2.8 Sub-page references

This is the wonderful code that Dave Love provided to make page references like 7a, 7b, and so on.

This code provides a mechanism for defining 'page sub-references' using `\sublabel{foo}` referenced with `\subpageref{foo}`. Sub-references will be numbered like these real examples: 18a, 18b, 18c etc. unless there is only one on the page, in which case the letter will be dropped like this: 4b.

To be able to use `\subpageref` we must define the label with `\sublabel`, used like label. (Using `\ref` with a label defined by `\sublabel` will produce the sub-reference number, by the way, and `\pageref` works as expected.) Note that `\subpageref` is robust and `\ref` and `\pageref` are redefined to be robust also, as they will be in future LaTeX releases. Incidentally, these expand to the relevant text plus `\null`—you might want to strip this off, e.g. for sorting lists.

There are various ways we could attack this task (which is made non-trivial by the well-known asynchrony of (La)TeX's output routine), but they all must depend on hacks in the `.aux` file or a similar one. Joachim Schrod's `fnpag.sty` does the same sort of thing differently to this LaTeX-specific approach. See `latex.tex` for enlightenment on the cross-referencing mechanism and the LaTeX internals used below. [DL: The internals change in LaTeX2e compared with LaTeX 2.09. The code here still works, though.]

The new-style LaTeX page-reference macros all work the same way: if the thing is undefined, barf. Otherwise, do the specified thing. We need to handle the fact that the expansion of the label may be two items or five items, depending on whether hypertext is used. Since we're only ever interested in the first two items, we use a hack—the "do the specified thing" must be defined as `\def\dome#1#2#3\\{...}` where `...` uses only the first two parameters.

18d ⟨*noweb.sty* 2b⟩+≡ ◁17c 18e▷

```
\newcommand\nw@genericref[2]{% what to do, name of ref
  \expandafter\nw@g@nericref\csname r@#2\endcsname#1{#2}}
\newcommand\nw@g@nericref[3]{% control sequence, what to do, name
  \ifx#1\relax
    \ref{#3}% trigger the standard 'undefined ref' mechanisms
  \else
    \expandafter#2#1.\\%
  \fi}
```

Much of what we want can be done by pulling out the first, second, or first and second elements of a ref.

18e ⟨*noweb.sty* 2b⟩+≡ ◁18d 19a▷

```
\def\nw@selectone#1#2#3\\{#1}
\def\nw@selecttwo#1#2#3\\{#2}
\def\nw@selectonetwo#1#2#3\\{{#1}{#2}}
```

The `\subpageref` macro first does a normal `\pageref`. If the reference is actually defined, it then goes on to check whether the control sequence `2on`⟨*page referenced*⟩ is defined and sets the `\ref` value to get `a` etc. if so. The magic, of course, is in defining the `2on` bit appropriately. `\subpageref` also tries to include the right hyperstuff for xhdvi.

19a ⟨*noweb.sty* 2b⟩+≡ ◁18e 19b▷
```
\newcommand{\subpageref}[1]{%
  \nwhyperreference{#1}{\nw@genericref\@subpageref{#1}}}
\def\@subpageref#1#2#3\\{%
  \@ifundefined{2on#2}{#2}{\nwthepagenum{#1}{#2}}}
```

`\subpagepair` produces a `{subpage}{page}` pair.

19b ⟨*noweb.sty* 2b⟩+≡ ◁19a 19c▷
```
\newcommand{\subpagepair}[1]{%  % produces {subpage}{page}
  \@ifundefined{r@#1}%
    {{0}{0}}%
    {\nw@genericref\@subpagepair{#1}}}
\def\@subpagepair#1#2#3\\{%
  \@ifundefined{2on#2}{{0}{#2}}{{#1}{#2}}}
```

`\sublabel` is like the `\label` command, except that it writes `\newsublabel` onto the `.aux` file rather than `\newlabel`. For hyperreferencing, all labels must be hypertext anchors, for which we use `\nwblindhyperanchor`.

19c ⟨*noweb.sty* 2b⟩+≡ ◁19b 19d▷
```
\newcommand{\sublabel}[1]{%
  \leavevmode % needed to make \@bsphack work
  \@bsphack
  \nwblindhyperanchor{#1}%
  \if@filesw {\let\thepage\relax
   \def\protect{\noexpand\noexpand\noexpand}%
   \edef\@tempa{\write\@auxout{\string
      \newsublabel{#1}{{}{\thepage}}}}%
   \expandafter}\@tempa
   \if@nobreak \ifvmode\nobreak\fi\fi\fi\@esphack}
```

`\nosublabel` creates a label with a sub-page part of 0.

19d ⟨*noweb.sty* 2b⟩+≡ ◁19c 20a▷
```
\newcommand{\nosublabel}[1]{%
  \@bsphack\if@filesw {\let\thepage\relax
   \def\protect{\noexpand\noexpand\noexpand}%
   \edef\@tempa{\write\@auxout{\string
      \newlabel{#1}{{0}{\thepage}}}}%
   \expandafter}\@tempa
   \if@nobreak \ifvmode\nobreak\fi\fi\fi\@esphack}
```

`\newsublabel` is the macro that does the important work. It is called with the same sort of arguments as `\newlabel`: the first argument is the label name and the second is `{`⟨*ref value* (never defined)⟩`}{`⟨*page number* (never defined)⟩`}`. (Note that the only definition here which needs to be global is the one which is, and that `\global` is redefined by `\enddocument`, which will bite you if you use it. . .)

20a ⟨*noweb.sty* 2b⟩+≡ ◃19d 21b▹
⟨*definition of* `\newsublabel` 20b⟩

Before we create a `\newsublabel` for the first time, we set the proper trailers.

20b ⟨*definition of* `\newsublabel` 20b⟩≡ (20a) 20c▹

```
\newcommand\newsublabel{%
  \nw@settrailers
  \global\let\newsublabel\@newsublabel
  \@newsublabel}
```

First we extract the page number into `\this@page`.

20c ⟨*definition of* `\newsublabel` 20b⟩+≡ (20a) ◃20b 20d▹

```
\newcommand{\@newsublabel}[2]{%
  \edef\this@page{\@cdr#2\@nil}%
```

Then we see whether it's changed from the value of `\last@page` which was stashed away by the last `\newsublabel` (or is `\relax` if this is the first one). If the page has changed, we reset the counter `\sub@page` telling us how many sub-labels there have been on the page.

20d ⟨*definition of* `\newsublabel` 20b⟩+≡ (20a) ◃20c 20e▹

```
  \ifx\this@page\last@page\else
    \sub@page=\z@
  \fi
  \edef\last@page{\this@page}
  \advance\sub@page by \@ne
```

If we've had at least two on the page, we define the `2on`⟨*page no.*⟩ macro to indicate the fact.

20e ⟨*definition of* `\newsublabel` 20b⟩+≡ (20a) ◃20d 21a▹

```
  \ifnum\sub@page=\tw@
    \global\@namedef{2on\this@page}{}%
  \fi
```

Then we write a normal `\newlabel` with the sub-reference as the normal reference value in the second argument. Unfortunately, if we want hypertext support, the second argument of `\newlabel` gets complicated. It is either

- `{`⟨*ref value* (never defined)⟩`}{`⟨*page number* (never defined)⟩`}`, when normal LaTeX is running, or
- `{`⟨*ref value* (never defined)⟩`}{`⟨*page number* (never defined)⟩`}{`⟨*text* (never defined)⟩`}{`⟨*hyper category* (never defined)⟩`}{`⟨*URL* (never defined)⟩`}`, when the `hyperref` package is running. (We actually detect this by looking for the `nameref` package, because that's the one that changes the use of labels.)

We unify these two things by producing `{`⟨*ref value* (never defined)⟩`}{`⟨*page number* (never defined)⟩`}\nw@labeltrailers`

We may have pending labels in support of `\nextchunklabel`, as defined in chunk 22a. Because we want to define all of the "pending sublabels" in exactly the same way, we do something a bit odd—we make the current label a pending label as well.

21a ⟨*definition of* `\newsublabel` 20b⟩+≡ (20a) ◁20e

```
\pendingsublabel{#1}%
\edef\@tempa##1{\noexpand\newlabel{##1}%
  {{\number\sub@page}{\this@page}\nw@labeltrailers}}%
\pending@sublabels
\def\pending@sublabels{}}
```

We can't use `\@ifpackageloaded` to see if `nameref` is loaded, because that is restricted to the preamble, and `\newsublabel` goes into the `.aux` file, which is executed after the whole document is processed. We therefore test for `\@secondoffive`. This is lame, but what else can we do?

21b ⟨*noweb.sty* 2b⟩+≡ ◁20a 22a▷

```
\newcommand\nw@settrailers{% -- won't work on first run
  \@ifpackageloaded{nameref}%
     {\gdef\nw@labeltrailers{{}{}{}}}%
     {\gdef\nw@labeltrailers{}}}
\renewcommand\nw@settrailers{%
  \@ifundefined{@secondoffive}%
     {\gdef\nw@labeltrailers{}}%
     {\gdef\nw@labeltrailers{{}{}{}}}}
```

Now we keep track of those pending guys. The goal here is to save them up until they're all equivalent to the label on the next chunk. We have to control expansion so chunks like 21a (21a) can be labelled twice.

22a ⟨*noweb.sty* 2b⟩+≡ ◁21b 22c▷
```
\newcommand{\nextchunklabel}[1]{%
  \nwblindhyperanchor{#1}%   % looks slightly bogus --- nr
  \@bsphack\if@filesw {\let\thepage\relax
      \edef\@tempa{\write\@auxout{\string\pendingsublabel{#1}}}%
      \expandafter}\@tempa
   \if@nobreak \ifvmode\nobreak\fi\fi\fi\@esphack}
\newcommand\pendingsublabel[1]{%
  \def\@tempa{\noexpand\@tempa}%
  \edef\pending@sublabels{\noexpand\@tempa{#1}\pending@sublabels}}
\def\pending@sublabels{}
```

22b ⟨*man page: noweb style control sequences* 22b⟩≡ 32b▷
```
.PP \" .TP will not work with the backslashes on the next line. Period.
\fB\\nextchunklabel{l}\fP
.RS
Associates label \fBl\fP
with the sub-page reference of the next code chunk.
Can be used in for concise chunk cross-reference with, e.g.,
\fBchunk~\\subpageref{l}\fP.
.RE
```

We need to define these.

22c ⟨*noweb.sty* 2b⟩+≡ ◁22a 23a▷
```
\def\last@page{\relax}
\newcount\sub@page
```

We no longer use Rainer's new expandable definitions of `\ref` and `\pageref` to minimise the risk of nasty surprises; enough time has elapsed that this should no longer be necessary.

22d ⟨*old noweb.sty* 22d⟩≡
```
% RmS 92/08/14: made \ref and \pageref robust
\def\ref#1{\@ifundefined{r@#1}{{\bf ??}⟨warn of undefined reference to #1 22e⟩}%
    {\expandafter\expandafter\expandafter
     \@car\csname r@#1\endcsname\@nil\null}}
\def\pageref#1{\@ifundefined{r@#1}{{\bf ??}⟨warn of undefined reference to #1 22e⟩}%
     {\expandafter\expandafter\expandafter
      \@cdr\csname r@#1\endcsname\@nil\null}}
\def\@refpair#1{\@ifundefined{r@#1}{{0}{0}⟨warn of undefined reference to #1 22e⟩}%
    {\@nameuse{r@#1}}}
```

22e ⟨*warn of undefined reference to* `#1` 22e⟩≡ (17d 22d)
```
\@warning{Reference `#1' on page \thepage \space undefined}
```

Here a a couple of hooks for formatting sub-page numbers, which can be alphabetic, numeric, or omitted.

hook

23a ⟨*noweb.sty* 2b⟩+≡ ◁22c 24a▷

```
\def\@alphasubpagenum#1#2{#2\ifnum#1=0 \else\@alph{#1}\fi}
\def\@nosubpagenum#1#2{#2}
\def\@numsubpagenum#1#2{#2\ifnum#1=0 \else.\@arabic{#1}\fi}
\def\nwopt@nosubpage{\let\nwthepagenum=\@nosubpagenum\nwopt@nomargintag}
\def\nwopt@numsubpage{\let\nwthepagenum=\@numsubpagenum}
\def\nwopt@alphasubpage{\let\nwthepagenum=\@alphasubpagenum}
\nwopt@alphasubpage
```

In rare cases, there may be more than 26 chunks on a page. In such a case, we need a sub-page numbering scheme that can go beyond "a to z." The scheme I have chosen is "a to z, then aa to zz, then aaa to zzz, etc." The conversion requires a bit of thought because it is *not* an ordinary conversion of integer to string as we usually think of such things. The problem is that the meaning of the letters depends on the position; the letter a acts like a zero in some positions or a one in others.

The solution I have implemented uses a variable `bound` which is always equal to 26^k for some k. If we write the recurrence $B_k = B_{k-1} + 26^k$, with $B_0 = 0$, we then use a string of k letters to represent numbers between B_{k-1} and B_k. Within that string, a's are 0's, and so on up to z's which are 25's, and we use standard integer-conversion methods to encode $n - B_{k-1}$.

The following Icon implementation may be more perspicuous than the TEX code actually used. Here the variable `bound` is 26^k, with $k = 1$ initially, and `n` is $n - B_{k-1}$. The first loop finds the right k, and the second does the usual string conversion.

23b ⟨*Icon code for subpage numbering* 23b⟩≡

```
procedure alphastring(n)
  bound := 26

  while n >= bound do {
    # invariant: bound = 26^(k+1) & n is initial n - B_k
    n -:= bound
    bound *:= 26
  }

  while bound > 1 do {
    bound /:= 26
    d := integer(n / bound)
    n -:= d * bound
    writes(&lcase[d+1])
  }
end
```

Here's TeX code to achieve the same end. The entire macro body is enclosed in braces, so that it can be used with `\loop` without picking up the wrong `\repeat`.

24a ⟨*noweb.sty* 2b⟩+≡ ◁23a 24b▷

```
\newcount\@nwalph@n
\let\@nwalph@d\@tempcnta
\let\@nwalph@bound\@tempcntb
\def\@nwlongalph#1{{%
  \@nwalph@n=#1\advance\@nwalph@n by-1
  \@nwalph@bound=26
  \loop\ifnum\@nwalph@n<\@nwalph@bound\else
     \advance\@nwalph@n by -\@nwalph@bound
     \multiply\@nwalph@bound by 26
  \repeat
  \loop\ifnum\@nwalph@bound>1
    \divide\@nwalph@bound by 26
    \@nwalph@d=\@nwalph@n\divide\@nwalph@d by \@nwalph@bound
    % d := d * bound ; n -:= d; d := d / bound --- saves a temporary
    \multiply\@nwalph@d by \@nwalph@bound
    \advance\@nwalph@n by -\@nwalph@d
    \divide\@nwalph@d by \@nwalph@bound
    \advance\@nwalph@d by 1 \@alph{\@nwalph@d}%
  \repeat
}}
```

2.9 WEB-like chunk numbering

Here's a righteous hack: we get the effect of WEB-like chunk numbers just by redefining `\sublabel` to use a counter instead of the current page number. Since the numbers are all distinct, no sub-page number is ever used.

24b ⟨*noweb.sty* 2b⟩+≡ ◁24a 25a▷

```
\newcount\nw@chunkcount
\nw@chunkcount=\@ne
\newcommand{\weblabel}[1]{%
  \@bsphack
  \nwblindhyperanchor{#1}%
  \if@filesw {\let\thepage\relax
   \def\protect{\noexpand\noexpand\noexpand}%
   \edef\@tempa{\write\@auxout{\string
      \newsublabel{#1}{{}{\number\nw@chunkcount}}}}%
   \expandafter}\@tempa
   \global\advance\nw@chunkcount by \@ne
   \if@nobreak \ifvmode\nobreak\fi\fi\fi\@esphack}
\def\nwopt@webnumbering{%
  \let\sublabel=\weblabel
  \def\nwpageword{chunk}\def\nwpagesword{chunks}%
  \def\nwpageprep{in}}
```