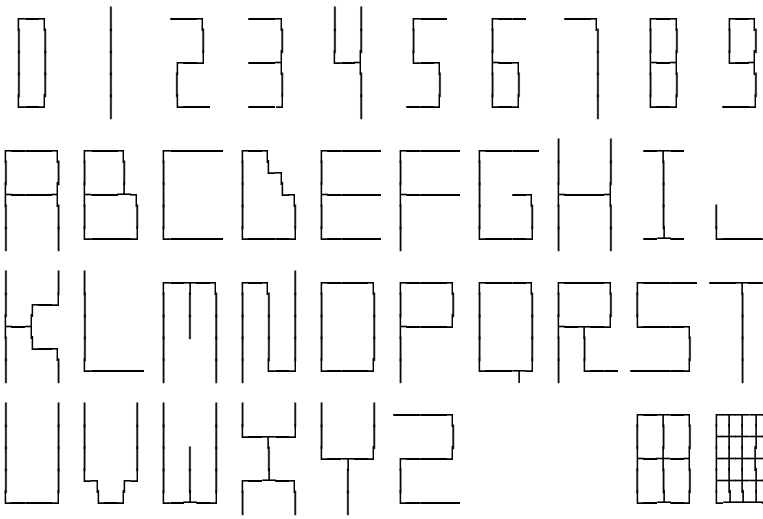


Figure 6.a. Example With "-g"

Argument -g works with '-M 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 in Noformat mode the whole file is a single picture. As another example, we want to print a small chessboard. Some remarks:

- '-M n' prevents formatting the file as text, but the backup is still performed because '-g' alters the text file
- 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 chessboard.pdf, so by printing it on an A4 sheet you get a square chessboard of about 18x18cm, 7x7in

```
$ 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 -M n -g -A 1 -X chessboard.txt
```
```

```
$ cat chessboard.txt
```

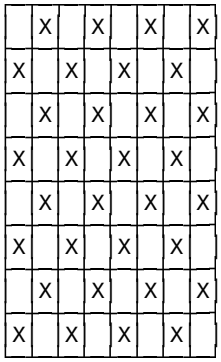


Figure 6.b. Example With "-M n -g -A 1"

7. PDF EXPORTING

In Format Noformat and Undo mode the text file is exported into a PDF file if argument -X is set:

- "-X, --export-pdf": after processing export and watch PDF file

then a PDF file is exported and viewed by a PDF browser (as atril evince or okular) for check preview and print.

To define path and name of the exported PDF file, set the argument:

- "-P, --file-pdf": exported PDF file (default: '%f.pdf')

Percent variables in -P are evaluated as explained in figure 5.c., 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 lowercase '.pdf'.

The resulting file name can be prefixed by a path or not. If path is given, it must exist, otherwise the directory containing text_file is assumed.

If the PDF file already exists, it is silently overwritten.

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

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

Value is an integer or float literal followed by a lowercase unit 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

You can also use comma ',' as decimal separator, '2,54cm' = '2.54cm'.

There is no default unit, the suffix is mandatory. 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 (characters per line) 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 body line length in file (page headers are not considered)
  - -W is computed from -w

Character proportions are defined by:

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

-A is a ratio, so it can be:

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

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

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

Format is portrait, in other words must width cannot be greater than height.

Values can be paper format names too, see the following table. These names are case-insensitive ('A4' or 'a4' is the same), while the 'x' separator and the unit suffix in the explicit value must be lowercase.

FORMAT 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

Figure 7.a. Symbolic Names For "-S"

Paper width and height can be 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 (default: '2cm' = min, unit: 'pt' 'in' 'mm' or 'cm')
- "-R, --right-margin": right margin (default: '2cm' = min, unit: 'pt' 'in' 'mm' or 'cm')
- "-T, --top-margin": top margin (default: '2cm' = min, unit: 'pt' 'in' 'mm' or 'cm')
- "-B, --bottom-margin": bottom margin (default: '2cm' = min, unit: 'pt' 'in' 'mm' or 'cm')

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

Left and right margins are affected by -a argument:

- if -a is off, left and right margins are swapped with each other on even pages
- if -a is on, left and right margins stay in place on odd and even pages

Computations about geometry of characters and pages are quite straightforward. When the number of characters per line is automatic, it has to be computed from the character width:

$$\text{chars-per-line} = \frac{\text{page-width} - \text{left-margin} - \text{right-margin}}{\text{char-width}}$$

When the character width is automatic, it has to be computed from the number of characters per line:

$$\text{char-width} = \frac{\text{page-width} - \text{left-margin} - \text{right-margin}}{\text{chars-per-line}}$$

Now we can compute the character height and the number of lines per page:

$$\text{char-height} = \frac{\text{char-width}}{\text{char-aspect}}$$

$$\text{lines-per-page} = \frac{\text{page-height} - \text{top-margin} - \text{bottom-margin}}{\text{char-height}}$$

8. RESERVED FILES

Functions of YAWP are supported by eight types of hidden reserved files, and this chapter ilustrates technicalities about these files.

FILE TYPE	FILE PATH AND NAME	SEE
session file	~/.yawp/.yawp.sess	8.1.
history file	~/.yawp/.yawp.hist	8.2.
correction file	~/.yawp/.yawp.corr	8.3.
log files	~/.yawp/.yawp.%Y.%m.%d-%H.%M.%S.log	8.4.
argument files	%P/.yawp.%f%e.args	8.5.
lock files	%P/.yawp.%f%e.lock	8.6.
temporary files	%P/.yawp.%f%e.temp	8.7.
backup files	%P/.yawp.%f%e.%Y.%m.%d-%H.%M.%S.back	8.8.

Figure 8.a. Hidden Reserved Files

Percent variables are explained in figure 5.c., '%P' '%f' and '%e' refer of course to the current text file.

8.1. SESSIONS AND SESSION FILE

Each execution of YAWP in GUI mode is a "session". Each session can process, one at a time, more text files. Path and name of the last text file, at session end, is saved into an unique session file:

~/.yawp/.yawp.sess

When YAWP is invoked in GUI mode again and next session begins:

- if YAWP is invoked with text_file argument, as in 'yawp exemple.txt', text file to be processed is taken from the argument
- if otherwise YAWP is invoked without text_file argument, as in 'yawp', text file to be processed is taken from the previously saved '~/.yawp/.yawp.sess' file, if any

In CLI modes the session file is not used.

8.2. RECENT BUTTON AND HISTORY FILE

When in GUI mode a file is processed by Edit or Format or Noformat or Undo action, path and name of the file are recorded into the history file

~/.yawp/.yawp.hist

Such entries are managed avoiding duplications, in descending chronological order, until a max limit of 20.

You can access history in order to select the next text file to be processed, or clear the history file, by the Recent button, see 2.2.2.

In CLI modes the history file is not used.

8.3. CORRECTIONS AND CORRECTION FILE

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" command the same thing happens. 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 errors are different, and so corrections must be different

A mechanism in YAWP tries to correct such errors. 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 correction points.

Corrections are controlled by -C argument:

- "-C, --correct-sizes": correct character size and page margins (default: 'd' = by default values, 'n' = no, 'f' = by correction file)

and by the correction file '~/.yawp/.yawp.corr', if exists:

- if -C is 'n', no correction takes place, and we can make experiments in order to define content of correction file
- if -C is 'd', the default corrections are applied
- if -C is 'f', correction point are read from correction file

An example for correction file follows, corresponding to the default corrections:

```
#
# ~/.yawp/.yawp.corr
#

plm  0mm  7mm # portrait left margin
plm 10mm 16mm
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 5mm 16mm # landscape left margin
llm 10mm 20.5mm
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 0mm 7mm # portrait right margin
prm 10mm 15mm
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 5mm 17mm # landscape left margin
lrm 10mm 22mm
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 0mm 2mm # portrait top margin
ptm 10mm 11.5mm
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 5mm 7mm # landscape top margin
ltm 10mm 11mm
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 0mm 16mm # portrait bottom margin
pbm 10mm 24mm
pbm 20mm 34mm
pbm 30mm 43mm
pbm 40mm 52.5mm
pbm 50mm 62mm
pbm 60mm 71mm
pbm 70mm 81mm
```

```
pbm 80mm 90mm
pbm 90mm 100mm
pbm 100mm 109.5mm

lbm 5mm 19mm # landscape bottom margin
lbm 10mm 24mm
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
```

Figure 8.3.a. Example Of Correction File

As you can see, standard Unix '#'-comments and empty lines are accepted. The file contains a set of correction points, in ascending order. Each "correction point" has the form 'k y x', where:

- k is a 3-letters lowercase "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 or centimeters
- x is an "obtained value" in points inches millimeters or centimeters

What does it mean "in ascending order"?

- For each key, first y value must be not less than zero, and each next y value must be strictly greater than the previous one
- For each key, first x value must be not less than zero, and each next x value must be strictly greater than the previous one

For instance, let's take the first correction point, 'plm 10mm 16mm'. This means that, by trying to print a page with -Z off, -C n, 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 should 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 correction points, 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 correction points, namely:

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

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

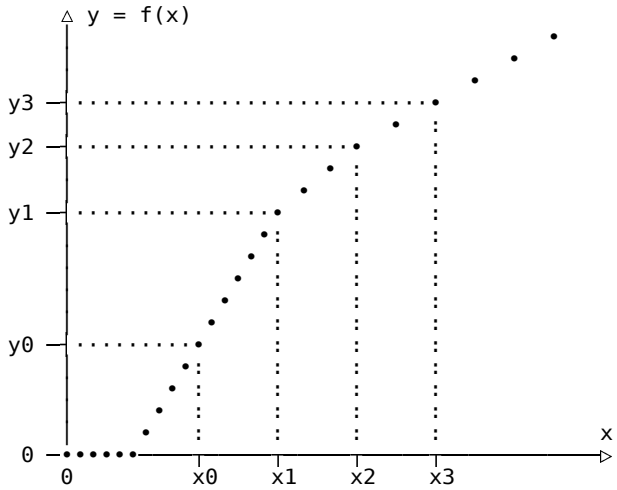


Figure 8.3.b. A Correction Function Defined By Four Correction Points

In GUI mode the correction file can be edited, reset to default values, or restored by the Correct button, see 2.2.3. With the due differences, the correction archive is managed in the same way as the text archives. Namely:

- if the Edit Restore or Undo buttons are clicked, the correction file is locked until end of the YAWP session by creating a lock file like:

  ~/.yawp/.yawp..yawp.corr.%Y.%m.%d-%H.%M.%S.lock

  and released at session end

- if the Edit or Restore buttons are clicked and the correction file is actually modified, its original content is saved in a backup file like:

  ~/.yawp/.yawp..yawp.corr.%Y.%m.%d-%H.%M.%S.back

8.4. MESSAGES AND LOG FILES

During execution YAWP can write various types of messages on the log file:

- an initial frame containing the log file name
- frames containing the operation performed and a datetime timestamp
- "information messages" saying what's going on
- "warning messages" (starting with 'WARNING:') saying what may be wrong, and YAWP processing continues
- "error messages" (starting with 'ERROR:') saying what's wrong, they can be:
  - "argument error messages", when an argument passed to YAWP is wrong or inconsistent
  - "correction error messages", when an error in correction file is found
  - "processing error messages", when an error raises during processing of the text file

Information and warning messages are displayed on stderr only if -v argument is on, except the initial frame containing the log file name, which is displayed anyway.

In case of an error message, file backup and rewriting do not take place. In CLI usage modes YAWP processing is terminated, while in GUI mode processing continues.

Correction error messages are followed by position and content of the offending line in the correction file.

Processing errors messages are followed by position and content of the offending line in the processed text file.

All three types of messages are written on stderr.

All messages (whether -v is on or off) are written to a timestamped log file, unique for each CLI or GUI execution, of the type:

~/.yawp/.yawp.%Y.%m.%d-%H.%M.%S.log

In GUI mode such a log file can be browsed through the Log button (see 2.2.2.).

A frame containing the log file name is always written on output, for further inspection.

An example of YAWP run with -v on follows.

```
$ yawp -M f -v -w 90 -p c -n -5 -E "YAWP 1.0.0b7 User Manual" -X -L 2.5cm yawp.txt
 /home/xxxx/.yawp/.yawp.2023.03.10-15.44.17.log

 Format     2023-04-01 15:44:17
Arguments:
    -h --help = False
    -V --version = False
    -H --view-manual = False
    -v --verbose = True
    -M --usage-mode = 'f'
    -y --text-editor = 'idle'
    -g --graphics = False
    -w --chars-per-line = '90'
    -l --just-left-only = False
    -c --contents-title = 'contents'
    -i --index-title = 'index'
    -f --figures-title = 'figures'
    -F --caption-prefix = 'figure'
    -p --page-headers = 'c'
    -e --even-left = '%n/%N'
    -E --even-right = 'yawp 1.0.0b7 User Manual'
    -o --odd-left = '%c'
    -O --odd-right = '%n/%N'
    -n --page-offset = '-5'
    -a --all-pages-E-e = False
    -X --export-pdf = True
    -Y --pdf-browser = 'xdg-open'
    -P --pdf-file = '%P/%f.pdf'
    -W --char-width = '0'
    -A --char-aspect = '3/5'
    -S --paper-size = 'A4'
    -Z --landscape = False
    -L --left-margin = '2.5cm'
    -R --right-margin = '2cm'
    -T --top-margin = '2cm'
    -B --bottom-margin = '2cm'
    -C --correct-sizes = 'd'
     text_file = '/home/xxxx/pypi/yawp/yawp.txt'
Read:    YAWP <⟵ /home/xxxx/pypi/yawp/yawp.txt
Compute: -W --char-width '5.196850pt = 0.072178in = 1.833333mm = 0.183333cm'
Correct: -W --char-width 5.489205pt = 0.076239in = 1.936470mm = 0.193647cm
Correct: -L 41.844769pt = 0.581177in = 14.761905mm = 1.476190cm (even pages)
Correct: -R 59.269864pt = 0.823193in = 20.909091mm = 2.090909cm (even pages)
Correct: -L 55.343082pt = 0.768654in = 19.523810mm = 1.952381cm (odd pages)
Correct: -R 42.519685pt = 0.590551in = 15.000000mm = 1.500000cm (odd pages)
Correct: -T 53.709076pt = 0.745959in = 18.947368mm = 1.894737cm
Correct: -B 14.173228pt = 0.196850in = 5.000000mm = 0.500000cm
Compute: char height 8.661417pt = 0.120297in = 3.055556mm = 0.305556cm
Compute: chars per inch 13.854545
Compute: lines per inch 8.312727
Correct: char height 9.179314pt = 0.127490in = 3.238258mm = 0.323826cm
Correct: chars per inch 13.116652
Correct: lines per inch 7.843723
Compute: lines per page 81
Before:
    header: 74 lines, 186 words, 6660 chars, max 90 chars per line, 38 pages
    body:   2687 lines, 15777 words, 108239 chars, max 90 chars per line
    total:  2761 lines, 15963 words, 114899 chars, max 90 chars per line
WARNING: Index line for 'text file' longer than -w --chars-per-line 90, truncated
WARNING: Index line for 'yawp' longer than -w --chars-per-line 90, truncated
Backup:  '/home/xxxx/pypi/yawp/yawp.txt' ⟶▷
         '/home/xxxx/pypi/yawp/.yawp.yawp.txt.2023.03.10-15.44.17.back'
Rewrite: YAWP ⟶▷ '/home/xxxx/pypi/yawp/yawp.txt'
After:
    header: 76 lines, 191 words, 6840 chars, max 90 chars per line, 39 pages
    body:   2682 lines, 15771 words, 108147 chars, max 90 chars per line
    total:  2758 lines, 15962 words, 114987 chars, max 90 chars per line
Export:  '/home/xxxx/pypi/yawp/yawp.txt' ⟶▷
         '/home/xxxx/pypi/yawp/yawp.pdf'
```

Figure 8.4.a. Example Of Verbose Output

This output has been written on terminal, thanks to -v argument, and recorded into:

    /home/xxxx/.yawp/.yawp.2023.03.10-15.44.17.log

If for any reason you want to remove all log files, type:

```
$ rm -fv ~/.yawp/.yawp.*.log
```

8.5. ARGUMENTS AND ARGUMENT FILES

If your text file is for example:

    ~/example.txt

then the associated arguments are saved into:

```
~/.yawp.example.txt.args
```

In CLI mode the arguments are:

- not loaded from a file but taken from the CLI interface at startup
- saved into .args file at finish

In GUI mode the arguments are:

- loaded from .args file at startup, if any (if otherwise the .args file is not found, arguments silently get their default values)
- saved into .args file and reloaded from another .args file whenever the current text file becomes another one
- saved into .args file at finish

This ensures argument continuity from a processing of a given text file to the next processing of the same text file:

- from CLI mode to GUI mode
- and from GUI mode to GUI mode

but not:

- from CLI mode to CLI mode
- or from GUI mode to CLI mode

8.6. LOCKS AND LOCK FILES

Both in CLI and in GUI mode, the text file is locked before processing in order to avoid interferences from other concurrent YAWP executions. Namely:

- in CLI mode the text file is locked before processing and unlocked at processing end
- in GUI mode any processed text file (by Edit Format Noformat or Undo action) is locked before processing, and stays locked until session end, so a single session can keep multiple text files locked at same time

Locking takes place through the creation of a lock file. For instance, in order to lock the text file:

```
~/yyyy/example.txt
```

a temporary hidden file:

```
~/yyyy/.yawp.example.txt.lock
```

is generated in the same directory and is removed at YAWP's end.

Path and name of log file are written into the lock file, this acts as a session signature and allows each YAWP session to distinguish its own lock files from those of others.

So two concurrent YAWP instances can process at the same time two distinct text files, but not both the same text file.

WARNING: Locks do not prevent locked text files from being modified by another concurrent non-YAWP application.

WARNING: Locks protect text files and correction file '~/.yawp/.yawp.corr' only, while '~/.yawp/.yawp.sess' and '~/.yawp/.yawp.hist' are not lock-protected.

It shouldn't happen, but if for any reason one or more text files in a given directory remain locked after running YAWP, jump to the directory and type:

```
| $ rm -fv .yawp.*.lock
```

8.7. FAKE MARGINS AND TEMPORARY FILES

When arguments -X and -K are both on, the PDF file is generated from the text file through an intermediate temporary file. This file contains empty lines at the top of the pages and blanks on the left of text lines, in order to simulate the top margin and the left margin respectively. For example the text file

```
~/yyyy/example.txt
```

will generate a temporary file:

```
~/yyyy/.yawp.example.temp
```

The temporary file is deleted when it is no longer needed. It shouldn't happen, but if for any reason one or more temporary files remain in a given directory after running YAWP, jump to the directory and type:

```
| $ rm -fv .yawp.*.temp
```

8.8. BACKUPS AND BACKUP FILES

The text file is backed up when needed into a timestamped hidden backup file, created in the same directory. For example the text file

```
~/yyyy/example.txt
```

will be backed up into a backup file of this kind:

```
~/yyyy/.yawp.example.txt.%Y.%m.%d-%H.%M.%S.back
```

Such a backup can take place after any operation likely to alter the text file, namely:

- GUI mode:
  - Edit button
  - Format button
  - Noformat button
- CLI modes:
  - Edit mode (yawp -M e)
  - Format mode (yawp -M f)
  - Noformat mode (yawp -M n)

In order to avoid proliferation of useless backup files, the backup is performed only if the Edit/Format/Noformat operation actually altered the content of the text file.

If for any reason you want to remove all backup files in a given directory, jump to the directory and type:

```
$ rm -fv .yawp.*.back
```

9. AFTERWORDS

9.1. VERSIONS

- version 1.0.0b7 - 2023-04-27 - on test.pypi.org
  - something fixed
- version 1.0.0b6 - 2023-04-18 - on test.pypi.org
  - something fixed
- version 1.0.0b5 - 2023-04-02 - on test.pypi.org
  - something fixed
- version 1.0.0b4 - 2023-04-01 - on test.pypi.org
  - something fixed
- version 1.0.0b3 - 2023-04-01 - on test.pypi.org
  - updated installation instructions
- version 1.0.0b2 - 2023-04-01 - on test.pypi.org
  - moved mkdir between constants
- version 1.0.0b1 - 2023-04-01 - on test.pypi.org
  - GUI mode, added and set as default usage mode
  - CLI Edit mode, added
  - several other additions and changes, too many to list here
- version 0.7.1 - 2022-06-25
  - experimental obsolete and deprecated
- version 0.6.1 - 2022-05-06
  - experimental obsolete and deprecated
- version 0.5.4 - 2022-04-04
  - experimental obsolete and deprecated
- version 0.5.3 - 2022-04-02
  - experimental obsolete and deprecated
- version 0.5.2 - 2022-04-02
  - experimental obsolete and deprecated
- version 0.5.1 - 2022-03-19
  - experimental obsolete and deprecated
- version 0.4.2 - 2022-01-04
  - experimental obsolete and deprecated
- version 0.4.1 - 2022-01-03
  - first version on https://pypi.org
  - experimental obsolete and deprecated

9.2. CREDITS

An analogous (but very different) program is the Unix command "fmt", for details type:

```
$ man fmt
```

We are not aware of any other program of this kind.

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

YAWP makes extensive use of f-strings, so it requires Python ≥ 3.6.

YAWP has been published on PyPI by flit 3.7.1:

https://pypi.org/project/flit

The GUI interface has been implemented thanks to PySimpleGUI 4.60.1:

https://pypi.org/project/PySimpleGUI

YAWP uses CUPS-PDF 3.0.1-13 to write PDF files, see:

https://www.cups-pdf.de

YAWP uses pdfrw 0.4 to manipulate PDF files, see:

https://pypi.org/project/pdfrw

Standard PDF browser in Xubuntu 21.10 is atril 1.26.0:

https://wiki.mate-desktop.org/mate-desktop/applications/atril

9.3. HOMONYMS

There are several homonyms online, with which this project has nothing to do. Some examples:

- Yale Alkohol Withdrawal Project
- Yet Another Wall Plotter
- Yet Another Weather Page
- Yet Another Web Playground
- Yet Another World Protector
- Young Adult Writing Program
- Young ArbitralWomen Practitioners

Anyway, this is the only YAWP you can find on https://pypi.org.

9.4. LICENSE

9.5. ACRONYMS

ACRONYM	MEANING
ASCII	American Standard Code for Information Interchange
CLI	Command Line Interface
CUPS	Common Unix Printing System
GNU	Gnu is Not Unix
GUI	Graphic User Interface
IDE	Integrated Development Environment
IDLE	Integrated Development and Learning Environment
PDF	Page Description Format
PyPI	Python Package Index
RAM	Random Access Memory
USA	United States of America
UTF8	Unicode Transformation Format 8 bits
WYSIWYG	What You See Is What You Get
YAWP	Yet Another Word Processor

Figure 9.5.a. Acronyms

10. UNICODE CHARACTERS

CHARACTER NAME	CHAR	HEX	DEC
horizontal tab	'\t'	0009	9
line feed	'\n'	000a	10
form feed	'\f'	000c	12
carriage return	'\r'	000d	13
space (or blank)	' '	0020	32
exclamation mark	'!'	0021	33
quotation mark (or double quote)	'"'	0022	34
number sign (or hash or pound)	'#'	0023	35
dollar sign	'$'	0024	36
percent sign	'%'	0025	37
ampersand	'&'	0026	38
apostrophe (or single quote)	"'"	0027	39
left parenthesis	'('	0028	40
right parenthesis	')'	0029	41
asterisk	'*'	002a	42
plus sign	'+'	002b	43
comma	','	002c	44
hyphen (or minus sign)	'-'	002d	45
decimal point (or full stop)	'.'	002e	46
slash	'/'	002f	47
digit zero	'0'	0030	48
...	...	...	...
digit nine	'9'	0039	57
colon	':'	003a	58
semicolon	';'	003b	59
less-than sign	'<'	003c	60
equal sign	'='	003d	61
greater-than sign	'>'	003e	62
question mark	'?'	003f	63
at sign	'@'	0040	64
latin capital letter A	'A'	0041	65
...	...	...	...
latin capital letter Z	'Z'	005a	90
left square bracket	'['	005b	91
back slash	'\\'	005c	92
right square bracket	']'	005d	93
circumflex accent	'^'	005e	94
low line (or underscore)	'_'	005f	95
back quote (or grave accent)	'‾'	0060	96
latin small letter a	'a'	0061	97
...	...	...	...
latin small letter z	'z'	007a	122
left curly bracket	'{'	007b	123
vertical bar	'|'	007c	124
right curly bracket	'}'	007d	125
tilde	'~'	007e	126
macron	'¯'	00af	175
middle dot	'·'	00b7	183
greek capital letter sigma	'Σ'	03a3	931
greek small letter pi	'π'	03c0	960
black small circle	'•'	2022	8226
infinity	'∞'	221e	8734
not equal sign	'≠'	2260	8800
less than or equal sign	'≤'	2264	8804
greater than or equal sign	'≥'	2265	8805
box drawings light horizontal	'─'	2500	9472
box drawings light vertical	'│'	2502	9474
box drawings light down and right	'┌'	250c	9484
box drawings light down and left	'┐'	2510	9488
box drawings light up and right	'└'	2514	9492
box drawings light up and left	'┘'	2518	9496
box drawings light vertical and right	'├'	251c	9500
box drawings light vertical and left	'┤'	2524	9508
box drawings light down and horizontal	'┬'	252c	9516
box drawings light up and horizontal	'┴'	2534	9524
box drawings light vertical and horizontal	'┼'	253c	9532
white square	'□'	25a1	9633
white up-pointing triangle	'△'	25b3	9651
white right-pointing triangle	'▷'	25b7	9655
white down-pointing triangle	'▽'	25bd	9661
white left-pointing triangle	'◁'	25c1	9665
white circle	'○'	25cb	9675
bullseye	'◎'	25ce	9678

Figure 10.a. Unicode Characters

11. CHEAT SHEET

- YAWP 1.0.0b7 - Yet Another Word Processor - https://pypi.org/project/yawp
- Carlo Alessandro Verre - carlo.alessandro.verre@gmail.com

Usage syntax:

```
$ yawp -H # show a help message and exit
$ yawp -V # show program's version number and exit
$ yawp -H [-Y pdf_browser] # browse the PDF YAWP User Manual and exit
$ yawp text_file # run YAWP in GUI mode, explicit text file, no arguments
$ yawp # run YAWP in GUI mode, text file from previous session, no arguments
$ yawp -M e [-y text_editor] text_file  # run YAWP in CLI Edit mode
$ yawp -M f [...arguments...] text_file # run YAWP in CLI Format mode
$ yawp -M n [...arguments...] text_file # run YAWP in CLI Noformat mode
$ yawp -M u [...arguments...] text_file # run YAWP in CLI Undo mode
```

Buttons in GUI Main window:

- 4 buttons dedicated to the selection of the text file:
  - New button: create a new empty text file
  - Open button: browse the file system to select the text file
  - Recent button: browse the list of recent files to select the text file
  - Saveas button: clone current text file into a new target text file
- Help button: browse the YAWP Manual through the PDF browser defined by -Y
- Exit button: save current text file with its arguments and finish
- 4 buttons dedicated to the processing of the text file:
  - Edit button: edit the text file through the text editor defined by -y
  - Format button: process the text file by formatting it
  - Noformat button: process the text file without formatting it
  - Undo button: restore the text file to its previous content
- Log button: browse the log file through the text editor defined by -y
- Correct button: manage the correction file (Edit Reset or Undo)

Usage arguments:

- -h, --help: show a help message and exit
- -V, --version: show program's version number and exit
- -H, --view-manual: view the YAWP-generated PDF YAWP User Manual and exit
- -v, --verbose: write all messages on stderr (default: write errors only)
- -M, --usage-mode: run YAWP in this usage mode (default: 'g' = GUI, 'e' = CLI Edit, 'f' = CLI Format, 'n' = CLI Noformat, 'U' = CLI Undo)

Format arguments:

- -y, --text-editor: editor for text files (default: 'idle')
- -g, --graphics: redraw '`'-segments and '^'-arrowheads
- -w, --chars-per-line: line width in characters per line (default: '0' = automatic)
- -l, --just-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')
- -f, --figures-title: title of Figures chapter (default: 'figures')
- -F, --caption-prefix: first word of figure captions (default: 'figure')

Paging arguments:

- -p, --page-headers: insert page headers (default: 'n' = no, 'f' = on full page, 'p' = and on broken pictures, 'c' = and on level-1 chapters)
- -e, --even-left: headers of even pages, left (default: '%n/%N')
- -E, --even-right: headers of even pages, right (default: '%F %Y-%m-%d')
- -o, --odd-left: headers of odd pages, left (default: '%c')
- -O, --odd-right: headers of odd pages, right (default: '%n/%N')
- -n, --page-offset: offset of page numbers (default: '0', min: '-3999', if negative it is the count of initial Roman numbers)
- -a, --all-pages-E-e: put in all page headers -E at left and -e at right

Export arguments:

- -X, --export-pdf: after processing export and browse PDF file
- -Y, --pdf-browser: browser for PDF files (default: 'xdg-open')
- -P, --file-pdf: exported PDF file (default: '%f.pdf')
- -W, --char-width: character width (default: '0' = automatic, 'pt' 'in' 'mm' or 'cm')
- -A, --char-aspect: character aspect ratio = char width / char height (default: '3/5', '1' = square grid)
- -S, --paper-size: portrait paper size width x height (default: 'A4' = '210x297mm', 'pt' 'in' 'mm' or 'cm')
- -Z, --landscape: turn page by 90 degrees (default: portrait)
- -L, --left-margin: left margin (default: '2cm' = min, 'pt' 'in' 'mm' or 'cm')
- -R, --right-margin: right margin (default: '2cm' = min, 'pt' 'in' 'mm' or 'cm')
- -T, --top-margin: top margin (default: '2cm' = min, 'pt' 'in' 'mm' or 'cm')
- -B, --bottom-margin: bottom margin (default: '2cm' = min, 'pt' 'in' 'mm' or 'cm')
- -C, --correct-sizes: correct character size and page margins (default: 'd' = by default values, 'n' = no, 'f' = by correction file)
- -K, --fake-margins: build left margin by blanks and top margin by empty lines

File argument:

- text_file: text file to be processed, ASCII or UTF-8-encoded

FIGURES

INDEX