1 Introduction

This is an internal package for storing common functions that are shared by more than one package in the AMS-LaTeX distribution. Some of these might eventually make it into the LaTeX kernel.

Standard package info. Using \ProvidesFile rather than \ProvidesPackage because the latter, when input by, e.g., amsbook, results in LaTeX warning: You have requested document class 'amsbook', but the document class provides 'amsgen'.

\NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
[1994/12/01]% LaTeX date must December 1994 or later
\ProvidesFile{amsgen.sty}[1999/11/30 v2.0 generic functions]

2 Implementation

Some general macros shared by amsart.dtx, amsmath.dtx, amsfonts.dtx, ...

\@saveprimitive

The amsmath package redefines a number of TeX primitives. In case some preceding package also decided to redefine one of those same primitives, we had better do some checking to make sure that we are able to save the primitive meaning for internal use. This is handled by the \@saveprimitive function. We follow the example of \@@input where the primitive meaning is stored in an internal control sequence with a @@ prefix. Primitive control sequences can be distinguished by the fact that \string and \meaning return the same information.

\providecommand\@saveprimitive[2]{\begingroup\escapechar'\relax
edef\@tempa{\string#1}\edef\@tempb{\meaning#1}\%
\ifx\@tempa\@tempb \global\let#2#1\%
\else

Check to see if \#2 was already given the desired primitive meaning somewhere else.

\edef\@tempb{\meaning#2}\
\ifx\@tempa\@tempb
  \else
    \latex@error{Unable to properly define \string#2; primitive \noexpand#1no longer primitive}@eha
  \fi
\fi
\endgroup}

Shorthands for long command names.
\let\@xp=\expandafter
\let\@nx=\noexpand
\@emptytoks A token register companion for \@empty. Saves a little main mem and probably makes initializations such as \toks@{} run faster too.
\newtoks\@emptytoks
\@oparg Use of \@oparg simplifies some constructions where a macro takes an optional argument in square brackets. We can’t use \newcommand here because this function might be previously defined by the amsmath package in a loading sequence such as
\usepackage{amsmath,amsthm}
\def\@oparg#1[#2]{\@ifnextchar[\@xp\@secondoftwo\@ifnextchar\{\@xp\@firstoftwo\fi
\@ifempty\@ifnotempty\@ifnotempty use category 11 @ characters to test whether the argument is empty or not, since these are highly unlikely to occur in the argument. As with \@oparg, there is a possibility that these commands were defined previously in amsmath.sty.
\long\def\@ifempty#1{\ifx\@let@token#1\@nil\else\@ifnextchar[\@xp\@ifnextchar\{\@xp\ifx\@empty\@empty\fi
\@ifnotempty is a shorthand that makes code read better when no action is needed in the empty case. At a cost of double argument-reading—so for often-executed code, avoiding \@ifnotempty might be wise.
\long\def\@ifnotempty#1{\ifx\@empty\@nil\else
Some abbreviations to conserve token mem.
\def\FW@{\futurelet}\def@token\def\DN@{\def\next@}
\def\RIN@{\relax@ifmmode}\def\setboxz@h{\setbox0hbox}\
\def\wdz@{\wd0}\def\boxz@{\box0}\def\relaxnext@{\let\@let@token\relax}
\new@ifnextchar

This macro is a new version of LATEX's \ifnextchar, macro that does not skip over spaces.
\long\def\new@ifnextchar#1#2#3{% 
By including the space after the equals sign, we make it possible for \new@ifnextchar to do look-ahead for any token, including a space!
\let\reserved@d= #1% 
\def\reserved@a{#2}\def\reserved@b{#3}% 
\futurelet\@let@token\new@ifnch 
}% \def\new@ifnch{% \ifx\@let@token\reserved@d \let\reserved@b\reserved@a \fi \reserved@b 
}%

\@ifstar

There will essentially never be a space before the *, so using \@ifnextchar is unnecessarily slow.
\def\@ifstar#1#2{\new@ifnextchar *{\def\reserved@a*{#1}\reserved@a}{#2}}

The hook \every@size was changed to \everymathsize in the December 1994 release of LATEX and its calling procedures changed. If \everymathsize is undefined it means the user has an older version of LATEX so we had better define it and patch a couple of functions (\glb@settings and \set@fontsize).
\@ifundefined{everymathsize}{% 
Reuse the same token register; since it was never used except for the purposes that are affected below, this is OK.
\let\everymathsize=\every@size 
\def\glb@settings{% 
\expandafter\ifx\csname S@f@size\endcsname\relax 
\calculate@math@sizes \fi 
\csname S@f@size\endcsname \ifmath@fonts 
% \ifnum \tracingfonts>\tw@
% \@font@info{Setting up math fonts for 
% \f@size/\f@baselineskip}\fi 
\begingroup 
\escapechar\m@ne 
\csname mv@math\endcsname \globaldefs\@ne 
\let \glb@currsize \f@size \math@fonts 
\endgroup 
\the\everymathsize 
\else 
% \ifnum \tracingfonts>\tw@
% \@font@info{No math setup for \f@size/\f@baselineskip}% 
\fi 
\fi 
\fi} 
\endgroup
```
% \fi
\fi
}

Remove \the\every@size from \size@update.

\def\set@fontsize#1#2#3{\@defaultunits\@tempdimb#2pt\relax\@nnil\edef\f@size{\strip@pt\@tempdimb}\@defaultunits\@tempskipa#3pt\relax\@nnil\edef\f@baselineskip{\the\@tempskipa}\edef\f@linespread{#1}\let\baselinestretch\f@linespread\let\size@update\relax}
\let\size@update\relax}

\ex@
The \ex@ variable provides a small unit of space for use in math-mode constructions, that varies according to the current type size. For example, the \pmb command uses \ex@ units. Since a macro or mu unit solution for the \langle \texttt{dimen} \rangle \ex@ won’t work without changing a lot of current code in the amsmath package, we set \ex@ through the \every@math@size hook. The value of \ex@ is scaled nonlinearly in a range of roughly 0.5pt to 1.5pt, by the function \compute@ex@.

\newdimen\ex@
\addto@hook\every@math@size{\compute@ex@}

\compute@ex@ computes \ex@ as a nonlinear scaling from 10pt to current font size (\f@size). Using .97 as the multiplier makes 1 \ex@ \approx .9pt when the current type size is 8pt and 1 \ex@ \approx 1.1pt when the current type size is 12pt.

The formula is essentially

\[ 1pt \pm (1pt - (.97)^{\lfloor|10 - n|\rfloor}) \]

where \( n \) = current type size, but adjusted to differentiate half-point sizes as well as whole point sizes, and there is a cutoff for extraordinarily large values of \f@size (> 20pt) so that the value of \ex@ never exceeds 1.5pt.

\def\compute@ex@{%
\begingroup
\dimen@-\f@size pt\ifdim\dimen@<20 pt
2. IMPLEMENTATION

Never make $\texttt{\textbackslash ex}$ larger than 1.5pt.

\begin{verbatim}
\global\texttt{\textbackslash ex} 1.5\texttt{p}\texttt{t}
\end{verbatim}

Adjust by the reference size and multiply by 2 to allow for half-point sizes.

\begin{verbatim}
\advance\texttt{dimen}10\texttt{p}\texttt{t} \multiply\texttt{dimen}2\texttt{w}
\end{verbatim}

Save information about the current sign of $\texttt{dimen}$.

\begin{verbatim}
\edef\texttt{\textbackslash tempa}{\ifdim\texttt{dimen}>\z\texttt{f}\fi}\
\end{verbatim}

Get the absolute value of $\texttt{dimen}$.

\begin{verbatim}
\texttt{dimen}\texttt{ifdim}\texttt{dimen}\texttt{z}\texttt{f} \texttt{dimen}\texttt{advance}\texttt{dimen}\texttt{w} \texttt{m sp} \texttt{fudge factor}
\end{verbatim}

Here we use $\texttt{vfuzz}$ merely as a convenient scratch register

\begin{verbatim}
\texttt{vfuzz}\texttt{p}\texttt{t}
\end{verbatim}

Multiply in a loop.

\begin{verbatim}
\def\do{\ifdim\texttt{dimen}>\z\texttt{v}\texttt{fuzz} .97\texttt{v}\texttt{fuzz} \texttt{advance}\texttt{dimen}\texttt{w}\texttt{p}\texttt{t} \%\texttt{message}\{\texttt{vfuzz: \texttt{thevfuzz}, \texttt{dimen}: \texttt{the\texttt{dimen}}\}\% \texttt{0xp}\texttt{do }\texttt{f}\texttt{i} }\% \texttt{do} \texttt{dimen}\texttt{p}\texttt{t} \texttt{advance}\texttt{dimen} \texttt{vfuzz} \texttt{global}\texttt{ex}\texttt{p}\texttt{t} \texttt{global}\texttt{advance}\texttt{ex}\texttt{p}\texttt{t} \texttt{tempa}\texttt{dimen}\texttt{w} \texttt{f}\texttt{i} \texttt{endgroup} \%\texttt{typeout}\{\texttt{string}\{\texttt{f}\texttt{size}: \texttt{f}\texttt{size}\}\texttt{showthe}\texttt{ex}\texttt{t}\}
\end{verbatim}

Tests of the $\texttt{compute@ex}$ function yield the following results:

\begin{verbatim}
\%\texttt{f@size} \texttt{ex} \texttt{f@size} \texttt{ex}
10 1.0pt 9.04089pt
11 1.05911pt 8.7 0.91266pt
12 1.11143pt 8.5 0.91266pt
14.4 1.23982pt 8.4 0.88527pt
17.28 1.36684pt 8 0.88527pt
20.74 1.5pt 7 0.83293pt
19.5 1.4395pt 6 0.78369pt
5 0.73737pt
1 0.57785pt
\end{verbatim}

Use of the $\texttt{@addpunct}$ function allows ending punctuation in section headings and elsewhere to be intelligently omitted when punctuation is already present.

\begin{verbatim}
\def\texttt{@addpunct#1}{\ifnum\texttt{spacefactor}>\z\texttt{else#1}\texttt{fi}}
\end{verbatim}

Change $\texttt{frenchspacing}$ to ensure that $\texttt{@addpunct}$ will continue to work properly even when ‘french’ spacing is in effect.

\begin{verbatim}
\def\texttt{frenchspacing}{\texttt{sfcode`}\texttt{.1006}\texttt{sfcode`}\texttt{?1005}\texttt{sfcode`}\texttt{!1004}\% \texttt{sfcode`}\texttt{:1003}\texttt{sfcode`}\texttt{;1002}\texttt{sfcode`}\texttt{,1001}
\end{verbatim}
2.1 Miscellaneous
\def\nomath@env{\@amsmath@err{\string\begin{\@currenvir} allowed only in paragraph mode}}\@ehb "You've lost some text"}

A trade-off between main memory space and hash size; using \texttt{Invalid@@} saves 14 bytes of main memory for each use of \texttt{Invalid@}, at the cost of one control sequence name. \texttt{Invalid@} is currently used about five times and \texttt{Invalid@@} is used by itself in some other instances, which means that it saves us more memory than \texttt{FN@}, \texttt{RIfM@}, and some of the other abbreviations above.
\def\Invalid@@{Invalid use of \string}

The usual \texttt{\endinput} to ensure that random garbage at the end of the file doesn’t get copied by docstrip.
\endinput