The cdcmd package

Wenjian Chern (Longaster[*])

Released October 12, 2021, version v1.0

Abstract

cdcmd is a package that allows you define 'polymorphic' command. Like styledcmd package, you can define `\protected` command, but cdcmd can define expandable conditional command as well.

1 Main Interface

`\newcondition`
`\setcondition`
`\clearcondition`

```
\newcondition {⟨identifier⟩} {⟨id(s)⟩}
\setcondition + {⟨identifier=ids list⟩}
\clearcondition[⟨identifier(s)⟩]
```

`\newcondition` new ⟨*identifier*⟩ and its ⟨*ids*⟩. The leading and trailing spaces in ⟨*identifier*⟩ will be removed.

`\setcondition` sets ⟨*ids*⟩ of ⟨*identifier*⟩ locally. The un-`+` version will clear ⟨*ids*⟩ formerly set.

Both ⟨*identifier*⟩ and ⟨*id*⟩ cannot be `*`.

`\clearcondition` will clear ids from given ⟨*identifiers*⟩ locally. Default value is `*`, that is, clear all.

`\conditionif`
`\conditioncmd`
`\econditionif`
`\econditioncmd`

```
\conditionif * [⟨identifier=ids list⟩] {⟨true⟩} {⟨false⟩}
\conditioncmd * [⟨identifier=ids list⟩] {⟨material⟩}
```

When the ⟨*identifier=ids list*⟩ makes `true` condition, leave ⟨*true*⟩/`{`⟨*material*⟩`}` in the input stream. Leaving ⟨*false*⟩ when the condition is `false`.

The starred version is `all`, unstarred version is any. See below for more details.

The `\econditionif` and `\econditioncmd` are expandable (`f`-expandable). `\conditionif`, `\conditioncmd` are `\protected`.

The default value of ⟨*identifier=id list*⟩ is `*`, will leave ⟨*true*⟩/⟨*material*⟩ in the input stream.

[*]Email: longaster@163.com

```
\conditioncase
\conditioncaseTF
\econditioncase
\econditioncaseTF
```

```
\conditioncaseTF * !
  {
    {⟨identifier=ids list case1⟩} {⟨code1⟩}
    {⟨identifier=ids list case2⟩} {⟨code2⟩}
    ...
    {⟨identifier=ids list casen⟩} {⟨coden⟩}
  }
  {⟨true code⟩}
  {⟨false code⟩}
```

Evaluates in turn each of the ⟨*identifier=ids list*⟩ until the first one that evaluates to `true` or to `false`, for un-`!` version or `!` version, respectively. The ⟨*code*⟩ associated to this first case is left in the input stream, followed by the ⟨*true code*⟩, and other cases are discarded. If none of the cases match then only the ⟨*false code*⟩ is inserted.

The unstarred version is `any`, starred version is `all`.

TEXhackers note: The process in ⟨*ids*⟩ is using `\clist_map_...` of LaTeX3. It will view `{,}` as empty, while `{{},}` are not. See interface3.pdf for more details.

Supposing following commands have been used.

```
\newcondition{defined}{}
\newcondition{paper}{a4,a5,b5}
\setcondition{paper={a5,b5}}
```

It will define an identifier named `defined`, which has not id. And define an identifier named `paper`, which has three ids: `a4, a5, b5`. Then set two ids: `a5,b5` for `paper` identifier.

`any` will be evaluated to `true` if ⟨*identifier=ids list*⟩ matches any of one statement described followed:

1. ⟨*identifier=ids list*⟩ is exactly `*`;
2. ⟨*identifier=ids list*⟩ is exactly a defined *identifier*, such as `paper`, or `defined`;
3. ⟨*identifier=ids list*⟩ is a defined *identifier*, and its ⟨*id*⟩ is `*`, such as `paper=*` or `defined=*`;
4. ⟨*identifier=ids list*⟩ is a defined *identifier*, and *one of* item in ⟨*ids*⟩ has been set, such as `paper=b5` or `paper={a5,b5}` or `paper={a5,a0}` (`a0` unset, but `a5` already set. Any id set to ⟨*identifier*⟩ `defined` will evaluate to `false`, except `*`, because the *identifier* never have defined id, even the ⟨*ids*⟩ is empty (`defined=`);
5. *Any* single item in ⟨*identifier=ids list*⟩ matches any statements listed above, such as `paper={a5,a0},undefined`.

`all` will be evaluated to true if ⟨*identifier=ids list*⟩ matches any of one statement described followed.

1. ⟨*identifier=ids list*⟩ is exactly `*`;
2. ⟨*identifier=ids list*⟩ is exactly a defined *identifier*, such as `paper`, or `defined`;
3. ⟨*identifier=ids list*⟩ is a defined *identifier*, and its ⟨*id*⟩ is `*`, such as `paper=*` or `defined=*`;

4. ⟨*identifier=ids list*⟩ is a defined *identifier*, and *all of* the ⟨*ids*⟩ has been set, such as `paper=b5` or `paper={a5,b5}`. The any id set to ⟨*identifier*⟩ `defined` will evaluate to `false`, except `*`, because the *identifier* never have defined id, even the ⟨*ids*⟩ is empty (`defined=`);

5. *All* items in ⟨*identifier=ids list*⟩ match any statements listed above, such as `paper={a5,b5},defined`.

```
\newconditioncommand
\renewconditioncommand
\provideconditioncommand
\declareconditioncommand
\neweconditioncommand
\reneweconditioncommand
\provideeconditioncommand
\declareeconditioncommand
```

`\newconditioncommand *` ⟨*function*⟩ `[`⟨*arg nums*⟩`] [`⟨*default*⟩`] {`⟨*code*⟩`}`
`\neweconditioncommand *` ⟨*function*⟩ `[`⟨*arg nums*⟩`] {`⟨*code*⟩`}`

Those commands are just like `\newcommand`, `\renewcommand`, etc. They will define command like `\foo+{`⟨*identifier=ids list*⟩`}`⟨*args*⟩. The optional argument cannot contain `\par`.

The `e`-version commands define expandable command, and cannot set default value. However you can use `xparse`-like command illustrated followed, which can set default value.

Unstarred version is `\long`, just like LaTeX's.

The new ⟨*function*⟩ will take one optional argument: `+`, the function is just like the `*` in `\conditionif`, etc. And one mandatory argument ⟨*identifier=ids list*⟩. After absorbing these two arguments, then absorb arguments of given ⟨*arg nums*⟩, or use ⟨*default*⟩, if given.

```
\NewConditionCommand
\RenewConditionCommand
\ProvideConditionCommand
\DeclareConditionCommand
\NewExpandableConditionCommand
\RenewExpandableConditionCommand
\ProvideExpandableConditionCommand
\DeclareExpandableConditionCommand
```

`\NewConditionCommand` ⟨*function*⟩ `{`⟨*arg spec*⟩`} {`⟨*code*⟩`}`

Those commands are just like `xparse`'s `\NewDocumentCommand`, etc. They will define command like `\foo+{`⟨*identifier=ids list*⟩`}`⟨*args*⟩.

⟨*arg spec*⟩ must follow the rules of the `xparse` package.

The new ⟨*function*⟩ will take one optional argument: `+`, the function is just like the `*` in `\conditionif`, etc. And one mandatory argument ⟨*identifier=ids list*⟩. After absorbing these two arguments, then absorb arguments of given ⟨*arg spec*⟩.

2 Examples

```
\newcondition{defined}{}
\newcondition{paper}{a4,a5,b5}
\setcondition{paper={a5,b5}}

\conditionif [*]{t}{f}:   t
\conditionif [defined]{t}{f}:   t
\conditionif [defined=]{t}{f}:   f
\conditionif [defined=*]{t}{f}:   t
\conditionif [defined=a]{t}{f}:   f
```

```
\conditionif [paper={a5,a0},undefined]{t}{f}:   t

\conditionif *[*]{t}{f}:   t
\conditionif *[defined]{t}{f}:   t
\conditionif *[defined={„}]{t}{f}:   f
\conditionif *[defined=*]{t}{f}:   t
\conditionif *[defined=a]{t}{f}:   f
\conditionif *[paper={a5,a0},undefined]{t}{f}:   f
\conditionif *[*,undefined]{t}{f}:   f
\conditionif *[paper={a5,b5}]{t}{f}:   t
\conditionif *[paper={a5„b5}]{t}{f}:   t
\conditionif *[paper={a5,b6,a5}]{t}{f}:   f
\conditionif *[paper={a5,{ },45}]{t}{f}:   f
\conditionif *[*,defined,paper={a5,b5}]{t}{f}:   t
```

```
\def\truetext{true} \def\falsetext{false}
\edef\testa{\econditionif[*]{true}{false}}
\ifx\testa\truetext t\else f\fi
\ifx\testa\falsetext t\else f\fi
\strcmp {\econditionif[*]{true}{false}} {true}
\strcmp {\econditionif[*]{true}{false}} {false}
\strcmp {\testa} {\truetext}
\strcmp {\testa} {\falsetext}
```

tftruetrue truefalse truetrue truefalse

```
\conditioncase{
  {paper=a3} {a3}
  {paper=a4} {a4}
  {paper,defined} {pd}
}
```

pd

```
\conditioncaseTF{
  {paper=a3} {a3}
  {paper=a4} {a4}
  {paper,defined} {pd}
}{true}{false}
```

pdtrue

```
\conditioncase!{
  {paper=a3} {a3}
  {paper=a4} {a4}
  {paper,defined} {pd}
}
```

a3

```
\conditioncaseTF!{
  {paper=a3} {a3}
  {paper=a4} {a4}
  {paper,defined} {pd}
}{true}{false}
```

a3true

```
\newconditioncommand\longprotectedcdcmd{longprotectedcdcmd}
\newconditioncommand\longprotectedcdcmdi[1]{longprotectedcdcmdi<#1>}
\newconditioncommand\longprotectedcdcmdio[1][DFT]{longprotectedcdcmdio<#1>}
\newconditioncommand*\shortprotectedcdcmd{shortprotectedcdcmd}
\newconditioncommand*\shortprotectedcdcmdi[1]{shortprotectedcdcmdi<#1>}
\newconditioncommand*\shortprotectedcdcmdio[1][DFT]{shortprotectedcdcmdio<#1>}

\setcondition{paper={a4,a5}}
\longprotectedcdcmd{*}
\longprotectedcdcmdi{*}{1\par arg}
\longprotectedcdcmdio{*}
\longprotectedcdcmdio{*}[1opt]
\longprotectedcdcmdio{paper=a4}[1opt a4]
\longprotectedcdcmdio+{paper={a4,a7}}[1opt a4a7]
\shortprotectedcdcmd{*}
\shortprotectedcdcmdi{*}{1\par arg}
\shortprotectedcdcmdio{*}
\shortprotectedcdcmdio{*}[1opt]
\shortprotectedcdcmdio{paper=a4}[1opt a4]
\shortprotectedcdcmdio+{paper={a4,a7}}[1opt a4a7]
```

```
longprotectedcdcmd
longprotectedcdcmdi<1
arg>
longprotectedcdcmdio<DFT>
longprotectedcdcmdio<1opt>
longprotectedcdcmdio<1opt a4>
shortprotectedcdcmd
shortprotectedcdcmdi<1arg>
shortprotectedcdcmdio<DFT>
shortprotectedcdcmdio<1opt>
shortprotectedcdcmdio<1opt a4>
```

3 For package authors

```
\cdcmd_any_if_p:n            ⋆
\cdcmd_any_if_p:(o|V|f)      ⋆
\cdcmd_any_if:nTF            ⋆
\cdcmd_any_if:(o|V|f)TF      ⋆
\cdcmd_all_if_p:n            ⋆
\cdcmd_all_if_p:(o|V|f)      ⋆
\cdcmd_all_if:nTF            ⋆
\cdcmd_all_if:(o|V|f)TF      ⋆
```

The meaning should be obvious.

```
\cdcmd_any_case_true:nTF
\cdcmd_any_case_false:nTF
\cdcmd_all_case_true:nTF
\cdcmd_all_case_false:nTF
```

The meaning should be obvious.

4 Implementation

```
⟨*package⟩
⟨@@=cdcmd⟩
\str_const:Nn \c_cdcmd_all_str { * }
\clist_new:N \g__cdcmd_clist
\bool_new:N \l__cdcmd_clear_set_bool
\msg_new:nnn { cdcmd } { condition-exist }
  { The~ condition~ '#1'~ you~ try~ to~ new~ already~ exists. }
\msg_new:nnn { cdcmd } { condition-not-exist }
  { The~ condition~ '#1'~ not~ exists. }
\msg_new:nnn { cdcmd } { condition-id-not-exist }
  { The~ id~ '#2' of~ condition~ '#1'~ not~ exists. }
```

`\cdcmd_if_exist_p:n`
`\cdcmd_if_exist:nTF`

Condition ⟨*indentifier*⟩ if exist.

```
\prg_new_conditional:Npnn \cdcmd_if_exist:n #1 { p, T, F, TF }
  {
    \clist_if_exist:cTF { c__cdcmd_condition@ #1 _clist }
      { \prg_return_true: } { \prg_return_false: }
  }
```

(*End definition for* `\cdcmd_if_exist:nTF`. *This function is documented on page* **??**.)

`\cdcmd_cd_id_if_exist:nnTF`

ID ⟨*id*⟩ of condition ⟨*indentifier*⟩ if exist.

```
\prg_new_conditional:Npnn \cdcmd_cd_id_if_exist:nn #1#2 { T, F, TF }
  {
    \clist_if_in:cnTF { c__cdcmd_condition@ #1 _clist } {#2}
      { \prg_return_true: } { \prg_return_false: }
  }
```

(*End definition for* `\cdcmd_cd_id_if_exist:nnTF`. *This function is documented on page* **??**.)

\cdcmd_new:nn
\cdcmd_set:nn
\cdcmd_set_cdcmd_single:nn
\cdcmd_set_cdcmd_all:n
\cdcmd_set:n
\cdcmd_clear_set:n

```
\cs_new_nopar:Npn \cdcmd_new:nn #1#2
  {
    \cdcmd_if_exist:nTF {#1}
      { \msg_error:nnn { cdcmd } { condition-exist } {#1} }
      {
        \clist_gput_right:Nn \g__cdcmd_clist {#1}
        \clist_const:cn { c__cdcmd_condition@ #1 _clist } {#2}
        \clist_new:c { l__cdcmd_curr_condition@ #1 _clist }
      }
  }
\cs_new_nopar:Npn \cdcmd_set:nn #1#2
  {
    \cdcmd_if_exist:nTF {#1}
      {
        \bool_if:NT \l__cdcmd_clear_set_bool
          { \clist_clear:c { l__cdcmd_curr_condition@ #1 _clist } }
        \clist_map_inline:nn {#2}
          {
            \str_if_eq:eeTF {##1} { \c_cdcmd_all_str }
              { \clist_map_break:n { \cdcmd_set_cdcmd_all:n {#1} } }
              { \cdcmd_set_cdcmd_single:nn {#1} {##1} }
          }
      }
      { \msg_warning:nnn { cdcmd } { condition-not-exist } {#1} }
  }
\cs_new_nopar:Npn \cdcmd_set_cdcmd_single:nn #1#2
  {
    \cdcmd_if_exist:nTF {#1}
      {
        \cdcmd_cd_id_if_exist:nnTF {#1} {#2}
          { \clist_put_right:cn { l__cdcmd_curr_condition@ #1 _clist } {#2} }
          { \msg_warning:nnnn { cdcmd } { condition-id-not-exist } {#1} {#2} }
      }
      { \msg_warning:nnn { cdcmd } { condition-not-exist } }
  }
\cs_new_nopar:Npn \cdcmd_set_cdcmd_all:n #1
  {
    \cdcmd_if_exist:nTF {#1}
      {
        \clist_set_eq:cc
          { l__cdcmd_curr_condition@ #1 _clist }
          { c__cdcmd_condition@ #1 _clist }
      }
      { \msg_warning:nnn { cdcmd } { condition-not-exist } {#1} }
  }
\cs_new_nopar:Npn \cdcmd_set:n
  { \keyval_parse:NNn \cdcmd_set_cdcmd_all:n \cdcmd_set:nn }
\cs_new_nopar:Npn \cdcmd_clear_set:n #1
  {
    \bool_set_true:N \l__cdcmd_clear_set_bool
    \keyval_parse:NNn \cdcmd_set_cdcmd_all:n \cdcmd_set:nn {#1}
    \bool_set_false:N \l__cdcmd_clear_set_bool
  }
```

(*End definition for* `\cdcmd_new:nn` *and others. These functions are documented on page* **??**.)

`\cdcmd_any_if_p:n`
`\cdcmd_any_if_p:o`
`\cdcmd_any_if_p:V`
`\cdcmd_any_if_p:f`
`\cdcmd_any_if:nTF`
`\cdcmd_any_if:oTF`
`\cdcmd_any_if:VTF`
`\cdcmd_any_if:fTF`

```
\cs_new:Npn \cdcmd_any_if:nTF #1
  {
    \bool_if:nTF
      {
        \keyval_parse:NNn
          \__cdcmd_any_i:n \__cdcmd_any_ii:nn {#1}
        \c_false_bool
      }
  }
\cs_new:Npn \cdcmd_any_if_p:n #1
  {
    \bool_if_p:n
      {
        \keyval_parse:NNn
          \__cdcmd_any_i:n \__cdcmd_any_ii:nn {#1}
        \c_false_bool
      }
  }
\cs_new:Npn \cdcmd_any_if:nT #1#2 { \cdcmd_any_if:nTF {#1} {#2} { } }
\cs_new:Npn \cdcmd_any_if:nF #1   { \cdcmd_any_if:nTF {#1} { } }
\cs_new:Npn \cdcmd_any_if:nFT #1#2#3 { \cdcmd_any_if:nTF {#1} {#3} {#2} }
\prg_generate_conditional_variant:Nnn \cdcmd_any_if:n { o, V, f } { p, T, F, TF }
\cs_new:Npn \__cdcmd_any_i:n #1
  {
    \str_if_eq:eeTF {#1} { \c_cdcmd_all_str }
      { \c_true_bool || }
      { \cdcmd_if_exist:nT {#1} { \c_true_bool || } }
  }
\cs_new:Npn \__cdcmd_any_ii:nn #1#2
  {
    \cdcmd_if_exist:nT {#1}
      {
        \clist_map_tokens:nn {#2}
          { \__cdcmd_any_ii_aux:nn {#1} }
      }
  }
\cs_new:Npn \__cdcmd_any_ii_aux:nn #1#2
  {
    \str_if_eq:eeTF {#2} { \c_cdcmd_all_str }
      { \clist_map_break:n { \tex_the:D \c_true_bool || } }
      {
        \__cdcmd_clist_if_in:cnT { l__cdcmd_curr_condition@ #1 _clist } {#2}
          { \clist_map_break:n { \tex_the:D \c_true_bool || } }
      }
  }
```

(*End definition for* `\cdcmd_any_if:nTF`*. This function is documented on page* 6.)

`\cdcmd_all_if_p:n`
`\cdcmd_all_if_p:o`
`\cdcmd_all_if_p:V`
`\cdcmd_all_if_p:f`
`\cdcmd_all_if:nTF`
`\cdcmd_all_if:oTF`
`\cdcmd_all_if:VTF`
`\cdcmd_all_if:fTF`

```
\cs_new:Npn \cdcmd_all_if:nTF #1
  {
    \bool_if:nTF
      {
        \keyval_parse:NNn
          \__cdcmd_all_i:n \__cdcmd_all_ii:nn {#1}
        \c_true_bool
      }
  }
\cs_new:Npn \cdcmd_all_if_p:n #1
  {
    \bool_if_p:n
      {
        \keyval_parse:NNn
          \__cdcmd_all_i:n \__cdcmd_all_ii:nn {#1}
        \c_true_bool
      }
  }
\cs_new:Npn \cdcmd_all_if:nT #1#2 { \cdcmd_all_if:nTF {#1} {#2} { } }
\cs_new:Npn \cdcmd_all_if:nF #1   { \cdcmd_all_if:nTF {#1} { } }
\cs_new:Npn \cdcmd_all_if:nFT #1#2#3 { \cdcmd_all_if:nTF {#1} {#3} {#2} }
\prg_generate_conditional_variant:Nnn \cdcmd_all_if:n { o, V, f } { p, T, F, TF }
\cs_new:Npn \__cdcmd_all_i:n #1
  {
    \str_if_eq:eeF {#1} { \c_cdcmd_all_str }
      { \cdcmd_if_exist:nF {#1} { \c_false_bool && } }
  }
\cs_new:Npn \__cdcmd_all_ii:nn #1#2
  {
    \cdcmd_if_exist:nTF {#1}
      {
        \bool_lazy_and_p:nn
          { \int_compare_p:n { \clist_count:n {#2} > 0 } }
          {
            \int_compare_p:n
              { \clist_map_tokens:nn {#2} { \__cdcmd_all_ii_aux:nn {#1} } 1 > 0 }
          } &&
      }
      { \c_false_bool && }
  }
\cs_new:Npn \__cdcmd_all_ii_aux:nn #1#2
  {
    \str_if_eq:eeF {#2} { \c_cdcmd_all_str }
      {
        \__cdcmd_clist_if_in:cnF { l__cdcmd_curr_condition@ #1 _clist } {#2}
          { \clist_map_break:n { - } }
      }
  }
```

(*End definition for* `\cdcmd_all_if:nTF`. *This function is documented on page 6.*)

__cdcmd_clist_if_in_p:Nn
__cdcmd_clist_if_in_p:NV
__cdcmd_clist_if_in_p:No
__cdcmd_clist_if_in_p:cn
__cdcmd_clist_if_in_p:cV
__cdcmd_clist_if_in_p:co
__cdcmd_clist_if_in:Nn*TF*
__cdcmd_clist_if_in:NV*TF*
__cdcmd_clist_if_in:No*TF*
__cdcmd_clist_if_in:cn*TF*
__cdcmd_clist_if_in:cV*TF*
__cdcmd_clist_if_in:co*TF*
__cdcmd_clist_if_in_p:nn
__cdcmd_clist_if_in_p:nV
__cdcmd_clist_if_in_p:no

```
\prg_new_conditional:Npnn \__cdcmd_clist_if_in:Nn #1#2 { p, T, F, TF }
  {
    \int_compare:nTF
      { 0 \clist_map_tokens:Nn #1 { \__cdcmd_if_eq_break:ee {#2} } > 0 }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__cdcmd_clist_if_in:Nn
  { NV, No, cn, cV, co } { p, T, F, TF }
\prg_new_conditional:Npnn \__cdcmd_clist_if_in:nn #1#2 { p, T, F, TF }
  {
    \int_compare:nTF
      { 0 \clist_map_tokens:nn {#1} { \__cdcmd_if_eq_break:ee {#2} } > 0 }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__cdcmd_clist_if_in:nn { nV, no } { p, T, F, TF }
\cs_new:Npn \__cdcmd_if_eq_break:ee #1#2
  {
    \str_if_eq:eeT {#1} {#2} { \clist_map_break:n { 1 } }
  }
```

(*End definition for* `\__cdcmd_clist_if_in:NnTF` *and* `\__cdcmd_clist_if_in:nnTF`.)

`\cdcmd_any_case_true:nTF`
`\cdcmd_any_case_false:nTF`
`\cdcmd_all_case_true:nTF`
`\cdcmd_all_case_false:nTF`

Conditional case, see also `\bool_case_true:n` and `\bool_case_false:n` in source3.pdf.

```
\scan_new:N \s__cdcmd_mark
\scan_new:N \s__cdcmd_stop
\cs_new:Npn \cdcmd_any_case_true:nTF { \exp:w \__cdcmd_any_case_true:nTF }
\cs_new:Npn \cdcmd_any_case_true:n #1 { \exp:w \__cdcmd_any_case_true:nTF {#1} { } { } }
\cs_new:Npn \cdcmd_all_case_true:nTF { \exp:w \__cdcmd_all_case_true:nTF }
\cs_new:Npn \cdcmd_all_case_true:n #1 { \exp:w \__cdcmd_all_case_true:nTF {#1} { } { } }
\cs_new:Npn \cdcmd_any_case_false:nTF { \exp:w \__cdcmd_any_case_false:nTF }
\cs_new:Npn \cdcmd_any_case_false:n #1 { \exp:w \__cdcmd_any_case_false:nTF {#1} { } { } }
\cs_new:Npn \cdcmd_all_case_false:nTF { \exp:w \__cdcmd_all_case_false:nTF }
\cs_new:Npn \cdcmd_all_case_false:n #1 { \exp:w \__cdcmd_all_case_false:nTF {#1} { } { } }
\cs_new:Npn \__cdcmd_any_case_true:nTF #1#2#3
  {
    \__cdcmd_case:Nw \cdcmd_any_if:nTF #1 { * } { }
      \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
  }
\cs_new:Npn \__cdcmd_all_case_true:nTF #1#2#3
  {
    \__cdcmd_case:Nw \cdcmd_all_if:nTF #1 { * } { }
      \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
  }
\cs_new:Npn \__cdcmd_any_case_false:nTF #1#2#3
  {
    \__cdcmd_case:Nw \cdcmd_any_if:nFT #1 { * } { }
      \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
  }
\cs_new:Npn \__cdcmd_all_case_false:nTF #1#2#3
  {
    \__cdcmd_case:Nw \cdcmd_all_if:nFT #1 { * } { }
      \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
  }
\cs_new:Npn \__cdcmd_case:Nw #1#2#3
  { #1 {#2} { \__cdcmd_case_end:nw {#3} } { \__cdcmd_case:Nw #1 } }
\cs_new:Npn \__cdcmd_case_end:nw #1#2#3 \s__cdcmd_mark #4#5 \s__cdcmd_stop
  { \exp_end: #1 #4 }
```

(*End definition for* `\cdcmd_any_case_true:nTF` *and others. These functions are documented on page 6.*)

`\newcondition`
`\setcondition`
`\clearcondition`

Conditional setting command for document.

```
\NewDocumentCommand \newcondition { >{ \TrimSpaces } m } { \cdcmd_new:nn {#1} }
\NewDocumentCommand \setcondition { t+ }
  { \IfBooleanTF {#1} { \cdcmd_set:n } { \cdcmd_clear_set:n } }
\NewDocumentCommand \clearcondition { !O{*} }
  {
    \clist_map_inline:nn {#1}
      {
        \str_if_eq:eeTF {##1} { \c_cdcmd_all_str }
          {
            \clist_map_break:n
              { \exp_after:wN \clearcondition \exp_after:wN [ \g__cdcmd_clist ] }
          }
          {
            \cdcmd_if_exist:nTF {##1}
              { \clist_clear:c { l__cdcmd_curr_condition@ ##1 _clist } }
              { \msg_warning:nnn { cdcmd } { condition-not-exist } {##1} }
          }
      }
  }
```

(*End definition for* `\newcondition`*,* `\setcondition`*, and* `\clearcondition`*. These functions are documented on page 1.*)

```
\NewExpandableDocumentCommand \econditionif { s O{*} +m +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_if:nTF }
      { \cdcmd_any_if:nTF }
        {#2} {#3} {#4}
  }
\NewExpandableDocumentCommand \econditioncmd { s O{*} +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_if:nTF }
      { \cdcmd_any_if:nTF }
        {#2} {#3} { }
  }
\NewExpandableDocumentCommand \econditioncase { s +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_case:n {#2} }
      { \cdcmd_any_case:n {#2} }
  }
\NewExpandableDocumentCommand \econditioncaseTF { s +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_case:nTF {#2} }
      { \cdcmd_any_case:nTF {#2} }
  }
\NewDocumentCommand \conditionif { s O{*} +m +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_if:nTF }
      { \cdcmd_any_if:nTF }
        {#2} {#3} {#4}
  }
\NewDocumentCommand \conditioncmd { s O{*} +m }
  {
    \IfBooleanTF {#1}
      { \cdcmd_all_if:nTF }
      { \cdcmd_any_if:nTF }
        {#2} {#3} { }
  }
\NewDocumentCommand \conditioncase { s t! +m }
  {
    \IfBooleanTF {#2}
      {
        \IfBooleanTF {#1}
          { \cdcmd_all_case_false:n {#3} }
          { \cdcmd_any_case_false:n {#3} }
      }
      {
        \IfBooleanTF {#1}
          { \cdcmd_all_case_true:n {#3} }
          { \cdcmd_any_case_true:n {#3} }
      }
  }
\NewDocumentCommand \conditioncaseTF { s t! +m }
  {
    \IfBooleanTF {#2}
      {
        \IfBooleanTF {#1}
          { \cdcmd_all_case_false:nTF {#3} }
          { \cdcmd_any_case_false:nTF {#3} }
      }
      {
        \IfBooleanTF {#1}
          { \cdcmd_all_case_true:nTF {#3} }
          { \cdcmd_any_case_true:nTF {#3} }
      }
  }
```

Define new xparse like conditional command.

```
\str_const:Nn \c_cdcmd_pair_u_str { cdcmd@u@ }
\str_const:Nn \c_cdcmd_pair_n_str { cdcmd@n@ }
\cs_new_nopar:Npn \__cdcmd_cs_pair_u:N #1
  { \c_cdcmd_pair_u_str \cs_to_str:N #1 }
\cs_new_nopar:Npn \__cdcmd_cs_pair_n:N #1
  { \c_cdcmd_pair_n_str \cs_to_str:N #1 }
\cs_new:Npn \__cdcmd_arg_spec_from_num:nn #1#2
  {
    \if_case:w 0#1 \exp_stop_f:
    \or: #2 \or: #2#2 \or: #2#2#2 \or: #2#2#2#2 \or: #2#2#2#2#2 \or: #2#2#2#2#2#2
    \or: #2#2#2#2#2#2#2 \or: #2#2#2#2#2#2#2#2 \else: #2#2#2#2#2#2#2#2#2 \fi:
  }
```

```
\cs_new_nopar:Npn \__cdcmd_cs_pair_u:Nn #1#2
  {
    \c_cdcmd_pair_u_str
    \cs_to_str:N #1 :
    \__cdcmd_arg_spec_from_num:nn {#2} { n }
  }
\cs_new_nopar:Npn \__cdcmd_cs_pair_n:Nn #1#2
  {
    \c_cdcmd_pair_n_str
    \cs_to_str:N #1 :
    \__cdcmd_arg_spec_from_num:nn {#2} { n }
  }
% do not check cs_if_free, let xparse do it
\cs_new:Npn \__cdcmd_new_cdcmd_command:NN #1#2
  {
    \cs_new_protected:Npn #1 ##1##2##3
      {
        #2 ##1 { t+ m }
          {
            \IfBooleanTF {####1}
              { \cdcmd_all_if:nTF }
              { \cdcmd_any_if:nTF }
                {####2}
                { \use:c { \__cdcmd_cs_pair_u:N ##1 } }
                { \use:c { \__cdcmd_cs_pair_n:N ##1 } }
          }
        \exp_args:Nc #2
          { \__cdcmd_cs_pair_u:N ##1 } {##2} {##3}
        \exp_args:Nc #2
          { \__cdcmd_cs_pair_n:N ##1 } {##2} { }
      }
  }
\seq_const_from_clist:Nn \c__cdcmd_Command_seq
  {
    \NewDocumentCommand ,
    \RenewDocumentCommand ,
    \ProvideDocumentCommand ,
    \DeclareDocumentCommand ,
    \NewExpandableDocumentCommand ,
    \RenewExpandableDocumentCommand ,
    \ProvideExpandableDocumentCommand ,
    \DeclareExpandableDocumentCommand ,
  }
\seq_const_from_clist:Nn \c__cdcmd_COMMAND_seq
  {
    \NewConditionCommand ,
    \RenewConditionCommand ,
    \ProvideConditionCommand ,
    \DeclareConditionCommand ,
    \NewExpandableConditionCommand ,
    \RenewExpandableConditionCommand ,
    \ProvideExpandableConditionCommand ,
    \DeclareExpandableConditionCommand ,
  }
```

```
\seq_mapthread_function:NNN
  \c__cdcmd_COMMAND_seq
  \c__cdcmd_Command_seq
  \__cdcmd_new_cdcmd_command:NN
```

(*End definition for* .)

Define LaTeX like command.

```
% do not check cs_if_free, let xparse do it
\cs_new:Npn \__cdcmd_new_cdcmd_cmd_no:nnn #1#2#3
  {
    \cs_new_protected:Npn #1 ##1##2##3
      {
        #3 ##1 { t+ m }
          {
            \IfBooleanTF {####1}
              { \cdcmd_all_if:nTF }
              { \cdcmd_any_if:nTF }
                { ####2 }
                { \use:c { \__cdcmd_cs_pair_u:Nn ##1 {##2} } }
                { \use:c { \__cdcmd_cs_pair_n:Nn ##1 {##2} } }
          }
        #2 { \__cdcmd_cs_pair_u:Nn ##1 {##2} } {##3}
        #2 { \__cdcmd_cs_pair_n:Nn ##1 {##2} } { }
      }
  }
\cs_generate_variant:Nn \__cdcmd_new_cdcmd_cmd_no:nnn { xxx }
\seq_const_from_clist:Nn \c__cdcmd_cmd_no_seq
  {
    \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
    \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
    \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
    \cs_set:cn , \cs_set_nopar:cn ,
    \cs_set:cn , \cs_set_nopar:cn ,
    \cs_set:cn , \cs_set_nopar:cn ,
  }
\seq_const_from_clist:Nn \c__cdcmd_Cmd_no_seq
  {
    \NewDocumentCommand , \NewDocumentCommand ,
    \RenewDocumentCommand , \RenewDocumentCommand ,
    \DeclareDocumentCommand , \DeclareDocumentCommand ,
    \NewExpandableDocumentCommand , \NewExpandableDocumentCommand ,
    \RenewExpandableDocumentCommand , \RenewExpandableDocumentCommand ,
    \DeclareExpandableDocumentCommand , \DeclareExpandableDocumentCommand ,
  }
\seq_const_from_clist:Nn \c__cdcmd_CMD_no_seq
  {
    \__cdcmd_new_cdcmd_p_l_num:Nnn , \__cdcmd_new_cdcmd_p_nl_num:Nnn ,
    \__cdcmd_renew_cdcmd_p_l_num:Nnn , \__cdcmd_renew_cdcmd_p_nl_num:Nnn ,
    \__cdcmd_declare_cdcmd_p_l_num:Nnn , \__cdcmd_declare_cdcmd_p_nl_num:Nnn ,
    \__cdcmd_new_cdcmd_np_l_num:Nnn , \__cdcmd_new_cdcmd_np_nl_num:Nnn ,
    \__cdcmd_renew_cdcmd_np_l_num:Nnn , \__cdcmd_renew_cdcmd_np_nl_num:Nnn ,
    \__cdcmd_declare_cdcmd_np_l_num:Nnn , \__cdcmd_declare_cdcmd_np_nl_num:Nnn ,
  }
```

```
\int_step_inline:nn { 6 }
  {
    \__cdcmd_new_cdcmd_cmd_no:xxx
      { \seq_item:Nn \c__cdcmd_CMD_no_seq {#1} }
      { \seq_item:Nn \c__cdcmd_cmd_no_seq {#1} }
      { \seq_item:Nn \c__cdcmd_Cmd_no_seq {#1} }
  }
\tl_new:N \l__cdcmd_arg_spec_tl
\cs_new:Npn \__cdcmd_generate_arg_spec:nnn #1#2#3
  {
    \tl_set:Nn \l__cdcmd_arg_spec_tl { O{#2} }
    \if_int_compare:w #1 > 1 \exp_stop_f:
      \int_step_inline:nn {#1-1} { \tl_put_right:Nn \l__cdcmd_arg_spec_tl {#3} }
    \fi:
  }
\cs_new:Npn \__cdcmd_new_cdcmd_cmd_o_aux:nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2##3##4##5
      {
        #2 ##1 { t+ m }
          {
            \IfBooleanTF{####1}
              { \cdcmd_all_if:nTF }
              { \cdcmd_any_if:nTF }
                {####2}
                { \use:c { \__cdcmd_cs_pair_u:N ##1 } }
                { \use:c { \__cdcmd_cs_pair_n:N ##1 } }
          }
        \__cdcmd_generate_arg_spec:nnn {##2} {##3} {##5}
        \exp_args:NcV #2 { \__cdcmd_cs_pair_u:N ##1 } \l__cdcmd_arg_spec_tl {##4}
        \exp_args:NcV #2 { \__cdcmd_cs_pair_n:N ##1 } \l__cdcmd_arg_spec_tl { }
      }
  }
\seq_const_from_clist:Nn \c__cdcmd_CMD_o_seq
  { \NewDocumentCommand , \RenewDocumentCommand , \DeclareDocumentCommand }
\seq_const_from_clist:Nn \c__cdcmd_cmd_o_seq
  {
    \__cdcmd_new_cdcmd_o_num:Nnnnn ,
    \__cdcmd_renew_cdcmd_o_num:Nnnnn ,
    \__cdcmd_declare_cdcmd_o_num:Nnnnn ,
  }
\seq_mapthread_function:NNN
  \c__cdcmd_cmd_o_seq
  \c__cdcmd_CMD_o_seq
  \__cdcmd_new_cdcmd_cmd_o_aux:nn
\cs_new_protected:Npn \__cdcmd_new_cdcmd_cmd_ne_aux:n #1
  {
    \exp_args:Nc \NewDocumentCommand { #1 conditioncommand } { s m O{0} o +m }
      {
        \IfBooleanTF{##1}
          {
            \IfNoValueTF{##4}
              { \use:c { __cdcmd_ #1 _cdcmd_p_nl_num:Nnn } ##2 {##3} {##5} }
              { \use:c { __cdcmd_ #1 _cdcmd_o_num:Nnnnn } ##2 {##3} {##4} {##5} { m } }
          }
          {
            \IfNoValueTF{##4}
              { \use:c { __cdcmd_ #1 _cdcmd_p_l_num:Nnn } ##2 {##3} {##5} }
              { \use:c { __cdcmd_ #1 _cdcmd_o_num:Nnnnn } ##2 {##3} {##4} {##5} { +m } }
          }
      }
  }
\clist_map_function:nN { new, renew, declare } \__cdcmd_new_cdcmd_cmd_ne_aux:n
\NewDocumentCommand \provideconditioncommand { s m O{0} o +m }
  {
    \cs_if_free:NT #2
      {
        \IfBooleanTF{#1}
          {
            \IfNoValueTF{#4}
              { \newconditioncommand * #2 [#3] {#5} }
              { \newconditioncommand * #2 [#3] [#4] {#5} }
          }
          {
            \IfNoValueTF{#4}
              { \newconditioncommand #2 [#3] {#5} }
              { \newconditioncommand #2 [#3] [#4] {#5} }
          }
      }
  }

\int_step_inline:nnnn { 7 } { 1 } { 12 }
  {
    \__cdcmd_new_cdcmd_cmd_no:xxx
      { \seq_item:Nn \c__cdcmd_CMD_no_seq {#1} }
      { \seq_item:Nn \c__cdcmd_cmd_no_seq {#1} }
      { \seq_item:Nn \c__cdcmd_Cmd_no_seq {#1} }
  }

\cs_new_protected:Npn \__cdcmd_new_cdcmd_cmd_e_no_aux:n #1
  {
    \exp_args:Nc \NewDocumentCommand { #1 econditioncommand } { s m O{0} +m }
      {
        \IfBooleanTF{##1}
          { \use:c { __cdcmd_ #1 _cdcmd_np_nl_num:Nnn } ##2 {##3} {##4} }
          { \use:c { __cdcmd_ #1 _cdcmd_np_l_num:Nnn } ##2 {##3} {##4} }
      }
  }
\clist_map_function:nN { new, renew, declare } \__cdcmd_new_cdcmd_cmd_e_no_aux:n
\NewDocumentCommand \provideeconditioncommand { s m O{0} +m }
  {
    \cs_if_free:NT #2
      {
        \IfBooleanTF{#1}
          { \neweconditioncommand * #2 [#3] {#4} }
          { \neweconditioncommand #2 [#2] {#4} }
      }
  }
```

(*End definition for .*)

532 ⟨/package⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.