The Wr document writer

Francisco J. Ballesteros
TR LSUB-14-5

ABSTRACT

To document Clive we had to write multiple documents including manual pages, papers, and HTML pages. This paper describes `wr`, a program used to write documents and generate the desired output format.

1. Motivation

All the documents we write for Clive are simple ones, regarding the format they use. Most use just simple typesetting marks, and keep a simple structure. For figures, we usually rely on `pic` to write graphics, or include an existing image.

Writing documents became a burden because for different purposes (manual pages, web pages, papers, slides, ...) we had to use different tools (LaTeX, roff, HTML, etc.).

`Wr` was written to put an end to this. It accepts a very simple input format which is enough for most of the documents we write. It can generate any of HTML, roff, text output (eg. man output), PostScript, LaTeX, and PDF.

2. Titles and paragraphs

A `wr` document is a text file, usually named with `.w` as the extension. If there is a title and author information, they are provided first, in lines starting with "`_ `", that is, an underscore and a space. The same goes for author and address information. For example, for this document, we used:

```
_ The Wr document writer
_ Francisco J. Ballesteros
_ TR LSUB-14-5
```

If there is no title there is no need to supply these lines. The first one is the title. The following ones are author, date, etc. The title line may be continued with extra lines like in

```
_ A title line
continued in another line
_ The author
...
```

This can be used to write a title in multiple lines while keeping control on where the line breaks are placed.

Copyright notices can be generated using a line like

```
(c) Yoyodine, Inc. 2064
```

right after the title lines (The "`(c)`" is the Unicode rune for *copyright*, not the three ASCII characters).

Sections are started using "`*  `", that is, a star and a space, followed by the section title. For example, this document has

```
* Abstract
To document Clive we had to write multiple documents
...
* Titles and paragraphs
A wr document
...
```

There are sub-sections and sub-sub-sections. The former uses `**` instead of `*` and the latter uses `***`.

For books, chapter titles are defined by lines starting with "`=  `", like in

```
= A tiny Chapter
```

In books, the title produces a title page and the generated roff and PDF (if PDF output is asked for) include a table of contents that is printed at the end (processing the source in a single pass). The PDF can be edited later on to move the TOC to the start of the book. If there is no chapter, the document is not a book.

Footnotes are supported. This was used to create the footnote in this document:

```
! An example footnote.
```

The footnote can be referenced like done here[1] using words contained in the note:

```
[foot: example footnote]
```

Paragraphs are delimited by empty lines. Indented paragraphs are simply indented using a tabulator character with respect to the surrounding text.

	For example, this paragraph is indented

and this is back to the first indentation level.

To do this, we wrote

```
Indented paragraphs are simply indented using a tabulator character
with respect to the surrounding text.
    For example, this paragraph is indented
and this is back to the first indentation level.
```

A line break,
like this one can be forced by writing `-` as the only text in the line.

3. Enumerations and item lists

Item lists are written as indented paragraphs that start with a `-` or # character. The first one indicates that the paragraph is an item and the second indicates that it is an enumerated item.

The list is made out of a sequence of items or enumerated items. An item may include one or more paragraphs, indicated by the indentation level of the paragraphs. For example

- This is an example item

1. An example footnote.

- And this is another
  1. including a first enumerated item

     a second paragraph of the first enumerated item
  2. and a second enumerated item
- And a final third item

To write the example, we wrote:

```
An item may include ... paragraphs. For example

    - This is an example item
    - And this is another
        # including a first enumerated item

        a second paragraph of the first enumerated item
        # and a second enumerated item
    - And a final third item
```

A description of items can be written as a list of items where each item has an indented description with one or more paragraphs. For example

- **wr**
  is a tool to write documents
- **roff**
  is a nice typesetting tool

can be generated writing:

```
For example

    - wr
        is a tool to write documents
    - roff
        is a nice typesetting tool
```

4. Fonts

In general, there is no control over fonts. However, some typesetting marks are understood. You can use ’’ to set words in **bold**, ’’ for `teletype`, and ’’ for *italics.*

Each one of these characters can be escaped by writing it twice. The teletype mark enters into verbatim mode and almost no other mark can be used inside it.

To write the previous paragraph, we wrote

```
You can use ’**’ to set words in *bold*, ’||’ for |teletype|, ’__’ for _italics_.
```

These marks may be written alone in a line to switch on/off the font change for several lines or paragraphs. Also, in description lists, the described item can be written in a different font using these. By default, de described items go in boldface. For example:

```
For example

    - |wr|
        is a tool to write documents
    - _roff_
        is a nice typesetting tool
```

yields

- `wr`
  is a tool to write documents
- *roff*
  is a nice typesetting tool

5. Figures, Tables, and other tools

To cite some document, like in wr(1) you can write `[cite: wr(1)]` inline with the rest of the text. Another variant places links to URLS, like when writing a like to the lsub page [http://lsub.org], which is generated by writing

```
[url: the lsub page|http://lsub.org]
```

Other figures, tables, etc. may be cited using the same syntax. For example:

```
is used to reference figure [fig: with caption] here.
```

is used to reference figure 1 here. Bibliography works in the same way. Using refer syntax for bib. files, you can cite documents like in

```
[bib: Clive operating system]
```

which matches the keywords against refer entries.

To write verbatim text use a bracketed description, like we did to generate the URL example above this paragraph:

```
To cite ... generated by writing
    [verb
        [url: the lsub page|http://lsub.org]
    ]
To write...
```

Using verbatim text is the suggested form to include source code listings. A tag may be given right after `verb` to name the source being listed. For example,

`[hi.sh]`:

```
#!/bin/sh
echo hi there
```

is produced using

```
[verb hi.sh
    #!/bin/sh
    echo hi there
]
```

A variant executes a command and takes its output as verbatim text. This indented paragraph

```
Mon Sep  5 17:27:28 CEST 2016
```

is generated when this document is compiled using

```
[sh
    date
]
```

A similar construct is a (floating) code listing figure. For example

Listing 1: `listing`

```
forever {
    do something;
}
```

Is generated writing

```
[code listing
    forever {
        do something;
    }
]
```

A figure relies on a similar construct, using the `fig` tag instead of `code`. For example, we can write

```
[fig
    inkdump.eps
with a caption
]
```

to generate one like in

Figures written in `pic` are easier, and can be written directly, like in the next one, which was written using:

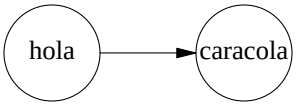

Figure 2: *the caption starts without indentation and may span several lines.*

```
[pic
    circle "hola"
    arrow
    circle "caracola"
the caption starts without indentation and
may span several lines.
]
```

Figures can be `grap` input, like in the plot that is shown using

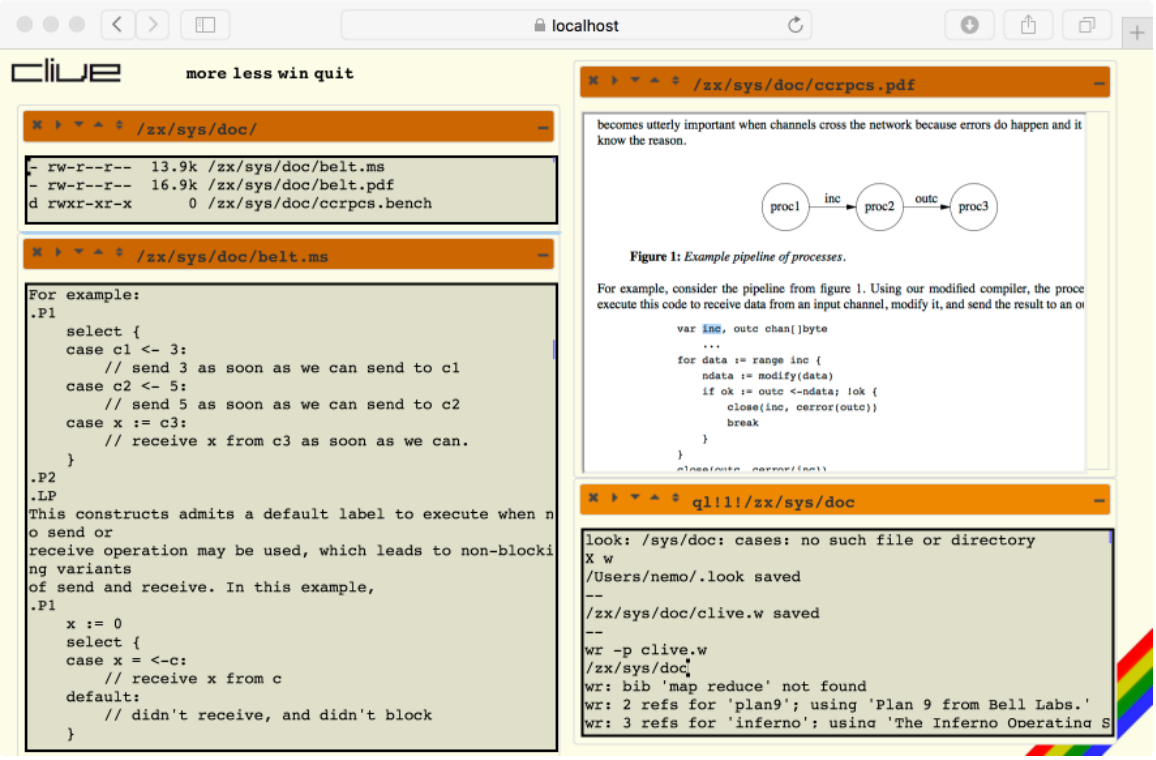

Figure 1: *with a caption*

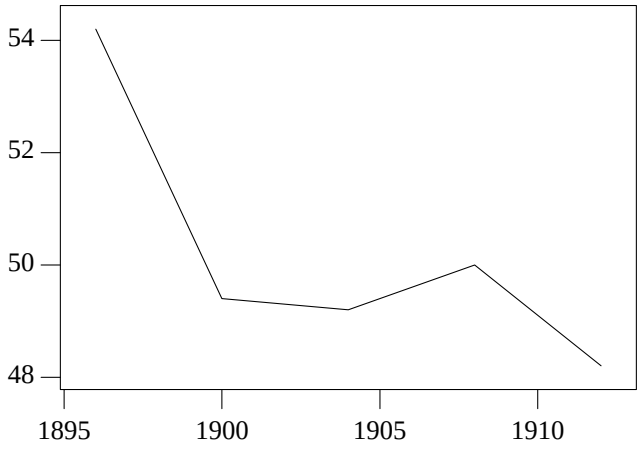

Figure 3: *A plot of data using grap*

```
[grap
    draw solid
    1896    54.2
    1900    49.4
    1904    49.2
    1908    50.0
    1912    48.2
A plot of data using grap
]
```

Finally, tables are tabular data, where the first row indicates alignment, the second row is the heading for

the rows, and the first column indicates names for the rows. For example:

	col2	col3
row1	11	12
row2	21	22

Table 1.

is written as

```
[tbl key
    unused    c    r
    unused    col2    col3
    row1    11    12
    row2    21    22
]
```

For more complex structures, do not use `wr`.

6. User's manual

The (now outdated) usage information for `wr` is:

```
wr [-DIPS] [-hlmpst] [-o out] file
```

The output file(s) are generated in the directory where the command is run, unless flag `-o` is used to indicate an output file. In this case, output file(s) are generated in the directory of the indicated output file.

The input file may have any name, but the suggested extension is `.w` for all `wr` input files.

These are the options:

- **-DIPS**
  debug flags.
- **-h**
  generate html.
- **-l**
  generate LaTeX.
- **-m**
  generate text (for manual pages).
- **-p**
  generate PDF.
- **-s**
  generate PS.
- **-t**
  generate [tn]roff.

And now some examples:

- Generate text output for a file:

  ```
  ; wr file
  ```

- Generate html and place the output at another directory (including any auxiliary files for figures):

  ```
  ; wr -h -o /dest/dir/out.html file
  ```

7. Remarks

There may be a few other features not documented here. You may refer to the `example` file in the source to see all the constructs understood by the implementation.