#*

`mark.latex.simple.pss` This version does not currently recognise lists.

This script attempts to parse a "markdown"-like syntax (this script header is an example of the document format) and to transform it into LaTeX source code. The script runs on on the "pep" parsing machine which is implemented at HTTP://BUMBLE.SF.NET/BOOKS/PARS/ . There are other ways, which may seem better or more straight-forward of achieving this. The ANTLR parsing system can be used to right grammars that can parse markdown like structures, or just regular expressions can be used. But this is a good exercise for the parse machine, and more complex structures can be recognised than with plain regular expressions.

Required tokens are, at least: ¿¿* —* codeline* codeblock* link* file* quoted* emline* uutext* uuword* nl* word* text* [[* (for images) We dont actually need heading* and subheading* tokens because they get transpiled (into LaTeX) as soon as they are seen in the document. Also, dont need link/file/quoted/star

1 Status

Producing nice readable output with pdflatex.

Would like to include date lists with descriptions. eg 24 july 2022 did something 26 jul 2022 more stuff

The script is now generating compilable tex output with:

```
pep -f eg/mark.latex.pss pars-book.txt > test.tex
```

```
pdflatex test.tex; pdflatex test.tex
```

This produces a pdf file "test.pdf" and the formatting includes lists.

A work in progress. The document parses as text, but not all structures are recognised. This file contains an interesting way of resolving or removing parse tokens when they are no longer significant- it uses a negative approach such as...

```
# codeblocks with no caption (description)
!"codeblock*".E"codeblock*".!B"emline*" {
  clear; get; add " "; ++; get; --; put; clear;
  add "text*"; push; .reparse
}
```

Another interesting idea: use the accumulator as a state marker when parsing in code-blocks*. So if acc¿0 then dont create uuword or other tokens.

2 Token list

Tokens currently used by this script

```
--- >> 4dots codeblock codeline emline nl text uutext uuword word
```

For lists need: - dash, bl, list eg: o/- -¿ olist, olist, text, nl, dash -¿ list list, text, bl -¿ But why have bl???? just used nl nl because nl/nl is reduced immediately.

star* has been eliminated by parsing immediately, and ¿¿ and — could also be elim-

inated. Probably need to add bl*=blankline dash* and ulist/olist/dlist for unordered lists, ordered lists and definition lists

Useful grammar analysis.

Get a unique list of tokens used during parsing
`pep -f eg/mark.latex.simple.pss pars-book.txt` sed '/

3 Bugs

Strange pep/nom segmentation fault with multiline comment.

4 Notes

Add images, datelists, inline code? Convert this grammar to generate html/markdown etc

Need to tidy up description lists.

Nested lists may just work- out of the box! But a different list terminator (not blankline) would be handy.

What about inline code? This script needs to parse ***any*** text successfully! Even text that is not in any particular format.

May need to add "quoted" to handle quoted text, but not really necessary at the moment.

Using o/- O/- u/- U/- d/- D/- to start ordered/unordered/description lists!

5 Document format

This is a document format I have used in many code and booklet files. It is a kind of markdown, with even less markup than markdown. It would be useful to try to transform it to markdown.

A Description of the format - ALL UPPER CASE LINE Is a heading or UPPER CASE WITH "QUOTES"

ALL UPPER CASE LINE WITH FOUR DOTS .... is a subheading
asterixes before and after a word eg *this* emphasises the word.
¿¿ a line starting with 2 ¿ is a code line
a block surrounded by — and ,,,, on their own lines is a code block
*** a line starting with a star** describes the code line or block which follows
urls and filenames should get a special format. empty

d/- : makes a description list
term : end with a colon
machine or with a newline
end the whole list with a blank line empty

1. or O/- or 0/- makes and ordered
2. list
3. end with a blank line

4. u/- makes an unordered list.

Key names are rendered as keys. eg Enter is rendered as a keyname. urls are turned into links. Filenames are made into fixed-pitch font.

6 Latex

Need to escape all these chars in latex

```
& % $ # _ { } ~ ^ \
```

The first 7 add a backslash eg \& The last 3 do \textasciitilde, \textasciicircum, and \textbackslash

\begin{verbatim}...\end{verbatim}

Look at books/format-book/booktocgi.sed for lots of latex tricks and tips

7 History

9 July 2022 ordered and unordered and description lists appear to work well. Made emphasis lines (starting with star) appear on a line by themselves. This means they can be used for as a simple list. Also, introduced a bl* blankline token which will be used with lists (to terminate a list). Made the parser more permissive. 8 July 2022 Removed the star* token and parsed immediately in the word block. But didnt remove ¿¿* because nl* tokens cause problems. 7 july 2022 A lot of progress. Had the idea to use the accumulator to count words on a line and so be able to tell what is the 1st word. This could allow to displense with a number of tokens, and simplify the grammar.

Output now seems acceptable apart from lists not working. Revisiting this to try to make a nice pep/nom book. Cant use a table in a figure environment. This version `mark.latex.simple.pss` is actually better than `mark.latex.pss`

17 June 2021 Adding emphasis words such as ***this*** . Need to do lists with - and maybe dates, images with floating and resizing. Better code listings

14 June 2021 Looked at how to escape chars. Done. We also escape chars that are in star lines and maybe codelines, but the \verbatim and \lstlisting words seem to take care of special characters in those blocks.

I should try

```
">>*star*" --> "starline*"
"starline*word*" --> "starline*"
"starline*uuword*" --> "starline*"
```

This is in contrast to the technique of immediately reading to the end of the line with "until" or "whilenot"

3 june more work. proceeding well.

2 june 2021 Much progress, a basic latex doc has been formed. Now to refine and introduce new tokens. Have done sections, subsections, and codeblocks. Found an elegant way to eliminate insignificant tokens using a "negative" logic.

Continuing to write this. The unstack;print;stack trick is very useful. Parsing and printing text with just upper case headings. This seems a promising incremental way to parse

1 June 2021 Try the unstack trick at the parse label to debug token reductions. I fixed the "stack" command in `object/machine.interp.c` so that it updates the tape pointer.

23 April 2021 Continuing to work on the script and look for ways to simply debug it. I have come to the idea that ".reparse" should also print the stack if a particular switch is set. This would allow the stack reductions to be easily followed.

21 April 2021 Script begun. This is a second attempt.

vvvvvvv