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>...]

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

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

<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>
: A string that interpolates %(fieldname) from the object pointed at by a ref being shown. If fieldname is prefixed with an asterisk (*) and the ref points at a tag object, the value for the field in the object tag refers is used. When unspecified, 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).

<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.

--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.

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.

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.

upstream
: The name of a local ref which can be considered "upstream" from the displayed ref. Respects :short 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). Has no effect if the ref does not have tracking information associated with it.

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

color
: Change output color. Followed by :<colorname>, where names are described in color.branch.*.

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.

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. Finally, the optional GPG signature is contents:signature.

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

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 one of :default, :relative, :short, :local, :iso8601, :rfc2822 or :raw to the end of the fieldname; e.g. %(taggerdate:relative).

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
```

SEE ALSO

git-show-ref(1)

GIT

Part of the **git(1)** suite