ConTEXt

Specials

category: ConTEXt Support Macros

version: 1997.07.05

date: March 19, 1998

author: Hans Hagen

```
1  \unprotect
```

This module implements some `\special` manipulation macros. I needed these when I implemented the code that handles the conversion of TPIC specials to PDF code.

```
2  \writestatus{loading}{Context Support Macros / Specials}
```

When interpreting specials we need to do some basic scanning. For the moment we distinguish between three cases. We need

```
\special{tag: arguments}
\special{tag arguments}
\special{tag}
```

We cannot be sure that the first case isn't

```
\special{tag:arguments}
```

So we have to take care of that one too.

`\redefines..` Specials that are to be interpreted are defined with commands like:

```
\redefinespecial a: \using#1\endspecial%
  {let's execute special 'a:' using '#1'}

\redefinespecial a  \using#1\endspecial%
  {let's execute special 'a' using '#1'}

\redefinespecial a  \using#1\endspecial%
  {let's execute special 'a' using nothing}
```

The first two always take an argument, the last one not. The definition of this redefinition macro is not that complex. The names are internally tagged with `\@rds@` which saves both time and space.

```
3  \def\@rds@{@rds@}

4  \def\redefinespecial #1 %
     {\setvalue{\@rds@#1}}
```

`\mimmicksp..` Mimmicking specials is activated by saying:

```
\mimmickspecials
```

This commands redefines the PLAIN TEX primitive `\special`.

```
5  \def\mimmickspecials%
     {\let\special=\domimmickspecial}
```

The special mimmicking macro first looks if it can find an colon terminated tag, next it searches for a tag that end with a space. If both cannot find, the tag itself is treated without argument.

```
6  \def\domimmickspecial#1%
     {\domimmickcolonspecial#1:\relax/:\relax/\end}

7  \def\domimmickcolonspecial#1:#2#3:\relax/#4\end%
     {\ifx#2\relax
        \domimmickspacespecial#1 \relax/ \relax/\end
      \else
        \dodomimmickspecial#1:\using#2#3\endspecial
      \fi}
```

```
\def\domimmickspacespecial#1 #2#3 \relax/#4\end%
  {\ifx#2\relax
     \dodomimmickspecial#1\using\endspecial
   \else
     \dodomimmickspecial#1\using#2#3\endspecial
   \fi}
```

```
\def\dodomimmickspecial#1\using#2\endspecial%
  {\expandafter\ifx\csname\@rds@#1\endcsname\relax % \doifdefinedelse
     \defaultspecial{#1 #2}%
   \else
     %\message{[mimmick special #1 with #2#3]}%
     \getvalue{\@rds@#1}\using#2\endspecial
   \fi}
```

Now let's show that things work the way we want, using the previous definitions of tag a.

```
\mimmickspecials
\special{a: 1 2 3 4 5}
\special{a: 1 2 3 4 5}
\special{a}
```

Which results in:

let's execute special 'a:' using '1 2 3 4 5'
let's execute special 'a:' using '1 2 3 4 5'
let's execute special 'a' using nothing

\mimmicksp.. When needed, one can call a mimmicked special directly by saying for instance:

```
\mimmickspecial a: \using...\endspecial
```

This can be handy when specials have much in common.

```
\def\mimmickspecial #1 %
  {\getvalue{\@rds@#1}}
```

\normalspe.. \defaultsp.. Unknown specials are passed to the default special handler. One can for instance ignore all further specials by saying \normalspecial:

```
\def\defaultspecial#1{}
```

But here we default to idle.

```
\let\normalspecial =\special
\let\defaultspecial=\special
```

```
\protect
```

```
\endinput
```

\defaultspecial *2*

\mimmickspecial *2*
\mimmickspecials *1*

\normalspecial *2*

\redefinespecial *1*