NAME

pud-base − a description of the Portable Unix Documentation base language

The macros in this package have been ported to both HTML and troff.

DESCRIPTION

pud-base − A description of the Portable Unix Documentation (PUD) base language. The macros in this package have been ported to both HTML and troff. This package is used by the PUD mini-language for authoring manual pages (**pud-man**) and the PUD language for FAQ authoring (**pud-faq**).

There is a small list of known issues in the **ISSUES** section, mostly concerning the troff device. These should generally be of no concern at all, but if you run into trouble look there first. A quick glance through the list *before* you run into trouble may be the wisest thing to do.

INTRODUCTION TO THE ITEMIZE ENVIRONMENT

The *itemize* environemnt is the PUD workhorse for lists, enumerations, itemizations, and other tailed creatures. A simple and valid use is for example

```
 \begin{itemize}
     \item{\bf{foo}}
        For I am foo.
     \items{
        {\bf{barra}}
        {\bf{zuttelezut}}
     }
        For we are bar and zut.
  \end{itemize}
```

This source result in the following output:

foo
    For I am foo.

barra
zuttelezut
    For we are bar and zut.

This is not impressive at all, but it gives an idea of how itemize works. The following example is a single itemize environment providing a rollercoasterride through most of the features of the itemize environment. As shown below, it is possible to change all the itemize settings and styles at will even within a single itemize instance. Of course this is not useful at all except for demonstrating the *itemize* capabilities, but it goes to show that the itemize macros are quite robust (by virtue of modularity).

NOTE
The entire listing below was put in PUD's *spacing* environment, described further below. The environment was used to create extra margins on the two sides. .

1 Several spacing modes (contiguous, compact, indent). The mode in the current list is both compact and contiguous. Compact means that the itemize token and the ensuing text are on the same line. Contiguous means that there is no paragraph skip between different item-description pairs. Below, compact mode is switched off (approximately) halfway.
2 Several item modes (custom, mark, enumeration).

3 Several enumeration modes (roman, arabic, alphabetic).
iv) The style of a list can be changed while in the middle of it.
v) Nuther item.
vi) The list can be 'interupted' and resumed (by means of the `\intermezzo#1` macro).

Perhaps you wonder what good is THAT for, and justly so. The `\intermezzo#1` macro should only be used inbetween different items, i.e. it should *not* split content belonging to a single item.

[7] Items can be optionally and automatically right and/or left delimited. The current item is delimited with square brackets.
[8] Items can be left or right aligned.
[9] Items can be stacked, which is supported only when compact is off.
[10]
Beginning with this item, compact is off.
Implying
That
Stacking
is now possible.
[12]
(back to right-align) The itemcounter just keeps running by the way.
[18] (back to compact) But the counter can be manipulated at will.

- A bullet item, with an ugly bullet in HTML. Unfortunately the `•` entity is not widely supported yet. I decided to stick with a plain asterisk until that time arrives.

- We now switch to contiguous=0 mode (affecting the current list), and start a new list that is contiguous to the present text (by setting margintop to 0).
  a. Hubris
  b. Laziness
  c. Impatience

  Are the three virtues of programming.

- This concludes a listing showing most of the itemize capabilities.

USING THE ITEMIZE ENVIRONMENT

You steer the itemize environment by providing it with tag-value pairs like so:

```
\begin{itemize}{
   {compact}{1}
   {contiguous}{1}
   {align}{right}
   {type}{abc}
   {rp}{.}
}
```

This is the list of tags that you may use.

margintop
Top of table, anomalous unit (ems), default 0.

contiguous
Don't put paragraph skips inbetween items, default 0 (do it).

compact
Put item and description on same line, default 0 (don't do it).

w1
Width of item column in ems.

w2
Width of separating column in ems.

mark
E.g. • (if type=mark), affects `\item`.

align
One of left or right (item alignment), default left.

debug
Shows underlying table structure in html.

lp
What's printed immediately to the left of an item.

rp
What's printed immediately to the right of an item.

type
One of mark, roman, abc, arabic, affects `\item`.

itemcount
The count of items seen so far, e.g. 11 right now.

You need to know that the itemize environment internally maps these tags to dollar keys simply by prepending a dollar. Thus, if you want to reset one of the values associated with such a tag, you need to do e.g.

```
\set{$align}{right}
\set{$itemcount}{30}
```

THE SPACING ENVIRONMENT

Its syntax is identical to that of the *itemize* environment. It accepts tags *left*, *right*, *top*, and *bottom*. These should receive numeric values. The associated unit is millimeter.

The troff device does not yet support the *top* and *bottom* tags.

MACROS

\enref#2
\iref#2
\lref#2
\aref#2
\httpref#1
\sibref#1
\sibref#2
\sibref#3

`\enref#2` *creates* a link for which the first argument is the anchor and for which the second argument is the content (which can be left empty). `\iref#2` takes such an anchor as the first argument and it takes content that carries the link as the second argument. `\lref#2` takes a file name (possibly including a relative or absolute path) as the first argument and content as the second argument. `\aref#2` takes a URL (later possibly a URI) as the first argument and content as the second argument. `\sibref#2` takes a label as argument which presumably is the name of some application. It may append an extension depending on the current device, and it assumes that `label + extension` is the name of a file in the current directory. The second argument is displayed in the text. For `\sibref#1` the displayed text is the same as the label. For `\sibref#3` the second argument is an additional anchor within the file being linked to, and the third argument is the displayed text. `\httpref#1` simply prints a URL which will be active when html is output.

\par#1
\cpar#2
\car#1
\ccar#2

These are all paragraph macros that carry the paragraph content as the last argument. The first argument of `\cpar#2` and `\ccar#2` is the caption. These macros will ensure wellformedness for devices that support it, such as HTML. For more information on the differences between these macros consult the entries below.

\par
\cpar#1
\car
\ccar#1

use `\car` where you don't need a paragraph skip, but just need to indicate that you are in text mode again. You can simply always use `\par` and never use `\car`. If you care about the details of spacing though, or if you have particular trouble for example in creating an itemize environment where you do not want top and bottom margins, then it could be worthwile to turn to `\car` under very specific circumstances. Examples for using `\car` are:

- After an environment that always carries a bottom margin.
- After a caption that always carries a bottom margin, such as most sectioning commands (e.g. `\sec` in the manual macros).
- After an environment that does *not* carry a bottom margin, and where you specifically want the environment to be contiguous to the enclosing text. The listing you are currently reading is an example of this.

As promised. The `\car` macro may feel a little unusual. If you don't mind standing the chance of a little spurious vertical white-space just use `\par` all the time. If you really need it, such as in an 'inline' listing as above, the `\car` macro is ready to do the job.

\bf#1
\it#1
\tt#1
\v#1
\ftinc#2
\ftdec#2

The first four items set their argument in the **font** *specified.* `\tt#1` and `\v#1` both denote a typewriter font. *These macros should not be nested if troff is to be among the output devices*. Support for the last two items is not yet very robust. They temporarily increment respectively decrement the font by the amount of the first argument and apply the resulting setting to the second argument.

\verbatim#1
\verbatix#1

Make the device output the contents verbatim in a mono-spaced font, obeying spaces and newlines. This does not prohibit expansion of zoem macros, use `\protect#1` for that. The macro `\verbatim#1` will create a non-breaking environment.

\"html::charset"

Set this e.g. to ISO-8859-5 or some other acceptable charset label. Do this *before* you use the `preamble` key from the base package that you are using (e.g. the `man.zmm` or `faq.zmm` macros).

ISSUES

Nesting

Do not nest `\bf#1`, `\it#1`, `\tt#1`, or `\v#1` macros if troff is among the output devices. It will yield unexpected results.

The rest of this list pertains to the *itemize* environment.

Boldness

There is currently not a way to get bold (Roman) numerals using the automatic enumeration mode within *itemize*, or to change the appearance of any of the other enumeration types.

Margins

If you use fractional values for `w1` and `w2` in the itemize environment, you may well run into a problem with troff (c.q. nroff) Suppose you use `w1=1.6` and `w2=0.8`. Nroff will round both these values as well as the sum of the two, thus incrementing by `2` and `1`, followed by a decrement of `2`.

Misc

Refrain from doing

```
\begin{itemize}
\begin{spacing}
\item{foo}
FOO
\item{bar}
BAR
\end{spacing}
\end{itemize}
```

Because the `\item#1` invocations will update keys in the dictionary pushed by the *spacing* environment rather than the dictionary beneath it that was pushed by the *itemize* environment. On the other hand, you *can* do this:

```
\begin{spacing}
\begin{itemize}
\item{foo}
FOO
\item{bar}
BAR
\end{itemize}
\end{spacing}
```

The reason being that the *spacing* environment does not facilitate associated keys within the dictionary scope it creates. Compare with this with the *itemize* environment, that facilitates the use of `\item#1`, `\item`, and `\items#1`.