Writing and Publishing with Linux

“”

Contents


```
MARKDOWN


  The markdown system uses a
   ⇒ minimalist structured text
   ⇒ format to produce
  html.  'markdown' is both a '
   ⇒ text format' and also a perl
   ⇒  script used to
  turn that format into html.
   ⇒ Using pandoc it is possible
   ⇒ to convert
  markdown formatted text into
   ⇒ other formats, such as pdf.
   ⇒ Markdown may fill
  a gap between enscript and latex
   ⇒ , where the user wants more
   ⇒ control over
  how the produced document looks
   ⇒ than can be obtained with '
   ⇒ enscript', but
  doesnt wish to become mired down
   ⇒  in the complexities of '
   ⇒ latex'


  @@ http://daringfireball.net/
   ⇒ projects/markdown/
    details about the format of
     ⇒ the source document (text
     ⇒ file).
  @@ http://daringfireball.net/
   ⇒ projects/markdown/dingus
    a page to actually try out
     ⇒ markdown text syntax
  @@ http://six.pairlist.net/
   ⇒ pipermail/markdown-discuss/
    the markdown discussion list
     ⇒ archives


CONVERTING TO HTML ....

 The 'markdown' perl script can
```

```
  ⇒ only create html but 'pandoc'
  ⇒  can produce
 other formats from the same
  ⇒ markdown source text document
  ⇒ . Markdown
 only seems to produce a 'fragment
  ⇒ ' of html (without head and
  ⇒ body tags)

 * display a text document
  ⇒ converted to html with
  ⇒ markdown
 >> markdown document.txt | less

 * convert a document to html and
  ⇒ save as 'doc.html'
 >> markdown doc.txt > doc.html

 * convert a document and add html
  ⇒  and body tags to make a
  ⇒ complete page
 >> (echo '<html><body>';markdown
  ⇒ doc.txt;echo '</body></html
  ⇒ >') > out.html
 >> markdown doc.txt|sed '1s/^/<
  ⇒ html><body>/;$s.$.</body></
  ⇒ html>.' > out.html
 >> pandoc -s -o out.html doc.txt
  ⇒  ##(the same but better,
  ⇒ using pandoc)

MARKDOWN SYNTAX ....

 The great advantage of markdown
  ⇒ syntax is that the source
  ⇒ document can be kept
 'clean' without distracting or
  ⇒ ugly markup codes.  Normal
  ⇒ html can be inserted
 anywhere in a markdown document.
  ⇒ If there is anything which
  ⇒ markdown
 cannot do (such as tables) just
  ⇒ use html tags instead.

DOCUMENT HEADINGS ....

  * create a 1st-level document
   ⇒ heading (underline the text
   ⇒ with '=' equals)
  \begin{lstlisting}
   The Heading
   ===========
```

* create a markdown 2nd-level document heading

```
   A Second Level Heading
   \begin{lstlisting}
```

* markup a 3rd level document heading 'Stone Fruit' ¿¿ Stone Fruit

* create a vim command ',ll' which underlines the current line with '-' dashes ¿¿ map ,ll yyp:s/[]*//|s/[]*// |s/[]/ - /g|s/ - / - -/g < cr > o(placein'vimrc'.Thismappinghelpstowritemarkdown2n

* create a vim command ',LL' which creates a markdown 1st level heading ¿¿ map ,ll yyp:s/[]*//|s/[]*// |s/[]/ = /g|s/ = / == /g < cr > o(Thismappingcommandunderlinesthecurrentlinewith' equals)

* create a 'blockquote', that is text more indented than surrounding paragraphs ¿¿ ¿ (blockquotes can contain headings and paragraphs

TEXT EMPHASIS ....

* make a word or phrase emphasised ¿¿ the following is *emphasised text* ¿¿ the following is $_e mphasisedtext_{(moreorlessthesame)}$

* strongly emphasise words or phrases ¿¿ the following is **strongly emphasised text** ¿¿ the following is $_s tronglyemphasisedtext$

LISTS ....

The list marker can also be a '+' or a '*'. There is no difference between these types of lists

* create an unordered (bulleted) list

```
- first item
- second item
- third and last
```

* create an nested unordered list

```
- first item
  * sublist item 1
  * sublist item 2
- second item
- third and last
```

* create an ordered list with a nested unordered list

```
 1.  First
 2.  Second:
     -   Fee
     -   Fie
     -   Foe
 3.  Third
```

* create an unordered list with links to other parts of the page

```
- [Chapter One](#chap1)
- [Chapter Two](#chap2)
- [Chapter Three](#chap3)
<h2 id='chap1'>...</h2>
<h2 id='chap2'>...</h2>
```

(notice how normal html is used to create the anchors)

* create an ordered list

```
1. first
2. second
3. third
```

LINKS ....
* create link in a document with the display text 'example link' ¿¿ This is an [example link](http://example.com/) (the html: this is an ¡a href="http://example.com/"¿example link¡/a¿.¡/p¿)
* create link with a display text and also a 'title' (when the mouse hovers over) ¿¿ [example link](http://example.com/ "With a Title")
* for pdf output the embedded code backtick may look best ¿¿ 'http://google.com' (displays the link in fixed pitch font in pdf)
* create and use a named link with display text 'Google'

```
 a search engine is [Google][1]
 [1]: http://www.google.com
  ⇒ "Google"
 [2]: http://www.yahoo.com
  ⇒ "Yahoo"

* create a reference link with a
  ⇒  space in the name
\begin{lstlisting}
 I start my morning with a cup
  ⇒ of coffee and
 [The New York Times][NY Times].

 [ny times]: http://www.nytimes.
  ⇒ com/
```

IMAGES ....
* embedd an image file in the document, with an optional title ![a tree](/path/to/img.jpg "Title") (if the image is not found show 'tree')
* embedd an image in the document using the 'reference' style syntax

```
![alt text][id]

[id]: /path/to/img.jpg "Title"
```

EMBEDDED CODE ....
* use backticks '`' to show code with special characters in it ¿¿ dont use the '¡blink¿' tag
* another example ¿¿ html entities can be named 'mdash;' or numbered '8212;'.
* use 4 spaces or 1 tab of indentation to display a paragraph of code
* use a backslash to escape special characters ¿¿ (this prints an asterix)
* create a horizontal rule using 3 or more dashes, asterixes etc ¿¿ — ¿¿ * * * (the same)

GOTCHAS ....
The table dashes must begin in the 5th column all later for the table to be formatted correctly.

PANDOC MARKDOWN SYNTAX ....
Pandoc allows a number of extensions to markdown syntax to allow for structures such as tables. These extensions can be turned off with the '–strict' switch @@ johnmacfarlane.net/pandoc/README an example of a pandoc markdown document @@ johnmacfarlane.net/pandoc/README.htmlpandocs-markdown-vs.standard-markdown details about the pandoc extensions to the markdown syntax
* create subscripts and superscripts ¿¿ H 2 O is a liquid. 2^10^is1024.
* create text P with 'a cat' superscripted ¿¿ P a cat (note the escaped backslash in the superscript)
* format text which is 'struck out', that is, has a bar through it ¿¿ This is deleted text.
* create compact definition lists

```
Term 1
  ~ Definition 1
Term 2
  ~ Definition 2a
  ~ Definition 2b
```

* single brace links (also allowed in modern markdown syntax)

```
1. Here's my [link]
2. Here's my [link][]
[link]: linky.com
```

* inline footnotes

```
Here is an inline note.^[
 ⇒ Inlines notes are easier
 ⇒ to write, since
you don't have to pick an
 ⇒ identifier and move down
 ⇒ to type the note.]
```

* footnotes

```
Here is a footnote reference
 ⇒ ,[^1] and another.[^
 ⇒ longnote]

[^1]: Here is the footnote.
[^longnote]: lots of text
```

* create a table with a caption (use the underline to dictate the alignment)

```
  Right      Left      Center
   ⇒      Default
-------      ------ ----------
 ⇒    -------
    12      12         12
     ⇒              12
   123      123        123
    ⇒           123
     1      1           1
      ⇒              1

Table:  Demonstration of
 ⇒ simple table syntax.
```

The table dashes must start in the 5th (?) or greater column of the source text file. The table is rendered in a fixed pitch font with the dashes.
* create a table with no headers

```
-------      ------ ----------
 ⇒    -------

    12      12          12
     ⇒              12

   123      123         123
    ⇒            123

     1      1            1
      ⇒               1
-------
```

A Table must end with a blank line. Multiline tables are also possible.
* source code blocks (begin and end with ¿3 tilde ' ' characters)

```
~~~
 code
~~~
```

* source code blocks with syntax highlighting and numbered lines

```
~~~~~~~~ {.haskell .numberLines}
qsort []     = []
qsort (x:xs) = qsort (filter (<
 ⇒   x) xs) ++ [x] ++
               qsort (filter
                ⇒ (>= x) xs)
~~~~~~~
```

Syntax highlighting may only work in html output files
* show what languages syntax highlighting pandoc supports ¿¿ pandoc –version
* provide document meta-information (title, author, date)

```
% Small Steps
% J Appleby, B Kernighan
% 21/4/2001
```

* html blocks within markdown documents

```
<table>
    <tr>
      <td>*one*</td>
      <td>[a link](http://
       ⇒ google.com)</td>
    </tr>
</table>
```

* links to the document sections automatically created ¿¿ See the section on [header identifiers](header-identifiers-in-html).
* The section heading 'Green Tree' becomes [link](green-tree)
* produce a 'presentation' slide show with pandoc and 's5'
In the morning
- Eat eggs - Drink coffee
In the evening
- Eat spaghetti - Drink wine
pandoc -w s5 -s eating.txt ¿ eating.html

Section 1

Pandoc

pandoc extends both the syntax of markdown (allowing tables for example) and also the number of formats to which markdown can be output, including most importantly 'pdf'. Pandoc can perform other nice tricks such as putting automatic tables of contents into documents based on the section headings.
Pandoc appears to be in active development (early 2010). Features such as 'tables' are only available in later versions of pandoc. And these versions are not available as debian packages.

`http://johnmacfarlane.net/pandoc/README.html`
: the good user guide for pandoc

`http://johnmacfarlane.net/pandoc/examples.html`
: example of converting markdown (et al) documents to other formats

`http://groups.google.com/group/pandoc-discuss`
: the pandoc discussion news group

pandoc output formats

HTML	Webpages
markdown	
LaTeX	The tex typesetting system
groff man	Unix manual pages
RTF	Can be opened in ms word, for example
DocBook XML	
texinfo	?
reStructuredText	
ConTeXt	
S5 HTML slide shows.	

1.1 Installing Pandoc

Install ghc6 which is a haskell compiler.
Install the latest pandoc using the haskell 'cabal' tool

```
cabal install pandoc
```

1.2 Basic Usage

Convert a 'markdown' text document to html format with pandoc

```
pandoc doc.txt
```

```
pandoc -o out.html doc.txt                  ~(the same, html is the default ⇒ )
```

```
pandoc doc.txt -o out.html                  ~(better?)
```

```
pandoc -f markdown doc.txt -o out.html      ~(the same, but unnecessary)
```

Show what header is added to a LaTeX output file

```
pandoc -D latex
```

Convert multiple markdown files and a references file to an html page

```
pandoc -s ch1.txt ch2.txt refs.txt > book.html
```

1.3 Converting Documents

If an output or input format is not given then pandoc will guess based on the file-names supplied.

pandoc output format (switches -t -w –to –write all equivalent)

`native`	Native haskell format
`markdown`	Markdown syntax
`man`	Unix groff man page
`rst`	Restructured text
`html`	Html
`LaTeX`	LaTeX typesetting format
`docbook`	The docbook xml format
`opendocument`	An xml document format a bit like docbook
`mediawiki`	The markup used with 'wikipedia'
`odt`	The open office document format
`s5`	An html and javascript slide show format (like powerpoint)
`rtf`	Rich text format for word processors (eg ms word)

Some of the output formats are documented in the user guide but not in the man page.
Convert a markdown document 'doc.txt' to xml docbook format saved in 'out.xml'

```
pandoc -t docbook -o out.xml doc.txt
```

```
pandoc --to docbook -o out.xml doc.txt      ~(the same)
```

```
pandoc -w docbook -o out.xml doc.txt        ~(the same again)
```

```
pandoc --write docbook -o out.xml doc.txt
```

DocBook XML:

```
pandoc -s -S -w docbook README -o example9.db
```

Convert 'pandoc.1.md' in markdown format to a unix man page 'eg10.1'

```
pandoc -s -w man pandoc.1.md -o eg10.1
```

Convert a 'markdown' text document to tex format with pandoc

```
pandoc -o out.tex doc.txt
```

Convert From LaTeX to markdown:

```
pandoc -s example4.tex -o example5.text
```

Convert a 'markdown' text file to a stand-alone LaTeX file

```
pandoc -s -o out.tex doc.txt
```

Convert the file 'doc.txt' in markdown format to an html webpage 'doc.html'

```
pandoc doc.txt
```

Convert an input stream into the LaTeX format saved as 'out.tex'

```
cat doc.txt | pandoc -o out.tex
```

1.4 Creating Pdf Files

Pandoc uses 'pdflatex' to create pdf files from the source text document.

Convert the file 'doc.txt' in markdown syntax to a printable pdf 'doc.pdf'

```
markdown2pdf doc.txt
```

Convert 'doc.txt' written in markdown format to the pdf file 'output.pdf'

```
markdown2pdf -o output.pdf doc.txt
```

Convert 'doc.txt' to a pdf file 'doc.pdf' which has a table of contents

```
markdown2pdf --toc doc.txt
```

Create a pdf file from 'doc.txt' with a table of contents and numbered sections

```
markdown2pdf -N --toc doc.txt
```

Create a pdf document called 'stdin.pdf' from the markdown document 'doc.txt'

```
cat doc.txt | markdown2pdf
```

Specify that the input format is html

```
pandoc -f html doc.txt
```

Convert the file 'input.txt' to pdf format on a non-utf8 system

```
iconv -t utf-8 input.txt | pandoc | iconv -f utf-8
```

```
iconv -t utf-8 input.txt | markdown2pdf | iconv -f utf-8   ~(better ??)
```

PDF with numbered sections and a custom LaTeX header, compilable with 'xetex'

```
markdown2pdf -N --template=mytemplate.tex --variable version=1.4 README
  ⇒  --xetex --toc -o example
```

(xetex may be good for international character sets)

1.5 Converting To Texinfo

Texinfo is the format of the free software foundation, also known as gnu. Its not a pleasant format

Convert markdown to a texinfo file

```
pandoc README -s -o example19.texi
```

Convert the texinfo source into 'info', html and pdf

```
makeinfo example19.texi -o example19.info
makeinfo example19.texi --html -o example19
texi2pdf example19.texi  # produces example19.pdf
```

1.6 Converting To Html

Convert the text file 'doc.txt' to html linking to css style sheet 'sty.css'

```
pandoc -c sty.css doc.txt   ~(doc.txt uses markdown syntax)
```

Display a markdown text document to an html page and send to standard output

```
pandoc -s -o - doc.txt
```

HTML with smart quotes, table of contents, CSS, and custom footer:

```
pandoc -s -S --toc -c pandoc.css -A footer.html README -o example3.html
```

1.7 Convert From Html

Convert a webpage to the markdown format and save as 'eg.txt'

```
html2markdown http://www.gnu.org/software/make/ -o eg.text
```

Section 2

Halibut

`http://www.chiark.greenend.org.uk/~sgtatham/halibut/`

Halibut is a system for creating formatted documents in a variety of output formats such as html (web-pages) pdf (printable) ... Halibut, uses a terse plain text input format. Halibut does not fully support unicode. Halibut may fill a gap between 'enscript' and 'LaTeX' or 'docbook' in that the user may quickly produce a typeset document with 'formatting'. An alternative may be to write a new 'style' for enscript.
Halibut produces nice pdf output, with automatic table of contents and cross-references. It is essentially a 'poor-mans' (and terse) LaTeX
Enscript seems to support the halibut format.

halibut output formats

```
                Plain ASCII text
                            HTML
          Unix man page format
               GNU info format
                           PDF
                    PostScript
 Old-style Windows Help (.HLP);
```

halibut markup codes (text is placed between the braces)

`\e{}`	Emphasize (eg italicize) the text between the braces
`\c{}`	Format as code (normally use a fixed-pitch font such as courier)
`\cw{}`	Format as weak code
`\cq{}`	Format as quoted code
`\k{} \K{}`	A reference to document section heading
`\W{http://www.google.com/}{Google}`	Url markup
`\i{words}`	Index the words within the braces
`\u00F6{oe}`	Embedd a unicode character in the text
`\q{}`	Quoted text
`\quote{}`	A long, maybe multi-paragraph quotations
`\rule`	A horizontal rule on the page (should go on a line by itself)
`\C \H \S \A \U \title`	Chapter and section headings

2.1 Gotchas

You cant have headings without chapters, etc (its not like html) Chapters start new pages and there doesnt seem to be any way around this Very few formatting codes in halibut, eg no 'bold' text, or font size I dont think that you can put images in the document

2.2 Formatting Examples

Display text between quotes

```
here is some \q{text in quotes}
```

Format code with some un-line-breakable hyphens and underscores

```
the Emacs command \c{M\-x\_psychoanalyze\-pinhead}
```

Halibut codes can be multi-line

```
\e{this is
emphasised}
```

Include an automatically generated date

```
This document was generated on \date
```

(the date looks like 'Thu Feb 1 12:20:43 2007')

Include an auto-date in a specific format

```
This document was generated on \date{%Y-%m-%d %H:%M:%S}
```

(this date looks like '2007-02-01 12:20:43')

View possible date format strings

```
man ? strftime
```

2.3 Urls

Format a url with the display string 'google'

```
Try searching on \W{http://www.google.com/}{Google}.
```

Display a url with itself as the display string

```
Google is at \W{http://www.google.com/}\cw{www.google.com}
```

For pdf output use code formatting for urls, if no link is required

```
Try searching on \c{http://www.google.com/}
```

```
Try searching on \cw{http://www.google.com/}    ~(nearly the same)
```

2.4 Unicode

Include a unicode character in the document

```
\u0041    ~(this would display 'A' since 41 is the code for 'A')
```

Include the 'euro' symbol or the text 'EUR_' if the symbol is not available

```
This is likely to cost \u20AC{EUR\_}2500 at least.
```

Include a comment in a document, which will be ommited in the output

```
The typical behaviour of an antelope \#{do I mean gazelle?} is..
```

Include code in a halibut source document

```
\c #include <stdio.h>
\c
\c int main(int argc, char **argv) {
\c     printf("hello, world\n");
\c     return 0;
\c }
```

2.5 Chapter And Section Headings

Create a title for the whole document

```
\title A Simple Car Manual
```

Create a chapter heading 'special tools' with a reference keyword 'tools'

```
\C{tools} Special Tools
```

Insert a cross-reference to the tools chapter created above

```
see the \k{tools} chapter
```

Create a heading (one level below chapters)

```
\H{tool-types} Tool Types
```

Create a section heading (two levels below a chapter)

```
\s{sec1} other concerns
```

Create a section which is called a 'question' in the table of contents

```
\S{question-about-fish}{Question} What about fish?
```

Create an appendix

```
\A
```

2.6 Lists

Include a list in a document

```
A list follows,

\b One.

\b Two.

\b Three.
```

Format a numbered list

```
A list follows,

\n One.

\n Two.

\n Three.
```

Format a list with a reference

```

\n One.

\n{this-one} Two.

\n Three.

\n go back to step \k{this-one}.
```

Create a description list

```

\dt Pelican

\dd This is a large bird with a big beak.

\dt Panda

\dd This isn't.
```

Create a horizontal rule

```
\rule
```

2.7 Cross References

Include references or links to document sections

```
\K{chap1} expands on \k{chap2}. ~('chap1' and 'chap2' are keywords)
```

(\K starts with a capital letter)

Define a bibliography entry

```
\B{freds-book} \q{The Taming Of The Mongoose}, by Fred Bloggs.

Published by Paperjam & Notoner, 1993.
```

Refer to a bibliography entry with the \k command

```
\k{freds-book}
```

Create a citation to a book

```
\nocite{abook}
```

2.8 The Index

Create an index entry in a source document

```
The \i{hippopotamus} is a particularly large animal.
```

(the index will include the word 'hippo' and the page where is occurs)

Create a multi-word index entry

```
we recommend a \i{torque wrench} for this job.
```

Create an index entry for a code term

```
\i\c{grep} command is what you want here.
```

Create an index for a word and emphasise it in the text

```
this is what we call a \i\e{paper jam}
```

Create an index entry which doesnt appear in the text

```
If your printer runs out of toner, \I{replacing toner cartridge} do
  ⇒ this:
```

2.9 Configuring Halibut

For producing pdf or postscript documents, all the margins and other spacings are configurable, using the \cfg command.

`http://www.chiark.greenend.org.uk/~sgtatham/halibut/doc-1.0/output.html#output-paper`
options to configure pdf output

Configure something

```
\cfg
```

Label all chapters as 'Book'

```
\cfg{chapter}{Book}
```

Define a macro

```
\define{eur} \u20AC{EUR\_}
```

Use the macro which was just defined

```
This is likely to cost \eur 2500 at least.
```

Configure the font size of the document text

```
\cfg{paper-base-font-size}{points} Specifies the font size of body text
  ⇒ .
```

Other config options

```
\cfg{paper-page-height}{points}
    Specify the absolute limits of the paper.
    \cfg{paper-left-margin}{points}
    \cfg{paper-top-margin}{points}
    \cfg{paper-right-margin}{points}
    \cfg{paper-bottom-margin}{points}
        Specify the margin
```

2.10 Compiling Documents

Produce html 'test.html' from the input text file 'file.txt'

```
halibut --html=test.html file.txt
```

Create a pdf file 'output.pdf' from the input file 'file.txt'

```
halibut --pdf file.txt
```

Section 3

Enscript

enscript is the simplest solution for creating a printed document, and such a handy way to quickly get some printed output without wasting precious paper. its great. Enscript may be able to produce postscript and pdf output quickly but it also has many options, which may make more complex typesetting tools unnecessary.

View a postscript file

```
gv file.ps
```

```
evince file.ps    ~(nicer than the ancient 'gv')
```

3.1 Basic Usage

The output file, if it exists, is overwritten.

Convert plain text 'file.txt' to a pdf file 'output.pdf'

```
enscript -p file.ps file.txt; ps2pdf file.ps
```

Create a pdf file from 'file.txt'

```
enscript -o - file.txt | ps2pdf - file.pdf
```

Create a postscript document from the '.txt' files in the current folder

```
enscript -p output.ps *.txt
```

Create a postscript document from 2 files, each line containing 'honey'

```
cat file1.txt file2.txt | grep -v honey | enscript -p output.ps
```

(using this method enscript cant put the filename in the header)

3.2 Headers And Footers

The header is what is printed at the top of each page of the outputted postscript document.

special enscript header/footer codes

$$	Character '$'
$%	Current page number
$=	Number of pages in the current file
$p	Number of pages processed so far
$(VAR)	Value of the environment variable VAR.
%c	Trailing component of the current working directory
%C	($C) current time (file modification time) in 'hh:mm:ss' format
%d	Current working directory
%D	($D) current date (file modification date) in 'yy-mm-dd' format
%D{string}	($D{string}) '%D{}' refers to the current date and '$D{}' to the input file's last modification da
%E	($E) current date (file modification date) in 'yy/mm/dd' format
%F	($F) current date (file modification date) in 'dd.mm.yyyy'
%H	Document title
$L	Number of lines in the current input file. This is valid strings.
%m	The hostname up to the first '.' character
%M	The full hostname
%n	The user login name
$n	Input file name without the directory part
%N	The user's pw_gecos field up to the first ',' character
$N	The full input file name
%t	Current time in 12-hour am/pm
$t	File modification time in 12-hour am/pm
%T	Current time in 24-hour format
$T	File modification time in 24-hour format
%*	($*) current time (file modification time) in 24-hour format
$v	The sequence number of the current input file
%W	($W) current date (file modification date) in 'mm/dd/yy' format

Create a pdf file with no "headers" called "output.pdf"

```
enscript -B -p output.ps file.txt; pd2pdf output.ps
```

```
enscript -Bp output.ps file.txt; pd2pdf output.ps    ~(the same)
```

```
enscript -Bo - file.txt | pd2pdf - output.ps         ~(the same)
```

Create a document with the text 'A summers day' at the top of each page

```
enscript -b 'The summers day' -p out.ps file.txt
```

```
enscript --header='The summers day' -p out.ps file.txt    ~(the same)
```

Output a document with header as filename, current date, and page numbers

```
enscript --header='$n %W Page $% of $=' -p out.ps file.txt
```

```
enscript -b '$n %W Page $% of $=' -p out.ps file.txt
```

(this would print 'out.ps 12/12/2003 Page 2 of 20' or something like that)

Create a document with a header with 'justified' fields (left, center, right)

```
enscript --header='$n|%W|Page $% of $=' -p out.ps file
```

Create a postscript file 'output.ps' with fancy grey headers

```
enscript -G -p output.ps list.txt
```

(the header contains the date, file name and the page number)

Create a ps document with a footer which is like 'Page 2 of 24'

```
enscript --footer='Page $% of $=' -p out.ps file.txt
```

3.3 Changing The Font

The text after the '-f' switch has to be a valid font specification,

Show all valid font names which can be used with the enscript -f option

```
less /usr/share/enscript/afm/font.map    ~(your mileage may vary)
```

```
find / -name 'font.map'     ~(if the above didnt work)
```

Create a postscript file "new.ps" using the font "Helvetica12"

```
enscript -f "Helvetica12" -p new.ps file.txt
```

```
enscript -f "Helvetica@12" -p new.ps file.txt    ~(the same)
```

```
enscript -fHelvetica12 -p new.ps file.txt  ~(this seems to work as well
  ⇒ )
```

These font names are case sensitive!!

```
enscript -f "helvetica12" -p new.ps file.txt
```

Create a postscript document using 8.5 point Times-Roman font

```
enscript -f 'Times-Roman8.5' -p new.ps file.txt
```

```
enscript -f 'Times-Roman@8.5' -p new.ps file.txt  ~(should be the same)
```

Create a document using a font which is 10 points wide and 14 points high

```
enscript -f 'Courier@10/14' -p new.ps file.txt
```

Create a postscript file with a smaller font

```
enscript -f "Helvetica8" -p new.ps file.txt
```

```
enscript -f "Helvetica6" -p new.ps file.txt  ~(only 6 points!)
```

```
enscript -f "Helvetica4" -p new.ps file.txt  ~(tiny, need a good
  ⇒ printer)
```

3.4 Creating Banner Text With Enscript

`http://www.linux.com/archive/articles/55835`
: how to make 'web banners' (an image to go at the top of a webpage) using enscript, by Michael Stutz, the same guy who wrote the Linux Cookbook for No Starch Press.

Make a large image of the text 'Green Tree'

```
echo "Green Tree" | enscript -B -f "Palatino-Bold48" -o blog.ps
```

```
echo "Green Tree" | enscript -Bf "Palatino-Bold48" -p blog.ps ~(same)
```

Do the whole thing in imagemagick

```
convert -font Helvetica-Bold -pointsize 48 -background black -fill red
  ⇒ label:'Hallo world' allo.png
```

Crop out all the unnecessary space around the text and convert to 'jpg'

```
convert -crop WxH+0+0 filename.ps filename.jpg  ~(width and height)
```

```
convert -trim +repage filename.ps filename.jpg  ~(untested)
```

All in one go

```
echo "Headlines for `date +'%B %e'`" | enscript -o - -B -f "Times-
  ⇒ Bold48" | convert -crop 200x600+0+0 - headlines.png  ~(untested)
```

3.5 Multiple Columns

Change the horizontal column height

```
--h-column-height=height
```

Make a postscript file (from 2 text files) with 3 columns, and view it.

```
cat file1.txt file2.txt | enscript -3p output.ps; evince output.ps
```

Create a 2 column postscript file with lines separating columns and a frame

```
enscript -2 -j -p file.ps file.txt
```

```
enscript -2jp file.ps file.txt     ~(the same, but better)
```

Create a pdf document with 20 columns and no 'headers' (thats right, 20)

```
enscript --columns=20 -2Bp file.ps file.txt; ps2pdf file.ps
```

(the 'header' is the text which appears at the top of each page)

Create a postscript document 'new.ps' with 9 columns from 'file.txt'

```
enscript -9p new.ps file.txt
```

3.6 Landscape Printing

Convert "file.txt" to pdf "file.pdf" landscape format with 2 columns

```
enscript -2r -p file.ps file.txt; ps2pdf file.ps
```

3.7 Changing The Margins

Set the margins in the document to those specified

```
enscript --margins=50:50:50:100 -p tree.ps tree.txt
```

In my version of enscript 1.6.4 the format for the margin command is

```
--margins=top:bottom:right:left which is a very unusual order
```

But in the documentation the format is stated to be

```
--margins=top:bottom:right:left which would be more logical
```

Set the left and right margins to 50 pnts with top and bottom default

```
enscript --margins=50:50 -p tree.ps tree.txt
```

Indent every line by 2 character widths

```
enscript -i 2l      ~(2 character widths)
```

```
enscript -i 20p     ~(indent by 20 points)
```

```
enscript -i 3i      ~(3 inches)
```

```
enscript -i 3c      ~(3 centimetres)
```

3.8 Underlayed Text

Enscript can print an 'underlayed' text which is like a watermark which goes underneath the text from the file.

Create a postscript file with a "watermark" large font text diagonally Down the page.

```
enscript -u"Resume" -p file.ps file.txt
```

3.9 Output Formats

Instead of creating postscript files, enscript can also create 'rtf' (rich text format) files (which can be opened in Microsoft Word, for example) and 'html' files, which can be viewed with a web-browser such as 'google chrome', 'firefox' or 'internet explorer'.

Create a rich text file 'output.rtf' from a text file with no headers

```
enscript -wrtf -B -p output.rtf file.txt
```

(rtf files can be opened in wysiwyg editors like MS Word)

Create a colourised webpage of the file "tree.txt"

```
enscript -whtml --color -E -p file.html tree.txt
```

3.10 Changing The Printout

Make a pdf file 'output.pdf' with each line preceded by the line number

```
enscript --line-numbers -p output.ps list.txt; ps2pdf output.ps
```

```
enscript -Cp output.ps list.txt; ps2pdf output.ps  ~(the same)
```

Create a document with each line numbered starting at line 20

```
enscript -C20 -p output.ps list.txt
```

3.11 The Page Size

Show what page sizes enscript can output to

```
enscript --list-media
```

Create a postscript document with an 'A5' page size

```
enscript -M A5 -p out.ps file.txt
```

```
enscript --media=A5 -p out.ps file.txt   ~(the same)
```

3.12 Making Booklets

By using the 'N-up' printing facilities of enscript it should be possible to create 'booklets', that is, postscript documents which when printed out and folded in half, have all the pages in the right place (just like a real book; this is also called 'quires'). N-up printing involves putting a multiple of 2 logical pages on each physical page.

Put 2 logical pages on each physical page

```
enscript -U2 -p out.ps file.txt
```

```
enscript -nup=2 -p out.ps file.txt  ~(the same)
```

Put 4 logical pages on each page

```
enscript -U4 -p out.ps file.txt
```

Other nup options which may do what we need

```
--nup-columnwise
     Change the layout of the sub-pages in the N-up printing from row-
     wise to columnwise.
--nup-xpad=num
     Set the page x-padding of the n-up  printing  to  num  PostScript
     points.  The default is 10 points.
 --nup-ypad=num
     Set  the  page  y-padding  of the n-up printing to num PostScript
     points.  The default is 10 points.
```

3.13 Printing Code

Enscript can be used to provide nice listing of software 'code'.

Show what code languages enscript can pretty-print

```
enscript --help-highlight | grep 'Name:' | less
```

```
enscript --help-highlight | less ~(show more detailed information)
```

Create a colourised webpage of the file "Point.java" with syntax highlighting

```
enscript -whtml --color -E -p file.html Point.java
```

Print a gaudy header, two columns, landscape, code highlighting, 2-up printing.

```
enscript -G2rE -U2 foo.c
```

When a line is too long and must be wrapped print an arrow at the line end

```
enscript --mark-wrapped-lines=arrow -p output.ps list.txt
```

```
--mark-wrapped-lines={plus|box|arrow}
```

Section 4

Other Plain Text Documentation Systems

This book tries to elucidate the forest of different ways and formats in which to write text documentation and the software which is used to transform the text into other digital formats and printed documents.
commandlinefu.com

Section 5

Man Pages

man: software <man>ual pages
Man ('manual') pages is the traditional Unix documentation format. The man page format is a simple plain text format using macros for the troff or groff document system. The Man documentation system is specifically designed for documenting software.

`http://www.schweikhardt.net/man_page_howto.html`
links to documents by Kernighan ...
`http://babbage.cs.qc.edu/courses/cs701/Handouts/man_pages.html`

5.1 Viewing Man Pages

Viewing a man page with 'man' (the normal way to view a man page)

```
man program    ~(the file program.n must be in the manpath)
```

Viewing the bobble manpage with groff (useful when writing a man page)

```
groff -man -Tascii bobble.1 | less
```

```
groff -man -Tutf8 bobble.1 | less
```

Viewing the 'bobble' program man page with nroff

```
nroff -man bobble.1 | less
```

View the raw groff text for the 'bobble' program man page.

```
zless .../man/man1/bobble.1.gz    ~(if the pages are compressed)
```

```
less .../man/man1/bobble.1        ~(if the pages are not compressed)
```

(useful to learn how to write a man page.) (type 'manpath' to find the path to the manpages)
Other viewing tools are xman, tkman, info

5.2 Writing Man Pages

See the conventions for writing a man page

```
man 7 man
```

See the formatting codes available

```
man groff_man
```

- ⋆
- ⋆ decide what section the page should go in (see:man page sections)
- ⋆ write the man page
- ⋆ name the file 'name.n' where 'n' is the section
- ⋆ place the file in .../man/manN/ where 'N' is the section (same as above)
- ⋆ if the page is not written in english, place the file in the folder .../man/aa/manN/ where 'aa' is a language or locale code, for example 'fr' for french, or 'es' for spanish. and 'N' is the section.
- ⋆

5.3 Man Formatting Conventions

While the author of a unix man page is free to format the document as she feels fit, certain conventions are time-tested and true.

In the synopsis section use bold alternating with italic for functions

```
.BI "myfunction(int " argc ", char **" argv );
```

File names in italic outside of the 'synopsis' section

```
.I /usr/include/stdio.h
```

In the synopsis section, make file names bold

```
.B #include <stdio.h>
```

Put references to other man pages in bold alternating with roman

```
.BR man (7)
```

Format acronymns in a small font

```
.SM UNIX
```

Summarize command options with .IP

```
.IP -b
```

5.4 Man Formatting Codes

View the man macro help for a list of formatting codes

```
man groff_man   ~(as usual, this contains no examples)
```

Create a document title

```
.TH FOO 1 "MARCH 2003" Linux "User Manuals"
```

(command name: foo, section: 1 (user commands)) (last revision: march 2003, other title: Linux ...)

Section heading

```
.SH heading
```

Subsection headings

```
.SS heading
```

Display a sentence in italics

```
.I "some words"
```

Put the word 'type' in italics

```
/fItype/fP          ~(groff codes)
```

Insert preformatted text in a man page (using groff requests)

```
.nf
  pre formatted text ...
.fi
```

Begin a new paragraph

```
.PP
```

Indent an option list with .TP (the first line after .TP is the label)

```
>
```

.TP \-h display a short help text .TP \-d use the given device <<< (.TP has the advantage over .IP that the paragraph label can be formatted)

Hanging left indented paragraph

```
.HP
```

Indented paragraph with label

```
.IP label
```

this can be used for summarizing options etc. If the label is long then the indented paragraph is begun on a new line (definition list style) other wise the indented paragraph is begun on the same line. (table style) example: .IP -b turn on debug mode .IP "-config file" load the specifed config file –;

Indent the following text by the default amount

```
.RS
```

Indent the following text by 10

```
.RS 10
```

Stop indenting text

```
.RE stop indenting
```

5.5 Standard Man Page Document Sections

A comment with the groff command

```
>
```

.\" Process this file with .\" groff -man -Tascii foo.1 .\" <<<

The title of the document

```
.TH FOO 1 "MARCH 1995" Linux "User Manuals"
```

Name, the name of the command etc

```
>
```

.SH NAME boggle \- frobnicate the bar library <<<

Synopsis, summary of the command usage

```
>
```

.SH SYNOPSIS .B boggle [-bar] [-c .I config-file .B] .I file .B ... <<<

Description, a complete description of what the command does

```
.SH DESCRIPTION
.B boggle
frobnicates the bar library by tweaking internal
symbol tables. By default it parses all baz segments
and rearranges them in reverse order by time for the
.BR xyzzy (1)
linker to find them. The symdef entry is then compressed
using the WBG (Whiz-Bang-Gizmo) algorithm.
All files are processed in the order specified.
```

Options, the options which the command accepts

```
.\" this could be formatted with a .TP list as well
.SH OPTIONS
.IP -b
Do not write 'busy' to stdout while processing.
.IP "-c config-file"
Use the alternate system wide
.I config-file
instead of
.IR /etc/foo.conf .
```

Files, other files which the command uses (a .TP list could be used)

```
>
```

.I /etc/foo.conf .RS The system wide configuration file. See .BR foo (5) .RE .I ~/.foorc .RS Per user configuration file. <<<

Environment, any environment variables which affect the program

```
>
```

.IP FOOCONF If non-null the full pathname for an alternate system wide .IR foo.conf . Overridden by the .B -c option. <<<

Diagnostics, messages which the program may display

```
.SH DIAGNOSTICS
The following diagnostics may be issued on stderr:
Bad magic number.
.RS
The input file does not look like an archive file.
.RE
Old style baz segments.
.RS
.B foo
can only handle new style baz segments. COBOL
object libraries are not supported in this version.
```

Bugs, problems with the program

```
.SH BUGS
The command name should have been chosen more carefully
to reflect its purpose.
```

Author

```
>
```

.SH AUTHOR bob good <who at server dot net> <<<

See also, other programs or documents relevant to this one.

```
>
```

.SH "SEE ALSO" .BR bar (1), .BR boggle (5), <<<

5.6 Standard Location Of Man Pages

Man pages should be placed in their appropriate section (subfolder) and with the correct extension

`http://www.schweikhardt.net/man_page_howto.html`
: a man page howto with links to all the manuals

See where the 'man' command searches for man pages

```
manpath
```

File name and location of the user command 'boggle' man page

```
.../man/man1/boggle.1
```

```
.../man/man1/boggle.1.gz   ~(man pages can be compressed as well)
```

(the file boggle.1 is a plain text file)

File name and location of the man page for the library function 'doogle'

```
.../man/man3/doogle.3
```

Location of the man page written in french for the library function 'doogle'

```
.../man/fr/man3/doogle.3
```

Search for man pages in 'folder' instead of in 'manpath'

```
man -M folder boggle
```

(useful while writing and testing man pages)

manual page sections

1	User commands executable from a shell
2	Kernel system calls
3	Library functions and calls (such as libc)
4	Special files such as in /dev
5	File formats and conventions (such as /etc/password)
6	Games
7	Conventions and miscellaneous, file system layout.
8	System management commands such as like mount(8)
9	Kernel routines. this is an obsolete manual section.

Automatically generate man pages from C, C++, Java, Python ... source code

```
doxygen    ~(doxygen can also generate documentation in html, latex etc)
```

A cgi script for generating html man pages

```
man2html
```

5.7 Converting Man Pages To Other Formats

Convert the 'doogle.1' man page to html

```
man2html doogle.1
```

Convert to html using groff

```
groff -Thtml /path/to/manpage.n
```

Rman can convert to LaTeX, rtf, sgml, html, ...

```
rman
```

Convert the 'ls' manpage to plain text, converting tabs to spaces (-x)

```
man ls | col -bx | less
```

Convert a man page to postscript

```
groff -man -Tps /path/to/manpage.n
```

Convert a man page to dvi

```
groff -man -Tdvi /path/to/manpage.1
```

Convert a man page to pdf

```
man -Tps ls | ps2pdf - ls_man.pdf
```

5.8 Converting To Man Pages From Other Formats

Convert a perl documentation doc (pod) to a man page

```
pod2man bogg.pod > bogg.1
```

Section 6

Groff And Troff

Troff: <T>ypesetting <R>un <Off> system Groff:
Groff is the Gnu version of the Troff document formatting and typesetting system, which was maintained for many years by Kernighan

`http://www.kohala.com/start/troff/troff.html`
: a list of resources and documents for troff

6.1 Using Tables

Use the tbl macros

6.2 Diagrams And Pictures With Groff

Creating certain sorts of diagrams with groff can be achieved with the 'pic' preprocessor.
http://www.kohala.com/start/troff/cstr116.ps Brian Kernighans manual for pic
http://www.kohala.com/start/troff/gpic.raymond.ps A manual for gpic the gnu version of 'pic'

View the help for the 'pic' diagram generator

```
man gpic
```

Convert a pic diagram to postscript

```
groff -e -p -ms pic.ms > file1.ps
```

6.3 Mathematical Equations

Use the 'eqn' macros

Section 7

Texinfo

Texinfo is the official documentation system of the Free Software Foundation.

Section 8

Docbook

Docbook is an XML based documentation system

Section 9

Halibut

```
http://www.chiark.greenend.org.uk/~sgtatham/halibut/
```

- ⋆ halibut, uses a terse plain text input format
- ⋆ the output formats for halibut are: (:Plain ASCII text, HTML, Unix man page format, GNU info format, PDF, PostScript, Old-style Windows Help (.HLP);
- ⋆ not fully unicode

halibut input format, (text is placed between the braces)

\e{}	Emphasis
\c{}	Code
\cw{}	Weak code
\cq{}	Quoted code
\W{http://www.google.com/}{Google}	Url markup

Section 10

Source Code Documentation Systems

10.1 Doxygen

10.2 Javadoc

Javadoc is the document system for the Java programming language. The output format for javadoc is normally HTML

Pod perl documentation