

The `amsgen` package

American Mathematical Society

Michael Downes

Version 2.0, 1999/11/30

1 Introduction

This is an internal package for storing common functions that are shared by more than one package in the $\mathcal{A}\mathcal{M}\mathcal{S}$ - \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'.`

```
1 \NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
2 [1994/12/01]% LaTeX date must December 1994 or later
3 \ProvidesFile{amsgen.sty}[1999/11/30 v2.0]
```

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.

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

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

```
8     \edef\@tempb{\meaning#2}%
9     \ifx\@tempa\@tempb
10    \else
11      \@latexerror{Unable to properly define \string#2; primitive
12        \noexpand#1no longer primitive}\@eha
13    \fi
14    \fi
15    \endgroup}
```

`\@xp` Shorthands for long command names.

```
\@nx 16 \let\@xp=\expandafter
      17 \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.

```
18 \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}
```

```
19 \def\@oparg#1[#2]{\@ifnextchar[#{1}{#1[#2]}}
```

`\@ifempty` `\@ifnotempty` and `\@ifempty` 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`.

```
20 \long\def\@ifempty#1{\@xifempty#1@@..\@nil}
21 \long\def\@xifempty#1#2@#3#4#5\@nil{%
22   \ifx#3#4\@xp\@firstoftwo\else\@xp\@secondoftwo\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.

```
23 \long\def\@ifnotempty#1{\@ifempty{#1}{}}
```

Some abbreviations to conserve token mem.

```
24 \def\FN@{\futurelet\@let@token}
25 \def\DN@{\def\next@}
26 \def\RIfM@{\relax\ifmmode}
27 \def\setboxz@h{\setbox\z@\hbox}
28 \def\wdz@{\wd\z@}
29 \def\boxz@{\box\z@}
30 \def\relaxnext@{\let\@let@token\relax}
```

`\new@ifnextchar` This macro is a new version of L^AT_EX's `\@ifnextchar`, macro that does not skip over spaces.

```
31 \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!

```
32   \let\reserved@d= #1%
33   \def\reserved@a{#2}\def\reserved@b{#3}%
34   \futurelet\@let@token\new@ifnch
35 }
36 %
37 \def\new@ifnch{%
38   \ifx\@let@token\reserved@d \let\reserved@b\reserved@a \fi
39   \reserved@b
40 }
```

`\@ifstar` There will essentially never be a space before the `*`, so using `\@ifnextchar` is unnecessarily slow.

```
41 \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 L^AT_EX and its calling procedures changed. If `\every@math@size`

is undefined it means the user has an older version of L^AT_EX so we had better define it and patch a couple of functions (`\glb@settings` and `\set@fontsize`).

```
42 \@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.

```
43 \let\every@math@size=\every@size
44 \def\glb@settings{%
45     \expandafter\ifx\csname S@\f@size\endcsname\relax
46         \calculate@math@sizes
47     \fi
48     \csname S@\f@size\endcsname
49     \ifmath@fonts
50 %         \ifnum \tracingfonts>\tw@
51 %             \@font@info{Setting up math fonts for
52 %                 \f@size/\f@baselineskip}\fi
53         \begingroup
54             \escapechar\m@ne
55             \csname mv@\math@version\endcsname
56             \globaldefs\@ne
57             \let \glb@currsiz \f@size
58             \math@fonts
59         \endgroup
60         \the\every@math@size
61     \else
62 %         \ifnum \tracingfonts>\tw@
63 %             \@font@info{No math setup for \f@size/\f@baselineskip}%
64 %         \fi
65     \fi
66 }
```

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

```
67 \def\set@fontsize#1#2#3{%
68     \@defaultunits\@tempdimb#2pt\relax\@nnil
69     \edef\f@size{\strip@pt\@tempdimb}%
70     \@defaultunits\@tempskipa#3pt\relax\@nnil
71     \edef\f@baselineskip{\the\@tempskipa}%
72     \edef\f@linespread{#1}%
73     \let\baselinestretch\f@linespread
74     \def\size@update{%
75         \baselineskip\f@baselineskip\relax
76         \baselineskip\f@linespread\baselineskip
77         \normalbaselineskip\baselineskip
78         \setbox\strutbox\hbox{%
79             \vrule\@height.7\baselineskip
80             \@depth.3\baselineskip
81             \@width\z@}%
82 %%%     \the\every@size
83     \let\size@update\relax}%
84 }
85 }{}% end \@ifundefined test
```

`\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 *<dimen>* `\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@`.

```
86 \newdimen\ex@
```

```
87 \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 $1\text{ ex@} \approx .9\text{pt}$ when the current type size is 8pt and $1\text{ ex@} \approx 1.1\text{pt}$ when the current type size is 12pt.

The formula is essentially

$$1\text{pt} \pm (1\text{pt} - (.97)^{\lfloor 10-n \rfloor})$$

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` ($> 20\text{pt}$) so that the value of `\ex@` never exceeds 1.5pt.

```
88 \def\compute@ex@{%
89   \begingroup
90   \dimen@-\f@size\p@
91   \ifdim\dimen@<-20\p@
```

Never make `\ex@` larger than 1.5pt.

```
92   \global\ex@ 1.5\p@
93   \else
```

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

```
94   \advance\dimen@10\p@ \multiply\dimen@ \tw@
```

Save information about the current sign of `\dimen@`.

```
95   \edef\@tempa{\ifdim\dimen@>\z@ -\fi}%
```

Get the absolute value of `\dimen@`.

```
96   \dimen@ \ifdim\dimen@<\z@ -\fi \dimen@
97   \advance\dimen@-\@m sp % fudge factor
```

Here we use `\vfuzz` merely as a convenient scratch register

```
98   \vfuzz\p@
```

Multiply in a loop.

```
99   \def\do{\ifdim\dimen@>\z@
100     \vfuzz=.97\vfuzz
101     \advance\dimen@ -\p@
102 %\message{\vfuzz: \the\vfuzz, \dimen@: \the\dimen@}%
103     \@xp\do \fi}%
104   \do
105   \dimen@\p@ \advance\dimen@-\vfuzz
106   \global\ex@\p@
107   \global\advance\ex@ \@tempa\dimen@
108   \fi
109   \endgroup
110 %\typeout{\string\f@size: \f@size}\showthe\ex@
111 }
```

Tests of the `\compute@ex@` function yield the following results:

<code>\f@size</code>	<code>\ex@</code>	<code>\f@size</code>	<code>\ex@</code>
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

G		
<code>\glb@currsiz</code>	57
<code>\glb@settings</code>	...	3, 44
<code>\globaldefs</code>	56
H		
<code>\hbox</code>	27, 78
I		
<code>\ifmath@fonts</code>	49
<code>\Invalid@</code>	119
M		
<code>\m@ne</code>	54
<code>\math@fonts</code>	58
<code>\math@version</code>	55
<code>\meaning</code>	1, 5, 8
<code>\message</code>	102
<code>\multiply</code>	94
N		
<code>\NeedsTeXFormat</code>	1
P		
<code>\new@ifnch</code>	34, 37
<code>\new@ifnextchar</code>	...	2, 31, 41
<code>\newcommand</code>	2
<code>\newdimen</code>	86
<code>\next@</code>	25
<code>\nomath@env</code>	115
R		
<code>\relaxnext@</code>	30
<code>\reserved@a</code>	...	33, 38, 41
<code>\reserved@b</code>	...	33, 38, 39
<code>\reserved@d</code>	32, 38
<code>\RIfM@</code>	26
S		
<code>\set@fontsize</code>	...	3, 67
T		
<code>\tracingfonts</code>	...	50, 62
<code>\tw@</code>	50, 62, 94
<code>\typeout</code>	110
V		
<code>\vfuzz</code>	4, 98, 100, 102, 105	
W		
<code>\wd</code>	28
<code>\wdz@</code>	28
<code>\setbox</code>	27, 78
<code>\setboxz@h</code>	27
<code>\sfcode</code>	113, 114
<code>\showthe</code>	110
<code>\size@update</code>	...	3, 74, 83
<code>\spacefactor</code>	112
<code>\string</code>	1
<code>\strip@pt</code>	69
<code>\strutbox</code>	78