NAME

git-for-each-ref − Output information on each ref

SYNOPSIS

git for−each−ref [−−count=<count>] [−−shell|−−perl|−−python|−−tcl]
                 [(−−sort=<key>)...] [−−format=<format>] [<pattern>...]
                 [−−points−at=<object>]
                 (−−merged[=<object>] | −−no−merged[=<object>])
                 [−−contains[=<object>]] [−−no−contains[=<object>]]

DESCRIPTION

Iterate over all refs that match **<pattern>** and show them according to the given **<format>**, after sorting them according to the given set of **<key>**. If **<count>** is given, stop after showing that many refs. The interpolated values in **<format>** can optionally be quoted as string literals in the specified host language allowing their direct evaluation in that language.

OPTIONS

<pattern>...

If one or more patterns are given, only refs are shown that match against at least one pattern, either using fnmatch(3) or literally, in the latter case matching completely or from the beginning up to a slash.

−−count=<count>

By default the command shows all refs that match **<pattern>**. This option makes it stop after showing that many refs.

−−sort=<key>

A field name to sort on. Prefix **−** to sort in descending order of the value. When unspecified, **refname** is used. You may use the −−sort=<key> option multiple times, in which case the last key becomes the primary key.

−−format=<format>

A string that interpolates **%(fieldname)** from a ref being shown and the object it points at. If **fieldname** is prefixed with an asterisk (*****) and the ref points at a tag object, use the value for the field in the object which the tag object refers to (instead of the field in the tag object). When unspecified, **<format>** defaults to **%(objectname) SPC %(objecttype) TAB %(refname)**. It also interpolates **%%** to **%**, and **%xx** where **xx** are hex digits interpolates to character with hex code **xx**; for example **%00** interpolates to **\0** (NUL), **%09** to **\t** (TAB) and **%0a** to **\n** (LF).

−−color[=<when>]

Respect any colors specified in the **−−format** option. The **<when>** field must be one of **always**, **never**, or **auto** (if **<when>** is absent, behave as if **always** was given).

−−shell, −−perl, −−python, −−tcl

If given, strings that substitute **%(fieldname)** placeholders are quoted as string literals suitable for the specified host language. This is meant to produce a scriptlet that can directly be 'eval'ed.

−−points−at=<object>

Only list refs which points at the given object.

−−merged[=<object>]

Only list refs whose tips are reachable from the specified commit (HEAD if not specified), incompatible with **−−no−merged**.

−−no−merged[=<object>]

Only list refs whose tips are not reachable from the specified commit (HEAD if not specified), incompatible with **−−merged**.

−−contains[=<object>]

Only list refs which contain the specified commit (HEAD if not specified).

−−no−contains[=<object>]
>   Only list refs which don't contain the specified commit (HEAD if not specified).

−−ignore−case
>   Sorting and filtering refs are case insensitive.

FIELD NAMES

Various values from structured fields in referenced objects can be used to interpolate into the resulting output, or as sort keys.

For all objects, the following names can be used:

refname
>   The name of the ref (the part after $GIT_DIR/). For a non−ambiguous short name of the ref append **:short**. The option core.warnAmbiguousRefs is used to select the strict abbreviation mode. If **lstrip=<N>** (**rstrip=<N>**) is appended, strips **<N>** slash−separated path components from the front (back) of the refname (e.g. **%(refname:lstrip=2)** turns **refs/tags/foo** into **foo** and **%(refname:rstrip=2)** turns **refs/tags/foo** into **refs**). If **<N>** is a negative number, strip as many path components as necessary from the specified end to leave **−<N>** path components (e.g. **%(refname:lstrip=−2)** turns **refs/tags/foo** into **tags/foo** and **%(refname:rstrip=−1)** turns **refs/tags/foo** into **refs**). When the ref does not have enough components, the result becomes an empty string if stripping with positive <N>, or it becomes the full refname if stripping with negative <N>. Neither is an error.

>   **strip** can be used as a synonym to **lstrip**.

objecttype
>   The type of the object (**blob**, **tree**, **commit**, **tag**).

objectsize
>   The size of the object (the same as *git cat−file −s* reports).

objectname
>   The object name (aka SHA−1). For a non−ambiguous abbreviation of the object name append **:short**. For an abbreviation of the object name with desired length append **:short=<length>**, where the minimum length is MINIMUM_ABBREV. The length may be exceeded to ensure unique object names.

upstream
>   The name of a local ref which can be considered "upstream" from the displayed ref. Respects **:short**, **:lstrip** and **:rstrip** in the same way as **refname** above. Additionally respects **:track** to show "[ahead N, behind M]" and **:trackshort** to show the terse version: ">" (ahead), "<" (behind), "<>" (ahead and behind), or "=" (in sync). **:track** also prints "[gone]" whenever unknown upstream ref is encountered. Append **:track,nobracket** to show tracking information without brackets (i.e "ahead N, behind M").

>   For any remote−tracking branch **%(upstream)**, **%(upstream:remotename)** and **%(upstream:remoteref)** refer to the name of the remote and the name of the tracked remote ref, respectively. In other words, the remote−tracking branch can be updated explicitly and individually by using the refspec **%(upstream:remoteref):%(upstream)** to fetch from **%(upstream:remotename)**.

>   Has no effect if the ref does not have tracking information associated with it. All the options apart from **nobracket** are mutually exclusive, but if used together the last option is selected.

push
>   The name of a local ref which represents the **@{push}** location for the displayed ref. Respects **:short**, **:lstrip**, **:rstrip**, **:track**, **:trackshort**, **:remotename**, and **:remoteref** options as **upstream** does. Produces an empty string if no **@{push}** ref is configured.

HEAD
>   * if HEAD matches current ref (the checked out branch), ' ' otherwise.

color

> Change output color. Followed by **:<colorname>**, where color names are described under Values in the "CONFIGURATION FILE" section of **git-config**(1). For example, **%(color:bold red)**.

align

> Left−, middle−, or right−align the content between %(align:...) and %(end). The "align:" is followed by **width=<width>** and **position=<position>** in any order separated by a comma, where the **<position>** is either left, right or middle, default being left and **<width>** is the total length of the content with alignment. For brevity, the "width=" and/or "position=" prefixes may be omitted, and bare <width> and <position> used instead. For instance, **%(align:<width>,<position>)**. If the contents length is more than the width then no alignment is performed. If used with **−−quote** everything in between %(align:...) and %(end) is quoted, but if nested then only the topmost level performs quoting.

if

> Used as %(if)...%(then)...%(end) or %(if)...%(then)...%(else)...%(end). If there is an atom with value or string literal after the %(if) then everything after the %(then) is printed, else if the %(else) atom is used, then everything after %(else) is printed. We ignore space when evaluating the string before %(then), this is useful when we use the %(HEAD) atom which prints either "*" or " " and we want to apply the *if* condition only on the *HEAD* ref. Append ":equals=<string>" or ":notequals=<string>" to compare the value between the %(if:...) and %(then) atoms with the given string.

symref

> The ref which the given symbolic ref refers to. If not a symbolic ref, nothing is printed. Respects the **:short**, **:lstrip** and **:rstrip** options in the same way as **refname** above.

In addition to the above, for commit and tag objects, the header field names (**tree**, **parent**, **object**, **type**, and **tag**) can be used to specify the value in the header field.

For commit and tag objects, the special **creatordate** and **creator** fields will correspond to the appropriate date or name−email−date tuple from the **committer** or **tagger** fields depending on the object type. These are intended for working on a mix of annotated and lightweight tags.

Fields that have name−email−date tuple as its value (**author**, **committer**, and **tagger**) can be suffixed with **name**, **email**, and **date** to extract the named component.

The complete message in a commit and tag object is **contents**. Its first line is **contents:subject**, where subject is the concatenation of all lines of the commit message up to the first blank line. The next line is **contents:body**, where body is all of the lines after the first blank line. The optional GPG signature is **contents:signature**. The first **N** lines of the message is obtained using **contents:lines=N**. Additionally, the trailers as interpreted by **git-interpret-trailers**(1) are obtained as **trailers** (or by using the historical alias **contents:trailers**). Non−trailer lines from the trailer block can be omitted with **trailers:only**. Whitespace−continuations can be removed from trailers so that each trailer appears on a line by itself with its full content with **trailers:unfold**. Both can be used together as **trailers:unfold,only**.

For sorting purposes, fields with numeric values sort in numeric order (**objectsize**, **authordate**, **committerdate**, **creatordate**, **taggerdate**). All other fields are used to sort in their byte−value order.

There is also an option to sort by versions, this can be done by using the fieldname **version:refname** or its alias **v:refname**.

In any case, a field name that refers to a field inapplicable to the object referred by the ref does not cause an error. It returns an empty string instead.

As a special case for the date−type fields, you may specify a format for the date by adding **:** followed by date format name (see the values the **−−date** option to **git-rev-list**(1) takes).

Some atoms like %(align) and %(if) always require a matching %(end). We call them "opening atoms" and sometimes denote them as %($open).

When a scripting language specific quoting is in effect, everything between a top−level opening atom and its matching %(end) is evaluated according to the semantics of the opening atom and only its result from the top−level is quoted.

EXAMPLES

An example directly producing formatted text. Show the most recent 3 tagged commits:

```
#!/bin/sh

git for−each−ref −−count=3 −−sort='−*authordate' \
−−format='From: %(*authorname) %(*authoremail)
Subject: %(*subject)
Date: %(*authordate)
Ref: %(*refname)

%(*body)
' 'refs/tags'
```

A simple example showing the use of shell eval on the output, demonstrating the use of −−shell. List the prefixes of all heads:

```
#!/bin/sh

git for−each−ref −−shell −−format="ref=%(refname)" refs/heads | \
while read entry
do
    eval "$entry"
    echo `dirname $ref`
done
```

A bit more elaborate report on tags, demonstrating that the format may be an entire script:

```
#!/bin/sh

fmt='
    r=%(refname)
    t=%(*objecttype)
    T=${r#refs/tags/}

    o=%(*objectname)
    n=%(*authorname)
    e=%(*authoremail)
    s=%(*subject)
    d=%(*authordate)
    b=%(*body)

    kind=Tag
    if test "z$t" = z
    then
        # could be a lightweight tag
```

```
            t=%(objecttype)
            kind="Lightweight tag"
            o=%(objectname)
            n=%(authorname)
            e=%(authoremail)
            s=%(subject)
            d=%(authordate)
            b=%(body)
      fi
      echo "$kind $T points at a $t object $o"
      if test "z$t" = zcommit
      then
            echo "The commit was authored by $n $e
at $d, and titled

  $s

Its message reads as:
"
            echo "$b" | sed −e "s/^/    /"
            echo
      fi
'

eval=`git for−each−ref −−shell −−format="$fmt" \
      −−sort='*objecttype' \
      −−sort=−taggerdate \
      refs/tags`
eval "$eval"
```

An example to show the usage of %(if)...%(then)...%(else)...%(end). This prefixes the current branch with a star.

git for−each−ref −−format="%(if)%(HEAD)%(then)* %(else) %(end)%(refname:short)" refs/heads/

An example to show the usage of %(if)...%(then)...%(end). This prints the authorname, if present.

git for−each−ref −−format="%(refname)%(if)%(authorname)%(then) Authored by: %(authorname)%(end)"

SEE ALSO
git-show-ref(1)
GIT
Part of the **git**(1) suite