box register; a box register that contains \vbox{} will not return true if tested with the \ifvoid test. So to decide whether \@tempboxa is empty we cannot use \ifvoid. Instead we employ the simple strategy of measuring the width of the box. This will not be 100% failsafe but the failure cases that I've been able to imagine are all rather exotic.

\ifdim\wd\@tempboxa=\z@
\setbox\@ne\hbox to\columnwidth{\hss\kern-6pc\box\@ne\hss}\fi

The \kern-6pc in the first branch is to offset the \moveright that is about to be done next. (If tortured, I would be forced to admit that it took me several attempts before I figured out the right amount for this kern and the proper place to put it.) Finally, we put the caption on the page, with a \vskip to separate it from the preceding or following material.

\ifnum\@tempcnta<64 %if it's a figure
\vskip 1pc%
\moveright 3pc\box\@ne
\else % if the float IS NOT a figure
\vskip 1pc%
\fi

By testing \@tempcnta we can tell whether the caption is being used in a figure environment or not; if so, we assume that the caption is placed below the artwork and hence put the \vskip above the caption; otherwise we assume the caption is at the top of the floating insertion and we put the \vskip below it.

\makecaption presents a few extra complications that have been omitted for the sake of simplicity; as given here, the caption will not be quite centered if the figure caption has no text, and so on.

Looking Ahead for a (box)
Sonja Maus

\TeX's primitive \afterassignment can be used for macros which first assign a value to a parameter, and then perform some actions using that value. For instance the plain \TeX macros \magnification and \hglue (see The \TeXbook, p. 364 and 352), assign a (number) or (glue) value to a variable and then use this value. They provide a user-friendly “syntax mimicry”: \magnification looks like an integer parameter in an assignment, and \hglue looks like the primitive command \hskip. There is another advantage to this method over the use of arguments with \#1: At the moment when \TeX looks at the tokens of the value, it already knows what kind of value it is looking for. This would be very useful when the value to be read is a (box), because an explicit \hbox or \vbox may contain \catcode changes and all tokens should not be read ahead.

There are seven ways to write a (box) (The \TeXbook, p. 278). The \afterassignment command behaves differently with the first four and the last three of these (box)es:

\afterassignment\t \setbox\@ne=\box\@ne results in \setbox\@ne=\box\@ne\t, whereas
\afterassignment\t \setbox\@ne=\hbox{h) results in \setbox\@ne=\hbox{t h}.

The macro \afterbox gives a substitute which is equally valid for all (box)es. Its syntax is

\afterbox\langle argument\rangle\box\langle argument\rangle

where \langle argument\rangle is an argument for an undelimited macro parameter (see The \TeXbook, p. 204), i.e. a single token or several tokens in explicit braces. \afterbox puts the (argument) aside (without the braces, if any), assigns the (box) to the register \box\afterbox, and then reads the (argument) again.

The definition must be read when \ is a letter:

\newbox\afterbox
\def\afterbox#1{\def\afterbox#1\chardef\next\{.\}
\def\afterbox#1\{\futurelet\next\afterbox#1test\}
\def\afterbox#1test\{
\ifcase\ifx\next\hbox\tw@\fi
\ifx\next\vbox\tw@\fi
\ifx\next\vtop\tw@\fi
\ifx\next\box\One\fi
\ifx\next\copy\One\fi
\ifx\next\vsplit\One\fi
\ifx\next\lastbox\One\fi
0\errmessage{No <box>}%
\or\afterassignment\afterbox#1}

Michael Downes
American Mathematical Society
201 Charles Street
Providence, RI 02904
mjd@Math.AMS.com
First, \afterbox puts the (argument) into \afbox. Then the \chardef command reads a (number) which turns out to be a (normal integer) with a (character token) (see The \TeXbook, p. 269). As the syntax of (number) requires, \TeX expands tokens and looks for (one optional space) which turns out (empty). This looks crazy, but it has the effect of unpacking the first non-expandable token of (box) if it was hidden behind expandable tokens like \null or \line (or \Boxit below). This non-expandable token’s meaning is then assigned to \next and tested by \afbox\test. It must be one of the seven primitives listed with the \if, and the cases 1 and 2 correspond to the two behaviours of \afterassignment mentioned above. In both cases, \afbox will reappear exactly at the time when the \setbox assignment is finished, e.g.:\afterbox \t \box1results in \setbox\afbox=\box1 \t, whereas\afterbox \t \hbox{h}\first becomes ...\hbox{\afbox h} and thenresults in \setbox\afbox=\hbox{h}\t.\def\Boxit{\hbox\bgroup\afterbox\vrule\dimen0=\dp\afbox\advance\dimen0 by3.4pt\lower\dimen0 vbox\hrule kern3pt\hbox{\kern3pt box\afbox/kern3pt}\kern3pt \hrule%\vrule \egroup}\Boxit{\hbox b}\Boxit{\hbox{c}}\byeYou notice that the same objects are treated in two different ways. The reason for this is that \TeX starts each job in vertical mode, that is, stacking material. In the second piece of input \TeX saw the character ‘a’ before it saw the boxes. A character is for \TeX the sign to switch to horizontal mode, that is, lining up material, and start building a paragraph.

Commands that can make \TeX switch to horizontal mode are called ‘horizontal commands’. As appeared from the above two examples characters are horizontal commands, but boxes are not. Let us now look at the two most obvious horizontal commands: \indent and \noindent.

1.1 \indent and \noindent

\indent is the command to start a paragraph with indentation. \TeX realizes the indentation by insert-