

HH Gets Carried Away

hhmuf, hhflxbox and hhcount

Herman Haverkort

herman@fgbbs.iaf.nl

March 1995

Abstract

This article presents hhmuf's multinotes: special cheery footnotes to be used in special situations, including so-called 'forbidden environments'. Then it presents hhflxbox's self-scaling frames: encircling macros are provided but you can define enwhatevering macros yourself by means of the macros provided by hhflxbox. Finally hhcount is presented: macros to handle simple and composite counters in a fancy way.

Keywords: circling, counter, dice, footnote, forbidden environment, frame, index

In this article I present some of the features of a few packages I wrote recently. However this presentation is far from complete. A more detailed manual and the packages itself can be obtained from FGBBS.¹ To use the packages described here additional packages are needed. You will get everything you need if you request the file hh.arj from FGBBS. I will try to submit the hh packages to CTAN as well. Note that L^AT_EX 2.0.9 versions are not available.



1 Some Features of the hhmuf Package

1.1 Reusable Footnotes and Recycled Markers

Suppose you have to typeset some tables containing many entries which are amplified by footnotes outside the table. Several entries, possibly but not necessarily in the same table, refer to the same footnote. If references to the same footnote appear in different tables on different pages, the footnote text should be set on all pages involved, while the footnote marker should be the same each time the footnote is set. The first to avoid unnecessary turning over, the second to avoid confusion.

Typesetting such tables is not very easy in basic L^AT_EX. A relatively easy way to do the job is probably as follows:

1. first get markers for all the footnotes and define macros to set the footnote markers;
2. then define a macro `\tablenotes` which typesets the footnotes;
3. then typeset the tables, using the macros defined in the steps mentioned above;

Here is some example input:

```
% step 1:
```

```
\newcommand\getfootnotemark[1]{%
  \stepcounter{footnote}%
  \newcounter{#1}%
  \setcounter{#1}{\value{footnote}}%
  \expandafter\newcommand\csname #1\endcsname
    {\footnotemark[\value{#1}]}%
  \getfootnotemark{notea}%
  \getfootnotemark{noteb}%
  \getfootnotemark{notec}%

% step 2:
\newcommand\tablenotes{%
  \footnotetext[\value{notea}]{%
    First example footnote}%
  \footnotetext[\value{noteb}]{%
    Second example footnote}%
  \footnotetext[\value{notec}]{%
    Third example footnote}}

% step 3:
\tablenotes
\begin{center}
\begin{tabular}{|l|l|}%
\hline
name & amount in \$ \\\
\hline
Achterberg & 100 & \\\
Bosman & 150\notea & \\\
Evers & 125\noteb & \\\
Gerritsen & 145 & \\\
Hooier & 170\notec & \\\
Jansen & 165\notea & \\\
\hline
\end{tabular}
\end{center}
```

¹FGBBS — tel. (085) 21 70 41

And the resulting output:

name	amount in \$
Achterberg	100
Bosman	150 ²
Evers	125 ³
Gerritsen	145
Hooier	170 ⁴
Jansen	165 ²

This article is too short to demonstrate it, but when typesetting multiple tables this way problems are likely to arise. If you typeset another table on the same page, calling `\tablenotes` again will cause the footnote texts to be typeset twice on the same page. If you typeset another table on another page, *all* table footnote texts will be typeset on that page, even if the table which is on that page does not refer to all of them.

The `hhmuf` package solves these problems, although the solution of the twice-on-the-same-page problem may be buggy in rare contexts. If you use the `hhmuf` package you can replace the previous listing by the following:

```
\mufhire note1:{First example footnote}%
\mufhire note2:{Second example footnote}%
\mufhire note3:{Third example footnote}%
%
\begin{center}
\begin{tabular}{|l|l|}%
\hline
name & amount in \$ & \\
\hline
Achterberg & 100 & \\
Bosman & 150\muf note1:{} & \\
Evers & 125\muf note2:{} & \\
Gerritsen & 145\muf:{Example
of an incidental note} & \\
Hooier & 170\muf note3:{} & \\
Jansen & 165\muf note1:{} & \\
\hline
\end{tabular}
\end{center}
```

And get this result:

name	amount in \$
Achterberg	100
Bosman	150 [△]
Evers	125 [⊖]
Gerritsen	145 [×]
Hooier	170 [♣]
Jansen	165 [△]

²First example footnote

³Second example footnote

⁴Third example footnote

△ First example footnote

⊖ Second example footnote

× Example of an incidental note

♣ Third example footnote

⁵which occurs frequently in my ever changing text

You will note that `\mufhire label:{footnote text}` is used to define footnotes. Its opponent is, of course, `\muffire label:`, which undefines footnotes, while `\muf label: { }` is used to set previously defined footnotes. As shown in the example, `\muf : {footnote text}` can be used to set incidental footnotes. `\muf : {footnote text}` actually acts as an abbreviation for hiring, typesetting and firing an incidental note. Thus it is well-suited for setting normal in-text footnotes.

`hhmuf` does not use common footnote markers. Most common sets of symbols have a well-defined order which makes them ill-suited for `hhmuf` since the `hhmuf` macros do not respect that order. While defining footnotes, markers are assigned in turn. Thus there is no need to restart footnote numbering every chapter or every page, because you never run out of markers, unless you hire a lot of them without ever firing any. Restarting footnote numbering does not make sense anyway because there is no such thing as a *first* marker.

The set of markers used by `hhmuf` can be fully specified by the user, either by selecting one of the predefined sets or by compiling a new one. The predefined sets are the following:

set name	markers included
black	•◆▼♣■▲◀♠
circlox	⊙⊘⊗⊕⊖⊗⊘⊙⊗⊕⊖
fuss	*◇⊗*#*♣♠∞⊙
geometry	◆□▼△■◀▲◇◀▼
misc	♠△⊖♣×◇⊗⊙∇⊕∞*⊙+◀T•∇
music	#b‡
strokes	T × Y + √ H ¯ }

For details see the documentation in the package file.

1.2 The Forbidden Environment Problem

Suppose I typed several pieces of text that have to be typeset all in the same special way. For that purpose I (re)defined an environment `specialtext`:

```
\renewenvironment{specialtext}%
{\par$\bigtriangledown$\par}%
{\par$\bigtriangleup$\par}
```

▽

If I have a piece of text like this paragraph, which refers to a footnote⁵, this should cause me no problems.

△

Then I defined a package option in the package I used to typeset my document. If I specified that option when loading the package, then `specialtext` would be defined as follows:

```
\renewenvironment{specialtext}%
  {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
  {\egroup
   \begin{center}\fbox{\box0}\end{center}}
```

Now problems arose. I typeset the same text again:

If I have a piece of text like this paragraph, which refers to a footnote⁶, this should cause me no problems.

The paragraph is shown framed all right, but something went wrong with the footnote. With the new definition of `specialtext`, the footnote suddenly appears in a ‘forbidden’ environment and therefore it actually disappears. Although the in-text marker is typeset, there is no note at the foot of the page.

While writing this article I discovered Kresten Krab Thorup’s style file `ftn.sty`⁷, which attempts to solve this problem but does so quite buggily. Footnotes disappear when forbidden environments are nested. When multiple footnotes are type-set in forbidden environments, footnotes are repeated and their numbering is wrong. A modified version of `ftn.sty` exists (by Zdenek Wagner), which solves the repetition problem correctly, and suppresses incorrect numbers by omitting them: at least that is what I got. I tried to contact Krab Thorup about making `ftn.sty` more robust, but did not succeed so far. Nevertheless Krab Thorup’s style file contained some very useful ideas, which I combined with my own ideas to construct a set of macros which seem to be quite robust. I will now show you the result: how easy it is to use *hhmuf*’s footnotes in forbidden environments.

The kind of unpleasant surprises presented above is easy to prevent when typesetting footnotes with `\muf` instead of `\footnote`, like in:

```
\begin{specialtext}
If I have a piece of text like this paragraph,
which refers to a footnote\muf:{which occurs
frequently in my ever changing text}, this
should cause me no problems.
\end{specialtext}
```

When using the first definition of `specialtext`, this yields:

▽

If I have a piece of text like this paragraph, which refers to a footnote⁸, this should cause me no problems.

⁷ Available at CTAN as `macros/latex209/contrib/misc/ftn.sty`

⁸ which occurs frequently in my ever changing text

△

In the following example the second definition of `specialtext` is used, but I added one line of code to ‘protect’ the environment:

```
\renewenvironment{specialtext}%
  {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
  {\egroup
   \begin{center}\fbox{\box0}\end{center}}
\mufoff[specialtext]
```

And here is the result:

If I have a piece of text like this paragraph, which refers to a footnote⁸, this should cause me no problems.

Without changing the text in my document, I could redefine `specialtext` to use forbidden environments in such a way that my footnotes did not disappear. `\mufoff` did the job.

1.3 Shortcomings: minipage Fans Beware!

hhmuf does not support `minipages` yet. If `\muf` is used in a `minipage` environment, the footnote will be placed at the foot of the ‘master page’ instead of under the `minipage`!



2 The *hhflxbox* Package

hhflxbox contains a number of boxing macros. The kernel consists of `\iframe`, which boxes things and sets self-scaling frames around, and `\sframe`, which sets more complex self-scaling and -stretching frames. Besides *hhflxbox* provides the encircling macros `\ringbox`, `\bellybox` and `\outringbox` (which use `\iframe`), the macros `\sepbox` and `\separbox`, which set empty space around boxes, and `\broadbox`, which boxes its argument in a `\vbox` of which the width is the line width minus some specified value.

2.1 `\sepbox` and `\separbox`

For introduction of `\sepbox` and `\separbox` it is convenient to introduce `\bellybox` first. `\bellybox` is one of the *hhflxbox* macros which can be used to encircle things, for example ③, which is set with: `\bellybox : { 3 }`.

You probably notice that the circle around the digit is somewhat tight. This problem can be solved by putting a `\separbox` around the digit, like in `\bellybox`

: `\separbox{1pt}{3}`, which yields: ③. Actually `\separbox{dimension}{stuff}` puts *dimension* wide empty space around *stuff* on all sides.

A more general form is:

`\sepbox(leftspace,topspace,rightspace,bottomspace){stuff}` which adds empty spaces of the specified widths to the sides of the box containing *stuff*.

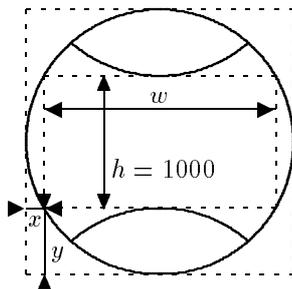
2.2 `\iframe`: Isomorphous Frames

`\iframe` is only a frame drawing *tool*: it does not draw frames itself but it can take care of the proper positioning and scaling of frames drawn by other macros. To explain the functioning of `\iframe` it is probably best to give an example of the development of a framing macro using `\iframe`.

Suppose we want to set self-scaling frames which have the following shape:



then we could imagine a box-shaped area in the frame which will contain the frame's contents (the inner dashed box in the figure below). Also we could imagine a box surrounding the frame (the outer dashed box in the figure).



Since `\iframe` expects the inner box height to be 1000 times the `\unitlength`, all dimensions have to be chosen so that the inner box height equals 1000 indeed. Then `\iframe` can scale the frame by setting the `\unitlength`. Furthermore `\iframe` expects the lower left corner of the outer box to have coordinates (0, 0). Taking these expectations in account we can design a macro which draws the frame:

```
\newcommand\sillyshape{%
\begin{picture}(2000,2000)
\thicklines
\put(1000,1000){\arc(0,1000){360}}
\put(1000,-498){\arc(662,749){83}}
\put(1000,2498){\arc(-662,-749){83}}
\end{picture}}
```

(`\arc` is defined in the `curves` package by I.L. Maclaine-cross.) Now we can define a silly shape framing macro by defining `\sillyframe` as: `\iframe\sillyshape(x,y){w}{0pt}\ifrch\ifrcv:{#1}`. Actually we will set `#1` in a `\sepbox(0pt,2pt,0pt,2pt){#1}` to prevent the frame from touching its contents. In the above example we have $x = 134$, $y = 500$ and $w = 1732$, so we write:

```
\newcommand\sillyframe[1]{%
\iframe\sillyshape(134,500){1732}{0pt}%
\ifrch\ifrcv:{\sepbox(0pt,2pt,0pt,2pt){#1}}}
```

Now we can put `\sillyframe{all}`, `\sillyframe{sorts}`, `\sillyframe{of}`, `\sillyframe{things}` in silly frames.

Now we can put ① ② ③ ④ in silly frames.

Note that I do not claim this kind of silly frame to be good-looking: it is just an example.

The dimension `0pt` in the example above determines the minimal height of the silly frame's inner box. Sometimes it is necessary to define it because \LaTeX 's picture environment suppresses small line segments.

The macro `\ifrch` determines what should be done if the frame's contents width/height ratio is too small. By specifying `\ifrch` we instruct `\iframe` to center the contents. Instead of `\ifrch` we could have specified `\ifrll` or `\ifrr` to have the contents flush left or right.

The macro `\ifrcv` determines what should be done if the frame's contents height/width ratio is too small. `\ifrcv` yields vertical centering, while `\ifrt` and `\ifrb` yield top and bottom flushing.

If we put frames around for example page numbers, then the self-scaling properties of isomorphous frames may have an unpleasant result: numbers of the same type, like page number ②① and page number ②⑤, might get differently sized frames because of their different natural sizes. This can be solved by redefining `\sillyframe` to specify a *unit name*, since all things typeset with the same unit name get equally sized frames. The unit name, for example `pagenr`, should be placed between the vertical alignment specification and the colon, like in:

```
\newcommand\pagenrframe[1]{%
\iframe\sillyshape(134,500){1732}%
{0pt}\ifrch\ifrcv pagenr:{%
\sepbox(0pt,2pt,0pt,2pt){#1}}}
```

framed numbers like `\pagenrframe{\oldstylenums{21}}` and `\pagenrframe{\oldstylenums{25}}`.

which yields (after compiling our document twice):

framed numbers like ②① and ②⑤.

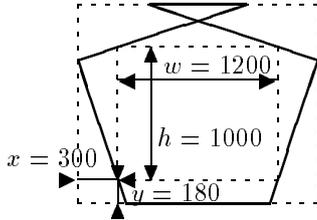
However, this is not fully satisfactory yet: now the frames are equally sized but the first frame is positioned higher than the second. This is no bug, it is a feature. No really, it is! It is, however, a sometimes unwanted feature. The solution is using `\lcenter` to center the frames on their line, like in:

```
pages \lcenter{\pagenrframe{\oldstylenums{21}}}
and \lcenter{\pagenrframe{\oldstylenums{25}}}
```

resulting in:

pages  and 

 As a final example of isomorphous frames, consider the following framing macro. Note that the inner box height is 1000 again, as expected by `\iframe`.



```
\newcommand\jarshape{%
  \begin{picture}(1800,1500)
    \thicklines
    \put(360,0){\line(-1,3){360}}
    \put(0,1080){\line(3,1){1260}}
    \put(540,1500){\line(1,0){720}}
    \put(1440,0){\line(1,3){360}}
    \put(1800,1080){\line(-3,1){1260}}
    \put(360,0){\line(1,0){1080}}
  \end{picture}}
```

```
\newcommand\jarframe[1]{%
  \iframe\jarshape(300,180){1200}{10pt}%
  \ifrch\ifrb:{\separbox{1pt}{#1}}}
```

2.3 `\ringbox`, `\bellybox` and `\outringbox`: Encircling

`\ringbox{optional unit name}: {stuff}` sets a circle around *stuff*. The specification of a unit name is optional; its use is explained above.

`\outringbox` is very much like `\ringbox`, but the following example demonstrates their difference:

```
1\ringbox:{2}3 and 1\outringbox:{2}3
```

yields:

1②3 and 1②B

If `\ringbox` is used, the circle contributes to the width, height and depth of the result. If `\outringbox` is used, the circle does not contribute any width, height or depth, so that the text is typeset as if the circle were not present and the circle were added after typesetting the text.

The result of `\bellybox` is a circle which contributes a bit to the dimensions of the encircled result but also sticks out a bit (by 10 percent of its radius to be sort of exact). So `\bellybox` is an intermediate form of `\ringbox` and `\outringbox`.

2.4 `\sframe`: Stretchable Frames

Putting `\sframe` to good use is a rather complex task. `\sframe` assembles user-defined frame components which actually are macros which set the values of a box and several dimension registers. Therefore I decided to

give only an example of what can be achieved in this article; for explanation see the manual and demo files available at FGBBS.

If one has defined suitable macros `\fancycolumn` and `\fancytympan`, one can define:

```
\newcommand\templebox[1]{\sframe
  [1]\fancycolumn [2]\fancytympan
  [1]\fancycolumn [-]\-%
  {\separbox{3pt}{#1}}}
```

which should be read as: after 3pt wide empty space is set around #1, first add columns to the left and the right, second put a tympan on top of the result, and never put anything at the foot. Then typing this:

```
\templebox{hello there!} and \templebox{%
  \vbox{\hbox{b}\hbox{y}\hbox{e}}}
```

will yield:



2.5 `\broadbox` for Setting Line Wide Frames

`\broadbox` can be useful to set frames that fill the line. Its use is best explained through an example. Suppose we want to set a paragraph of text in a line wide temple box. Then the lines will be filled by (from left to right): a column, empty space added by `\separbox`, text, empty space and a column. In other words: the whole line is available for setting text, except for the space needed by the columns and the empty space set by `\separbox`. The columns are 12pt each while `\separbox` adds 3pt wide empty space to the left and the right: that makes a total of 30pt. So we write: `\templebox{\broadbox{30pt}{\<broadbox> can be ...\textit{dimension}.}}`, which yields a paragraph typeset like this. So `\broadbox{dimension}{stuff}` sets *stuff* in a `\vbox` which has width line width minus *dimension*.

2.6 Environment Versions

Some of the macros defined in `hhflxbox` are available as L^AT_EX environments. For example: instead of `\broadbox{30pt}{text to be boxed}` one could also use `\begin{boxed}{30pt} text to be boxed\end{boxed}`. Similarly one could use the environments `sepboxed`, `separboxed` and `sframed` instead of the macros `\seplib`, `\separbox` and `\sframe`.

Actually I have to confess something: I lied to you about the typesetting of the section about `\broadbox`. I did it with:

```
\newenvironment{templeboxed}{%
  \begin{sframed}%
    [1]\fancycolumn [2]\fancytympan
    [1]\fancycolumn [-]\-
    \begin{separboxed}{3pt}
      \begin{broadboxed}{30pt}
    }{%
      \end{broadboxed}%
    \end{separboxed}%
  \end{sframed}%
}

\begin{templeboxed}%
  \langlebroadbox\rangle can be useful to set frames that
  : : : : :
  width line width minus \textit{dimension}.
\end{templeboxed}
```

I hope you will forgive me my cheating. I mean... without using the environments typesetting verbatim stuff is so troublesome...



3 Some Features of the hhcount Package

3.1 Simple Number Formatting

Let us start by summarizing the simple number formatting macros which are provided by hhcount:

example input	corresp. output	other example output
<code>\fctabdigit{2}</code>	2	29
<code>\fcolddigit{2}</code>	2	29
<code>\fcloweralpha{2}</code>	b	cc
<code>\fcbigalpha{2}</code>	B	CC
<code>\fcsmallalpha{2}</code>	B	CC
<code>\fclowerroman{2}</code>	ii	xxix
<code>\fcbigroman{2}</code>	II	XXIX
<code>\fcsmallroman{2}</code>	II	XXIX
<code>\fcbigromanlined{2}</code>	II	XXIX
<code>\fcsmallromanlined{2}</code>	II	XXIX
<code>\fcbigdice{2}</code>	□	□□□□□□
<code>\fcsmalldice{2}</code>	□	□□□□□□
<code>\fcbigscore{2}</code>		
<code>\fcsmallscore{2}</code>		
<code>\fcfnsymbol{2}</code>	†	

The next step in complexity are number formatting macros which give context-dependent output. This is implemented by using the *context switches* `\if@fcolddstyle` and `\if@fcsmall`, which are set by context switching macros like `\fcinheading` and `\fcintext`. We say that a context switching macro is active if it was the last one to affect the context switches.

example input	output when <code>\fcinheading</code> is active	output when <code>\fcintext</code> is active
<code>\fcdigit{14}</code>	14	14
<code>\fcalpha{14}</code>	N	N
<code>\fcroman{14}</code>	XIV	XIV
<code>\fcromanlined{14}</code>	XIV	XIV
<code>\fcdice{14}</code>	□□□□□□	□□□□□□
<code>\fcscore{14}</code>		

By default `\fcinheading` is active; `\fcintext` is active when using `\ref` or `\pageref` (those two macros are redefined by hhcount).

3.2 How to Define Composite Counters

I will now try to give an impression of the way in which composite counters can be defined using hhcount. However, this is *not* a manual. After reading the following paragraphs you may be able to hack a composite counter together yourself, by imitating what is done below and experimenting with some small modifications of your own. If you want to be taught how to use hhcount efficiently and effectively, then you should read the manual.

Suppose we want to set up a three-level section numbering system for some sub-document in another document, for example the rules of a club embedded in some booklet about that club. The section numbers should be composed from the values of three (sub-)counters: `ruleschapter`, `rulessection` and `rulesparagraph`. Chapter numbers should be represented by capital alphabetic characters; elementary section and paragraph numbers by arabic digits. What should be done?

First we select a *series identifier* for our composite counter. Series identifiers are natural numbers which are assigned to composite counters. Each composite counter should be assigned a unique identifier. Because identifiers 1 to 8 and 12 are reserved for common purposes we select 9 for our rules section numbers.

Then we define a macro which expands to the series identifier:

```
\def\rulesseries{9}
```

Next we specify how the three sub-counters are to be combined:

```
\combinecounters\rulesseries{%
  \{\ruleschapter}%
  \{\rulessection}%
  \{\rulesparagraph}}
```

And finally we define how the counter is to be formatted:

```
\setcounterformat\rulesseries{#1-#2-#3}{%
  \fcorfinally
  % capitals for chapter numbers:
  \fcformat{#1}{\fcalpha}%
  % digits for section numbers:
  \fcformat{#2}{\fcdigit}%
```

```
% a period to separate section and paragraph
% numbers:
\fcformat{#3}[.]%
% digits for paragraph numbers:
  {\fcdigit}%
\fcordespair}
```

If you want to understand the definition above, read the manual.

3.3 The Result

Now the composite counter can be accessed by the macros `\theruleschapter`, `\therulessection` and `\therulesparagraph`, which give results like: ‘A’, ‘A2’ and ‘A2.3’. The macros `\stepcounter{ruleschapter}`, `\stepcounter{rulessection}` and `\stepcounter{rulesparagraph}` can be used to step the counter.

When `\fcinheading` is active, rules paragraph numbers will be set like ‘A2.3’, but when `\fcintext` is active, the same number will be set like ‘A2.3’.

More complex distinctions in representation of counters are possible. `hhcount` provides a set of macros which can be used in the last argument of `\setcounterformat`. Those macros enable definition of counters which are set like ‘A2.3’ in headings and like ‘section A2, par. 3’ in text etc. For details see the manual.

3.4 `hhcount` and `makeindex`

Composite section numbers like ‘A2.3’ cannot be handled by the `makeindex` program. Besides `makeindex` has problems with sorting alphabetic numbers since it cannot determine whether or not it are roman numbers. `hhcount` provides a way to get around these problems.

All composite numbers defined by `hhcount` constructs are internally represented by a sequence of natural numbers, separated by hyphens and embedded in a macro call. A typical example is `\fancycounter 9-1-2-3-!`. The first number represents the series identifier (9 in the example), while the following numbers represent the values of the relevant sub-counters.

`hhcount` provides macros `\initfancycounters`, `\indextolabels` and `\indextopages`. The first redefines the section and page numbering systems to use `hhcount`’s composite counters. `\indextolabels` sort of redefines `\index` to use the redefined section numbers and strip the `\fancycounter` and the `-!` off the composite counter representation. `\indextopages` does the similar thing for page numbers. In both cases the result is a sequence of natural numbers, separated by hyphens, which can be handled perfectly well by `makeindex`. By

embodying the appropriate definitions in your index style (`.ist`) file `makeindex` will undo the stripping after sorting the page or section numbers, so that your index entries will still be typeset as defined by means of `hhcount` macros. Thus section numbers like ‘A2.3’ can be used for references in the index. Inserting equation, table and figure numbers in the index is just as easy. It is even possible to have different kind of composite numbers in the same index, for example page as well as section numbers, because the series identifiers are not stripped off so that it remains possible to determine the proper series and formatting of each composite number. For details see the manual.

3.5 Bugs and Deficiencies

Class files tend to make the \TeX compiler show on your terminal which chapter of your book or report is being processed. Error messages often contain the page number. When using `hhcount` there is a chance that the chapter and page numbers shown on your terminal look weird: you will be shown the internal representation of your counter (`\fancycounter 9-1-0-0-!` for example). This is caused by an incorrect timing of macro expansion: in this case `\fancycounter` is expanded too late, that is: not at all.

Late expansion with `hhcount` is typically a problem with error and other messages: I would be highly surprised if someone discovers something like `\fancycounter 9-1-0-0-!` outside verbatim environments in a typeset document. However, when compiling your document you might run into early expansion, which causes severe errors. With the latest version of `hhcount` this problem does not seem to emerge in ‘usual’ contexts; however I am not sure.

Front matter, appendix and back matter peculiarities (with respect to page and section numbering) are not automatically supported by `hhcount`. Class files are too different in that respect. If `hhcount` is to be used to handle the section and page numbering in documents containing front matter and appendices, it would probably be best to incorporate `hhcount` in the class file, instead of loading it as an additional package.

3.6 Gamesters Page Numbers

The following redefines the page counter so that page numbers will be set as dice (I designed this for a gamesters society):

```
\def\fcpageseries{12}
\combinecounters\fcpageseries{\{page}}
\setcounterformat\fcpageseries{#1}{%
  \fcorfinally
  \fcformat{#1}\fcdice
  \fcordespair}
```

I could not help to give you this as an final example.