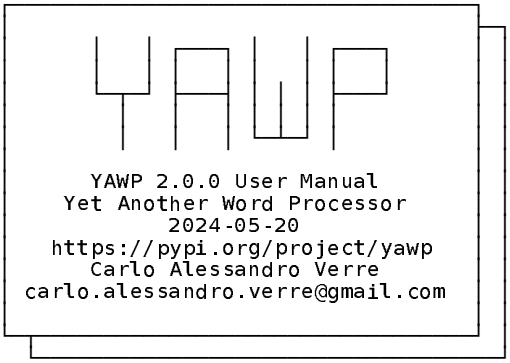


YAWP 2.0.0 User Manual
Yet Another Word Processor
2024-05-20
https://pypi.org/project/yawp
Carlo Alessandro Verre
carlo.alessandro.verre@gmail.com

I sound my barbaric yawp over the roofs of the world

Walt Whitman

CONTENTS

0. FOREWORDS

0.1. INTRODUCTION

"YAWP" here means Yet Another Word Processor, and YAWP is a pure-Python Linux-only free and open-source word processor for plain text files, with PDF export. If you really need all the endless 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 like this, give YAWP a try.

YAWP's main features are:

- YAWP has a GUI interface, but can be used as a CLI command too
- YAWP processes in place a single plain text file, hereinafter referred to simply as the "text file"
- YAWP's default text editor allows you to edit the text file and check its spelling correctness
- 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 text processing 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 schemes, 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 graphics capabilities, you can sketch pictures with horizontal and vertical segments (by '``') and arrowheads (by '^'), YAWP redraws them by suitable Unicode graphic characters
- pictures can also be left-aligned, centered or right-aligned
- 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 can export 2, 4 or 8 pages on each paper sheet side, allowing you to create in folio, in quarto and in octavo booklets
- YAWP writes information and error messages on terminal and on the log file of the text file
- 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 locks text files while they are processed, in order to avoid interference by other concurrent YAWP executions
- YAWP in GUI mode keeps a distinct argument file for each text file
- YAWP in GUI mode saves the name of the last processed text file and restores it at next invocation
- YAWP in GUI mode saves a list of the 25 most recent processed text files, among which you can select the next current text file

As an example of CLI usage, this YAWP User Manual has been generated as

    'YAWP 2.0.0 User Manual.txt.pdf'

from the text file

    'YAWP 2.0.0 User Manual.txt'

by typing at terminal:

```
$ yawp -M f -v -w 97 -m -1 -p c -n -4 -X e -L 3cm 'YAWP 2.0.0 User Manual.txt'
```

where arguments mean:

- -M f: run YAWP in CLI Format mode
- -v: write messages not only into log file but also on terminal
- -w 97: set 97 characters per line
- -m -1: let numbered chapters start from 0 instead of 1
- -p c: insert a page header on page full, on picture break and before level-1 chapters
- -n -4: first four pages are numbered by Roman numbers
- -X e: export in PDF format
- -L 3cm: set left margins on odd pages and right margins on even pages to 3 cm
- 'YAWP 2.0.0 User Manual.txt': the text file to process

0.2. INSTALLATION

If your Linux belongs to the Debian family, type at terminal:

```
$ sudo apt install KWrite universal-ctags atril printer-driver-cups-pdf pipx
```

On other platforms you will use the specific installer instead, for instance:

```
$ sudo yum install …        # on RHEL/CentOS/Fedora/Rocky/Alma Linux
$ sudo emerge -a sys-apps/… # on Gentoo Linux
$ sudo apk add …            # on Alpine Linux
$ sudo pacman -S …          # on Arch Linux
$ sudo zypper install …     # on OpenSUSE Linux
```

Then run:

```
$ pipx ensurepath
$ pipx install yawp
```

Now you can close the terminal, open another one, and call YAWP, see 1.1. for syntax.

Later you can type:

```
$ pipx upgrade yawp
```

in order to upgrade YAWP to a later version.

0.3. QUICK START

To start using YAWP you don't need to read this entire User Manual:

- read 4.2. Best Practices chapter
- keep 11. Cheat Sheet chapter handy
- start YAWP in GUI mode:

  ```
  $ yawp
  ```

- begin to explore arguments and action buttons

0.4. GLOSSARY

In this document four terms about character strings are marked by an asterisk '*' because they are used with an uncommon meaning.

A character string is said to be "shrunk*" when all leading and trailing blanks are removed, and any internal sequence of blanks is reduced to a single blank. Example:

```
'   -aBc   dEf   ' → '-aBc dEf'
```

A character string is said to be "uppercased*" when its words made of alphabetic characters only are translated into capital letters, while its words containing numeric or special characters are left unchanged. In the next example, the word '-aBc' is not altered because contains the special character '-':

```
'-aBc dEf' → '-aBc DEF'
```

A character string is said to be "titlecased*" when, in its words made of alphabetic characters only, the first letter is translated into a capital letter and the remaining letters are translated into small letters, while its words containing numeric or special characters are left unchanged. Example:

```
'-aBc dEf' → '-aBc Def'
```

Two character strings are said to be "equivalent*" if when both shrunk* and uppercased* they become the same. As an example, these two stings are equivalent*:

```
'   -aBc   dEf   ' ≡ '-aBc Def   '
```

because both strings, when shrunk* and uppercased*, become '-aBc DEF'.

1. USAGE

1.1. USAGE MODES

After installation, you can open a terminal and call YAWP, syntax is:

```
$ yawp -h               # show a help message and exit
$ yawp -V               # show program's version number and exit
$ yawp -H […arguments…] # browse the PDF YAWP User Manual and exit
$ yawp text_file # run YAWP in GUI mode, explicit text file, no other arguments
$ yawp           # run YAWP in GUI mode, text file from previous session, no arguments
$ yawp -M c […arguments…] text_file target_file # run YAWP in CLI Copy mode
$ yawp -M m […arguments…] text_file target_file # run YAWP in CLI Move mode
$ yawp -M d […arguments…] text_file             # run YAWP in CLI Delete mode
$ yawp -M e […arguments…] 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 Noform mode
$ yawp -M u […arguments…] text_file             # run YAWP in CLI Undo mode
```

Figure 1.1.a. Invocation Syntax

Both in GUI mode and in CLI modes, all user-supplied arguments are shrunk*. General behavior is controlled by the following arguments:

- "-h", "--help": show a help message and exit
- "-V", "--version": show program's version number and exit
- "-H", "--browse-manual": browse the YAWP-generated PDF YAWP User Manual and exit
- "-v", "--verbose": write information messages on stdout too (CLI modes only)
- "-M", "--usage-mode": run YAWP in this usage mode (default: 'g'=GUI, 'c'=CLI Copy, 'm'=CLI Move, 'd'=CLI Delete, 'e'=CLI Edit, 'f'=CLI Format, 'n'=CLI Noform, 'u'=CLI Undo)

The last argument is positional text_file, which indicates the text file to process. It is optional in GUI mode, mandatory in CLI modes. A leading path (absolute or relative) is optional, default is the current directory. A '.txt' extension is recommended but not mandatory. The file can contain ASCII characters, or any other UTF-8-encoded Unicode character. A second text file argument (target_file) is required in CLI Copy mode (-M c) and CLI Move mode (-M m).

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 and Apple)
- carriage return '\r' (Apple old MacOS pre-OSX)
- carriage return + line feed '\r\n' (MS-Windows)

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

1.2. TEXT EDITOR

YAWP uses a text editor, you can choose your preferred one. To define the text editor, set the argument:

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

Such an editor is used:

- to edit the text file by GUI Edit button, or by CLI Edit mode
- to browse the log file of the current text file by GUI Log button

You can choose your favorite graphical text editor, but YAWP is not compatible with command-line text editors, such as vi vim micro nano or pico. Avoid also tabbed text editors such as gedit, because they can generate a bothersome entanglement between two concurrent YAWP instances, as in:

```
$ yawp -M e -y gedit one.txt & yawp -M e -y gedit two.txt
```

which does not produce two distinct edit windows, but a single window with two edit tabs inside.

KWrite was chosen because it doesn't have the aforementioned flaw, and has many options, including an out-of-the-box spell checker. If the text editor is KWrite, the first time you should change the screen font in order to get a correct rendering of Unicode graphic characters, click:

- Settings
- Configure KWrite
- General
- Editor Font…
  - Font: DejaVu Sans Mono
  - Font style: Book
  - Size: (as you like it)
  - OK
- Save
- Close

In order to use the KWrite's spell checker, click:

- Tools
- Spelling
  - Automatic Spell Checking (Ctrl-Shift-O) to toggle on/off the red underlining of the wrong words
  - Spelling… to start a check on the entire text file
  - Spelling (from Cursor)… to start a check from the current character
  - Change Dictionary to select a different language

2. "GUI" MODE

YAWP running in GUI mode is a graphic front-end between user and CLI modes. In GUI mode the text_file argument is optional and the other arguments (including -M) are not allowed.

```
$ yawp text_file # run YAWP in GUI mode, explicit text file, no other arguments
$ yawp           # run YAWP in GUI mode, text file from previous session, no arguments
```

If text_file is not given, path and name of the text file to process are read from a session file, written at end of previous YAWP's execution. If such a file is not found, the text file is silently set to undefined.

If the text file is undefined or does not exist or exists but is locked, it is provisionally accepted, but it will raise an error during subsequent processing, namely in Format and Noform actions.

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 session and argument files, see 9.1. and 9.4.

Then the Main window appears.

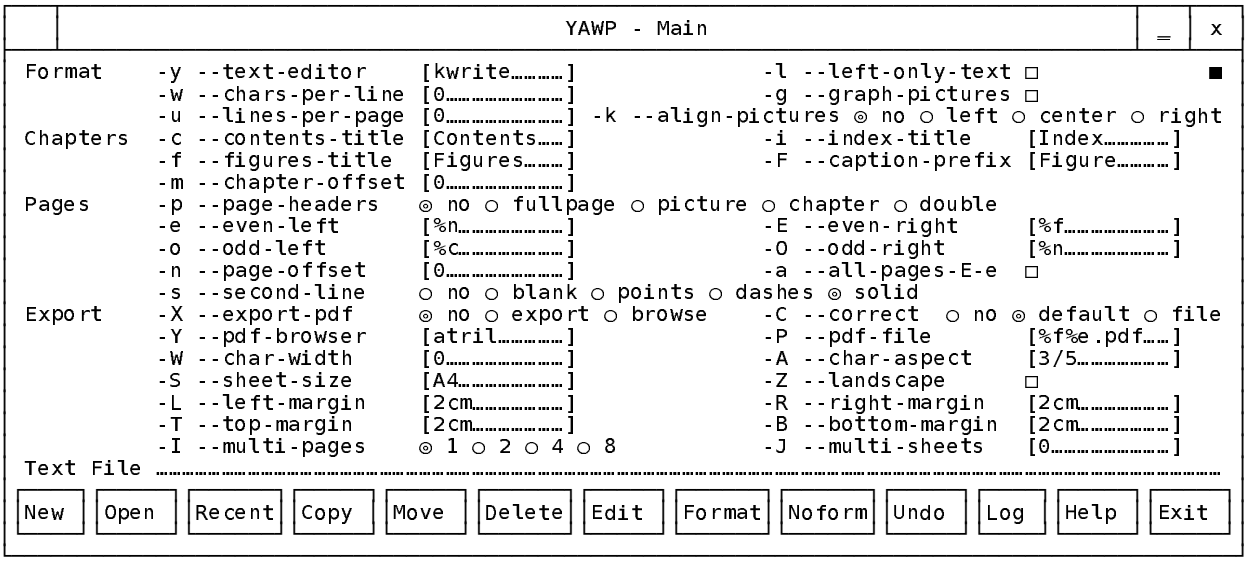


Figure 2.a. Main Window With Default Argument Values

The Main window contains:

- 18 lines with 33 arguments, of 3 types:
  - boolean (checkbox) arguments (-l -g -a and -Z), they can be on or off, but they are all turned off by default
  - radio button arguments (-k -p -s -X -C and -I)
  - field arguments (all the others)
- a line with the current text_file argument as a read-only field (it can be changed by New Open Recent Copy or Move buttons, or cleared by Delete button)
- a line with 13 action buttons, each corresponding to an action to perform:
  - text file definition:
    - New: create a new empty text file with default arguments
    - Open: browse the file system to select an existing text file
    - Recent: browse the list of recent files to select an existing text file
    - Copy: copy the current text file with its argument file
    - Move: move the current text file with its argument log and backup files
    - Delete: delete the current text file with its argument log and backup files
  - text file processing:
    - Edit: edit the current text file by the text editor defined by -y
    - Format: format the current text file and redraw and align pictures and export PDF
    - Noform: don't format current text file but redraw and align pictures and export PDF
    - Undo: restore the current text file to its previous content and export PDF
    - Log: browse the log file of current text file by the text editor defined by -y
  - other actions:
    - Help: browse the YAWP-generated YAWP Manual by the PDF browser defined by -Y
    - Exit: quit YAWP
- at right end of the first button line is also present a read-only two-values "status indicator", a colored square character '■':
  - GREEN while YAWP is idle, waiting for the user to request an action by clicking a button
  - RED while YAWP is busy, performing an action requested by the user

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

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

- move window focus, forward by Tab, backward by Shift-Tab
- when focus is on a field argument, update its value
- 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 its siblings off
- when focus is on an action button, press Space bar to trigger the associated action
- when you want to exit from YAWP, you can also press Alt-F4

2.1. NEW BUTTON

Create a new empty text file with default arguments:

- starting from the directory containing the current text file, browse the file system and click the Save button to define the new text file:
  - the new file may and may not exist, but if it already exists:
    - you are asked for confirmation
    - the new file can't be the current text file and must not be locked
    - its argument log and backup files (if any) are removed
  - the new text file is created as an empty file with default arguments
  - the new text file becomes the current text file
- otherwise press the Cancel button to go back without doing anything

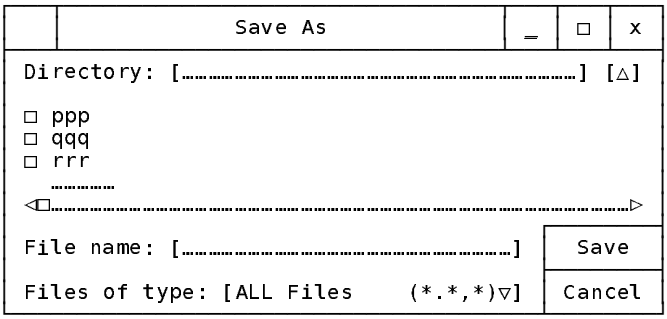


Figure 2.1.a. New Button, New File Definition Window

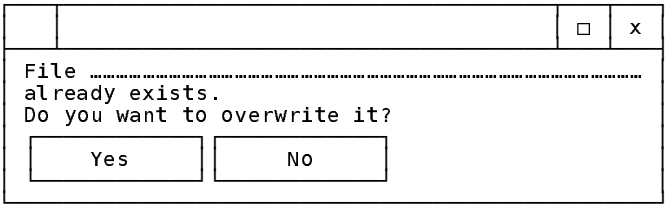


Figure 2.1.b. New Button, Overwriting Confirmation Window

2.2. OPEN BUTTON

Browse the file system to select an existing text file:

- starting from the directory containing the current text file, browse the file system and click the Open button to select an existing file:
  - the selected file can't be the current text file and must not be locked
  - the selected file becomes the new current text file
- otherwise press the Cancel button to go back without doing anything

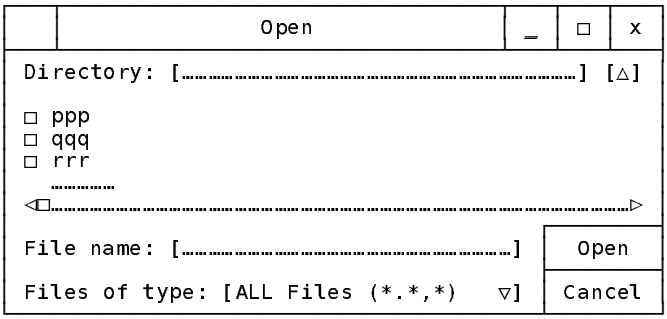


Figure 2.2.a. Open Button, File Selection Window

2.3. RECENT BUTTON

Browse the list of recent files to select an existing text file:

- view the list of most recently processed text files (max 25, the most recent one on top) and click a file button to select an existing file:
  - the selected file can't be the current text file and must not be locked
  - the selected file becomes the new current text file
- otherwise click the Clear button to clear (after confirmation) the list of recent files
- otherwise press the Cancel button to go back without doing anything

The selected text file can not be the current text file. For further details about history file containing the list of recent text files, see 9.2.

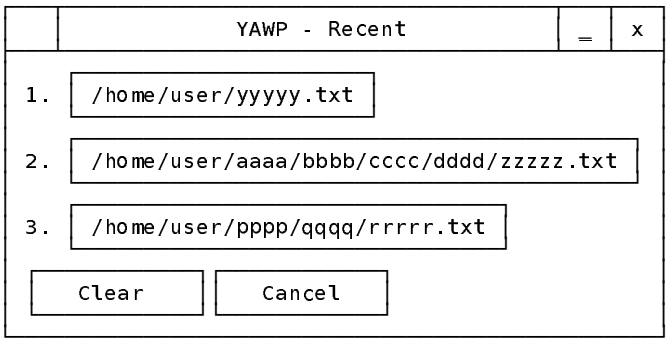


Figure 2.3.a. Recent Button, File Selection Window

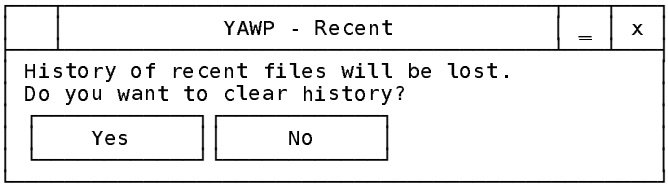


Figure 2.3.b. Recent Button, Clear Confirmation Window

2.4. COPY BUTTON

Copy the current text file into a new target text file:

- the current text file must exist and not be locked
- starting from the directory containing the current text file, browse the file system and click the Save button to select a target text file:
  - the target file may and may not exist, but if it already exists:
    - you are asked for confirmation
    - the new file can't be the current text file and must not be locked
  - copy current text file into target text file
  - copy argument file of current text file into argument file of target text file
  - the old current text file and its reserved files are not altered
  - the target text file becomes the new current text file, with no log file and no backup files
- otherwise press the Cancel button to go back without doing anything

WARNING: PDF files previously exported from the copied text file are not copied.

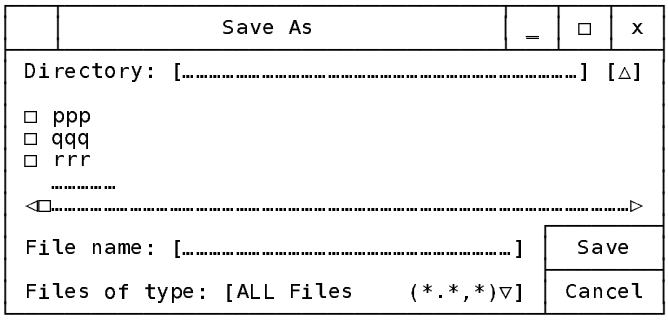


Figure 2.4.a. Copy Button, Target File Definition Window

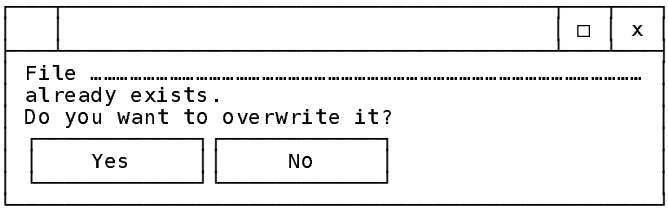


Figure 2.4.b. Copy Button, Overwriting Confirmation Window

2.5. MOVE BUTTON

Move current text file into a new target text file:

- the current text file must exist and not be locked
- starting from the directory containing the current text file, browse the file system and click the Save button to select a target text file:
  - the target file may and may not exist, if it already exists:
    - you are asked for confirmation
    - the new file can't be the current text file and must not be locked

        • move current text file to target text file
        • move argument log and backup files of current text file into argument log and backup files of target text file, if they exist they are overwritten
        • the old current text file and its argument log and backup files no longer exist
        • the target text file becomes the new current text file
    • otherwise press the Cancel button to go back without doing anything

WARNING: PDF files previously exported from the moved text file are not moved.

Figure 2.5.a. Move Button, Target File Definition Window

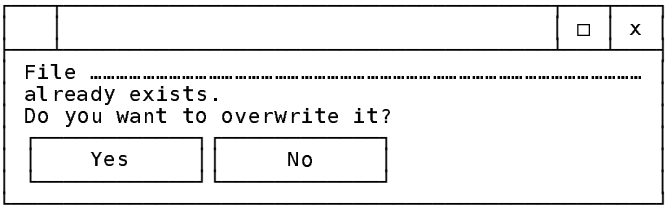


Figure 2.5.b. Move Button, Overwriting Confirmation Window

2.6. DELETE BUTTON

Permanently delete the text file with all its associated files:

    • the current text file may or may not exist, but if it does it must not be locked
    • you are asked for confirmation
    • delete the current text file (if exists) and all its argument log lock and backup files
    • set the current text file to undefined

WARNING: the Delete operation cannot be undone, the text file and its reserved files are lost forever.

WARNING: PDF files previously exported from the deleted text file are not deleted.

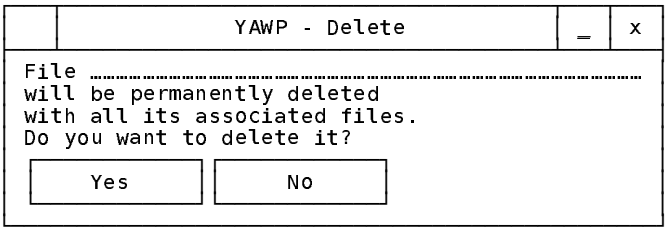


Figure 2.6.a. Delete Button, Delete Confirmation Window

2.7. EDIT BUTTON

Edit the current text file by the text editor defined by -y:

    • the current text file may and may not exist:
        • if it exists, it must not be locked
        • if it does not exist, create it (after confirmation) as an empty text file with default arguments
    • edit the current text file by the text editor defined by -y
    • if the text file has been altered, backup its original content into a timestamped copy (see 9.5.)

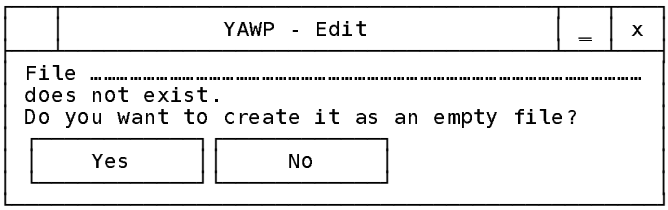


Figure 2.7.a. Edit Button, Creation Confirmation Window

2.8. FORMAT BUTTON

Format the current text file and redraw and align pictures and export PDF:

- the current text file to process must exist
- read the text file, eliminating in each line:
  - all horizontal tab '\t' characters, each replaced by four blanks
  - all zero width characters '\u200b', '\u200c' and '\u200d'
  - all trailing blanks
- format the text file content:
  - remove page headers and content of automatic chapters (Contents Index and Figures)
  - justify the text (see 4.1.)
  - if -p is not '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 6.)
  - perform automatic multi-level renumbering of chapters and refill (if needed) the automatic Contents chapter in the text file (see 5.3.1.)
  - recognize relevant subjects and refill (if needed) the automatic Index chapter in the text file (see 5.3.2.)
  - perform automatic multi-level renumbering of figure captions and refill (if needed) the automatic Figures chapter in the text file (see 5.3.3.)
- if -g is on, redraw horizontal and vertical segments and arrowheads by suitable Unicode graphic characters (see 7.1.)
- if -k ≠ 'n', align pictures at left/center/right (see 7.2.)
- if text has been altered:
  - save the previous content of the text file into a timestamped backup file (see 9.5.)
  - rewrite the text file
- if -X ≠ 'n':
  - export the text file into the PDF file (see 8.)
- if -X = 'b':
  - browse the PDF file by the PDF browser defined by -Y

2.9. NOFORM BUTTON

Don't format current text file but redraw and align pictures and export PDF:

- the current text file to process must exist
- read the text file, eliminating in each line:
  - all horizontal tab '\t' characters, each replaced by four blanks
  - all zero-width characters '\u200b', '\u200c' and '\u200d'
  - all trailing blanks
- don't not format the text file content
- if -g is on, redraw horizontal and vertical segments and arrowheads by suitable Unicode graphic characters (see 7.1.)
- if -k ≠ 'n', align pictures at left/center/right (see 7.2.)
- if text has been altered:
  - save the previous content of the text file into a timestamped backup file (see 9.5.)
  - rewrite the text file
- if -X ≠ 'n':
  - export the text file into the PDF file (see 8.)
- if -X = 'b':
  - browse the PDF file by the PDF browser defined by -Y

2.10. UNDO BUTTON

Restore the text file to its previous content and export PDF:

- the current text file to be restored may and may not exist, if it exists, ask for confirmation
- if confirmed, restore the current text file to its previous version from its most recent backup file
- if -X ≠ 'n':
  - export the text file into the PDF file (see 8.)
- if -X = 'b':
  - browse the PDF file by the PDF browser defined by -Y

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

Argument file of text file is not altered. For further details about backup files see 9.5.

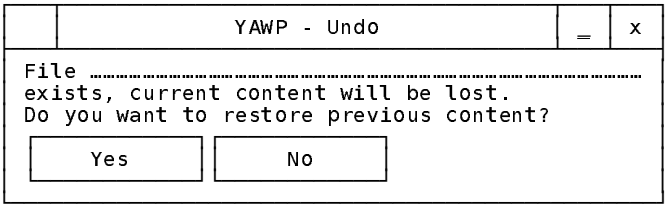


Figure 2.10.a. Undo Button, Restore Confirmation Window

2.11. LOG BUTTON

Browse the log file of the current text file by the text editor defined by -y.

Only Copy Move Delete Edit Format Noform Undo and Log actions are logged. For further details about log files see 9.6.

2.12. HELP BUTTON

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

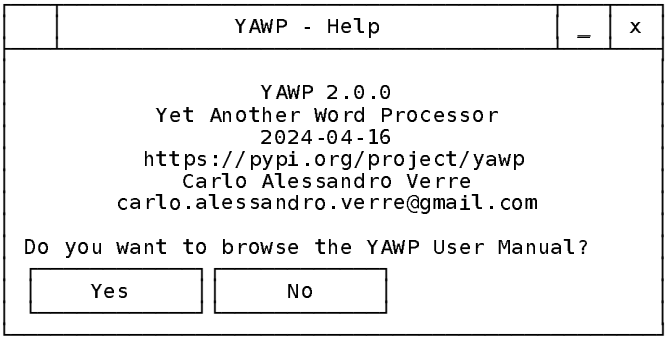


Figure 2.12.a. Help Button, Manual Confirmation Window

2.13. EXIT BUTTON

Quit YAWP:

- no risk of data loss, therefore no confirmation request
- save argument file of current text file
- unlock current text file
- write path of text file into session file
- close main window
- finish YAWP execution

You can get the same effect by clicking the 'x' button in main window's top right corner or by pressing Alt-F4 on keyboard.

2.14. INFORMATIONS WARNINGS AND ERRORS

In GUI mode, informations about text file processing and error/warning messages are written into the log file of text file. Nothing is written in either stdout or stderr.

An error is reported when something went wrong and you cannot proceed. An error window with one button appears, click the 'OK' button to interrupt processing, text file is not modified and control goes back to main window.

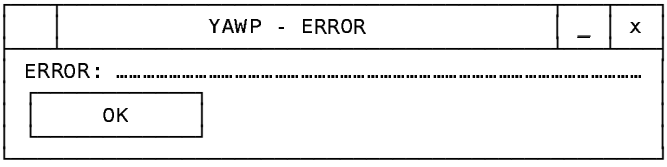


Figure 2.14.a. Error Window

A warning is reported when something went wrong but the user can choose whether to proceed or not. A warning window with two buttons appears, click:

- the 'Yes' button to continue processing (at your own risk)
- or the 'No' button to interrupt processing, text file is not modified and control goes back to Main Window

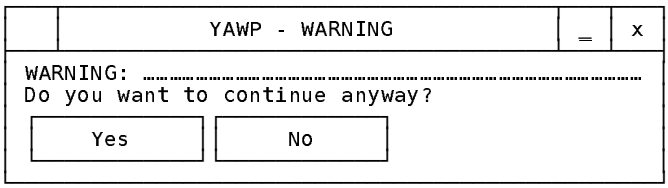


Figure 2.14.b. Warning Window

The program's return code when you quit YAWP is always 0.

3. "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 argument file is not used. The 7 CLI modes are analogous to the corresponding GUI buttons, but without any confirmation request.

3.1. COPY MODE

```
$ yawp -M c […arguments…] text_file target_file
```

Copy text_file into target_file, together with its argument file:

- text_file must exist and not be locked
- target_file may or may not exist:
  - if it exists, it must not be locked and it is silently overwritten

3.2. MOVE MODE

```
$ yawp -M s […arguments…] text_file target_file
```

Move text_file into target_file, together with its argument log and backup files:

- text_file must exist and not be locked
- target_file may or may not exist:
  - if it exists, it must not be locked and it is silently overwritten

3.3. DELETE MODE

```
$ yawp -M d […arguments…] text_file
```

Permanently delete the text file with all its argument log lock and backup files:

- text_file may or may not exist:
  - if it exists, it must not be locked and it is silently deleted

3.4. EDIT MODE

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

Edit the text file by the text editor defined by -y:

- text_file may or may not exist:
  - if it exists, it must not be locked
  - if it does not exist, it is created as an empty file with default arguments

3.5. FORMAT MODE

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

Format the current text file and redraw and align pictures and export PDF:

- text_file must exist and not be locked

3.6. NOFORM MODE

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

Don't format the current text file but redraw and align pictures and export PDF:

- text_file must exist and not be locked

3.7. UNDO MODE

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

Restore the text file to its previous content and export PDF:

- text_file to be restored may and may not exist, anyway it is silently restored from its most recent backup file

3.8. INFORMATIONS AND ERRORS

In CLI mode, informations about text file processing and error messages are written to the log file of current text file, furthermore:

- error messages are also written to stderr (always)
- information messages are also written to stdout (as long as -v is on)

In CLI mode, warnings don't exist, any error is a fatal error and ends the execution, the text file is not altered. The program's return code at processing end is:

- 0 on normal completion
- 1 in case of error

4. TEXT AND PICTURES

4.1. ALGORITHM

As said before, in Format mode each "page header line" inserted by YAWP 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 file text rewriting)
  - a blank ' '
  - a non-blank character
- 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

Position of the dot character in a dot line must not exceed the half of -w (chars per line argument).

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

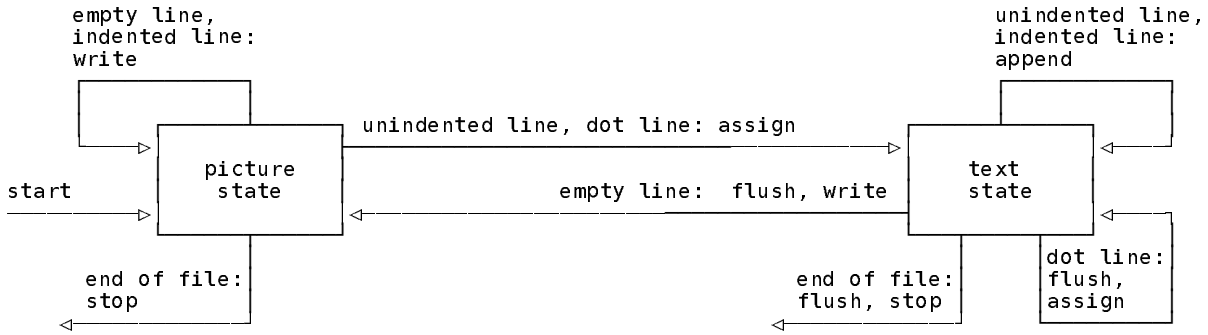


Figure 4.1.a. Formatting State Diagram

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
- 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 character plus two
- end of input file: paragraph buffer is flushed, processing is ended

Max number of characters in a line is controlled by:

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

-w must be an unsigned integer. When a paragraph is flushed, it is sliced into lines no longer than -w, then spaces in each line are left-to-right doubled until the desired length is reached. To avoid such a right-alignment, turn on:

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

If -w -u and -W arguments are all zero hence automatic, -w is taken from max text line length in text file, page header lines are not considered. Then, -w -u and -W interact with each other in various ways, depending on which of them is zero and which is not, see 8.1. for details.

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 e banner.txt; cat banner.txt
BANNER
```

Figure 4.1.b. banner.txt Example, -M = 'n', -w = '1'

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

4.2. BEST PRACTICES

Now we can define some 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 indented paragraphs are not implemented, but they can be created by hand:

- 1. first…
- 2. second…
- 3. third…

You can not get bold letters, but as a substitute you can use CAPITAL letters.

5. CHAPTERS

The text file can be partitioned in chapters by chapter lines. Each "chapter line" must be a one-line unindented text paragraph, in other words:

- 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

Each chapter except the first one is preceded by a chapter line, so the total number of chapters is equal to the number of chapter lines plus one. 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 by Format action:
  - the Contents chapter, the list of chapters
  - the Index chapter, the list of subjects
  - the Figures chapter, the list of figure captions

CHAPTER	LEVEL	AUTOMATIC	HOW MANY	WHERE	CHAPTER LINE	IN CONTENTS	FORM FEED	SEE
Nameless	= 0	no	max one	file top	no	no	no	5.1.
Numbered	= 1	no	no limit	everywhere	yes	yes	yes	5.2.
Numbered	> 1	no	no limit	everywhere	yes	yes	no	5.2.
Contents	= 1	yes	max one	everywhere	yes	no	yes	5.3.1.
Index	= 1	yes	max one	everywhere	yes	yes	yes	5.3.2.
Figures	= 1	yes	max one	everywhere	yes	yes	yes	5.3.3.

Figure 5.a. Chapter Types

Nameless chapter is level-0, level of numbered chapters can be 1 or more, automatic chapters are level-1.

Each kind of chapter, 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.

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

All level-1 chapters trigger a form feed if -p is 'c' or 'd'.

5.1. NAMELESS CHAPTER

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.

5.2. 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 '0'…'9'
- one 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
- 1.345. A Level-2 Numbered Chapter
- 1.3.5. A Level-3 Numbered Chapter

Numbered chapter lines must follow a couple of quite obvious sequence rules:

- first Numbered chapter in the text file and all Numbered chapters next to an automatic chapter must be level-1
- 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 6.)

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 it, they will be automatically renumbered.

Level-1 chapter numbers can be incremented by:

- "-m", "--chapter-offset": offset of level-1 numbered chapters (default: 0, min: -1)

This can be useful when the text file is a chapter of a larger document (see also -n).

If you want chapters numbered starting from 0, -m can be -1, but not less, because negative chapter numbers are not allowed.

Example:

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

32.33. bbb bbb bbb

0.0. ccc ccc ccc

0. ddd ddd ddd

0.1000. eee eee eee

9.9.9. fff fff fff
```

```
$ yawp -M f -w 40 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
```

```
$ yawp -M f -w 40 -m -1 chapters.txt; cat chapters.txt
0. AAA AAA AAA

0.1. BBB BBB BBB

0.2. CCC CCC CCC

1. DDD DDD DDD

1.1. EEE EEE EEE

1.1.1. FFF FFF FFF
```

```
$ yawp -M f -w 40 -m 7777 chapters.txt; cat chapters.txt
7778. AAA AAA AAA

7778.1. BBB BBB BBB

7778.2. CCC CCC CCC

7779. DDD DDD DDD

7779.1. EEE EEE EEE

7779.1.1. FFF FFF FFF
```

Figure 5.2.a. chapters.txt Example

5.3. AUTOMATIC CHAPTERS

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

- the text 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

When -p is 'n' (see 6.), page headers are not inserted, pages are not numbered, and the entire file is treated as a single page. So, if an automatic chapter is requested, it will still appear, but without page numbers aside.

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 action.

5.3.1. 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 6.)

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

0.1000. eee eee eee

9.9.9. fff fff fff
```

```
$ yawp -M f -w 40 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

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
```

Figure 5.3.1.a. contents.txt Example

Another example of Contents chapter is at page ii of this Manual.

5.3.2. 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 '"' or "'" 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 6.)

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 in Index chapter is too long, it's truncated and terminated by an ellipsis '…'.

An example of Index chapter is at the end of this Manual.

5.3.3. 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 6.)

YAWP recognizes a figure caption by -F:

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

```
$ yawp -F 'image' …
```

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

A line is a "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 'a'…'z', meaning caption position in containing chapter
    - a decimal point '.'
  - the following words, if any, are the caption title

Examples of captions lines are:

- Figure a. The First Caption In The Nameless Chapter
- Figure 7.c. The Third Caption In Chapter 7.
- Figure 7.9.b. The Second Caption In Chapter 7.9.
- Figure 7.9.8.z. The 26th Caption In Chapter 7.9.8.

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

The "level" of a caption line is the count of decimal points in its label, hence is equal to the level of the containing chapter plus one.

The input caption label must be a correct one but can be any, of any level, it will be 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

An example of Figures chapter is at the end of this Manual.

6. PAGES

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 chapters, 'd'=double if level-1 on even page)

So:

- with '-p n', page headers are not inserted, hence if you want to eliminate page headers from your file, just run Format action with '-p n'.
- with '-p f', page headers are inserted when current line does not fit into current page
- with '-p p', page headers are inserted as above and when current picture does not fit into current page
- with '-p c', page headers are inserted as above and when current line is a level-1 or automatic chapter line
- with '-p d', page headers are inserted as above and when a level-1 or automatic chapter happens to start on an even page, in such a case an intentionally blank page is inserted in order to get all level-1 or automatic chapters starting on an odd page

If -p is 'p' 'c' or 'd', when a picture is too long to fit into the current page, a page eject is forced. Two consecutive pictures, separated by an empty line, may be split onto two consecutive pages. But if a single picture is higher than one page, such a page eject is useless and does not take place.

Max number of lines in a page (including one or two lines of the page header, if any) is controlled by:

- "-u", "--lines-per-page": page height in lines per page (default: 0=automatic)

-u must be an unsigned integer.

If -w -u and -W arguments are all zero hence automatic, -w is taken from max text line length in text file, page header lines are not considered. Then, -w -u and -W interact with each other in various ways, depending on which of them is zero and which is not, see 8.1. for details.

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 of one or two header lines:

- the first header line, starting with a form feed '\f' character, contains data such as file name, chapter title, date, time or page number, under control of -e -E -o -O -n and -a arguments
- the second header line is optional, when it is present, it is a kind of underline under the first header line, as defined by -s argument

Page footers are not implemented. 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 text editor, it can be invisible or appear as a white rectangle '▯' or a white square '□' with 'FF' or '000c' written inside or as a quarter note '♩'.

WARNING: never start a text line with a form feed or no-break space or dot above or macron or overline character, after a YAWP run in Format mode such a line would disappear.

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

- "-e", "--even-left": first line of headers of even pages, left (default: '%n')
- "-E", "--even-right": first line of headers of even pages, right (default: '%f')
- "-o", "--odd-left": first line of headers of odd pages, left (default: '%c')
- "-O", "--odd-right": first line of headers of odd pages, right (default: '%n')

Each "percent variable" in the first page header line is evaluated as follows, no other percent variable is allowed.

%VAR	VALUE
'%%'	'%'
'%i'	current process PID
'%P'	text file long path, with final '/'
'%p'	text file short path, with final '/'
'%f'	text file name, without extension
'%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
'%u'	current microsecond, 6 digits
'%n'	current page number
'%N'	total number of pages
'%c'	current level-1/Contents/Figures/Index chapter

Figure 6.a. Percent Variables

If for instance the current text file is:

/home/user/yyyy/zzz.www.txt

then we get:

- '%P' → '/home/user/yyyy/'
- '%p' → '~/yyyy/'
- '%f' → 'zzz.www'
- '%e' → '.txt'
- '%P%f%e' → '/home/user/yyyy/zzz.www.txt'
- '%p%f%e' → '~/yyyy/zzz.www.txt'

Actual value of '%n' is affected by:

- "-n", "--page-offset": offset of page numbers (default: 0, if negative: 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 6.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 a chapter of a larger document (see also -m)
- 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 can get any negative value, but Roman numbers greater than 'mmmcmxcix' ≡ 3999 are not allowed.

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

If you don't 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 also 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 remain in place on both even and odd pages

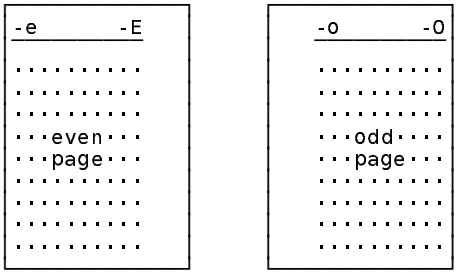


Figure 6.c. -L > -R, -a Off

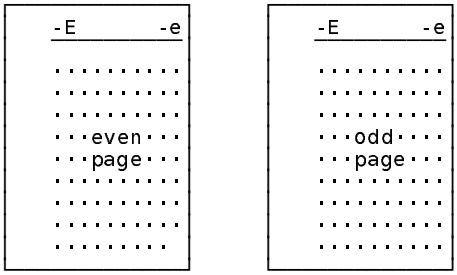


Figure 6.d. -L > -R, -a On

Of course if -n is odd, the Arabic %n is odd on even pages, and even on odd pages.

Second lines of page headers are controlled by:

- "-s", "--second-line": second line of page headers (default: 's'=solid, 'n'=no, 'b'=blanks, 'p'=points, 'd'=dashes)

So:

- if -s = 'n', the second line is not written
- if -s = 'b', the second line is a blank line, made of no-break space characters: ' '
- if -s = 'p', the second line is a dotted line, made of dot above characters: '˙˙˙˙˙˙'
- if -s = 'd', the second line is a dashed line, made of macron characters: '¯¯¯¯¯¯'
- if -s = 's', the second line is a solid line, made of overline characters: '‾‾‾‾‾‾'

7. PICTURE PROCESSING

In Format and Noform operation, you can modify your pictures by -g and -k arguments, but remember:

- in Format operation, -g and -k act separately on each picture and don't touch text or chapter titles or figure captions
- in Noform operation, the entire text file is treated as a single picture, hence -g and -k act on the entire text file as a single whole

7.1. GRAPHICS

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

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

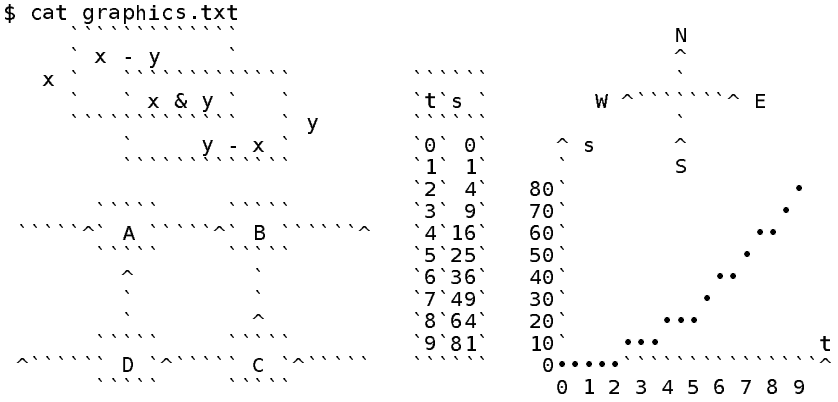
If you turn on the argument:

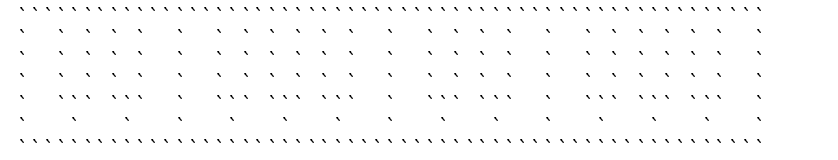
- "-g", "--graph-pictures": redraw '`'-segments and '^'-arrowheads

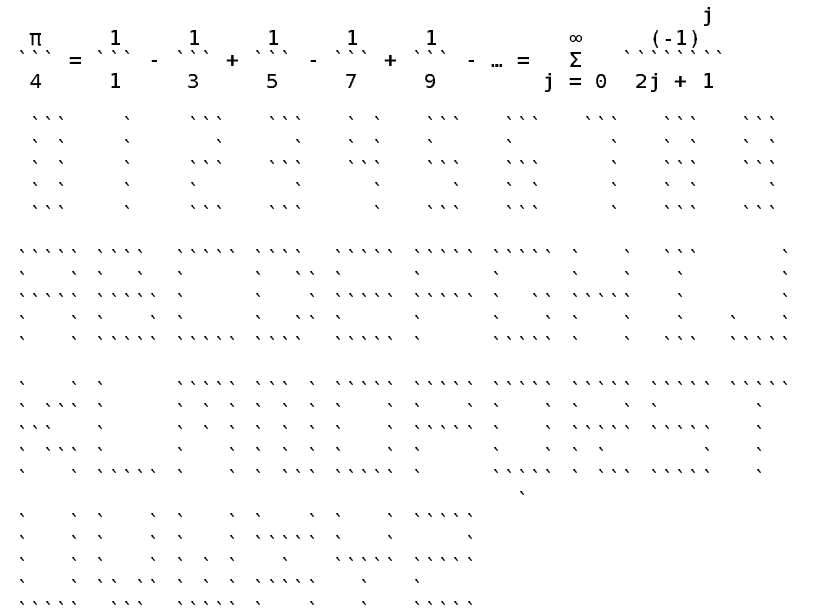
then YAWP will redraw the pictures by replacing the draw characters with suitable graphic ones, depending on the other four characters around (left right above and below).

An example:

```
$ cat graphics.txt
```







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

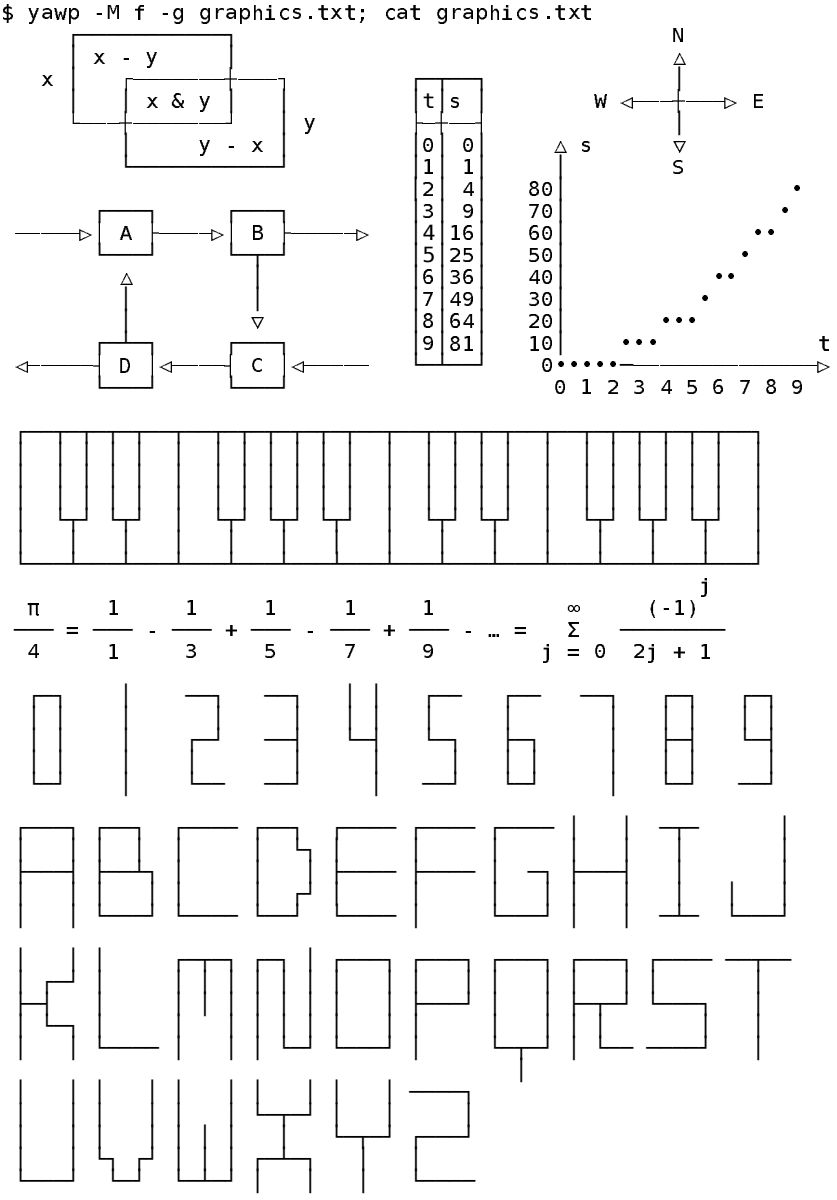


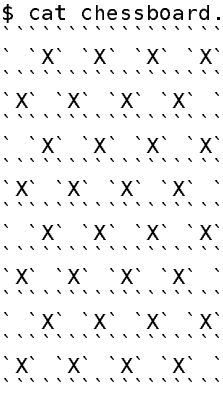
Figure 7.1.a. graphics.txt Example, -g On

All lines in graphics.txt are indented lines, hence the whole file is a single picture.

Argument -g works with '-M n' too, so you can redraw without the need for a blank at beginning of the lines, and without YAWP trying to format the file. We can say that in Noform 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.txt.pdf, so by printing it on an A4 sheet you get a square chessboard, about 18x18cm or 7x7in

```
$ cat chessboard.txt
```



```
$ yawp -M n -g -A 1 -X e chessboard.txt; cat chessboard.txt
```

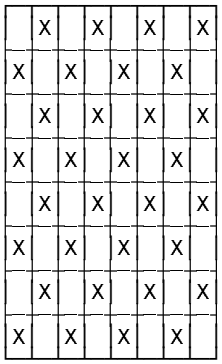


Figure 7.1.b. chessboard.txt Example, -g On, -M = 'n', -A = 1

7.2. ALIGNMENT

Pictures in text file can be shifted left and right, into the page width defined by -w:

- "-k", "--align-pictures": align pictures (default: 'n'=no, 'l'=left, 'c'=center, 'r'=right)

So:

- with '-k n' pictures are not shifted
- with '-k l' pictures are left-aligned, but keeping one space at the beginning of each line, so pictures remain pictures even after subsequent processing
- with '-k c' pictures are centered
- with '-k r' pictures are right-aligned

See the following example:

```
$ cat align.txt
```

```
                      ``````
                      `    `
                      `    `
                      ``````

   figure x. little square

     ````````````````
     `              `
     `              `
     `              `
     `              `
     `              `
     `              `
     `              `
     `              `
     ````````````````

               figure x. big square
```

```
$ yawp $1 -M f -w 40 -g -k l align.txt; cat align.txt
```

Figure a. Little Square

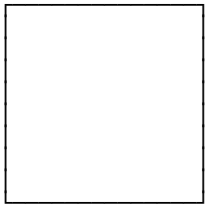


Figure b. Big Square

Figure 7.2.a. align.txt Example, -g On, -k = 'l'

```
$ yawp $1 -M f -w 40 -g -k c align.txt; $ cat align.txt
```

Figure a. Little Square

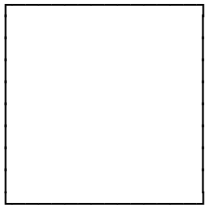


Figure b. Big Square

Figure 7.2.b. align.txt Example, -g On, -k = 'c'

```
$ yawp $1 -M f -w 40 -g -k r align.txt; cat align.txt
```

Figure a. Little Square

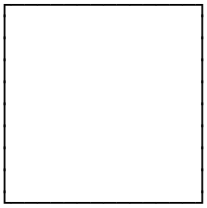


Figure b. Big Square

Figure 7.2.c. align.txt Example, -g On, -k = 'r'

As you can see, captions are always centered.

8. EXPORT

After Format Noform or Undo processing, the text file can be exported into PDF format, under control of:

- "-X", "--export-pdf": export and browse PDF file (default: 'n'=no, 'e'=export, 'b'=export and browse)

So:

- with '-X n', no PDF exporting takes place
- with '-X e', the text file is exported into a PDF file
- with '-X b', the text file is exported into a PDF file, and such a PDF file is browsed by the PDF browser defined by argument -Y

To define the PDF browser, set the argument:

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

Atril is the system standard PDF browser on Debian Linux. You can indicate your favorite PDF browser, such as okular evince foxit xpdf etc. Such a PDF browser is also used to browse the User Manual by -H argument or Help Button.

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

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

Percent variables in -P are evaluated as explained in figure 5.a., 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
- if path is not given, the directory containing the current text file is assumed

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

8.1. GEOMETRY

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

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

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

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

so for instance all these give the same value of one inch:

- -W 72pt
- -W 1.0in
- -W 25.4mm
- -W 2.54cm

Among various existing kinds of typographical point, YAWP's point is the so-called DTP point.

You can also use a 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.

Character proportions are defined by:

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

-A is a ratio, so it can be:

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

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

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

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

-S value can be a paper format name too, see the following table.

FORMAT NAME	VALUE	H/W	WxH(cm²)	WxH(in²)
HALF-LETTER	5.5x8.5in	1.5455	301.61	46.75
LETTER	8.5x11in	1.2941	603.22	93.50
LEGAL	8.5x14in	1.6471	767.74	119.00
JUNIOR-LEGAL	5x8in	1.6000	258.06	40.00
LEDGER	11x17in	1.5455	1206.45	187.00
TABLOID	11x17in	1.5455	1206.45	187.00
A0	841x1189mm	1.4138	9999.49	1549.92
A1	594x841mm	1.4158	4995.54	774.31
A2	420x594mm	1.4143	2494.80	386.69
A3	297x420mm	1.4141	1247.40	193.35
A4	210x297mm	1.4143	623.70	96.67
A5	148x210mm	1.4189	310.80	48.17
A6	105x148mm	1.4095	155.40	24.09
A7	74x105mm	1.4189	77.70	12.04
A8	52x74mm	1.4231	38.48	5.96
A9	37x52mm	1.4054	19.24	2.98
A10	26x37mm	1.4231	9.62	1.49
B0	1000x1414mm	1.4140	14140.00	2191.70
B1	707x1000mm	1.4144	7070.00	1095.85
B1+	720x1020mm	1.4167	7344.00	1138.32
B2	500x707mm	1.4140	3535.00	547.93
B2+	520x720mm	1.3846	3744.00	580.32
B3	353x500mm	1.4164	1765.00	273.58
B4	250x353mm	1.4120	882.50	136.79
B5	176x250mm	1.4205	440.00	68.20
B6	125x176mm	1.4080	220.00	34.10
B7	88x125mm	1.4205	110.00	17.05
B8	62x88mm	1.4194	54.56	8.46
B9	44x62mm	1.4091	27.28	4.23
B10	31x44mm	1.4194	13.64	2.11
C0	917x1297mm	1.4144	11893.49	1843.49
C1	648x917mm	1.4151	5942.16	921.04
C2	458x648mm	1.4148	2967.84	460.02
C3	324x458mm	1.4136	1483.92	230.01
C4	229x324mm	1.4148	741.96	115.00
C5	162x229mm	1.4136	370.98	57.50
C6	114x162mm	1.4211	184.68	28.63
C7	81x114mm	1.4074	92.34	14.31
C8	57x81mm	1.4211	46.17	7.16
C9	40x57mm	1.4250	22.80	3.53
C10	28x40mm	1.4286	11.20	1.74
C11	22x32mm	1.4545	7.04	1.09
C12	16x22mm	1.3750	3.52	0.55

$$H/W = \frac{height}{width}$$

WxH = width x height

Figure 8.1.a. Paper Format Names For -S

These names are case-insensitive ('A4' ≡ 'a4'), while the 'x' separator and the unit suffix in the explicit value must be lowercase.

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

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

Unprintable margins around the paper sheet are controlled by:

- "-L", "--left-margin": left margin (default: 2cm, unit: pt/in/mm/cm)
- "-R", "--right-margin": right margin (default: 2cm, unit: pt/in/mm/cm)
- "-T", "--top-margin": top margin (default: 2cm, unit: pt/in/mm/cm)
- "-B", "--bottom-margin": bottom margin (default: 2cm, unit: pt/in/mm/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 on even pages (see Figure 6.c.)
- if -a is on, left and right margins stay in place on odd and even pages (see Figure 6.d.)

If -w -u and -W arguments are all zero hence automatic, -w is taken from max text line length in text file, page header lines are not considered. Then, -w -u and -W interact with each other in various ways, depending on which of them is zero and which is not, so we have $2^3 = 8$ possible cases, as shown by the following Python pseudocode.

```
def automatic_wuW(w, u, W):
    # usage: w, u, W = automatic_wuW(w, u, W)
    # w:  line length in chars (from -w argument)
    # u:  page height in lines (from -u argument)
    # W:  char width in inches (from -W argument)
    # A:  char aspect ratio = width / height (from -A argument)
    # Sw: paper sheet width in inches (from -S argument)
    # Sh: paper sheet height in inches (from -S argument)
    # L:  left margin in inches (from -L argument)
    # R:  right margin in inches (from -R argument)
    # T:  top margin in inches (from -T argument)
    # B:  bottom margin in inches (from -B argument)
    pw = Sw - L - R # print width in inches
    ph = Sh - T - B # print height in inches
    def w2W(w): return pw / w          # convert w to W
    def u2W(u): return ph * A / u      # convert u to W
    def W2w(W): return int(pw / W)     # convert W to w
    def W2u(W): return int(ph * A / W) # convert W to u
    if W == 0.0 and u == 0 and w == 0:
        w = max_text_line_length_in(text_file)
        W = w2W(w)
        u = W2u(W)
    elif W == 0.0 and u == 0 and w > 0:
        W = w2W(w)
        u = W2u(W)
    elif W == 0.0 and u > 0 and w == 0:
        W = u2W(u)
        w = W2w(W)
    elif W == 0.0 and u > 0 and w > 0:
        W = min(w2W(w), u2W(u))
    elif W > 0.0 and u == 0 and w == 0:
        w = W2w(W)
        u = W2u(W)
    elif W > 0.0 and u == 0 and w > 0:
        W = min(W, w2W(w))
        u = W2u(W)
    elif W > 0.0 and u > 0 and w == 0:
        W = min(W, u2W(u))
        w = W2w(W)
    else: # W > 0.0 and u > 0 and w > 0
        W = min(W, w2W(w), u2W(u))
    return (w, u, W)
```

Figure 8.1.b. Automatic -w -u And -W

8.2. MULTIPLE PAGES

You can print more than one page on each paper side, under control of two arguments:

- "-I", "--multi-pages": pages on each side of paper sheets (default: 1, values: 1/2/4/8)
- "-J", "--multi-sheets": paper sheets gathered together (default: 0=export sequentially)

-I > 1 is not allowed when -p is 'n'. -J must be an unsigned integer. If -J = 0, the pages are exported in sequential order. Otherwise, the pages are shuffled and possibly overturned so that each odd page has its even counterpart on the other side of the sheet, and we need to fold and cut the paper sheet before to bind it. In such a case -J defines how many sheets are to be inserted within each other. If -J is too high, it is automatically reduced to the minimum necessary to accommodate all pages.

In the following we suppose:

- the PDF file is printed double-sided
- the PDF file is to be read left-to-right
- leaves are to be bound on the left edge, in both portrait and landscape orientation

Note that:

- with '-I 2', each leaf is 1/2 of the original size, reduction factor is √2, if for instance the paper sheet is A4, the resulting leaves are A5
- with '-I 4', each leaf is 1/4 of the original size, reduction factor is 2, if for instance the paper sheet is A4, the resulting leaves are A6
- with '-I 8', each leaf is 1/8 of the original size, reduction factor is 2√2, if for instance the paper sheet is A4, the resulting leaves are A7

Hence if -I > 1, you get better results when

$$\frac{\text{sheet-height}}{\text{sheet-width}} \approx \sqrt{2} = 1.4142\ldots$$

Pages with page number marked by an asterisk '*' are printed upside down.

Sheet layout depends on page orientation, portrait (-Z off) or landscape (-Z on), as follows.

8.2.1. PORTRAIT

If -I = 1 and -J ≥ 0 or -I > 1 and -J = 0, the pages are sequentially exported in their natural order.

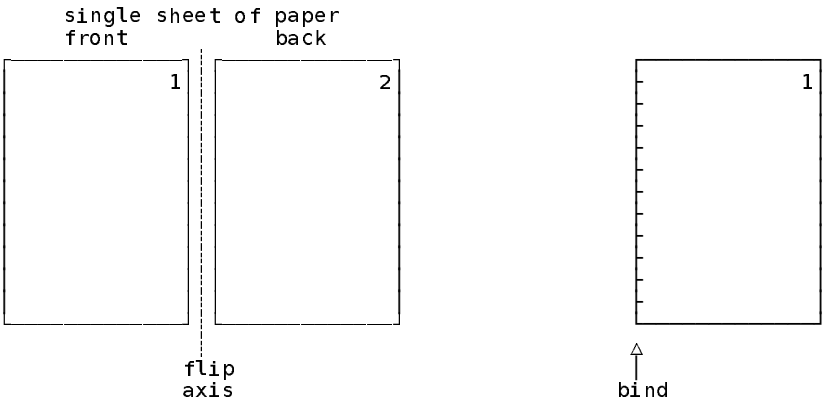


Figure 8.2.1.a. -Z Off, -I = 1, -J ≥ 0

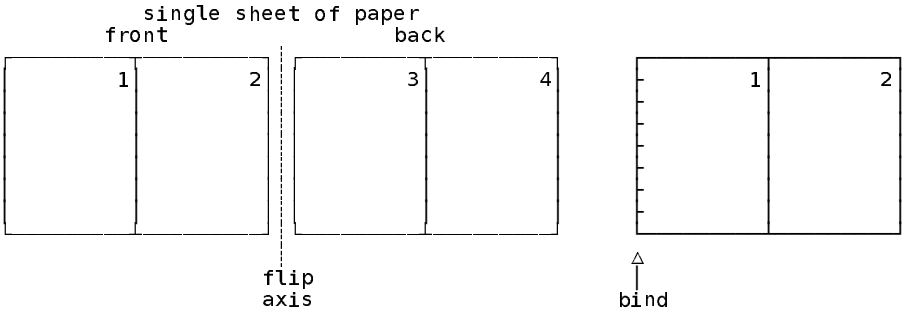


Figure 8.2.1.b. -Z Off, -I = 2, -J = 0

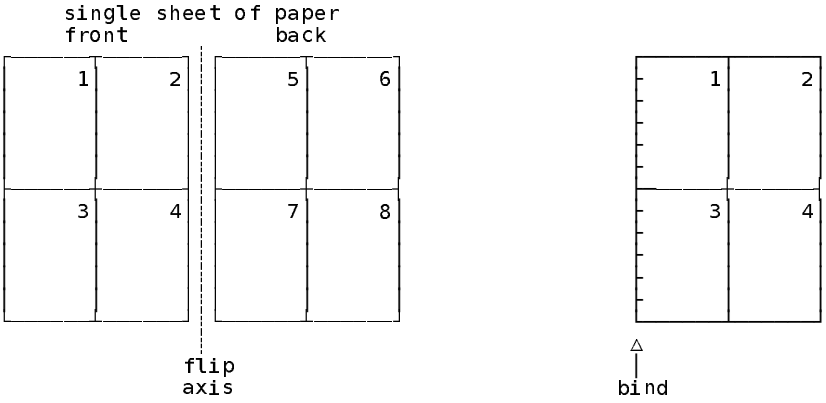


Figure 8.2.1.c. -Z Off, -I = 4, -J = 0

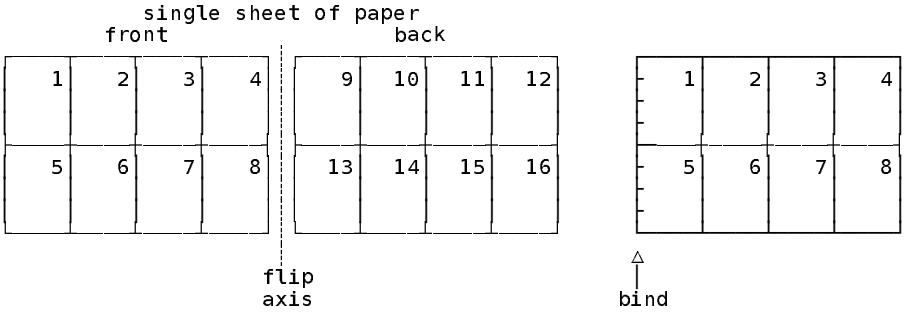


Figure 8.2.1.d. -Z Off, -I = 8, -J = 0

If -I > 1 and -J > 0, the pages are swapped and possibly overturned so that each odd page has its even counterpart on the other side of the sheet. If -I = 2 and -J = 1, you get 2 pages per side, the so-called "in folio" format. So from each paper sheet you get a booklet of 2 leaves and 4 pages.

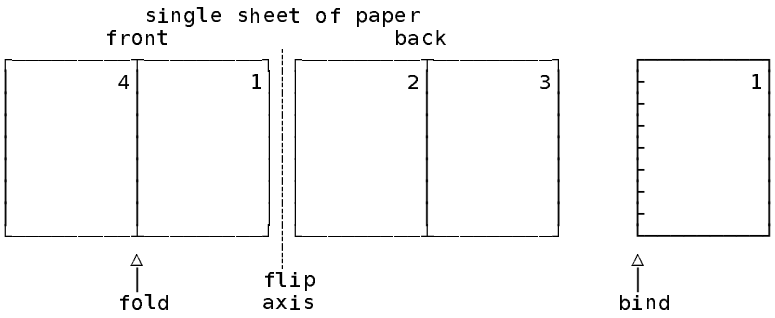


Figure 8.2.1.e. -Z Off, -I = 2, -J = 1

If -I = 2 and -J > 1, two or more folios can be inserted within each other. In the next example, from each couple of paper sheets you get a booklet of 4 leaves and 8 pages.

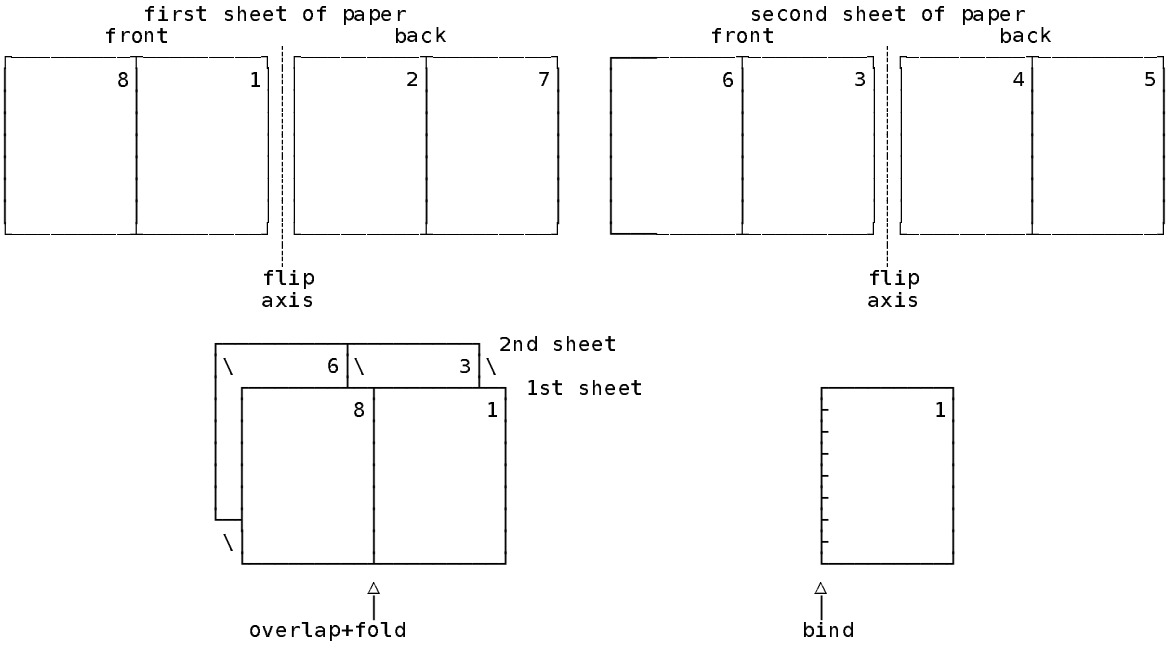


Figure 8.2.1.f. -Z Off, -I = 2, -J = 2

If you want all folios inserted within each other, set -J to a very high value. In the next example, the text file contains 20 pages, and we have 2 pages per sheet side, 2 * 2 = 4 pages per sheet. Hence we need 20 / 4 = 5 sheets, and -J is automatically reduced from 999 to 5. The result is a booklet of 5 sheets and 20 pages. If the number of pages had not been a multiple of 4, a convenient number of blank pages would have been appended.

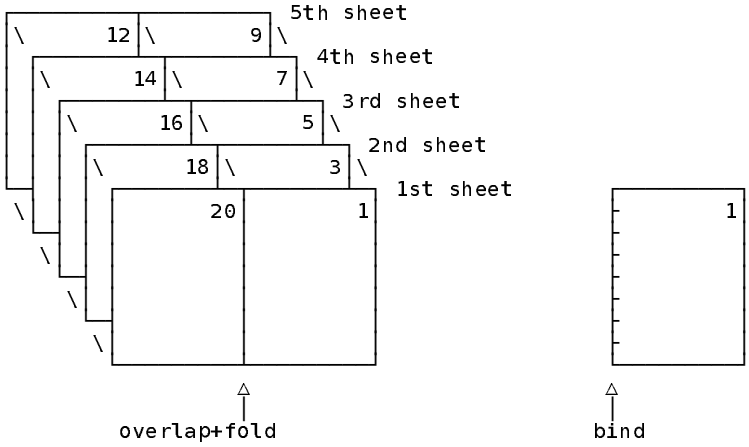


Figure 8.2.1.g. 20 pages, -Z Off, -I = 2, -J = 999

If -I = 4 and -J = 1, you get 4 pages per side, 8 pages per sheet, the so-called "in quarto" format. So from each paper sheet you get a booklet of 4 leaves and 8 pages.

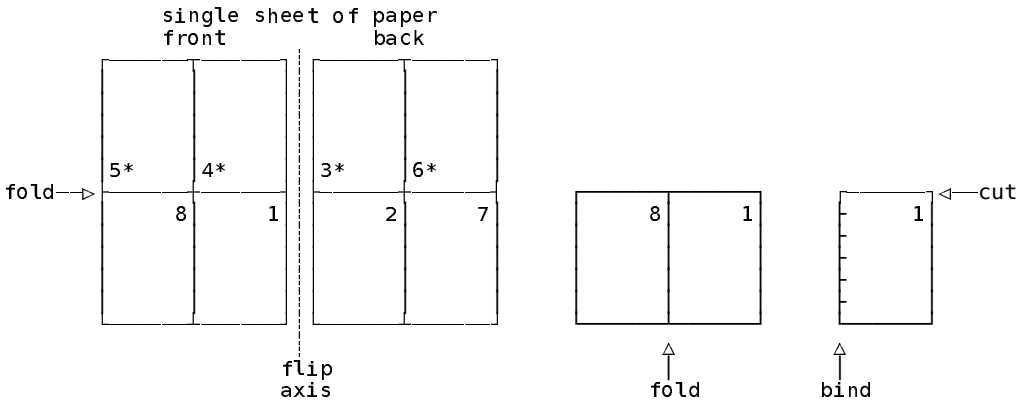


Figure 8.2.1.h. -Z Off, -I = 4, -J = 1

If -I = 4 and -J > 1, two or more of these booklets can be inserted within each other. In the next example, from each couple of paper sheets you get a booklet of 8 leaves and 16 pages.

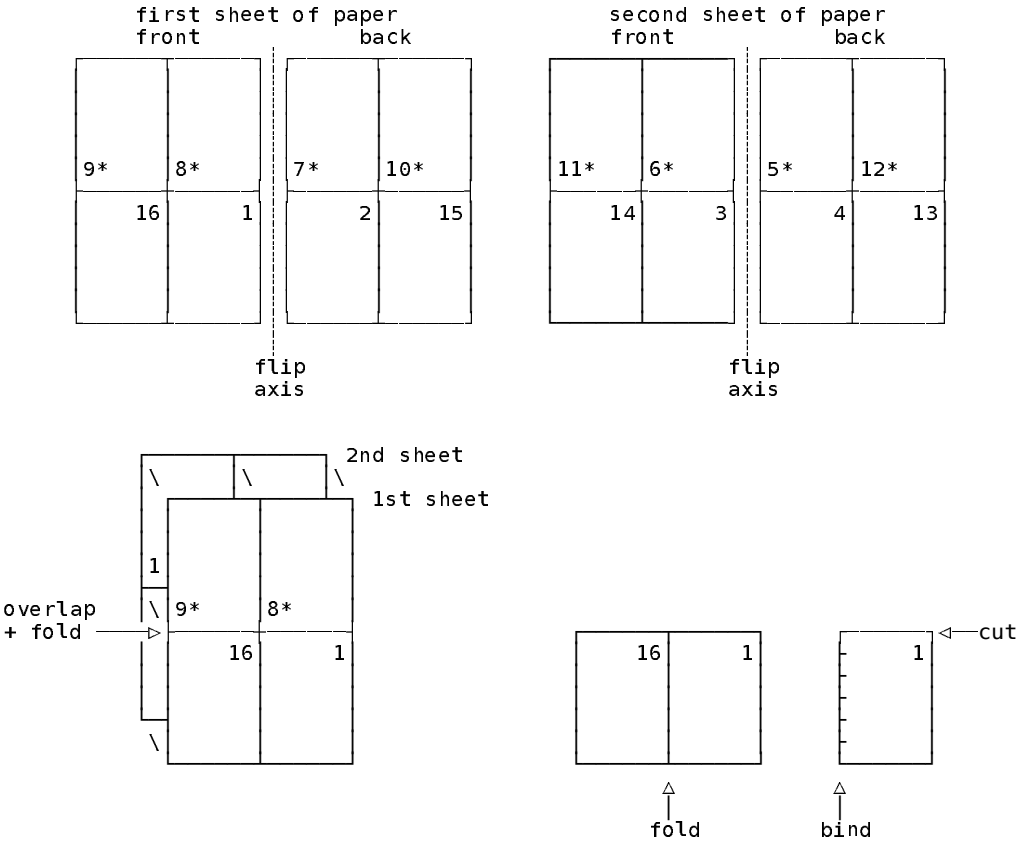


Figure 8.2.1.i. -Z Off, -I = 4, -J = 2

If -I = 8 and -J = 1, you get 8 pages per side, 16 pages per sheet, the so-called "in octavo" format. So from each paper sheet you get a booklet of 8 leaves and 16 pages.

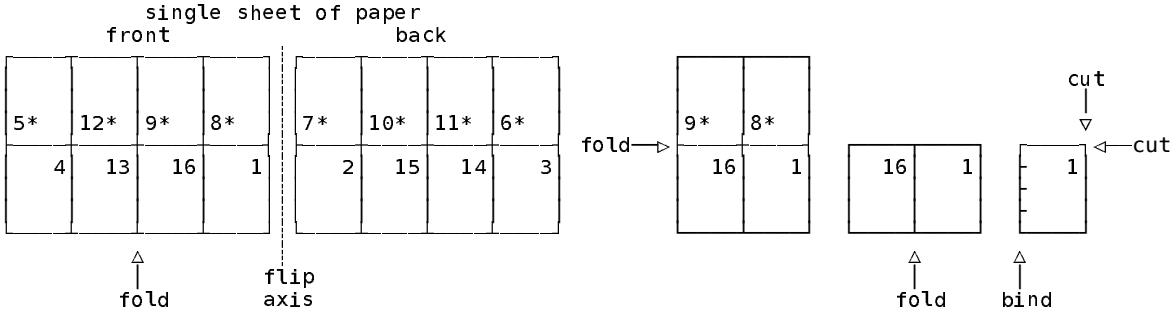


Figure 8.2.1.j. -Z Off, -I = 8, -J = 1

If -J > 1, two or more of these booklets can be inserted within each other. In the next example, from each couple of paper sheets you get a booklet of 16 leaves and 32 pages.

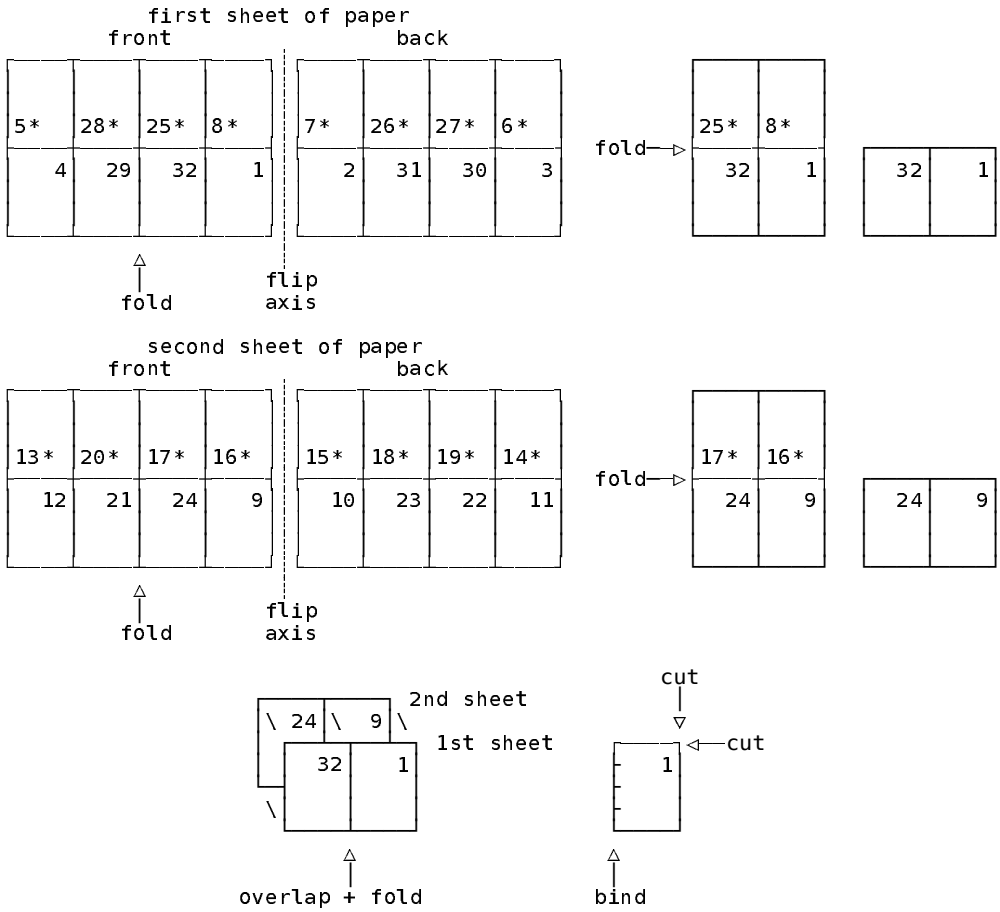


Figure 8.2.1.k. -Z Off, -I = 8, -J = 2

8.2.2. LANDSCAPE

If -I = 1 and -J ≥ 0 or -I > 1 and -J = 0, the pages are sequentially exported in their natural order.

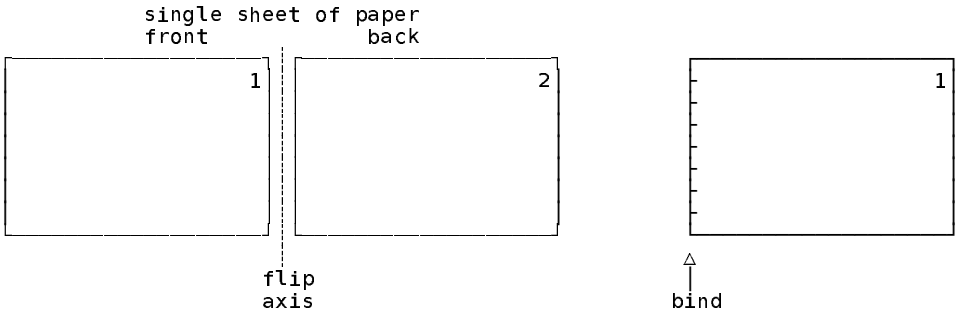


Figure 8.2.2.a. -Z On, -I = 1, -J ≥ 0

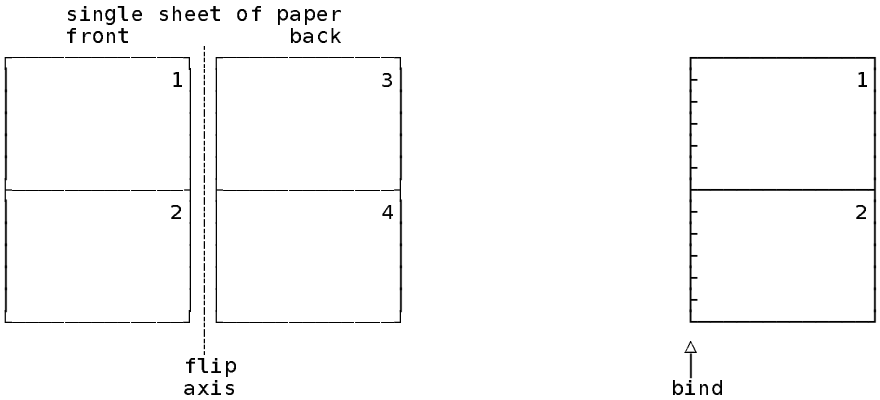


Figure 8.2.2.b. -Z On, -I = 2, -J = 0

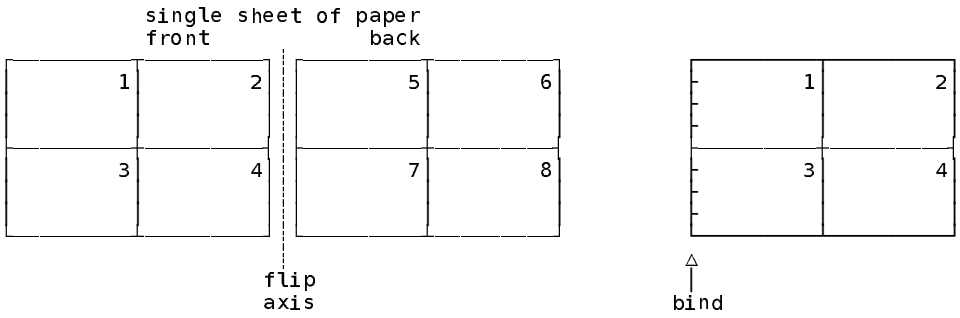


Figure 8.2.2.c. -Z On, -I = 4, -J = 0

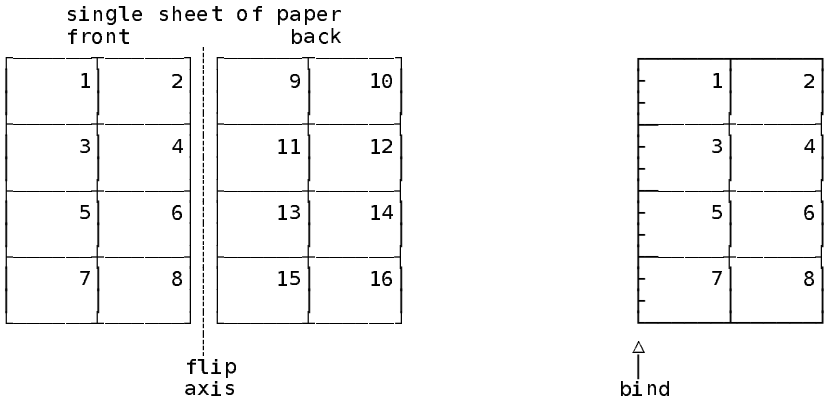


Figure 8.2.2.d. -Z On, -I = 8, -J = 0

If -I > 1 and -J > 0, the pages are swapped and possibly overturned so that each odd page has its even counterpart on the other side of the sheet. If -I = 2 and -J = 1, you get 2 pages per side. So from each paper sheet you get a booklet of 2 leaves and 4 pages. If -I = 2 and -J > 1, you get the same result.

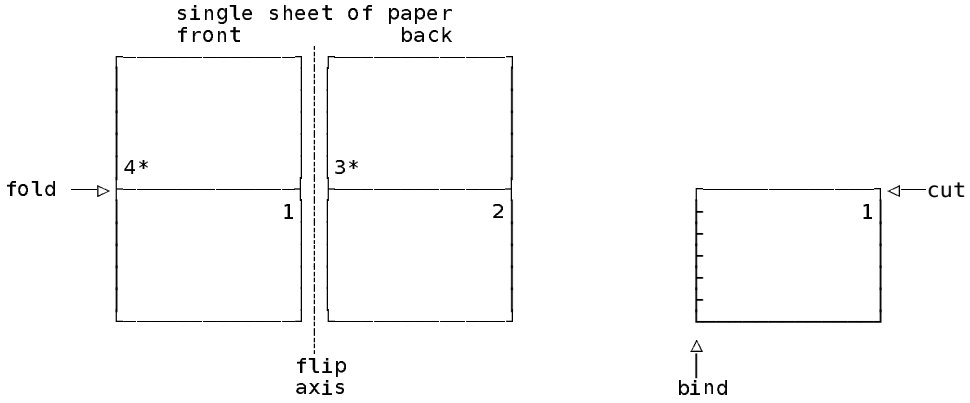


Figure 8.2.2.e. -Z On, -I = 2, -J ≥ 1

If -I = 4 and -J = 1, you get 4 pages per side, 8 pages per sheet. So from each paper sheet you get a booklet of 4 leaves and 8 pages.

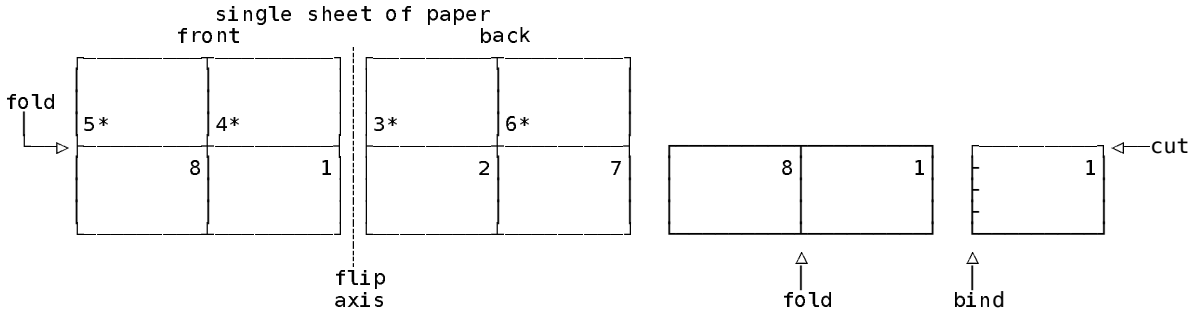


Figure 8.2.2.f. -Z On, -I = 4, -J = 1

If -I = 4 and -J > 1, two or more of these booklets can be inserted within each other. In the next example, from each couple of paper sheets you get a booklet of 8 leaves and 16 pages.

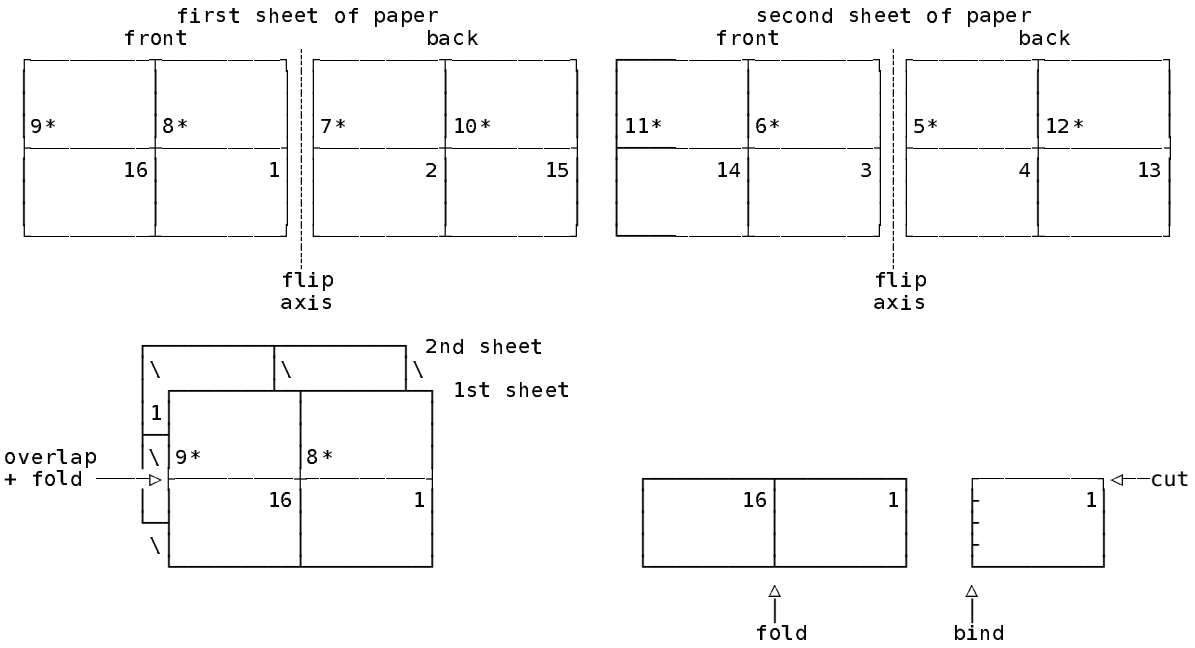


Figure 8.2.2.g. -Z On, -I = 4, -J = 2

If -I = 8 and -J = 1, you get 8 pages per side, 16 pages per sheet. So from each paper sheet you get a booklet of 8 leaves and 16 pages.

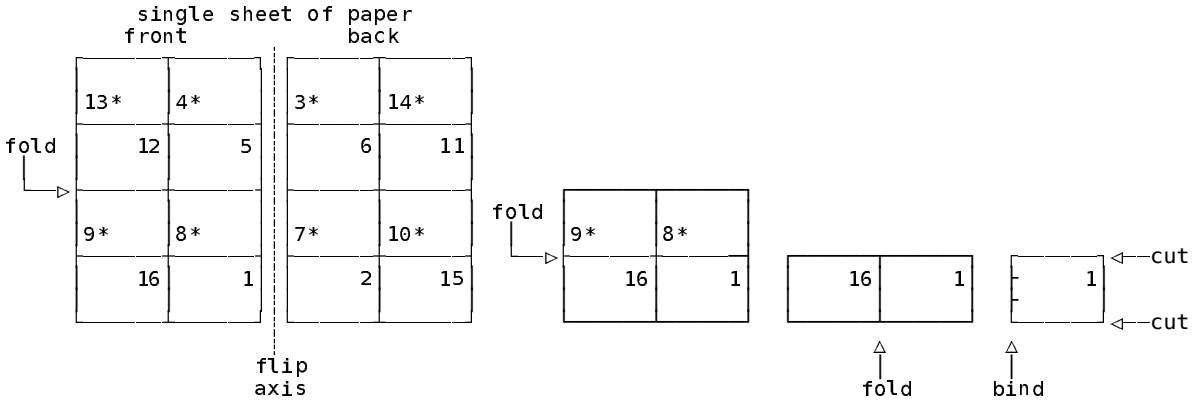
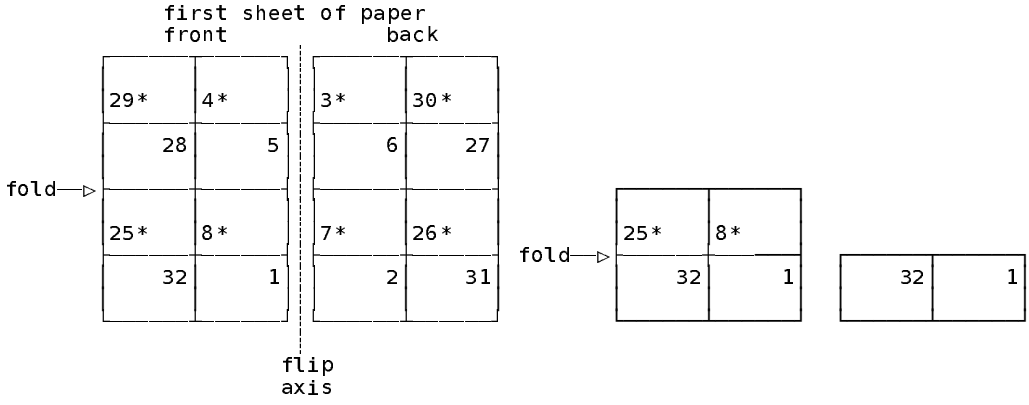


Figure 8.2.2.h. -Z On, -I = 8, -J = 1

If -I = 8 and -J > 1, two or more of these booklets can be inserted within each other. In the next example, from each couple of paper sheets you get a booklet of 16 leaves and 32 pages.



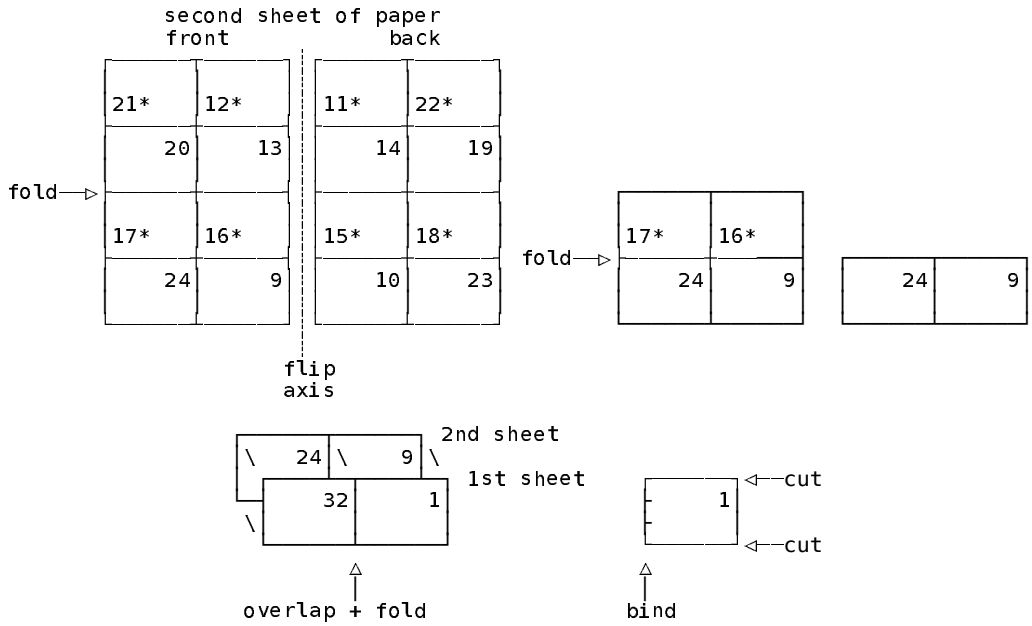


Figure 8.2.2.i. -Z On, -I = 8, -J = 2

9. RESERVED FILES

This chapter illustrates technicalities about the nine types of reserved files which support YAWP's processing.

FILE TYPE	FILE PATH AND NAME	GUI	CLI	SEE
session file	~/.config/yawp/yawp.sess	YES	NO	9.1.
history file	~/.config/yawp/yawp.hist	YES	NO	9.2.
correction file	~/.config/yawp/yawp.corr	YES	YES	9.3.
argument files	%P/.yawp.%f%e.args	YES	NO	9.4.
backup files	%P/.yawp.%f%e.%Y-%m-%d.%H-%M-%S.back	YES	YES	9.5.
log files	%P/.yawp.%f%e.log	YES	YES	9.6.
lock files	%P/.yawp.%f%e.lock	YES	YES	9.7.
temporary files	~/PDF/.yawp.%i.%Y-%m-%d.%H-%M-%S.%u.temp	YES	YES	9.8.
temp PDF files	~/PDF/.yawp.%i.%Y-%m-%d.%H-%M-%S.%u.temp*.pdf	YES	YES	9.9.

Figure 9.a. Reserved Files

All such files are text files except the PDF files.

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

Session, history and argument files are used in GUI mode but not in CLI modes.

Being text files, reserved files can be processed by YAWP itself, hence inspected by Edit action or printed by Noform action. Updating is deprecated, with the exception of the correction file. Format action on reserved files is never allowed.

9.1. 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:

```
~/.config/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 example.txt', current text file to process is taken from the argument
- if otherwise YAWP is invoked without text_file argument, as in 'yawp', current text file to process is taken from the previously saved session file (if such a file is not found, or points to a non-existent text file, the current text file remains undefined, and will need to be defined by New Open Recent or Copy buttons before being processed by Edit Format Noform or Undo buttons)

Of course, if you start multiple concurrent YAWP sessions, as in:

```
$ yawp one.txt & yawp two.txt
```

then the final content of the session file will be determined by the session you closed last.

In CLI modes the text_file argument is mandatory and the session file is not used.

To remove the session file, type:

```
$ rm ~/.config/yawp/yawp.sess
```

and after next argumentless YAWP GUI invocation the current text file will be undefined.

9.2. RECENT BUTTON AND HISTORY FILE

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

```
~/.config/yawp/yawp.hist
```

Such entries are managed avoiding duplications, in descending chronological order, until a max limit of 25.

By the Recent button you can access history in order to select the next text file to process, or clear the history file, see 2.3.

In CLI modes the history file is not used.

To remove the history file, type:

```
$ rm ~/.config/yawp/yawp.hist
```

and after next YAWP GUI invocation the list of recent files will be empty.

9.3. 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, among which:

- -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, hence corrections too 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 HP 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 '~/.config/yawp/yawp.corr', so:

- with '-C n', no correction takes place, this is useful to make experiments to fill the correction file
- with '-C d', corrections are performed by default values
- with '-C f', corrections are performed by values read from correction file

Default values have been detected with the default paper size 'A4' = '210x297mm', both portrait and landscape, on a HP Officejet 2620 printer. With another format or another printer you may get different results, and so you may need different corrector values. Being a text file, correction file can be processed by YAWP itself:

```
$ yawp ~/.config/yawp/yawp.corr
```

so it can be:

- edited by the Edit button
- printed by Noform button
- restored to its previous content by the Undo Button

As with all other reserved files, the Format action is not allowed.

Furthermore, if you remove the correction file:

```
$ rm ~/.config/yawp/yawp.corr
```

then it will be automatically reset to its default content at next YAWP invocation.

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

```
$ cat ~/.config/yawp/yawp.corr

#
#  ~/.config/yawp/yawp.corr
#

plm 0mm 6mm # 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 0mm 12mm # landscape left margin
llm 5mm 16mm
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 8.5mm # 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 0mm 12mm # landscape left margin
lrm 5mm 17mm
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 0mm 2mm # landscape top margin
ltm 5mm 7mm
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 0mm 14.5mm # landscape bottom margin
lbm 5mm 19mm
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 9.3.a. Correction File With Default Values

As you can see, standard Unix '#'-comments and empty lines are accepted. The file contains a set of correction points. 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 desired value in points inches millimeters or centimeters
    - x is an obtained value in points inches millimeters or centimeters

For instance, let's take the first correction point, 'plm 0mm 6mm'. This means that, by trying to print a page with -Z off, -C = 'n', and -L = '0mm' (y = 0mm), we have empirically obtained an actual measured left margin of 6mm on paper (x = 6mm). Therefore if we wish an actual measured left margin of 6mm on paper (x = 6mm), we should give to lp command an argument '-o page-left' equivalent to 0mm (y = 0mm). So, given a wanted left margin x and a set of corresponding correction points, when page orientation is portrait (-Z off), which is the function y = fplm(x) telling us what argument y provide to lp?

For each key, say 'plm', we can have zero, one, two or more correction points, namely:

- zero correction points, no correction:

  $$y = fplm(x) = x$$

- one correction point, say [(y0, x0)], correction function is given by the straight line passing by (0, 0) and (x0, y0):

  $$y = fplm(x) = x \frac{y0}{x0}$$

- two correction points, say [(y0, x0), (y1, x1)], correction function is given by the straight line passing by these two points:

  $$y = fplm(x) = y0 + (x - x0) \frac{y1 - y0}{x1 - x0}$$

- three or more correction points, say [(y0, x0), (y1, x1), (y2, x2), …], correction function y = fplm(x) is approximated by the least-squares line defined by points (x0, y0), (x1, y1), (x2, y2), …

Finally, if the result is less than zero:

- if we are correcting a margin, and we are in GUI mode, the user is notified that the requested margin cannot be obtained, and is asked whether to proceed or not, if answer is:
    - yes, the fplm(x) is set to zero and the processing continues
    - no, the processing is interrupted with an error message
- otherwise the processing is anyway interrupted with an error message

So with portrait orientation, -L will be replaced by fplm(-L).

To remove the correction file, type:

```
$ rm ~/.config/yawp/yawp.corr
```

and at next YAWP GUI invocation the correction file will be automatically created again with its default content.

9.4. ARGUMENTS AND ARGUMENT FILES

In GUI mode, if your text file is for example:

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

then the associated arguments are saved into:

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

The arguments are:

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

This ensures the argument continuity between GUI processing of a given text file and the next GUI processing of the same text file.

Only arguments with non-default values are written.

In CLI mode arguments are read from the CLI interface and the argument file is not used.

To remove all argument files from the current directory, type:

```
$ rm .yawp.*.args
```

and at next YAWP GUI invocation any text file in the current directory will get default values.

If you create a program writing a text file to be processed by yawp, the same program can write the associated argument file too. Such a file will contain an assignment statement per line, one for each non-default argument, as in the following example:

```
$ cat .yawp.example.txt.args
# argument file for example.txt

just_left_only = True # boolean argument, must not be quoted
chars_per_line = '97' # numeric argument, must be quoted
caption_prefix = 'Image' # string argument, must be quoted
```

Figure 9.4.a. Example Of Argument File

As you can see, '#'-comments and blank lines are accepted. Each assignment statement has three blank-separated parts:

- an internal argument name, it is the long argument name with leading '--' removed and internal '-' replaced by '_', example: '--just-left-only' → 'just_left_only'
- a '=' sign
- a value:
  - an unquoted value in the case of a boolean (checkbox) argument
  - a value quoted by "'" in all other cases

9.5. 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
```

The backup can take place after any operation (Edit/Format/Noform) likely to alter the text file. In order to avoid proliferation of useless backup files, the backup is performed only if the Edit/Format/Noform operation actually altered the content of the text file.

To remove all backup files from the current directory, type:

```
$ rm .yawp.*.back
```

and at next YAWP invocation an Undo action on a file in the current directory will result in an error.

9.6. MESSAGES AND LOG FILES

Simultaneous YAWP instances can process distinct text files at the same time, hence each text file deserves its distinct log file. If your text file is for example:

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

then the associated log file will be:

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

During execution YAWP can write two types of messages:

- error messages (starting with 'ERROR:') saying what's wrong during any action
- information messages (all the others) saying what's going on during Edit Format Noform and Undo actions

Both in CLI and GUI mode, all messages are written into log file. In GUI mode such a log file can be browsed by the Log button.

In GUI mode, no message is written into stdout or stderr. In case of error an error window appears, then execution continues. At execution end, the return code is always 0.

In CLI mode, error messages are always written on stderr, while warning and information messages are written on stdout or not, depending on -v argument on or off. Return code is 0 on normal completion, 1 in case of error.

An example of YAWP run with -v set to on follows, output on terminal is equal to the content appended to log file.

```
$ yawp -M f -v -w 97 -n -4 -p c -m -1 -X e -L 3cm 'YAWP 2.0.0 User Manual.txt'
 Format - 2024-05-20 15:03:33
Non-default arguments:
    chapter_offset = '-1'
    chars_per_line = '97'
    export_pdf     = 'b'
    left_margin    = '3cm'
    page_headers   = 'c'
    page_offset    = '-4'
    pdf_file       = 'yawp/docs/%f%e.pdf'
    text_editor    = 'idle'
Read:
    YAWP ← '/home/user/YAWP 2.0.0 User Manual.txt'
Before:
    header1:   51 lines,    51 words,   4947 chars, max 97 chars per line, 52 pages
    header2:   51 lines,    51 words,   4947 chars, max 97 chars per line
    body:    3913 lines, 23921 words, 186757 chars, max 97 chars per line
    total:   4015 lines, 24023 words, 196651 chars, max 97 chars per line
Computations:
    print_width    = 453.543307pt = 6.299213in = 160.0mm = 16.0cm
    print_height   = 728.503937pt = 10.11811in = 257.0mm = 25.7cm
    chars_per_line = 97
    lines_per_page = 93
    char_width     = 4.675704pt = 0.06494in = 1.649485mm = 0.164948cm
    char_height    = 7.79284pt = 0.108234in = 2.749141mm = 0.274914cm
    chars_per_inch = 15.39875
    lines_per_inch = 9.23925
Backup:
    copy '/home/user/YAWP 2.0.0 User Manual.txt' →
         '/home/user/.yawp.YAWP 2.0.0 User Manual.txt.2024-05-20.15-03-33.back'
Rewrite:
    YAWP → '/home/user/YAWP 2.0.0 User Manual.txt'
After:
    header1:   51 lines,    51 words,   4947 chars, max 97 chars per line, 52 pages
    header2:   51 lines,    51 words,   4947 chars, max 97 chars per line
    body:    3913 lines, 23920 words, 186753 chars, max 97 chars per line
    total:   4015 lines, 24022 words, 196647 chars, max 97 chars per line
Corrections:
    left_margin    = 72.530191pt = 1.007364in = 25.58704mm = 2.558704cm
    right_margin   = 39.080132pt = 0.54278in = 13.786602mm = 1.37866cm
    top_margin     = 53.905646pt = 0.74869in = 19.016714mm = 1.901671cm
    bottom_margin  = 14.827379pt = 0.205936in = 5.23077mm = 0.523077cm
    char_width     = 4.938742pt = 0.068594in = 1.742278mm = 0.174228cm
    char_height    = 8.258802pt = 0.114706in = 2.913522mm = 0.291352cm
    chars_per_inch = 14.578613
    lines_per_inch = 8.717972
Export:
    '/home/user/YAWP 2.0.0 User Manual.txt' →
    '/home/user/yawp/docs/YAWP 2.0.0 User Manual.txt.pdf'
```

Figure 9.6.a. Example Of Verbose Output

This output has been written on terminal, thanks to -v argument, and appended to log file:

```
/home/user/pypi/yawp/.yawp.yawp.txt.log
```

To remove all log files from the current directory, type:

```
$ rm .yawp.*.log
```

and at next YAWP GUI invocation a Log action on a file in the current directory will show an empty log file.

9.7. 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 a text file is locked when the user invokes Edit Format Noform Undo or Log action, and remains locked while is the current one, hence until another text file takes its place, or until YAWP finishes

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

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

a temporary hidden file containing a session signature is created in the same directory:

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

In order to ensure uniqueness such a signature has the format '%i %Y-%m-%d %H:%M:%S.%u', example:

```
$ cat ~/yyyy/.yawp.example.txt.lock
32856 2024-05-20 12:26:58.055814
```

So each YAWP session can distinguish its own lock files from those of others, and 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.

The lock file is deleted when the locked text file is released. It should not't be necessary, but to remove all lock files from the current directory, type:

```
$ rm .yawp.*.lock
```

9.8. TEMPORARY FILES

The target PDF file is generated from the text file through an intermediate temporary file possibly containing blank pages at end, with format '~/PDF/.yawp.%i.%Y-%m-%d.%H-%M-%S.%u.temp', for example:

```
~/PDF/.yawp.32856.2024-05-20.12-26-58.055814.temp
```

To ensure uniqueness, the temporary filename contains the process PID and a microsecond-precision timestamp.

Such temporary files are removed when they are no longer needed. It should not be necessary, but to remove all temporary files, type:

```
$ rm ~/PDF/.yawp.*.temp
```

9.9. TEMPORARY "PDF" FILES

In order to build the target PDF file, one or two temporary PDF files are generated, with format '~/PDF/.yawp.%i.%Y-%m-%d.%H-%M-%S.%u.temp*.pdf', for example:

```
~/PDF/.yawp.32856.2024-05-20.12-26-58.055814.temp__ted_files-job_3183.pdf
```

Such temporary PDF files are removed when they are no longer needeed. It should not be necessary, but to remove all temporary PDF files, type:

```
$ rm ~/PDF/.yawp.*.pdf
```

10. AFTERWORDS

10.1. VERSIONS

- version 2.0.0 - 2024-05-20 - production/stable
  - GUI Saveas button, removed
  - GUI Correct button, removed
  - GUI Noformat button, renamed as Noform
  - GUI Copy button and CLI Copy mode, added
  - GUI Move button and CLI Move mode, added
  - GUI Delete button and CLI Delete mode, added
  - argument -v --verbose, removed from GUI main window
  - argument -K --fake-margins, removed
  - argument -u --lines-per-page, added
  - argument -k --align-pictures, added
  - argument -m --chapter-offset, added
  - argument -s --second-line, added
  - argument -I --multi-pages, added
  - argument -J --multi-sheets, added
  - argument -p --page-headers, option 'd' added
  - argument -X --export-pdf, options 'n' 'e' and 'b' added
  - status indicator field in GUI main window, added
  - removing of spurious empty pages in PDF files, added
  - empty arguments, allowed
  - treatment of undefined text files, redesigned
  - logic of log files, redesigned
  - logic of correction file, redesigned
  - logic of temporary files, redesigned
  - main window layout and window theme colors, changed for enhanced readability
  - max number of items in history of recent files, raised from 20 to 25
  - configuration path changed from '~/.yawp' to '~/.config/yawp'
  - session signature for locks, changed, now contains process id and microsecond-precision timestamp
  - PySimpleGUI version, stuck on 4.60.5 (because PySimpleGUI 5 requires a license)
  - default text editor, changed from IDLE to KWrite
  - bug: in New action, argument log and backup files (if any) of the new text file must be removed, fixed
  - bug: in Copy action, log and backup files (if any) of the new text file must be removed, fixed
  - bug: argument files are not properly managed, fixed
  - bug: lpr and lpq Unix commands are not present in every Linux Distribution, replaced respectively by lp command and by an internal Python function
  - bug: temporary PDF files must be removed, fixed
  - bug: 'yawp x.txt' does not record x.txt into history of recent files, fixed
  - some other minor quirks, fixed
  - 99% back-compatible with previous version (in CLI call convert '- X' into '-X b')
- version 1.0.0 - 2023-05-18 - production/stable - deprecated
- version 1.0.0b10 - 2023-05-18 - beta release - deprecated
  - GUI mode, added and set as default usage mode
  - CLI Edit mode, added
  - -n --page-offset argument, added
  - several other additions and changes, too many to list here
  - not compatible with previous versions
- version 0.7.1 - 2022-06-25 - experimental - deprecated
- version 0.6.1 - 2022-05-06 - experimental - deprecated
- version 0.5.4 - 2022-04-04 - experimental - deprecated
- version 0.5.3 - 2022-04-02 - experimental - deprecated
- version 0.5.2 - 2022-04-02 - experimental - deprecated
- version 0.5.1 - 2022-03-19 - experimental - deprecated
- version 0.4.2 - 2022-01-04 - experimental - deprecated
- version 0.4.1 - 2022-01-03 - first version on pypi.org - experimental - deprecated

10.2. DIMENSIONS

WHAT	LINES	WORDS	CHARS
Python code	3790	16930	160702
User Manual	4029	24248	252469
TOTAL	7819	41178	413171

Figure 10.2.a. Dimensions Of YAWP-2.0.0 Project

10.3. BUGS

Sometimes the second line of page headers is not rendered correctly in the exported PDF file, this seems to be due to a bug in printer-driver-cups-pdf.

Please report bugs and comments by an e-mail to:

carlo.alessandro.verre@gmail.com

10.4. 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 2.0.0 has been developed under Debian 12:

http://www.debian.org

by Python 3.11.2 (and KWrite 3.11.2, the Python's IDE):

https://www.python.org

but minimum required Python is 3.6.

YAWP has been published on PyPI by flit 3.9.0:

https://pypi.org/project/flit

The GUI interface has been implemented thanks to PySimpleGUI 4.60.5:

https://pypi.org/project/PySimpleGUI/4.60.5

4.60.5 has been the last published FOSS version of PySimpleGUI, hence YAWP does not use PySimpleGUI 5, which requires a license.

YAWP uses printer-driver-cups-pdf 3.0.1-14 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

Default YAWP's text editor is KWrite:

https://apps.kde.org/KWrite

Default YAWP's PDF browser is atril:

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

10.5. 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 Weather Plasmoid
- Yet Another Web Page
- Yet Another Web Playground
- Yet Another Web Programming
- Yet Another Webservice Provider
- Yet Another Webshare Plugin
- Yet Another Winning Post
- Yet Another Wireless Problem
- Yet Another Wonderful Protocol
- Yet Another World Protector
- Yet Another Worthless Publication
- Yet Another WSE Presentation
- Young Adult Writing Program
- Young ArbitralWomen Practitioners
- Young Asian With Power

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

10.6. ACRONYMS

The following table shows acronyms contained in this YAWP User Manual:

ACRONYM	MEANING
ASCII	American Standard Code for Information Interchange
CLI	Command Line Interface
CUPS	Common Unix Printing System
DTP	DeskTop Publishing
FOSS	Free and Open Source Software
FSF	Free Software Foundation
GNU	Gnu is Not Unix
GPL	General Public License
GUI	Graphic User Interface
HP	Hewlett-Packard
IDE	Integrated Development Environment
IDLE	Integrated Development and Learning Environment
KDE	Kool Desktop Environment
MS	MicroSoft
PDF	Portable Document Format
PID	Process IDentifier
PyPI	Python Package Index
RAM	Random Access Memory
stderr	standard error file
stdin	standard input file
stdout	standard output file
USA	United States of America
UTF-8	Unicode Transformation Format 8 bits
WP	Word Processor
WYSIWYG	What You See Is What You Get
XFCE	XForms Common Environment
YAWP	Yet Another Word Processor

Figure 10.6.a. Acronyms

10.7. LICENSE

This program is free software, you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program (see the text file LICENSE), if not, see:

    https://www.gnu.org/licenses/

or contact Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02111-1301 USA.

10.8. CHARACTERS

If you want to insert into your text file some character that appears as a space, but prevents automatic line break and automatic space insertion or deletion by text formatting, you can insert one or more "no-break space" characters (codepoint: hexadecimal 00a0, decimal 160). There are several ways to achieve this, we will mention a rather simple one:

- press Ctrl-Shift-U
- type the hexadecimal codepoint of the character, namely '00a0', or simply 'a0'
- press Enter

With this method you can insert any Unicode character you want, as long as you know its codepoint. YAWP can not include images in text file, but Unicode characters offer a lot of graphic capabilities you can explore, see for instance:

    https://symbl.cc/en/unicode-table

The following tables show a choice of relevant characters, with their code points next to them in hexadecimal and decimal format:

CHARACTER NAME	CHAR	HEX	DEC
null (or nul)	'\0'	0000	0
…	…	…	…
bell	'\a'	0007	7
backspace	'\b'	0008	8
horizontal tab (or tab)	'\t'	0009	9
line feed	'\n'	000a	10
vertical tab	'\v'	000b	11
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 (or pound or hash or sharp)	'#'	0023	35
dollar	'$'	0024	36
percent	'%'	0025	37
ampersand	'&'	0026	38
apostrophe (or single quote)	"'"	0027	39
left parenthesis	'('	0028	40
right parenthesis	')'	0029	41
asterisk	'*'	002a	42
plus	'+'	002b	43
comma	','	002c	44
hyphen (or minus)	'-'	002d	45
decimal point (or dot or period or full stop)	'.'	002e	46
slash (or solidus)	'/'	002f	47
digit zero	'0'	0030	48
…	…	…	…
digit nine	'9'	0039	57
colon	':'	003a	58
semicolon	';'	003b	59
less than	'<'	003c	60
equal	'='	003d	61
greater than	'>'	003e	62
question mark	'?'	003f	63
at	'@'	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
delete (or del or rubout)	'\u007f'	007f	127

Figure 10.8.a. "ASCII" Characters

CHARACTER NAME	CHAR	HEX	DEC
no-break (or required or hard or fixed) space	'\u00a0'	00a0	160
macron	'¯'	00af	175
degree	'°'	00b0	176
superscript two	'²'	00b2	178
superscript three	'³'	00b3	179
middle dot (or center dot)	'·'	00b7	183
dot above (or diacritica)	'˙'	02d9	729
greek capital letter sigma	'Σ'	03a3	931
greek small letter pi	'π'	03c0	960
zero width space	'\u200b'	200b	8203
zero width non-joiner	'\u200c'	200c	8204
zero width joiner	'\u200d'	200d	8205
black small circle (or bullet)	'•'	2022	8226
horizontal ellipsis (or ellipsis)	'…'	2026	8230
overline	'‾'	203e	8254
leftwards arrow	'←'	2190	8592
rightwards arrow	'→'	2192	8594
square root	'√'	221a	8730
infinity	'∞'	221e	8734
almost equal	'≈'	221e	8776
not equal	'≠'	2260	8800
identical (or equivalent or congruent)	'≡'	2261	8801
less than or equal	'≤'	2264	8804
greater than or equal	'≥'	2265	8805
box drawings light horizontal	'─'	2500	9472
box drawings light vertical	'│'	2502	9474
box drawings light triple dash horizontal	'┄'	2504	9476
box drawings light quadruple dash vertical	'┆'	250a	9482
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
black square	'■'	25a0	9632
white square	'□'	25a1	9633
white vertical rectangle	'▯'	25af	9647
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
quarter note	'♩'	2669	9833

Figure 10.8.b. Other Unicode Characters

11. CHEAT SHEET

- YAWP 2.0.0 - Yet Another Word Processor - 2024-05-20 - 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 YAWP's version number and exit
$ yawp -H […arguments…] # browse the PDF YAWP User Manual and exit
$ yawp text_file # run YAWP in GUI mode, explicit text file, no other arguments
$ yawp           # run YAWP in GUI mode, text file from previous session, no arguments
$ yawp -M c […arguments…] text_file target_file # run YAWP in CLI Copy mode
$ yawp -M m […arguments…] text_file target_file # run YAWP in CLI Move mode
$ yawp -M d […arguments…] text_file             # run YAWP in CLI Delete mode
$ yawp -M e […arguments…] 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 Noform mode
$ yawp -M u […arguments…] text_file             # run YAWP in CLI Undo mode
```

Buttons in GUI Main Window:

- New: create a new empty text file with default arguments
- Open: browse the file system to select an existing text file
- Recent: browse the list of recent files to select an existing text file
- Copy: copy the current text file with its argument file
- Move: move the current text file with its argument log and backup files
- Delete: delete the current text file with its argument log and backup files
- Edit: edit the current text file by the text editor defined by -y
- Format: format the current text file and redraw and align pictures and export PDF
- Noform: don't format current text file but redraw and align pictures and export PDF
- Undo: restore the current text file to its previous content and export PDF
- Log: browse the log file of current text file by the text editor defined by -y
- Help: browse the YAWP-generated YAWP Manual by the PDF browser defined by -Y
- Exit: quit YAWP

Arguments:

- Usage:
  - -h, --help: show a help message and exit
  - -V, --version: show YAWP version number and exit
  - -H, --browse-manual: browse the YAWP-generated PDF YAWP User Manual and exit
  - -v, --verbose: write information messages on stdout too (CLI modes only)
  - -M, --usage-mode: run YAWP in this usage mode (default: 'g'=GUI, 'c'=CLI Copy, 'm'=CLI Move, 'd'=CLI Delete, 'e'=CLI Edit, 'f'=CLI Format, 'n'=CLI Noform, 'u'=CLI Undo)
- Format:
  - -y, --text-editor: editor for text files (default: 'kwrite')
  - -l, --left-only-text: justify text lines at left only (default: at left and right)
  - -w, --chars-per-line: line width in characters per line (default: 0=automatic)
  - -g, --graph-pictures: redraw '`'-segments and '^'-arrowheads
  - -u, --lines-per-page: page height in lines per page (default: 0=automatic)
  - -k, --align-pictures: align pictures (default: 'n'=no, 'l'=left, 'c'=center, 'r'=right)
- Chapters:
  - -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')
  - -m, --chapter-offset: offset of level-1 numbered chapters (default: 0, min: -1)
- Pages:
  - -p, --page-headers: insert page headers (default: 'n'=no, 'f'=on full page, 'p'=and on broken pictures, 'c'=and on level-1 chapters, 'd'=double if level-1 on even page)
  - -e, --even-left: first line of headers of even pages, left (default: '%n')
  - -E, --even-right: first line of headers of even pages, right (default: '%f')
  - -o, --odd-left: first line of headers of odd pages, left (default: '%c')
  - -O, --odd-right: first line of headers of odd pages, right (default: '%n')
  - -n, --page-offset: offset of page numbers (default: 0, if negative: Roman numbers)
  - -a, --all-pages-E-e: put in all page headers -E at left and -e at right
  - -s, --second-line: second line of page headers (default: 's'=solid, 'n'=no, 'b'=blanks, 'p'=points, 'd'=dashes)
- Export:
  - -X, --export-pdf: export and browse PDF file (default: 'n'=no, 'e'=export, 'b'=export and browse)
  - -C, --correct: correct character size and page margins (default: 'd'=by default values, 'n'=no, 'f'=by correction file)
  - -Y, --pdf-browser: browser for PDF files (default: 'atril')
  - -P, --pdf-file: exported PDF file (default: '%f%e.pdf')
  - -W, --char-width: character width (default: '0'=automatic, unit: pt/in/mm/cm)
  - -A, --char-aspect: character aspect ratio=width/height (default: 3/5, 1=square chars)
  - -S, --sheet-size: portrait paper sheet size, width x height (default: 'A4'≡'210x297mm', unit: pt/in/mm/cm)
  - -Z, --landscape: turn page by 90° (default: portrait)
  - -L, --left-margin: left margin (default: 2cm, unit: pt/in/mm/cm)
  - -R, --right-margin: right margin (default: 2cm, unit: pt/in/mm/cm)
  - -T, --top-margin: top margin (default: 2cm, unit: pt/in/mm/cm)
  - -B, --bottom-margin: bottom margin (default: 2cm, unit: pt/in/mm/cm)
  - -I, --multi-pages: pages on each side of paper sheets (default: 1, values: 1/2/4/8)
  - -J, --multi-sheets: paper sheets gathered together (default: 0=export sequentially)
- Text Files:
  - text_file: text file to process, ASCII or UTF-8-encoded Unicode
  - target_file: destination text file for CLI Copy mode (-M c) and CLI Move mode (-M m)

FIGURES

INDEX

- in folio iii, "26"
- in octavo iii, "28"
- in quarto iii, "27"
- inches "23", 24, 36
- indented line "10"
- int-dot couples "12"
- level "12", "15"
- millimeters "23", 24, 36
- no-break space 16, 18, "42"
- page header line "10", 16
- percent variable "16"
- picture state "10"
- points 15, 18, "23", 24, 33, 34, 35, 36, 42, 45
- shrunk* "iv", 1, 10, 12, 14, 15
- status indicator "2", 40
- subject "14"
- text file "iii", 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21, …
- text state "10"
- titlecased* "iv", 12, 14, 15
- unindented line "10", 11
- uppercased* "iv", 12, 14, 15