

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 `styled-cmd` package, you can define `\protected` command, but `cdcmd` can define expandable conditional command as well.

1 Main Interface

<code>\newcondition</code>	<code>\newcondition {⟨identifier⟩} {⟨id(s)⟩}</code>
<code>\setcondition</code>	<code>\setcondition + {⟨identifier=ids list⟩}</code>
<code>\clearcondition</code>	<code>\clearcondition[⟨identifier(s)⟩]</code>

`\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.

<code>\conditionif</code>	<code>\conditionif * [⟨identifier=ids list⟩] {⟨true⟩} {⟨false⟩}</code>
<code>\conditioncmd</code>	<code>\conditioncmd * [⟨identifier=ids list⟩] {⟨material⟩}</code>

<code>\econditionif</code>	When the <code>⟨identifier=ids list⟩</code> makes <code>true</code> condition, leave <code>⟨true⟩/⟨material⟩</code> in the input stream. Leaving <code>⟨false⟩</code> when the condition is <code>false</code> .
<code>\econditioncmd</code>	

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

<code>\conditioncase</code> <code>\conditioncaseTF</code> <code>\econditioncase</code> <code>\econditioncaseTF</code>	<code>\conditioncaseTF * !</code> <code>{</code> <code> {<identifier=ids list case₁>} {<code₁>}</code> <code> {<identifier=ids list case₂>} {<code₂>}</code> <code> ...</code> <code> {<identifier=ids list case_n>} {<code_n>}</code> <code>}</code> <code>{<true code>}</code> <code>{<false code>}</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 L^AT_EX3. 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. $\langle identifier=ids list \rangle$ is a defined *identifier*, and *all* of the $\langle ids \rangle$ has been set, such as `paper=b5` or `paper={a5,b5}`. The any id set to $\langle identifier \rangle$ defined will evaluate to `false`, except `*`, because the *identifier* never have defined id, even the $\langle ids \rangle$ is empty (`defined=`);
5. All items in $\langle identifier=ids list \rangle$ match any statements listed above, such as `paper={a5,b5},defined`.

```

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

```

```

\newconditioncommand *  $\langle function \rangle$  [ $\langle arg nums \rangle$ ] [ $\langle default \rangle$ ] { $\langle code \rangle$ }
\neweconditioncommand *  $\langle function \rangle$  [ $\langle arg nums \rangle$ ] { $\langle code \rangle$ }

```

Those commands are just like `\newcommand`, `\renewcommand`, etc. They will define command like `\foo+{ $\langle identifier=ids list \rangle$ } $\langle args \rangle$` . 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 L^AT_EX's.

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

```

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

```

```

\NewConditionCommand  $\langle function \rangle$  { $\langle arg spec \rangle$ } { $\langle code \rangle$ }

```

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

$\langle arg spec \rangle$ must follow the rules of the `xparse` package.

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

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

<code>\cdcmt_any_if_p:n</code>	★	The meaning should be obvious.
<code>\cdcmt_any_if_p:(o V f)</code>	★	
<code>\cdcmt_any_if:nTF</code>	★	
<code>\cdcmt_any_if:(o V f)TF</code>	★	
<code>\cdcmt_all_if_p:n</code>	★	
<code>\cdcmt_all_if_p:(o V f)</code>	★	
<code>\cdcmt_all_if:nTF</code>	★	
<code>\cdcmt_all_if:(o V f)TF</code>	★	

<code>\cdcmt_any_case_true:nTF</code>	The meaning should be obvious.
<code>\cdcmt_any_case_false:nTF</code>	
<code>\cdcmt_all_case_true:nTF</code>	
<code>\cdcmt_all_case_false:nTF</code>	

4 Implementation

```

1 <*package>
2 <@@=cdcmt>

3 \str_const:Nn \c_cdcmt_all_str { * }
4 \clist_new:N \g__cdcmt_clist
5 \bool_new:N \l__cdcmt_clear_set_bool
6 \msg_new:nnn { cdcmt } { condition-exist }
7   { The~ condition~ ‘#1’~ you~ try~ to~ new~ already~ exists. }
8 \msg_new:nnn { cdcmt } { condition-not-exist }
9   { The~ condition~ ‘#1’~ not~ exists. }
10 \msg_new:nnn { cdcmt } { condition-id-not-exist }
11   { The~ id~ ‘#2’ of~ condition~ ‘#1’~ not~ exists. }

\cdcmt_if_exist_p:n Condition <identifier> if exist.
\cdcmt_if_exist:nTF
12 \prg_new_conditional:Npnn \cdcmt_if_exist:n #1 { p, T, F, TF }
13   {
14     \clist_if_exist:cnTF { c__cdcmt_condition@ #1 _clist }
15     { \prg_return_true: } { \prg_return_false: }
16   }
```

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

```

\cdcmt_cd_id_if_exist:nnTF ID <id> of condition <identifier> if exist.
17 \prg_new_conditional:Npnn \cdcmt_cd_id_if_exist:nn #1#2 { T, F, TF }
18   {
19     \clist_if_in:cnTF { c__cdcmt_condition@ #1 _clist } {#2}
20     { \prg_return_true: } { \prg_return_false: }
21   }
```

(End definition for `\cdcmt_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

22 \cs_new_nopar:Npn \cdcmd_new:nn #1#2
23 {
24   \cdcmd_if_exist:nTF {#1}
25   { \msg_error:nnn { cdcmd } { condition-exist } {#1} }
26   {
27     \clist_gput_right:Nn \g__cdcmd_clist {#1}
28     \clist_const:cn { c__cdcmd_condition@ #1 _clist } {#2}
29     \clist_new:c { l__cdcmd_curr_condition@ #1 _clist }
30   }
31 }
32 \cs_new_nopar:Npn \cdcmd_set:nn #1#2
33 {
34   \cdcmd_if_exist:nTF {#1}
35   {
36     \bool_if:NT \l__cdcmd_clear_set_bool
37     { \clist_clear:c { l__cdcmd_curr_condition@ #1 _clist } }
38     \clist_map_inline:nn {#2}
39     {
40       \str_if_eq:eeTF {##1} { \c_cdcmd_all_str }
41       { \clist_map_break:n { \cdcmd_set_cdcmd_all:n {#1} } }
42       { \cdcmd_set_cdcmd_single:nn {#1} {##1} }
43     }
44   }
45   { \msg_warning:nnn { cdcmd } { condition-not-exist } {#1} }
46 }
47 \cs_new_nopar:Npn \cdcmd_set_cdcmd_single:nn #1#2
48 {
49   \cdcmd_if_exist:nTF {#1}
50   {
51     \cdcmd_cd_id_if_exist:nnTF {#1} {#2}
52     { \clist_put_right:cn { l__cdcmd_curr_condition@ #1 _clist } {#2} }
53     { \msg_warning:nnnn { cdcmd } { condition-id-not-exist } {#1} {#2} }
54   }
55   { \msg_warning:nnn { cdcmd } { condition-not-exist } }
56 }
57 \cs_new_nopar:Npn \cdcmd_set_cdcmd_all:n #1
58 {
59   \cdcmd_if_exist:nTF {#1}
60   {
61     \clist_set_eq:cc
62     { l__cdcmd_curr_condition@ #1 _clist }
63     { c__cdcmd_condition@ #1 _clist }
64   }
65   { \msg_warning:nnn { cdcmd } { condition-not-exist } {#1} }
66 }
67 \cs_new_nopar:Npn \cdcmd_set:n
68 { \keyval_parse:NNn \cdcmd_set_cdcmd_all:n \cdcmd_set:nn }
69 \cs_new_nopar:Npn \cdcmd_clear_set:n #1
70 {
71   \bool_set_true:N \l__cdcmd_clear_set_bool
72   \keyval_parse:NNn \cdcmd_set_cdcmd_all:n \cdcmd_set:nn {#1}
73   \bool_set_false:N \l__cdcmd_clear_set_bool
74 }

```

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

```

\cdcmd_any_if_p:n
\cdcmd_any_if_p:o 75 \cs_new:Npn \cdcmd_any_if:nTF #1
\cdcmd_any_if_p:V 76 {
\cdcmd_any_if_p:f 77   \bool_if:nTF
\cdcmd_any_if:nTF 78   {
\cdcmd_any_if:oTF 79     \keyval_parse:NNn
\cdcmd_any_if:VTF 80     \__cdcmd_any_i:n \__cdcmd_any_ii:nn {#1}
\cdcmd_any_if:fTF 81     \c_false_bool
82   }
83 }
84 \cs_new:Npn \cdcmd_any_if_p:n #1
85 {
86   \bool_if_p:n
87   {
88     \keyval_parse:NNn
89     \__cdcmd_any_i:n \__cdcmd_any_ii:nn {#1}
90     \c_false_bool
91   }
92 }
93 \cs_new:Npn \cdcmd_any_if:nT #1#2 { \cdcmd_any_if:nTF {#1} {#2} { } }
94 \cs_new:Npn \cdcmd_any_if:nF #1 { \cdcmd_any_if:nTF {#1} { } }
95 \cs_new:Npn \cdcmd_any_if:nFT #1#2#3 { \cdcmd_any_if:nTF {#1} {#3} {#2} }
96 \prg_generate_conditional_variant:Nnn \cdcmd_any_if:n { o, V, f } { p, T, F, TF }
97 \cs_new:Npn \__cdcmd_any_i:n #1
98 {
99   \str_if_eq:eeTF {#1} { \c_cdcmd_all_str }
100   { \c_true_bool || }
101   { \cdcmd_if_exist:nT {#1} { \c_true_bool || } }
102 }
103 \cs_new:Npn \__cdcmd_any_ii:nn #1#2
104 {
105   \cdcmd_if_exist:nT {#1}
106   {
107     \clist_map_tokens:nn {#2}
108     { \__cdcmd_any_ii_aux:nn {#1} }
109   }
110 }
111 \cs_new:Npn \__cdcmd_any_ii_aux:nn #1#2
112 {
113   \str_if_eq:eeTF {#2} { \c_cdcmd_all_str }
114   { \clist_map_break:n { \tex_the:D \c_true_bool || } }
115   {
116     \__cdcmd_clist_if_in:cnT { l__cdcmd_curr_condition@ #1 _clist } {#2}
117     { \clist_map_break:n { \tex_the:D \c_true_bool || } }
118   }
119 }

```

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

```

\cdcmd_all_if_p:n
\cdcmd_all_if_p:o 120 \cs_new:Npn \cdcmd_all_if:nTF #1
\cdcmd_all_if_p:V 121 {
\cdcmd_all_if_p:f
\cdcmd_all_if:nTF
\cdcmd_all_if:oTF
\cdcmd_all_if:VTF
\cdcmd_all_if:fTF

```



```

122     \bool_if:nTF
123     {
124         \keyval_parse:NNn
125         \__cdcmd_all_i:n \__cdcmd_all_ii:nn {#1}
126         \c_true_bool
127     }
128 }
129 \cs_new:Npn \cdcmd_all_if_p:n #1
130 {
131     \bool_if_p:n
132     {
133         \keyval_parse:NNn
134         \__cdcmd_all_i:n \__cdcmd_all_ii:nn {#1}
135         \c_true_bool
136     }
137 }
138 \cs_new:Npn \cdcmd_all_if:nT #1#2 { \cdcmd_all_if:nTF {#1} {#2} { } }
139 \cs_new:Npn \cdcmd_all_if:nF #1 { \cdcmd_all_if:nTF {#1} { } }
140 \cs_new:Npn \cdcmd_all_if:nFT #1#2#3 { \cdcmd_all_if:nTF {#1} {#3} {#2} }
141 \prg_generate_conditional_variant:Nnn \cdcmd_all_if:n { o, V, f } { p, T, F, TF }
142 \cs_new:Npn \__cdcmd_all_i:n #1
143 {
144     \str_if_eq:eeF {#1} { \c_cdcmd_all_str }
145     { \cdcmd_if_exist:nF {#1} { \c_false_bool && } }
146 }
147 \cs_new:Npn \__cdcmd_all_ii:nn #1#2
148 {
149     \cdcmd_if_exist:nTF {#1}
150     {
151         \bool_lazy_and_p:nn
152         { \int_compare_p:n { \clist_count:n {#2} > 0 } }
153         {
154             \int_compare_p:n
155             { \clist_map_tokens:nn {#2} { \__cdcmd_all_ii_aux:nn {#1} } 1 > 0 }
156         } &&
157     }
158     { \c_false_bool && }
159 }
160 \cs_new:Npn \__cdcmd_all_ii_aux:nn #1#2
161 {
162     \str_if_eq:eeF {#2} { \c_cdcmd_all_str }
163     {
164         \__cdcmd_clist_if_in:cnF { l__cdcmd_curr_condition@ #1 _clist } {#2}
165         { \clist_map_break:n { - } }
166     }
167 }

```

(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:NnTF
\__cdcmd_clist_if_in:NVTF
\__cdcmd_clist_if_in:NoTF
\__cdcmd_clist_if_in:cnTF
\__cdcmd_clist_if_in:cVTF
\__cdcmd_clist_if_in:coTF
\__cdcmd_clist_if_in_p:nn
\__cdcmd_clist_if_in_p:nV
\__cdcmd_clist_if_in_p:no

```

```

172     { \prg_return_true: } { \prg_return_false: }
173   }
174   \prg_generate_conditional_variant:Nnn \__cdcmd_clist_if_in:Nn
175   { NV, No, cn, cV, co } { p, T, F, TF }
176   \prg_new_conditional:Npnn \__cdcmd_clist_if_in:nn #1#2 { p, T, F, TF }
177   {
178     \int_compare:nTF
179     { 0 \clist_map_tokens:nn {#1} { \__cdcmd_if_eq_break:ee {#2} } > 0 }
180     { \prg_return_true: } { \prg_return_false: }
181   }
182   \prg_generate_conditional_variant:Nnn \__cdcmd_clist_if_in:nn { nV, no } { p, T, F, TF }
183   \cs_new:Npn \__cdcmd_if_eq_break:ee #1#2
184   {
185     \str_if_eq:eeT {#1} {#2} { \clist_map_break:n { 1 } }
186   }

```

(End definition for __cdcmd_clist_if_in:NnTF and __cdcmd_clist_if_in:nnTF.)

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

```

\cdcmd_any_case_false:nTF
\cdcmd_all_case_true:nTF
\cdcmd_all_case_false:nTF
187 \scan_new:N \s__cdcmd_mark
188 \scan_new:N \s__cdcmd_stop
189 \cs_new:Npn \cdcmd_any_case_true:nTF { \exp:w \__cdcmd_any_case_true:nTF }
190 \cs_new:Npn \cdcmd_any_case_true:n #1 { \exp:w \__cdcmd_any_case_true:nTF {#1} { } { } }
191 \cs_new:Npn \cdcmd_all_case_true:nTF { \exp:w \__cdcmd_all_case_true:nTF }
192 \cs_new:Npn \cdcmd_all_case_true:n #1 { \exp:w \__cdcmd_all_case_true:nTF {#1} { } { } }
193 \cs_new:Npn \cdcmd_any_case_false:nTF { \exp:w \__cdcmd_any_case_false:nTF }
194 \cs_new:Npn \cdcmd_any_case_false:n #1 { \exp:w \__cdcmd_any_case_false:nTF {#1} { } { } }
195 \cs_new:Npn \cdcmd_all_case_false:nTF { \exp:w \__cdcmd_all_case_false:nTF }
196 \cs_new:Npn \cdcmd_all_case_false:n #1 { \exp:w \__cdcmd_all_case_false:nTF {#1} { } { } }
197 \cs_new:Npn \__cdcmd_any_case_true:nTF #1#2#3
198 {
199   \__cdcmd_case:Nw \cdcmd_any_if:nTF #1 { * } { }
200   \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
201 }
202 \cs_new:Npn \__cdcmd_all_case_true:nTF #1#2#3
203 {
204   \__cdcmd_case:Nw \cdcmd_all_if:nTF #1 { * } { }
205   \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
206 }
207 \cs_new:Npn \__cdcmd_any_case_false:nTF #1#2#3
208 {
209   \__cdcmd_case:Nw \cdcmd_any_if:nFT #1 { * } { }
210   \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
211 }
212 \cs_new:Npn \__cdcmd_all_case_false:nTF #1#2#3
213 {
214   \__cdcmd_case:Nw \cdcmd_all_if:nFT #1 { * } { }
215   \s__cdcmd_mark {#2} \s__cdcmd_mark {#3} \s__cdcmd_stop
216 }
217 \cs_new:Npn \__cdcmd_case:Nw #1#2#3
218 { #1 {#2} { \__cdcmd_case_end:nw {#3} } { \__cdcmd_case:Nw #1 } }
219 \cs_new:Npn \__cdcmd_case_end:nw #1#2#3 \s__cdcmd_mark #4#5 \s__cdcmd_stop
220 { \exp_end: #1 #4 }

```

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

```

\newcondition Conditional setting command for document.
\setcondition 221 \NewDocumentCommand \newcondition { >{ \TrimSpaces } m } { \cdcmt_new:nn {#1} }
\clearcondition 222 \NewDocumentCommand \setcondition { t+ }
223 { \IfBooleanTF {#1} { \cdcmt_set:n } { \cdcmt_clear_set:n } }
224 \NewDocumentCommand \clearcondition { !O{*} }
225 {
226   \clist_map_inline:nn {#1}
227   {
228     \str_if_eq:eeTF {##1} { \c_cdcmt_all_str }
229     {
230       \clist_map_break:n
231       { \exp_after:wN \clearcondition \exp_after:wN [ \g_cdcmt_clist ] }
232     }
233     {
234       \cdcmt_if_exist:nTF {##1}
235       { \clist_clear:c { l__cdcmt_curr_condition@ ##1 _clist } }
236       { \msg_warning:nnn { cdcmt } { condition-not-exist } {##1} }
237     }
238   }
239 }

```

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

```

240 \NewExpandableDocumentCommand \econditionif { s O{*} +m +m }
241 {
242   \IfBooleanTF {#1}
243   { \cdcmt_all_if:nTF }
244   { \cdcmt_any_if:nTF }
245   {#2} {#3} {#4}
246 }
247 \NewExpandableDocumentCommand \econditioncmd { s O{*} +m }
248 {
249   \IfBooleanTF {#1}
250   { \cdcmt_all_if:nTF }
251   { \cdcmt_any_if:nTF }
252   {#2} {#3} { }
253 }
254 \NewExpandableDocumentCommand \econditioncase { s +m }
255 {
256   \IfBooleanTF {#1}
257   { \cdcmt_all_case:n {#2} }
258   { \cdcmt_any_case:n {#2} }
259 }
260 \NewExpandableDocumentCommand \econditioncaseTF { s +m }
261 {
262   \IfBooleanTF {#1}
263   { \cdcmt_all_case:nTF {#2} }
264   { \cdcmt_any_case:nTF {#2} }
265 }
266 \NewDocumentCommand \conditionif { s O{*} +m +m }
267 {

```

```

268 \IfBooleanTF {#1}
269 { \cdcmd_all_if:nTF }
270 { \cdcmd_any_if:nTF }
271 {#2} {#3} {#4}
272 }
273 \NewDocumentCommand \conditioncmd { s O{*} +m }
274 {
275 \IfBooleanTF {#1}
276 { \cdcmd_all_if:nTF }
277 { \cdcmd_any_if:nTF }
278 {#2} {#3} { }
279 }
280 \NewDocumentCommand \conditioncase { s t! +m }
281 {
282 \IfBooleanTF {#2}
283 {
284 \IfBooleanTF {#1}
285 { \cdcmd_all_case_false:n {#3} }
286 { \cdcmd_any_case_false:n {#3} }
287 }
288 {
289 \IfBooleanTF {#1}
290 { \cdcmd_all_case_true:n {#3} }
291 { \cdcmd_any_case_true:n {#3} }
292 }
293 }
294 \NewDocumentCommand \conditioncaseTF { s t! +m }
295 {
296 \IfBooleanTF {#2}
297 {
298 \IfBooleanTF {#1}
299 { \cdcmd_all_case_false:nTF {#3} }
300 { \cdcmd_any_case_false:nTF {#3} }
301 }
302 {
303 \IfBooleanTF {#1}
304 { \cdcmd_all_case_true:nTF {#3} }
305 { \cdcmd_any_case_true:nTF {#3} }
306 }
307 }

```

Define new xparse like conditional command.

```

308 \str_const:Nn \c_cdcmd_pair_u_str { cdcmd@u@ }
309 \str_const:Nn \c_cdcmd_pair_n_str { cdcmd@n@ }
310 \cs_new_nopar:Npn \__cdcmd_cs_pair_u:N #1
311 { \c_cdcmd_pair_u_str \cs_to_str:N #1 }
312 \cs_new_nopar:Npn \__cdcmd_cs_pair_n:N #1
313 { \c_cdcmd_pair_n_str \cs_to_str:N #1 }
314 \cs_new:Npn \__cdcmd_arg_spec_from_num:nn #1#2
315 {
316 \if_case:w 0#1 \exp_stop_f:
317 \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
318 \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:
319 }

```

```

320 \cs_new_nopar:Npn \__cdcmd_cs_pair_u:Nn #1#2
321 {
322   \c_cdcmd_pair_u_str
323   \cs_to_str:N #1 :
324   \__cdcmd_arg_spec_from_num:nn {#2} { n }
325 }
326 \cs_new_nopar:Npn \__cdcmd_cs_pair_n:Nn #1#2
327 {
328   \c_cdcmd_pair_n_str
329   \cs_to_str:N #1 :
330   \__cdcmd_arg_spec_from_num:nn {#2} { n }
331 }
332 % do not check cs_if_free, let xparse do it
333 \cs_new:Npn \__cdcmd_new_cdcmd_command:NN #1#2
334 {
335   \cs_new_protected:Npn #1 ##1##2##3
336   {
337     #2 ##1 { t+ m }
338     {
339       \IfBooleanTF {###1}
340       { \cdcmd_all_if:nTF }
341       { \cdcmd_any_if:nTF }
342       {###2}
343       { \use:c { \__cdcmd_cs_pair_u:N ##1 } }
344       { \use:c { \__cdcmd_cs_pair_n:N ##1 } }
345     }
346     \exp_args:Nc #2
347     { \__cdcmd_cs_pair_u:N ##1 } {##2} {##3}
348     \exp_args:Nc #2
349     { \__cdcmd_cs_pair_n:N ##1 } {##2} { }
350   }
351 }
352 \seq_const_from_clist:Nn \c__cdcmd_Command_seq
353 {
354   \NewDocumentCommand ,
355   \RenewDocumentCommand ,
356   \ProvideDocumentCommand ,
357   \DeclareDocumentCommand ,
358   \NewExpandableDocumentCommand ,
359   \RenewExpandableDocumentCommand ,
360   \ProvideExpandableDocumentCommand ,
361   \DeclareExpandableDocumentCommand ,
362 }
363 \seq_const_from_clist:Nn \c__cdcmd_COMMAND_seq
364 {
365   \NewConditionCommand ,
366   \RenewConditionCommand ,
367   \ProvideConditionCommand ,
368   \DeclareConditionCommand ,
369   \NewExpandableConditionCommand ,
370   \RenewExpandableConditionCommand ,
371   \ProvideExpandableConditionCommand ,
372   \DeclareExpandableConditionCommand ,
373 }

```

```

374 \seq_mapthread_function:NNN
375   \c__cdcmd_COMMAND_seq
376   \c__cdcmd_Command_seq
377   \__cdcmd_new_cdcmd_command:NN

```

(End definition for .)

Define L^AT_EX like command.

```

378 % do not check cs_if_free, let xparse do it
379 \cs_new:Npn \__cdcmd_new_cdcmd_cmd_no:nnn #1#2#3
380 {
381   \cs_new_protected:Npn #1 ##1##2##3
382   {
383     #3 ##1 { t+ m }
384     {
385       \IfBooleanTF {###1}
386       { \cdcmd_all_if:nTF }
387       { \cdcmd_any_if:nTF }
388       { ###2 }
389       { \use:c { \__cdcmd_cs_pair_u:Nn ##1 {##2} } }
390       { \use:c { \__cdcmd_cs_pair_n:Nn ##1 {##2} } }
391     }
392     #2 { \__cdcmd_cs_pair_u:Nn ##1 {##2} } {##3}
393     #2 { \__cdcmd_cs_pair_n:Nn ##1 {##2} } { }
394   }
395 }
396 \cs_generate_variant:Nn \__cdcmd_new_cdcmd_cmd_no:nnn { xxx }
397 \seq_const_from_clist:Nn \c__cdcmd_cmd_no_seq
398 {
399   \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
400   \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
401   \cs_set_protected:cn , \cs_set_protected_nopar:cn ,
402   \cs_set:cn , \cs_set_nopar:cn ,
403   \cs_set:cn , \cs_set_nopar:cn ,
404   \cs_set:cn , \cs_set_nopar:cn ,
405 }
406 \seq_const_from_clist:Nn \c__cdcmd_Cmd_no_seq
407 {
408   \NewDocumentCommand , \NewDocumentCommand ,
409   \RenewDocumentCommand , \RenewDocumentCommand ,
410   \DeclareDocumentCommand , \DeclareDocumentCommand ,
411   \NewExpandableDocumentCommand , \NewExpandableDocumentCommand ,
412   \RenewExpandableDocumentCommand , \RenewExpandableDocumentCommand ,
413   \DeclareExpandableDocumentCommand , \DeclareExpandableDocumentCommand ,
414 }
415 \seq_const_from_clist:Nn \c__cdcmd_CMD_no_seq
416 {
417   \__cdcmd_new_cdcmd_p_l_num:Nnn , \__cdcmd_new_cdcmd_p_nl_num:Nnn ,
418   \__cdcmd_renew_cdcmd_p_l_num:Nnn , \__cdcmd_renew_cdcmd_p_nl_num:Nnn ,
419   \__cdcmd_declare_cdcmd_p_l_num:Nnn , \__cdcmd_declare_cdcmd_p_nl_num:Nnn ,
420   \__cdcmd_new_cdcmd_np_l_num:Nnn , \__cdcmd_new_cdcmd_np_nl_num:Nnn ,
421   \__cdcmd_renew_cdcmd_np_l_num:Nnn , \__cdcmd_renew_cdcmd_np_nl_num:Nnn ,
422   \__cdcmd_declare_cdcmd_np_l_num:Nnn , \__cdcmd_declare_cdcmd_np_nl_num:Nnn ,
423 }

```

```

424 \int_step_inline:nn { 6 }
425 {
426   \__cdcmd_new_cdcmd_cmd_no:xxx
427   { \seq_item:Nn \c__cdcmd_CMD_no_seq {#1} }
428   { \seq_item:Nn \c__cdcmd_cmd_no_seq {#1} }
429   { \seq_item:Nn \c__cdcmd_Cmd_no_seq {#1} }
430 }
431 \tl_new:N \l__cdcmd_arg_spec_tl
432 \cs_new:Npn \__cdcmd_generate_arg_spec:nnn #1#2#3
433 {
434   \tl_set:Nn \l__cdcmd_arg_spec_tl { 0{#2} }
435   \if_int_compare:w #1 > 1 \exp_stop_f:
436     \int_step_inline:nn {#1-1} { \tl_put_right:Nn \l__cdcmd_arg_spec_tl {#3} }
437   \fi:
438 }
439 \cs_new:Npn \__cdcmd_new_cdcmd_cmd_o_aux:nn #1#2
440 {
441   \cs_new_protected:Npn #1 ##1##2##3##4##5
442   {
443     #2 ##1 { t+ m }
444     {
445       \IfBooleanTF{##1}
446       { \cdcmd_all_if:nTF }
447       { \cdcmd_any_if:nTF }
448       {##2}
449       { \use:c { \__cdcmd_cs_pair_u:N ##1 } }
450       { \use:c { \__cdcmd_cs_pair_n:N ##1 } }
451     }
452     \__cdcmd_generate_arg_spec:nnn {##2} {##3} {##5}
453     \exp_args:NcV #2 { \__cdcmd_cs_pair_u:N ##1 } \l__cdcmd_arg_spec_tl {##4}
454     \exp_args:NcV #2 { \__cdcmd_cs_pair_n:N ##1 } \l__cdcmd_arg_spec_tl { }
455   }
456 }
457 \seq_const_from_clist:Nn \c__cdcmd_CMD_o_seq
458 { \NewDocumentCommand , \RenewDocumentCommand , \DeclareDocumentCommand }
459 \seq_const_from_clist:Nn \c__cdcmd_cmd_o_seq
460 {
461   \__cdcmd_new_cdcmd_o_num:Nnnnn ,
462   \__cdcmd_renew_cdcmd_o_num:Nnnnn ,
463   \__cdcmd_declare_cdcmd_o_num:Nnnnn ,
464 }
465 \seq_mapthread_function:NNN
466 \c__cdcmd_cmd_o_seq
467 \c__cdcmd_CMD_o_seq
468 \__cdcmd_new_cdcmd_cmd_o_aux:nn
469 \cs_new_protected:Npn \__cdcmd_new_cdcmd_cmd_ne_aux:n #1
470 {
471   \exp_args:Nc \NewDocumentCommand { #1 conditioncommand } { s m 0{0} o +m }
472   {
473     \IfBooleanTF{##1}
474     {
475       \IfNoValueTF{##4}
476       { \use:c { __cdcmd_ #1 _cdcmd_p_n1_num:Nnn } ##2 {##3} {##5} }
477       { \use:c { __cdcmd_ #1 _cdcmd_o_num:Nnnnn } ##2 {##3} {##4} {##5} { m } }

```

```

478     }
479     {
480         \IfNoValueTF{##4}
481         { \use:c { __cdcmd_ #1 _cdcmd_p_l_num:Nnn } ##2 {##3} {##5} }
482         { \use:c { __cdcmd_ #1 _cdcmd_o_num:Nnnnn } ##2 {##3} {##4} {##5} { +m } }
483     }
484 }
485 }
486 \clist_map_function:nN { new, renew, declare } \__cdcmd_new_cdcmd_cmd_ne_aux:n
487 \NewDocumentCommand \provideconditioncommand { s m O{0} o +m }
488 {
489     \cs_if_free:NT #2
490     {
491         \IfBooleanTF{#1}
492         {
493             \IfNoValueTF{#4}
494             { \newconditioncommand * #2 [#3] {#5} }
495             { \newconditioncommand * #2 [#3] [#4] {#5} }
496         }
497         {
498             \IfNoValueTF{#4}
499             { \newconditioncommand #2 [#3] {#5} }
500             { \newconditioncommand #2 [#3] [#4] {#5} }
501         }
502     }
503 }
504
505 \int_step_inline:nnnn { 7 } { 1 } { 12 }
506 {
507     \__cdcmd_new_cdcmd_cmd_no:xxx
508     { \seq_item:Nn \c__cdcmd_CMD_no_seq {#1} }
509     { \seq_item:Nn \c__cdcmd_cmd_no_seq {#1} }
510     { \seq_item:Nn \c__cdcmd_Cmd_no_seq {#1} }
511 }
512
513 \cs_new_protected:Npn \__cdcmd_new_cdcmd_cmd_e_no_aux:n #1
514 {
515     \exp_args:Nc \NewDocumentCommand { #1 econditioncommand } { s m O{0} +m }
516     {
517         \IfBooleanTF{##1}
518         { \use:c { __cdcmd_ #1 _cdcmd_np_nl_num:Nnn } ##2 {##3} {##4} }
519         { \use:c { __cdcmd_ #1 _cdcmd_np_l_num:Nnn } ##2 {##3} {##4} }
520     }
521 }
522 \clist_map_function:nN { new, renew, declare } \__cdcmd_new_cdcmd_cmd_e_no_aux:n
523 \NewDocumentCommand \provideeconditioncommand { s m O{0} +m }
524 {
525     \cs_if_free:NT #2
526     {
527         \IfBooleanTF{#1}
528         { \neweconditioncommand * #2 [#3] {#4} }
529         { \neweconditioncommand #2 [#2] {#4} }
530     }
531 }

```


(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.

B

bool commands:

`\bool_case_false:n` 10
`\bool_case_true:n` 10
`\bool_if:N` 36
`\bool_if:N` 77, 122
`\bool_if_p:n` 86, 131
`\bool_lazy_and_p:nn` 151
`\bool_new:N` 5
`\bool_set_false:N` 73
`\bool_set_true:N` 71
`\c_false_bool` 81, 90, 145, 158
`\c_true_bool` 100, 101, 114, 117, 126, 135

C

cdcmd commands:

`\cdcmd_all_case:n` 257
`\cdcmd_all_case:nTF` 263
`\cdcmd_all_case_false:n` 285
`\cdcmd_all_case_false:nTF` 6, 187, 299
`\cdcmd_all_case_true:n` 290
`\cdcmd_all_case_true:nTF` . 6, 187, 304
`\cdcmd_all_if:nTF` 140, 214
`\cdcmd_all_if:nTF` 6, 120, 138, 139,
204, 243, 250, 269, 276, 340, 386, 446
`\cdcmd_all_if_p:n` 6, 120
`\c_cdcmd_all_str`
..... 3, 40, 99, 113, 144, 162, 228
`\cdcmd_any_case:n` 258
`\cdcmd_any_case:nTF` 264
`\cdcmd_any_case_false:n` 286
`\cdcmd_any_case_false:nTF` 6, 187, 300
`\cdcmd_any_case_true:n` 291
`\cdcmd_any_case_true:nTF` . 6, 187, 305
`\cdcmd_any_if:nTF` 95, 209
`\cdcmd_any_if:nTF` ... 6, 75, 93, 94,
199, 244, 251, 270, 277, 341, 387, 447
`\cdcmd_any_if_p:n` 6, 75
`\cdcmd_cd_id_if_exist:nnTF` .. 17, 51
`\cdcmd_clear_set:n` 22, 223
`\cdcmd_if_exist:nTF` 12,
24, 34, 49, 59, 101, 105, 145, 149, 234
`\cdcmd_if_exist_p:n` 12

`\cdcmd_new:nn` 22, 221
`\c_cdcmd_pair_n_str` ... 309, 313, 328
`\c_cdcmd_pair_u_str` ... 308, 311, 322
`\cdcmd_set:n` 22, 223
`\cdcmd_set:nn` 22
`\cdcmd_set_cdcmd_all:n` 22
`\cdcmd_set_cdcmd_single:nn` 22
cdcmd internal commands:
`__cdcmd_all_case_false:nTF`
..... 195, 196, 212
`__cdcmd_all_case_true:nTF`
..... 191, 192, 202
`__cdcmd_all_i:n` 125, 134, 142
`__cdcmd_all_ii:nn` 125, 134, 147
`__cdcmd_all_ii_aux:nn` 155, 160
`__cdcmd_any_case_false:nTF`
..... 193, 194, 207
`__cdcmd_any_case_true:nTF`
..... 189, 190, 197
`__cdcmd_any_i:n` 80, 89, 97
`__cdcmd_any_ii:nn` 80, 89, 103
`__cdcmd_any_ii_aux:nn` 108, 111
`__cdcmd_arg_spec_from_num:nn` ...
..... 314, 324, 330
`\l_cdcmd_arg_spec_tl`
..... 431, 434, 436, 453, 454
`__cdcmd_case:Nw`
..... 199, 204, 209, 214, 217, 218
`__cdcmd_case_end:nw` 218, 219
`\l_cdcmd_clear_set_bool` 5, 36, 71, 73
`\g_cdcmd_clist` 4, 27, 231
`__cdcmd_clist_if_in:NnTF`
..... 116, 164, 168
`__cdcmd_clist_if_in:nnTF` 168
`__cdcmd_clist_if_in_p:Nn` 168
`__cdcmd_clist_if_in_p:nn` 168
`\c_cdcmd_CMD_no_seq` .. 415, 427, 508
`\c_cdcmd_Cmd_no_seq` .. 406, 429, 510
`\c_cdcmd_cmd_no_seq` .. 397, 428, 509
`\c_cdcmd_CMD_o_seq` 457, 467
`\c_cdcmd_cmd_o_seq` 459, 466
`\c_cdcmd_COMMAND_seq` 363, 375
`\c_cdcmd_Command_seq` 352, 376

_cdcmd_cs_pair_n:N		\clist_map_inline:nn	38, 226
.....	312, 344, 349, 450, 454		\clist_map_tokens:Nn	171
_cdcmd_cs_pair_n:Nn	. 326, 390, 393		\clist_map_tokens:nn	.. 107, 155, 179	
_cdcmd_cs_pair_u:N		\clist_new:N	4, 29
.....	310, 343, 347, 449, 453		\clist_put_right:Nn	52
_cdcmd_cs_pair_u:Nn	. 320, 389, 392		\clist_set_eq:NN	61
_cdcmd_declare_cdcmd_np_l-			\conditioncase	2, 280
num:Nnn 422		\conditioncaseTF	2, 294
_cdcmd_declare_cdcmd_np_nl-			\conditioncmd	1, 273
num:Nnn 422		\conditionif	1, 3, 266
_cdcmd_declare_cdcmd_o-			cs commands:		
num:Nnnnn 463		\cs_generate_variant:Nn	396
_cdcmd_declare_cdcmd_p_l-			\cs_if_free:NTF	489, 525
num:Nnn 419		\cs_new:Npn	75, 84, 93, 94, 95, 97, 103,	
_cdcmd_declare_cdcmd_p_nl-				111, 120, 129, 138, 139, 140, 142,	
num:Nnn 419			147, 160, 183, 189, 190, 191, 192,	
_cdcmd_generate_arg_spec:nnn	..			193, 194, 195, 196, 197, 202, 207,	
.....	432, 452			212, 217, 219, 314, 333, 379, 432, 439	
_cdcmd_if_eq_break:nn	171, 179, 183		\cs_new_nopar:Npn	22,
_cdcmd_new_cdcmd_cmd_e_no-				32, 47, 57, 67, 69, 310, 312, 320, 326	
aux:n 513, 522		\cs_new_protected:Npn	
_cdcmd_new_cdcmd_cmd_ne_aux:n	.			335, 381, 441, 469, 513	
.....	469, 486		\cs_set:Nn	402, 403, 404
_cdcmd_new_cdcmd_cmd_no:nnn	..		\cs_set_nopar:Nn	402, 403, 404
.....	379, 396, 426, 507		\cs_set_protected:Nn	.. 399, 400, 401	
_cdcmd_new_cdcmd_cmd_o_aux:nn	.		\cs_set_protected_nopar:Nn	
.....	439, 468			399, 400, 401	
_cdcmd_new_cdcmd_command:NN	..		\cs_to_str:N	311, 313, 323, 329
.....	333, 377				
_cdcmd_new_cdcmd_np_l_num:Nnn	420		D		
_cdcmd_new_cdcmd_np_nl_num:Nnn	420		\DeclareConditionCommand	3, 368
_cdcmd_new_cdcmd_o_num:Nnnnn	. 461		\declareconditioncommand	3
_cdcmd_new_cdcmd_p_l_num:Nnn	. 417		\DeclareDocumentCommand	.. 357, 410, 458	
_cdcmd_new_cdcmd_p_nl_num:Nnn	417		\declareeconditioncommand	3
_cdcmd_renew_cdcmd_np_l-			\DeclareExpandableConditionCommand	.	
num:Nnn 421			3, 372
_cdcmd_renew_cdcmd_np_nl-			\DeclareExpandableDocumentCommand	..	
num:Nnn 421			361, 413
_cdcmd_renew_cdcmd_o_num:Nnnnn	462		E		
_cdcmd_renew_cdcmd_p_l_num:Nnn	418		\econditioncase	2, 254
_cdcmd_renew_cdcmd_p_nl-			\econditioncaseTF	2, 260
num:Nnn 418		\econditioncmd	1, 247
\clearcondition 1, 221		\econditionif	1, 240
clist commands:			else commands:		
\clist_clear:N 37, 235		\else:	318
\clist_const:Nn 28		exp commands:		
\clist_count:n 152		\exp:w	
\clist_gput_right:Nn 27			189, 190, 191, 192, 193, 194, 195, 196	
\clist_if_exist:NTF 14		\exp_after:wN	231
\clist_if_in:NnTF 19		\exp_args:Nc	346, 348, 471, 515
\clist_map... 2		\exp_args:NcV	453, 454
\clist_map_break:n		\exp_end:	220
.....	41, 114, 117, 165, 185, 230		\exp_stop_f:	316, 435
\clist map function:nN 486, 522				

F		<code>\ProvideConditionCommand</code> 3, 367
fi commands:		<code>\provideconditioncommand</code> 3, 487
<code>\fi:</code>	318, 437	<code>\ProvideDocumentCommand</code> 356
I		<code>\provideeconditioncommand</code> 3, 523
if commands:		<code>\ProvideExpandableConditionCommand</code> 3, 371
<code>\if_case:w</code>	316	<code>\ProvideExpandableDocumentCommand</code> . 360
<code>\if_int_compare:w</code>	435	
<code>\IfBooleanTF</code>	223, 242, 249, 256, 262, 268, 275, 282, 284, 289, 296, 298, 303, 339, 385, 445, 473, 491, 517, 527	
<code>\IfNoValueTF</code>	475, 480, 493, 498	
int commands:		
<code>\int_compare:nTF</code>	170, 178	
<code>\int_compare_p:n</code>	152, 154	
<code>\int_step_inline:nn</code>	424, 436	
<code>\int_step_inline:nnnn</code>	505	
K		
keyval commands:		
<code>\keyval_parse:NNn</code>	68, 72, 79, 88, 124, 133	
M		
msg commands:		
<code>\msg_error:nnn</code>	25	
<code>\msg_new:nnn</code>	6, 8, 10	
<code>\msg_warning:nnn</code>	45, 55, 65, 236	
<code>\msg_warning:nnnn</code>	53	
N		
<code>\newcondition</code>	1, 221	
<code>\NewConditionCommand</code>	3, 365	
<code>\newconditioncommand</code>	3, 494, 495, 499, 500	
<code>\NewDocumentCommand</code>	3, 221, 222, 224, 266, 273, 280, 294, 354, 408, 458, 471, 487, 515, 523	
<code>\neweconditioncommand</code>	3, 528, 529	
<code>\NewExpandableConditionCommand</code>	3, 369	
<code>\NewExpandableDocumentCommand</code>	240, 247, 254, 260, 358, 411	
O		
or commands:		
<code>\or:</code>	317, 318	
P		
prg commands:		
<code>\prg_generate_conditional_-variant:Nnn</code>	96, 141, 174, 182	
<code>\prg_new_conditional:Npnn</code>	12, 17, 168, 176	
<code>\prg_return_false:</code>	15, 20, 172, 180	
<code>\prg_return_true:</code>	15, 20, 172, 180	
R		
<code>\RenewConditionCommand</code>	3, 366	
<code>\renewconditioncommand</code>	3	
<code>\RenewDocumentCommand</code>	355, 409, 458	
<code>\reneweconditioncommand</code>	3	
<code>\RenewExpandableConditionCommand</code>	3, 370	
<code>\RenewExpandableDocumentCommand</code>	359, 412	
S		
scan commands:		
<code>\scan_new:N</code>	187, 188	
scan internal commands:		
<code>\s__cdcmd_mark</code>	187, 200, 205, 210, 215, 219	
<code>\s__cdcmd_stop</code>	188, 200, 205, 210, 215, 219	
seq commands:		
<code>\seq_const_from_clist:Nn</code>	352, 363, 397, 406, 415, 457, 459	
<code>\seq_item:Nn</code>	427, 428, 429, 508, 509, 510	
<code>\seq_mapthread_function:NNN</code>	374, 465	
<code>\setcondition</code>	1, 221	
str commands:		
<code>\str_const:Nn</code>	3, 308, 309	
<code>\str_if_eq:nnTF</code>	40, 99, 113, 144, 162, 185, 228	
T		
TeX and L ^A T _E X 2 _ε commands:		
<code>\long</code>	3	
<code>\newcommand</code>	3	
<code>\par</code>	3	
<code>\protected</code>	1	
<code>\renewcommand</code>	3	
tex commands:		
<code>\tex_the:D</code>	114, 117	
tl commands:		
<code>\tl_new:N</code>	431	
<code>\tl_put_right:Nn</code>	436	
<code>\tl_set:Nn</code>	434	
<code>\TrimSpaces</code>	221	
U		
use commands:		
<code>\use:N</code>	343, 344, 389, 390, 449, 450, 476, 477, 481, 482, 518, 519	