The cleveref-usedon package [*]

Sven Pistre [†]

Released 2023-04-21

Abstract

This package adds “forward-referencing” to the cleveref package. Any label can be referenced with the new optional argument `UsedOn` passed to `\cref`. Doing so, will print an info message at the original label location (in a theorem environment, say) which reads “*Used on pages ⟨list of pages⟩.*”. This functionality is complementary to hyperref’s `pagebackref` or biblatex’s `backref` option for the bibliography. It might be useful for authors of longer texts such as textbooks or theses, where a lot of supplementary results and information are given in early chapters, appendices or exercises. The message on which pages these results will be used can be a helpful information for the reader of the final text. Additionally, a bug in cleveref v0.21.4 is patched.

Contents

1 Introduction

Imagine you are reading a long mathematical text such as a text book or a thesis. There are plenty of supplementary lemmas, propositions, theorems and/or exercises throughout the whole text. You ask yourself “Gosh, while Lemma 1.12 is certainly an interesting result *where* is this result used later on in this long text? I really would find that helpful to decide *why* I should read the proof.” You can, of course, use the PDF search function of your viewer to look up the string “Lemma 1.12” but wouldn’t it be more helpful if Lemma 1.12 already indicates all or at least its most useful/crucial applications via an info message?

This is what the package cleveref-usedon tries to address. The info message “*Used on p. 40, 43-45 and 101.*” would then be printed to the header of Lemma 1.12. For example, we have given the following theorem the label

```
\label{thm:SqrtTwoIrrational}.
```

Theorem 1.1. *The number* $\sqrt{2}$ *is irrational.*

Now we can reference this theorem via

```
\cref[UsedOn]{thm:SqrtTwoIrrational}:
```

[*]This document corresponds to cleveref-usedon v0.4.0, last revised 2023-04-21.

[†]E-mail: cleveref-usedon@sven-pistre.com

A proof of **??** can be traced back to Euclid.

We will now reference this theorem without the optional argument `[UsedOn]`. So let's clear the page of this PDF, so that we can see the effects of calling

```
    \cref{thm:SqrtTwoIrrational}
```

more clearly.

Note that the current page number 3 is not included in the list of page references in the header of ??.

2 Usage

The cleveref-usedon package uses cleveref v0.21.4 as its base. To freely cite from the cleveref documentation:
The cleveref-usedon package is loaded in the usual way, by putting the line

```
\usepackage{cleveref-usedon}
```

in your document's preamble. However, care must be taken when using cleveref in conjunction with other packages that modify LaTeX's referencing system (see Section 13 of cleveref's documentation). Basically, cleveref-usedon must be loaded *last* but definitely AFTER hyperref.

\cref `\cref[⟨Option⟩]{⟨LabelName⟩}`
\Cref `\Cref[⟨Option⟩]{⟨LabelName⟩}`

The `\cref` macro can be called with options `UsedOn` (see ??), `UsedBy` (experimental, see ??) and `UsedByAndOn` (experimental, see ??) or their short forms `uo`, `ub`, `ubao`. This is case-insensitive, i.e. you could also write[1]

`\cref[UsEdOn]{`⟨*LabelName*⟩`}`,
`\cref[uO]{`⟨*LabelName*⟩`}`.

The package cleveref-usedon is implemented using the LaTeX3 programming layer `expl3`. If you are interested, I have spent some time to document and comment on the implementation in ??. On an abstract level the implementation is as follows: Whenever the label ⟨*LabelName*⟩ gets referenced with one of the options at some location via `\cref[`⟨*Option*⟩`]{`⟨*LabelName*⟩`}`, an additional auxiliary label is created at this very location. This auxiliary label has the form ⟨*Option*⟩@⟨*LabelName*⟩@⟨*Counter*⟩ where ⟨*Counter*⟩ is an integer that counts how often the label ⟨*LabelName*⟩ has been referenced with ⟨*Option*⟩. At the end of the LaTeX run, the final value of this counter is written to the .aux file as a key-value pair:

⟨*Option*⟩@⟨*LabelName*⟩ = ⟨*MaxCounter*⟩

In the second LaTeX run, we read this counter from the .aux file. Then, at the original location of the referenced label ⟨*LabelName*⟩, we can now pass the list of auxiliary labels

⟨*Option*⟩@⟨*LabelName*⟩@1, ..., ⟨*Option*⟩@⟨*LabelName*⟩@⟨*MaxCounter*⟩

to `\cpageref` (and `\cref` for the experimental options) and write the forward-referencing info message.

[1]But why would you want to?

2.1 The option `[`$\langle$*UsedOn*$\rangle$`]`

`UsedOn` This option adds the message

> *(Used on page(s) $\langle$list of page(s)$\rangle$.)*

The text is followed by a line break and is set after the original location of the referenced label $\langle$*LabelName*$\rangle$. If hyperref has been loaded, there will also be hyperlinks to the corresponding pages from where the label has been referenced.

If the original label has been set in a theorem-like environment such as

```
\begin{theorem}   \label{thm:SqrtTwoIrrational}
    The number $\sqrt{2}$ is irrational.
\end{theorem}
```

then the info message is printed in the header of this theorem-like environment. The same functionality can be used for `\Cref`.

2.2 The experimental options `[`$\langle$*UsedBy*$\rangle$`]` and `[`$\langle$*UsedByAndOn*$\rangle$`]`

`UsedBy`
`UsedByAndOn` The option `[`$\langle$*UsedBy*$\rangle$`]` adds the message

> *(Used by $\langle$list of theorem-like destination(s)$\rangle$.)*

The option `[`$\langle$*UsedByAndOn*$\rangle$`]` adds the message

> *(Used by $\langle$list of theorem-like destination(s)$\rangle$ on page(s) $\langle$list of page(s)$\rangle$.)*

Each text is followed by a line break and is set after the original location of the referenced label $\langle$*LabelName*$\rangle$. If hyperref has been loaded, there will also be hyperlinks to the destinations.

For example, suppose we have the following lemma.

Lemma 2.1. *Any smooth function $f\colon \mathbb{R} \to \mathbb{R}$ is continuous.*

And we will use it in the proof of the following result.

Corollary 2.2. *Suppose $f\colon \mathbb{R} \to \mathbb{R}$ is smooth. The derivative $f'\colon \mathbb{R} \to \mathbb{R}$ is continuous.*

Proof. The derivative of a smooth map is itself smooth. Hence, the claim follows by **??**. □

The previous result will in turn be used in the proof of the next one.

Corollary 2.3. *Suppose $f\colon \mathbb{R} \to \mathbb{R}$ is smooth and $k \in \mathbb{N}$. The kth derivative $f^{(k)}\colon \mathbb{R} \to \mathbb{R}$ is continuous.*

Proof. This follows from **??** by induction. □

The code for the above examples is as follows:

```
\begin{lemma}        \label{lemma:SmoothFunction}
    Any smooth function $f\colon \mathbb{R}\to \mathbb{R}$
    is continuous.
\end{lemma}

\begin{corollary}   \label{cor:DerivativeContinuous}
    Suppose $f\colon \mathbb{R}\to \mathbb{R}$ is smooth.
    The derivative $f^{\prime}\colon \mathbb{R}\to \mathbb{R}$
    is continuous.
\begin{proof}
    The derivative of a smooth map is itself smooth.
    Hence, the claim follows by \cref[UsedBy]{lemma:SmoothFunction}.
\end{proof}
\end{corollary}

\begin{corollary}   \label{cor:AllDerivativesContinuous}
    Suppose $f\colon \mathbb{R}\to \mathbb{R}$ is smooth and
    $k\in\mathbb{N}$. The $k$th derivative
    $f^{(k)}\colon \mathbb{R}\to \mathbb{R}$
    is continuous.
\begin{proof}
    This follows from
    \cref[UsedByAndOn]{cor:DerivativeContinuous} by induction.
\end{proof}
\end{corollary}
```

Unfortunately, due to how this package is currently implemented, to get these experimental options to work it is necessary to abuse the usage of proof environments. Namely, one needs to nest the proof environment *inside* the theorem-like environment. Note carefully how the proof environments are (ab)used in the above code example.

This is – as far is I know – not how these environments are supposed to be used. In particular, placing text between theorem-like environment and the corresponding proof, as is often common, will result in a wrong reference. Namely, instead of referencing the theorem-like environment by name only the corresponding section name would be printed, e.g. "*Used by* **??**.". You can see this for yourself, if you move the proof environment out of the theorem-like environment in the above examples. Hence, using proof environments correctly results in messages which are less helpful to the reader. On the other hand, using this experimental functionality to help the reader forces users (i.e. authors) of this package to use proof environments incorrectly. This sounds like a No-Free-Lunch theorem... Therefore, use these two experimental options at your own discretion!

3 Hints and tips

If you use the `capitalise` option for cleveref, you might want to revert this capitalisation for page references for more visual appeal by putting

```
\crefname{page}{page}{pages}
```

in your document's preamble, after loading cleveref-usedon.

It is recommended to not use the optional arguments for equation-like environments such as **??** because sometimes[2] the info message will — unhelpfully — be printed inside the equation environment, like so (this might or might not[3] show undesired behaviour):

$$\int_M \mathrm{d}\omega = \int_{\partial M} \omega. \tag{1}$$

So, one should use this functionality only for theorem-like environments such as theorems, lemmas and exercises etc.

If one references the same label multiple times but with different options, say `UsedOn` and `UsedBy`, then *both* info messages are printed after the original label location. This is not how this functionality was intended and you shouldn't use it like that. I am not going to implement a check which various combinations of these options are used for the same label.

3.1 Editing the info messages

```
\UsedOnMessage        \UsedOnMessage{⟨PageList from cpageref⟩}
\UsedByMessage        \UsedByMessage{⟨EnvironmentList from cpageref⟩}
\UsedByAndOnMessage   \UsedByAndOnMessage{⟨EnvironmentList from cpageref⟩}{⟨PageList from cpageref⟩}
```

The standard messages which get printed to the first line of the labelled environment are

(Used on ⟨PageList⟩.),
(Used by ⟨EnvironmentList⟩.),
(Used by ⟨EnvironmentList⟩) on ⟨PageList⟩.),

respectively — followed by a line break — where ⟨*PageList*⟩ is generated internally by cleveref via `\cpageref` and ⟨*EnvironmentList*⟩ is generated internally by cleveref via `\cref`. You can change these behaviours by redefining the macros `\UsedOnMessage`, `\UsedByMessage` and `\UsedByAndOnMessage`, e.g. as

```
\RenewDocumentCommand{\UsedOnMessage}{ m }{
    \emph{(Will be cited on #1.)} \newline
}
\RenewDocumentCommand{\UsedByMessage}{ m }{
    \emph{(Will be applied in #1.)} \newline
}
\RenewDocumentCommand{\UsedByAndOnMessage}{ m m }{
    \emph{(Will be applied in #1 on #2.)} \newline
}
```

4 Interaction with other packages

All interactions with other packages mentioned in Section 13 of cleveref's documentation also apply to cleveref-usedon. In fact (if cleveref-usedon is loaded last), ntheorem's `\thref`

[2]I haven't quite tracked down this bug.
[3]In version 0.2.0 of this package, the text "*Used on page 2.*" was printed right after the formula in the equation environment.

and varioref's `\vref` also obtain the additional `UsedOn` functionality because cleveref redefines these macros to be aliases for `\cref`. Of course, they need to be loaded in the correct order, i.e.

```
\usepackage{ntheorem}
\usepackage{hyperref}
\usepackage{cleveref-usedon}
```

or

```
\usepackage{varioref}
\usepackage{hyperref}
\usepackage{cleveref-usedon}
```

5 Future features

For all feature requests, either create a github issue or send me an email.

Let's just reference **??** one last time for the fun of it, check **??** again to see the effect to the reference list in the header of **??**.

6 Implementation

Start the DocStrip guards.

```
⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention).

```
⟨@@=UsedOn⟩
```

If the LaTeX version is too old, then abort loading and show an error message. The macro `\IfFormatAtLeastTF` is not be available on LaTeX versions older than 2020-10-01 so we need to provide an internal workaround.

```
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\IfFormatAtLeastTF{2021-06-01}{%
    % LaTeX2e version new enough
}{%
    \PackageError{cleveref-usedon}{%
        Mismatched~LaTeX~support~files~detected.\MessageBreak
        Your~LaTeX~format~is~dated~\fmtversion,\MessageBreak
        but~the~package~cleveref-usedon\MessageBreak
        requires~at~least~2021-06-01.\MessageBreak
        Update~your~TeX~distribution.\MessageBreak
        \MessageBreak
        Loading~cleveref-usedon~will~abort!}%
        {Update~your~TeX~distribution~using~your~TeX~package~manager.}%
}
```

If the version of the LaTeX3 kernel is too old, then abort loading and show an error message. The macro `\IfExplAtLeastTF` currently does not exist at all[4], but we can provide an internal workaround.

```
\providecommand\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
\RequirePackage{expl3}[2021-05-16]
```

[4]But might in the future, see github issue #1004.

```
\IfExplAtLeastTF{2021-05-16}{%
    % expl3 version new enough
}{%
    \PackageError{cleveref-usedon}{%
        Support~package~expl3~too~old.\MessageBreak
        The~L3~programming~layer~in~the~LaTeX~format\MessageBreak
        is~dated~\ExplLoaderFileDate,\MessageBreak
        but~the~package~cleveref-usedon\MessageBreak
        requires~at~least~2021-05-16.\MessageBreak
        Update~your~TeX~distribution.\MessageBreak
        \MessageBreak
        Loading~cleveref-usedon~will~abort!}%
        {Update~your~TeX~distribution~using~your~TeX~package~manager.}%
}
```

6.1 Options and requirements

Below are the available package options.

```

\keys_define:nn { cleveref-usedon }
  {
    , Default     .tl_set:N   = \l__UsedOn_default_tl
    , Default     .initial:n  = { }
    , Default     .value_required:n = true
    , UsedOn      .meta:n     = { Default = UsedOn }
    , usedon      .meta:n     = { Default = UsedOn }
    , uo          .meta:n     = { Default = UsedOn }
    , UsedBy      .meta:n     = { Default = UsedBy }
    , usedby      .meta:n     = { Default = UsedBy }
    , ub          .meta:n     = { Default = UsedBy }
    , UsedByAndOn .meta:n     = { Default = UsedByAndOn }
    , usedbyandon .meta:n     = { Default = UsedByAndOn }
    , ubao        .meta:n     = { Default = UsedByAndOn }

```

All other package options get passed on to cleveref which is the base for the current package here.

```
    , unknown    .code:n     =
        { \PassOptionsToPackage { \CurrentOption } { cleveref } }
  }
\ProcessKeyOptions [ cleveref-usedon ]

\RequirePackage{cleveref}[2018/03/27]
```

6.2 Patches of known bugs to cleveref

The following fixes the range bug for `\cpageref` in cleveref v0.21.4. See https://tex.stackexchange.com/a/620066/267438. (We need to temporarily reset the DocStrip guards.)

```
⟨@@=⟩
\newcommand*{\@setcpagerefrange}[3]{%
    \@@setcpagerefrange{#1}{#2}{cref}{#3}}
\newcommand*{\@setCpagerefrange}[3]{%
```

```
    \@@setcpagerefrange{#1}{#2}{Cref}{#3}}
\newcommand*{\@setlabelcpagerefrange}[3]{%
    \@@setcpagerefrange{#1}{#2}{labelcref}{#3}}
⟨@@=UsedOn⟩
```

6.3 Overloading of label and cref

We need variants of `\str_case:nn` which expand the input string token. This will be used to match options for `\__UsedOn_Processor`.

```
\prg_generate_conditional_variant:Nnn \str_case:nn { x } { T, TF }
\cs_generate_variant:Nn \str_case:nn { x }
```

`\g__UsedOn_k_seq` Let's initialise a global key sequence for those label names that have been referenced via `[UsedOn]`, `[UsedBy]` or `[UsedByAndOn]`.

```
\seq_new:N \g__UsedOn_k_seq
```

(*End of definition for* `\g__UsedOn_k_seq`.)

`\g__UsedOn_kv_prop` And we'll also create a global key-value property list with label names as keys and the maximal amount of times they have been referenced via `[UsedOn]` as values (possibly known from the last pdflatex run).

```
\prop_new:N \g__UsedOn_kv_prop
```

(*End of definition for* `\g__UsedOn_kv_prop`.)

`\g__UsedOn_Options_clist` This `clist` contains all options that can be passed to `\cref` which are currently implemented.

```
\clist_new:N \g__UsedOn_Options_clist
\clist_set:Nn \g__UsedOn_Options_clist { UsedOn, UsedBy, UsedByAndOn }
```

(*End of definition for* `\g__UsedOn_Options_clist`.)

_new:Nn @_UsedOnMessage:n _new:Nn @_UsedByMessage:n _new:Nn @_-UsedByAndOnMessage:nn

`\DefineUsedOnMessage`

```
\NewDocumentCommand \DefineUsedOnMessage { m }
  {
    \cs_set:Nn \__UsedOn_UsedOnMessage:n { #1 }
  }
```

(*End of definition for* `\DefineUsedOnMessage`. *This function is documented on page* **??**.)

`\DefineUsedByMessage`

```
\NewDocumentCommand \DefineUsedByMessage { m }
  {
    \cs_set:Nn \__UsedOn_UsedByMessage:n { #1 }
  }
```

(*End of definition for* `\DefineUsedByMessage`. *This function is documented on page* **??**.)

\DefineUsedByAndOnMessage

```
\NewDocumentCommand \DefineUsedByAndOnMessage { m }
  {
    \cs_set:Nn \__UsedOn_UsedByAndOnMessage:nn { #1 }
  }
```

(*End of definition for* `\DefineUsedByAndOnMessage`*. This function is documented on page* **??***.*)

```
\RequirePackage{iflang}

\DefineUsedOnMessage
  {
    \IfLanguageName{english}
        { (Used~on~#1.) } {}
    \IfLanguageName{french}
        { (Apparaît~en~#1.) } {}
    \IfLanguageName{ngerman}
        { (Wird~auf~#1.) } {}
    \IfLanguageName{spanish}
        { (Aparece~en~#1.) } {}
  }
\DefineUsedByMessage
  {
    \IfLanguageName{english}
        { (Used~by~#1.) } {}
    \IfLanguageName{french}
        { (Apparaît~dans~#1.) } {}
    \IfLanguageName{ngerman}
        { (Wird~in~#1.) } {}
    \IfLanguageName{spanish}
        { (Aparece~en~#1.) } {}
  }
\DefineUsedByAndOnMessage
  {
    \IfLanguageName{english}
        { (Used~by~#1~on~#2.) } {}
    \IfLanguageName{french}
        { (Apparaît~dans~#1~en~#2.) } {}
    \IfLanguageName{ngerman}
        { (Wird~in~#1~auf~#2.) } {}
    \IfLanguageName{spanish}
        { (Aparece~en~#1~en~#2.) } {}
  }
```

\UsedOnMessage The following are the standard texts that get printed in the first line of the labelled environment which later gets referenced with `[UsedOn]`, `[UsedBy]` or `[UsedByAndOn]`.

```
\NewDocumentCommand \UsedOnMessage { m }
  {
    \emph{ \__UsedOn_UsedOnMessage:n { #1 } } \\[.3\baselineskip]
  }
```

(*End of definition for* `\UsedOnMessage`*. This function is documented on page* **??***.*)

\UsedByMessage

```
\NewDocumentCommand \UsedByMessage { m }
```

```
  {
    \emph{ \__UsedOn_UsedByMessage:n { #1 } } \\[.3\baselineskip]
  }
```

(*End of definition for* `\UsedByMessage`*. This function is documented on page* **??**.)

`\UsedByAndOnMessage`

```
\NewDocumentCommand \UsedByAndOnMessage { m m }
  {
    \emph{ \__UsedOn_UsedByAndOnMessage:nn { #1 } { #2 } } \\[.3\baselineskip]
  }
```

(*End of definition for* `\UsedByAndOnMessage`*. This function is documented on page* **??**.)

`\__UsedOn_Printer` Given a ⟨*LabelName*⟩, the following command records all references via the optional `cref` arguments `[UsedOn]`, `[UsedBy]` or `[UsedByAndOn]`, i.e. when the user called `\cref[`⟨*Option*⟩`]{`⟨*LabelName*⟩`}`. They are recorded in a temporary comma-separated list (a `clist` in `expl3` speak). This `clist` is then passed to cleveref's `cpageref` or `cref` which in turn is passed to `\UsedOnMessage`, `\UsedByMessage` or `\UsedByAndOnMessage` to be printed after the original label.

```
\NewDocumentCommand \__UsedOn_Printer { m m }
  {
```

First, we will check if the reference ⟨*Option*⟩`@`⟨*LabelName*⟩`@1` exists. Here, the `@1` means that ⟨*LabelName*⟩ has been referenced with option `[`⟨*Option*⟩`]` at least once. If this reference does not exist, nothing happens.

```
    \cs_if_exist:cT {r@#1@#2@1}
      {
```

Next, we store all the references of the form ⟨*Option*⟩`@`⟨*LabelName*⟩`@`⟨*Number*⟩ in a temporary comma-separated list (`clist`). We do this by looping from 1 to the value of `LastRun@`⟨*Option*⟩`@`⟨*LabelName*⟩ (if the latter value exists, otherwise we set it to 1). Initially, this will need two consecutive runs of pdflatex.

```
        \cs_if_free:cTF {c@LastRun@#1@#2}
          { \int_set:Nn \l_tmpa_int { 1 } }
          { \int_set:Nn \l_tmpa_int { \value{LastRun@#1@#2} } }
        \int_set:Nn \l_tmpb_int { 1 }
        \int_while_do:nn { \l_tmpb_int <= \l_tmpa_int }
          {
            \clist_put_right:Nx \l_tmpa_clist { #1@#2@\int_use:N \l_tmpb_int }
            \int_incr:N \l_tmpb_int
          }
```

Finally, we print the message that was set in the macro `\UsedOnMessage`.

```
        \str_case:xn { \str_foldcase:n { #1 } }
          {
            {usedon}
              { \UsedOnMessage { \cpageref{\l_tmpa_clist} } }
            {usedby}
              { \UsedByMessage { \cref{\l_tmpa_clist} } }
            {usedbyandon}
              { \UsedByAndOnMessage { \cref{\l_tmpa_clist} } { \cpageref{\l_tmpa_clist} } }
          }
      }
  }
```

(*End of definition for* `\__UsedOn_Printer`.)

`\__UsedOn_PrintMessage` This macro prints the corresponding `UsedOn` message after the original label.

```
\NewDocumentCommand \__UsedOn_PrintMessage { m }
  {
    \clist_map_inline:Nn \g__UsedOn_Options_clist
      {
        \__UsedOn_Printer{##1}{#1}
      }
  }
```

(*End of definition for* `\__UsedOn_PrintMessage`.)

`\l__UsedOn_Option_str` This variable will be used to store the used option in a `\cref` call.

```
\str_new:N \l__UsedOn_Option_str
```

(*End of definition for* `\l__UsedOn_Option_str`.)

`\__UsedOn_Processor` This macro takes an optional argument (a case-insensitive version of the options or their shortform) and a mandatory argument (a single `{`⟨*LabelName*⟩`}` or a `clist` {⟨*LabelName1*⟩,⟨*LabelName2*⟩,...}).

```
\NewDocumentCommand \__UsedOn_Processor { o m }
  {
    \IfValueT{#1}{
```

First, we check if the option `[UsedOn]` or `[uo]` (case-insensitive) was used.

```
      \str_case:xnTF { \str_foldcase:n { #1 } }
        {
            {usedon}      {\str_set:Nn \l__UsedOn_Option_str {UsedOn}}
            {uo}          {\str_set:Nn \l__UsedOn_Option_str {UsedOn}}
            {usedby}      {\str_set:Nn \l__UsedOn_Option_str {UsedBy}}
            {ub}          {\str_set:Nn \l__UsedOn_Option_str {UsedBy}}
            {usedbyandon} {\str_set:Nn \l__UsedOn_Option_str {UsedByAndOn}}
            {ubao}        {\str_set:Nn \l__UsedOn_Option_str {UsedByAndOn}}
        }
        {
            {
```

Loop through the (potential) label list in the mandatory argument of `\cref` (or `\Cref`) which gets passed as the mandatory argument of the current macro.

```
              \seq_set_from_clist:Nn \l_tmpa_seq {#2}
              \seq_map_inline:Nn \l_tmpa_seq
                {
```

If the label has *not* been referenced yet via the option `#1` where `#1` is one of the current available options in {UsedOn,UsedBy,UsedByAndOn}, create a counter for the current run `ThisRun@<Option>@##1`. If we are not in the initial run anymore, there should be a counter `LastRun@<Option>@##1` which contains the maximal amount this specific label has been referenced via `UsedOn`. If we are in the initial run, we need to create this counter as well. Then save the label in the global container `\g__UsedOn_k_seq`.

```
                  \seq_if_in:NxF \g__UsedOn_k_seq { \l__UsedOn_Option_str @##1 }
                    {
                        \newcounter{ ThisRun@ \l__UsedOn_Option_str @##1 }
                        \cs_if_free:cT {c@LastRun@ \l__UsedOn_Option_str @##1}
```

```
             {
              \newcounter{LastRun@ \l__UsedOn_Option_str @##1}
             }
            \seq_gput_right:Nx \g__UsedOn_k_seq
             { \l__UsedOn_Option_str @##1}
          }
```

Increase the counter for the current run by 1 and set the counter for last run (containing the maximal amount of `UsedOn`-`\cref`'s) to...the maximal amount of `UsedOn`-`\cref`'s.

```
         \stepcounter{ThisRun@ \l__UsedOn_Option_str @##1}
         \setcounter{LastRun@ \l__UsedOn_Option_str @##1}
          {
           \fp_eval:n
            {
             max(
              \value{ThisRun@ \l__UsedOn_Option_str @##1} ,
              \value{LastRun@ \l__UsedOn_Option_str @##1}
             )
            }
          }
```

Store the value of the max counter `LastRun@<Option>@##1` in the global container `\g__UsedOn_kv_prop`.

```
          \prop_gput:Nxx \g__UsedOn_kv_prop
           { \l__UsedOn_Option_str @##1}
           {\arabic{LastRun@ \l__UsedOn_Option_str @##1}}
```

Now we create a numbered auxiliary label. This label is issued at the location where we referenced the original label via `\cref[UsedOn]`⟨*LabelName*⟩. The new auxiliary label has the prefix `UsedOn@`, `UsedBy@` or `UsedByAndOn@` and the suffix `@\arabic{ThisRun@<Option>@##1}`, e.g. `UsedOn@thm:Pythagoras@4` if it is the fourth time that we called
`\cref[UsedOn]{thm:Pythagoras}`.

```
          \__UsedOn_origlabel
           {
            \l__UsedOn_Option_str @##1@
            \arabic{ThisRun@ \l__UsedOn_Option_str @##1}
           }
        }
      }
    }
    {
```

Throw an error, if an unrecognised option was used for the optional argument to this macro.

```
      \msg_new:nnn {cleveref-usedon} { OptionSpellingError }
       {
        \MessageBreak
        Spelling~error~\msg_line_context:
        \MessageBreak
        Did~you~mean~to~pass~option\MessageBreak
        'UsedOn'~to~cref~or~Cref?
       }
      \msg_fatal:nn { cleveref-usedon } { OptionSpellingError }
    }
    }
  }
```

(*End of definition for* `\__UsedOn_Processor`.)

`\__UsedOn_cref` This is just a wrapper around cleveref's `\cref`. Additionally the `\__UsedOn_Processor` gets called.

```
\bool_new:N \l__UsedOn_already_specified_bool
\clist_new:N \l__UsedOn_args_clist
\cs_new:Nn \__UsedOn_define_from_orig:NN
  {
    \NewDocumentCommand #1 { s o m }
      {
        \IfValueTF { ##2 }
          {
            \bool_set_false:N \l__UsedOn_already_specified_bool
            \clist_set:Nn \l__UsedOn_args_clist { ##2 }
            \clist_put_right:No \l__UsedOn_args_clist { \l__UsedOn_default_tl }
            \clist_map_inline:Nn \l__UsedOn_args_clist
              {
                \str_case:xnT { \str_foldcase:n { ####1 } }
                  {
                    {usedon} {}
                    {uo}     {}
                    {usedby} {}
                    {ub}     {}
                    {usedbyandon} {}
                    {ubao}        {}
                  }
                  {
                    \clist_remove_all:Nn \l__UsedOn_args_clist { ####1 }
                    \bool_if:NF \l__UsedOn_already_specified_bool
                      {
                        \__UsedOn_Processor[####1]{##3}
                      }
                    \bool_set_true:N \l__UsedOn_already_specified_bool
                  }
              }
            \clist_if_empty:NTF \l__UsedOn_args_clist
              {
                \IfBooleanTF{##1}{ #2*{##3} }{ #2{##3} }
              }
              {
                \IfBooleanTF{##1}{ #2*[\l__UsedOn_args_clist]{##3} }{ #2[\l__UsedOn_args_clist
              }
          }
          {
            \tl_if_blank:VF \l__UsedOn_default_tl
              {
                \clist_clear:N \l__UsedOn_args_clist
                \clist_put_right:No \l__UsedOn_args_clist { \l__UsedOn_default_tl }
                \clist_map_inline:Nn \l__UsedOn_args_clist
                  {
                    \__UsedOn_Processor[####1]{##3}
                    }
                }
            \IfBooleanTF{##1}{ #2*{##3} }{ #2{##3} }
          }
      }
  }
\__UsedOn_define_from_orig:NN \__UsedOn_cref \__UsedOn_origcref
\__UsedOn_define_from_orig:NN \__UsedOn_Cref \__UsedOn_origCref
\__UsedOn_define_from_orig:NN \__UsedOn_labelcref \__UsedOn_origlabelcref
```

(*End of definition for* `\__UsedOn_cref`.)

`\__UsedOn_ReadFromAux` From the .aux file we will read the contents of the global container `\g__UsedOn_kv_prop`. This is a key-value property list and we create and set a for each label (key) and the maximal amount (value) it was called in the last run.

```
\NewDocumentCommand \__UsedOn_ReadFromAux { }
  {
    \prop_map_inline:Nn \g__UsedOn_kv_prop
      {
          \newcounter{LastRun@##1}
          \setcounter{LastRun@##1}{##2}
      }
  }
```

(*End of definition for* `\__UsedOn_ReadFromAux`.)

`\__UsedOn_WriteToAux` For each label we write a line in the .aux file of the form:
⟨*LabelName*⟩ = ⟨*Maximal references via UsedOn in last run*⟩.
This information can be constructed from the global container `\g__UsedOn_k_seq` and the counters with prefix `ThisRun@` we set earlier. We need to wrap this in the on/off switch for `expl3` functionality.

```
\NewDocumentCommand \__UsedOn_WriteToAux { }
  {
```

First, we clear the global key-value prop list `\g__UsedOn_kv_prop` and then we rebuild it with the information from the current run.

```
    \prop_clear:N \g__UsedOn_kv_prop
    \seq_map_inline:Nn \g__UsedOn_k_seq
      { \prop_gput:Nxx \g__UsedOn_kv_prop {##1}{\arabic{ThisRun@##1}} }
    \iow_now:cx { @auxout }
      { \token_to_str:N \ExplSyntaxOn }
```

Loop through the key-val `proplist` and write contents to .aux file.

```
    \prop_map_inline:Nn \g__UsedOn_kv_prop
      {
        \tl_set:Nn \l_tmpa_tl { ##1 }
        \tl_replace_all:Nnn \l_tmpa_tl { ~ } { \c_tilde_str }
        \iow_now:cx { @auxout }
          {
            \prop_gput_from_keyval:Nn \token_to_str:N \g__UsedOn_kv_prop
              { {\l_tmpa_tl} = ##2 }
          }
      }
    \iow_now:cx { @auxout }
      { \token_to_str:N \ExplSyntaxOff }
}%
```

(*End of definition for* `\__UsedOn_WriteToAux`*.*)

At the hook `\AtBeginDocument` we read from the .aux file and patch commands.

```
\hook_gput_code:nnn { begindocument } { cleveref-usedon }
  {
    \__UsedOn_ReadFromAux
```

Patch label and cref to include the new `[UsedOn]` capabilities.

```
    \NewCommandCopy \__UsedOn_origlabel \label
    \NewCommandCopy \__UsedOn_origcref  \cref
    \NewCommandCopy \__UsedOn_origCref  \Cref
    \NewCommandCopy \__UsedOn_origlabelcref \labelcref
    \RenewDocumentCommand \label { m }
      {
        \__UsedOn_origlabel{#1}\__UsedOn_PrintMessage{#1}
      }
    \RenewCommandCopy \cref \__UsedOn_cref
    \RenewCommandCopy \Cref \__UsedOn_Cref
    \RenewCommandCopy \labelcref \__UsedOn_labelcref
  }
```

At the hook `\AtEndDocument` we write to the .aux file.

```
\hook_gput_code:nnn { enddocument } { cleveref-usedon }
  {
    \__UsedOn_WriteToAux
  }
⟨/package⟩
```