Myglob search

# Objectives

It's about an efficient glob file search, limiting file system access, avoiding useless access. Some specific design goals:

* Avoid exploration of C:\Development\$RECYCLE.BIN\... for a search such C:\Development\\*\*\cargo.toml, since it's useless, returns unwanted files, and can take time if the recycle bin is large.
* When exploring slow network filesystems, limit network round trips to the strict minimum (accessing metainformation for each file/folder is really expensive)
* Ideally support extended globs (\*\*, \*, ?, [], {}), knowing that \*\* and braced alternatives are not part of POSIX rules. (character classes support should be specified). Note that contrary to dos/windows, \*.\* match only files with at least one extension, not all files. \* matches all files/folders regardless of extension(s) or not.
* For this version, assume that we're only searching for files, myglob search will only return files, never paths.
* Do not hardcode $RECYCLE.BIN in exploration code, but build a folder exclusion HashSet that could potentially other folders such as .git
* Possibly add options to explorer, such as case sensitive, or ignore hidden files (warning: probably very costly regarding execution time)

# Prior Art

* **glob** crate works fine, but doesn't propose filtering and searches in $RECYCLE.BIN. It doesn't support glob braces pattern.
* **ignore** crate filers $RECYCLE.BIN and other unwanted files, but I've not been able do a positive search to include only \*\*\cargo.toml for instance (either adding a specific .ignore file with !\*\*\cargo.toml, or a specific file filter, it always returns everything.
* **glob-match** (https://github.com/devongovett/glob-match) only contains a matching function, no filesystem exploration. Can still be useful though for efficient one glob segment match, but only works with / separator, and only case sensitive…
* **rust\_search** crate (https://github.com/ParthJadhav/rust\_search) seems efficient, but it's a filtering explorer, and does not support glob searches. Filtering function .custom\_filter() mentioned in the doc doesn't exist, and calling filter(xx::xx::Custom(closure)) doesn't work because it's using a private member… Either I miss something or it's not tested/documented correctly.
* **globmatch** crate supports glob searches (separating root from patterns, that's Ok), but actual filtering to exclude $RECYCLE.BIN is processed on iterator output, that is, AFTER exploring $RECYCLE.BIN, making it inefficient for my use case.

# Algorithm

## Path preparation

Decompose glob pattern in segments separated by \ or /:

* **Constant** segment that contain no \* ? [ {
* **Recurse** segment containing only \*\*
* **Filter** segment, containing \* ? [ { and possibly do a syntax validation on filter

Remove and concatenate all constant segments at the beginning to build root path. If there's no constant segment at the beginning, use . as root path.

Possibly, If last segment is \*\* which only match folders, append a \* segment to match all files of deepest folder since we only return files. If we decide to add an option to return folders only, this rule may be revised.

Mark last segment with Last flag, to know that this segment can only match files. Or maybe this may be derived automatically by code without the need for an actual flag.

## Search

Special case, if all segments are constant, just do a direct search for the specified file, and return file or nothing, otherwise just call myglobsearch(root, Vec<Segment>, false).

Fn myglobsearch(root: Path, segments: &[Segment], recurse: bool)

match Segment[0]:

\* Recurse segment: call myglobsearch(root, &Segment[1..], true)

\* Constant segment:

* If final segment, direct search for file, return it if found.
* if not final segment, direct search for folder, and if found, call myglobsearch(root+folder, &Segment[1..], false)
* if recurse, for each folder excluding $RECYCLE.BIN, call myglobsearch(root+folder, segment, true)

\* Filter segment:

* if final segment, search for all files, return the ones matching filter.
* if not final segment, search for all folders, and for each one matching filter, excluding $RECYCLE.BIN, call myglobsearch(root+folder, &Segment[1..], false)
* if recurse, for each folder excluding $RECYCLE.BIN, call myglobsearch(root+folder, segment, true). Be sure to enumerate folders only once.

Use glob\_match library for filter segment matching, manually managing ignore case

## Tests

Glob pattern **C:\Development\\*\*\projects\\*\*\target\release\rsgrep.d**

Match: C:\Development\GitHub\Projects\10\_RsGrep\target\release\rsgrep.d

No match: C:\Development\GitHub\Projects\10\_RsGrep\target\example\release\rsgrep.d

But glob pattern **C:\Development\\*\*\projects\*\*\target\\*\release\rsgrep.d** should match the second, but not the first.

# MyGlob extension: autorecurse

## Objective

Automatically transform some glob expressions to add a recurse search. This optional transformation is implemented in MyGlobSearch crate to be sure the rules are consistent across applications using it.

This is not the default, recurse search can always be explicitly included in a glob expression.

Some applications may chose to always add autorecurse optional transformation because it fits more naturally with some use cases, or only add this transformation if a specific option such as -r indicates an explicit recursive request.

## Mechanism

### Applicability

The autorecurse only applies to valid non-recursive glob expressions, that is, valid glob expressions not containing a \*\* segment. Any glob expression containing \*\* is unchanged through this transformation, since it’s already recursive.

Invalid glob expressions such as “\folder\\*\*pom” still cause an error to be returned when calling compile() on a MyGlobBuilder object, regardless of autorecurse transformation.

### Case of constant glob expressions

A constant glob expression is only composed of a root path, and no filter or recurse segment. Only constant glob expressions referencing a valid folder are transformed.

* If the root path is a valid folder (or a symbolic link to a folder), then automatically append “\\*\*\\*” to search for all files in root folder and its subfolders. Note that for Glob expressions ending with \*\*, thus unmodified by autorecurse mechanism, a \\* is already automatically appended.
* Other root paths (pointing to a file, a symlink to a file, or pointing to nothing existing are unchanged. Exploring glob will either return 0 file on an error (invalid drive letter or network path for instance), it’s the caller responsibility to handle these cases.

### Case of glob expressions where last segment is a filter

Examples: “C:\Development\Git\*\\*.tar”, “src\cargo.{toml,lock}” or just “\*.cb[rz]” (in this last example, root path is ‘.’).

In this case, a \*\* is automatically inserted before the last segment to extend search to all subfolders of the parent path. Here are how these examples are transformed:

Original glob expression After autorecurse transformation

C:\Development\Git\*\\*.tar C:\Development\Git\*\\*\*\\*.tar

src\cargo.{toml,lock} src\\*\*\cargo.{toml,lock}

\*.cb[rz] \*\*\\*.cb[rz]

## API update

Add “pub fn autorecurse(mut self) -> Self” to MyGlobBuilder

Maybe: Add a member function to MyGlobSearch to retrieve the actual glob string used for search, including transformation effects if applied.