1 Introduction
This is an internal package for storing common functions that are shared by more than one package in the AMSG-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, ...
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\@noexpand\@noexpand#1no longer primitive}\@eha
\fi
\fi
\endgroup}

\@xp Shorthands for long command names.
\@nx \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[\def\@oparg#1[#2]{#1}[#2]}
\@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#1\@nil\@nil\else\@ifnextchar[\def\@ifempty#1[\def\@ifempty#1]{#1}[\def\@ifempty#1]}}
\long\def\@ifnotempty#1{\@ifempty{#1}{}\@ifnextchar[\def\@ifnotempty#1[\def\@ifnotempty#1]{#1}[\def\@ifnotempty#1]}
\@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{\@ifempty#1\@ifempty{#1}}
\@ifempty\@ifnotempty\@ifempty
\@ifnotempty Some abbreviations to conserve token mem.
\def\FN@{\futurelet\@let@token}
\def\DN@{\def\next@}
\def\RIfM@{\relax\ifmmode}
\def\setboxz@h#1{\setbox\z@\hbox{#1}}
\def\wdz@{\wd\z@}
\def\boxz@{\box\z@}
\def\relaxnext@{\let\@let@token\relax}
This macro is a new version of \LaTeX’s \ifnextchar, macro that does not skip over spaces.

\long\def\new@ifnextchar#1#2#3{\futurelet\@let@token\new@ifnch

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 \every@math@size in the December 1994 release of \LaTeX and its calling procedures changed. If \every@math@size is undefined it means the user has an older version of \LaTeX so we had better define it and patch a couple of functions (\global@settings and \set@fontsize).

\@ifundefined{every@math@size}{% Reuse the same token register; since it was never used except for the purposes that are affected below, this is OK.
\let\every@math@size=\every@size
\def\global@settings{%
  \expandafter\ifx\csname S@\f@size\endcsname\relax
    \calculate@math@sizes
  \fi
  \csname S@\f@size\endcsname
  \ifmath@fonts
    % \ifnum \tracingfonts>	w@
    % \@font@info{Setting up math fonts for \f@size/\f@baselineskip}\fi
    \begingroup
      \escapechar\m@ne
      \csname mv@\math@version \endcsname
      \globaldefs\@ne
      \let \global@currsize \f@size
      \math@fonts
    \endgroup
    \the\every@math@size
  \else
    % \ifnum \tracingfonts>	w@
    % \@font@info{No math setup for \f@size/\f@baselineskip}%
    \fi
  \fi
}\fi
}
THE AMS\textsc{gen} PACKAGE

\% \fi
\fi
}
\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 
\def\size@update{ 
\baselineskip\f@baselineskip\relax 
\baselineskip\f@linespread\baselineskip 
\normalbaselineskip\baselineskip 
\setbox\strutbox\hbox{ 
\vrule\@height.7\baselineskip 
\@depth.3\baselineskip 
\@width\z@} 
%% \the\every@size 
\let\size@update\relax}\
}\message{}
}
\let\ex@\relax
\begin{footnotesize}
\footnotesize
\pagestyle{empty}
\vspace{50pt}
\noindent
\begin{center}
\textbf{\texttt{\textbackslash ex@}}
\end{center}
\noindent
\begin{footnotesize}
\footnotesize
\end{footnotesize}

The \texttt{\textbackslash 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 \texttt{\textbackslash pmb} command uses \texttt{\textbackslash ex@} units. Since a macro or mu unit solution for the \texttt{(dimen)} \texttt{\textbackslash ex@} won’t work without changing a lot of current code in the \texttt{amsmath} package, we set \texttt{\textbackslash ex@} through the \texttt{\textbackslash every@math@size} hook. The value of \texttt{\textbackslash ex@} is scaled nonlinearly in a range of roughly 0.5pt to 1.5pt, by the function \texttt{\textbackslash compute@ex@}.

\begin{verbatim}
\def\compute@ex@{% 
\begingroup 
\dimen@-\f@size\p@ 
\ifdim\dimen@<-20\p@
\fi
\endgroup
\addto@hook\every@math@size{\compute@ex@}
\end{verbatim}

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

The formula is essentially

\[ 1\text{pt} \pm (1\text{pt} - (.97)^{|10-\text{\texttt{\textbackslash f@size}}|}) \]

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 \texttt{\textbackslash f@size} (> 20pt) so that the value of \texttt{\textbackslash ex@} never exceeds 1.5pt.

\begin{verbatim}
\def\compute@ex@{% 
\begingroup 
\dimen@-\f@size\p@ 
\ifdim\dimen@<-20\p@
\fi
\endgroup
\addto@hook\every@math@size{\compute@ex@}
\end{verbatim}
2. IMPLEMENTATION

Never make $\texttt{\char14 ex}$ larger than 1.5pt.

\begin{verbatim}
\global\texttt{\char14 ex} 1.5pt
\else
\end{verbatim}

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

\begin{verbatim}
\advance\texttt{\char14 dimen} 10pt \multiply\texttt{\char14 dimen} 2
\end{verbatim}

Save information about the current sign of $\texttt{\char14 dimen}$.

\begin{verbatim}
\edef\texttt{\char14 @tempa}{\ifdim\texttt{\char14 dimen}>0 -\fi}\
\end{verbatim}

Get the absolute value of $\texttt{\char14 dimen}$.

\begin{verbatim}
\texttt{\char14 dimen} \ifdim\texttt{\char14 dimen}<0 -\fi \texttt{\char14 dimen}
\advance\texttt{\char14 dimen} -\texttt{\char14 m sp} % fudge factor
\end{verbatim}

Here we use $\texttt{\char14 v fuzz}$ merely as a convenient scratch register

\begin{verbatim}
\texttt{\char14 v fuzz} pt
\end{verbatim}

Multiply in a loop.

\begin{verbatim}
\def\do{\ifdim\texttt{\char14 dimen}>0\texttt{\char14 v fuzz} .97\texttt{\char14 v fuzz}}
\advance\texttt{\char14 dimen} -\texttt{\char14 pt} %
%\message{\texttt{\char14 v fuzz}: \the\texttt{\char14 v fuzz}, \texttt{\char14 dimen}: \the\texttt{\char14 dimen}}%
% \expdo \fi}\
\do
\texttt{\char14 dimen} pt \advance\texttt{\char14 dimen} -\texttt{\char14 v fuzz}
\global\texttt{\char14 ex} pt \texttt{\char14 @tempa}\texttt{\char14 dimen}
\texttt{\char14 fi}
\end{verbatim}

%\typeout{\string{\char14 f@size}: \f@size}\showthe\texttt{\char14 ex}

Tests of the $\texttt{\char14 compute\char14 ex}$ function yield the following results:

\begin{verbatim}
\f@size \texttt{\char14 ex}\f@size \texttt{\char14 ex}
10 1.0pt 9 0.94089pt
11 1.05911pt 8.7 0.91266pt
12 1.11473pt 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}

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

\begin{verbatim}
\def\@addpunct#1{\ifnum\spacefactor>1000 \else#1\fi}
\end{verbatim}

\frenchspacing Change $\texttt{\char14 frenchspacing}$ to ensure that $\@addpunct$ will continue to work properly even when ‘french’ spacing is in effect.

\begin{verbatim}
\def\frenchspacing{
\sfcode\texttt{.}1006\sfcode\texttt{?}1005\sfcode\texttt{!}1004%
\sfcode\texttt{\char13}1003\sfcode\texttt{;}1002\sfcode\texttt{,}1001 }
\end{verbatim}
2.1 Miscellaneous
\def\nomath@env\@anmath@err{%
\string\begin{\currenenvr} allowed only in paragraph mode%
}\@ehb% "You've lost some text"
}

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

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