This page contains my idea for extending Total Commander's functionality which (I hope) will be implemented someday.

---

1. Introduction

This issue has been discussed couple of times but I would like to describe my idea of implementing that.

Everyone knows the possibilities of "Custom view", QuickSearch and "Find Files" itself. I think that gathering a few of theirs features would be really nice.

First thing to remember is that live filter should work on visible files / folders list only. The second one: because it differs from other searching methods this one can be used additionally with others. There is nothing against using LiveFilter together with "Custom view" or after "Feed to listbox" functions. Of course, having this feature implemented makes no difference between Total Commander and other file managers. The biggest advantage of it is having really good parser behind whole filtering issue.

Another thing worth to mention is that Windows operating system places restrictions on characters allowed to be used in file / folder names. Because of that we already have set of meta-characters for further use.

2. Placement

Because everyone would like to have clear TC interface, there are not so many possible solutions for placement issue.

Three possible places come to mind:

- integrate filter syntax into path fields above file list (like for Custom view),

- give dynamic "status bar like" control which will appear when it's needed,
- use similar approach as for **QuickSearch** (but with always on top option).

3. Basic filter

The most primitive filtering which comes to mind is just searching for typed string existence in file name. Parser should forget about casing or filter chain position.

Sample Flash movie can be seen here .

4. Advanced filter

4.1. Custom view sets.

For more advanced filtering, the functionality of "Custom view" could be used.

When user put > sign he can see the list of suggested filters taken from rules saved in "Find Files" or "Custom view".

Sample Flash movie can be seen here .

4.2. Regular expressions.

LiveFilter could mean nothing without the possibility of using regular expressions.

Placing ? character at the beginning of the search string could switch filtering to RegEx mode. This way the rest of the string would be interpreted as regular expression pattern. Automatic beginning of the line ("^") placing should be optional (configurable via INI file). In the same matter using modifiers (case-sensitive matching, greedy mode) can be solved also.

Sample Flash movie can be seen here .

5. Columns filtering.

5.1. Basic rules.

All sections above are meant to process "Name" column only but there is one place more to go: filtering the rest of columns.

Basically it is possible to search for files with special attributes using "Find Files" window so once again I will write: it might be easier to do it with LiveFilter. Because LF operates on visible list only, it should be consistent in this matter too.

As I wrote above: we have plenty of possible characters to use for other columns searching. My proposition is to treat character as column indicator (the number of back-slashes will inform about column's number).

Example 1.

- LF for: **ai**
- is the same as for: **ai**
- It means: look for **ai** string in the content of the first column.

It is obvious than:

- \ - will be threaten as second column mark,
- \ - third,
- … and so on.

Example 2.

- **aibj** - will search for items with **ai** in first column and **bj** in second,
- **?.b.*** - will search for items with second column's content matching regular expression **.b.***

5.2. Column types filter strings.

TC allows to use different types of columns for displaying files properties. Some of them could allow to use algorithmic operations like "more than" or "less than". The easiest way to achieve that is using similar scope indicating as RegEx has.

My idea is to use the syntax like below:

- **<X,Y>** - value in the range **X** <= value <= **Y**,
- **<X,>** - value greater than **X**,
- **<,Y>** - value smaller than **Y**,
- **<X>** - value is equal **X**.

For numeric fields with available units, its usage could be enhanced:

- **<10kB, 1MB>** - value greater than **10KB** and smaller than **1MB**,
- **<,1GB>** - value smaller than **1GB**.

In the same way, the date and time fields should give proper post-fixes parsing:

- **<1Y1D,>** - values newer than **one year and one day**,
- **<2Y>** - all values matching **two years ago** rule.

Upper case indicators should indicate (Y - year, M - month, D - day, H - hour, Mi - minute) the time counted from now.

Lower case indicators should search for exact date value (y - year, m - month, d - day, h - hour, mi - minute):

- **<2007y12m, 2008y3m>** - all values from **December, 2007** to **March, 2008**,
- **<, 2008y3m>** - all values younger than **March, 2008**,

- **<9m>** - all values from **Septembers** (BUT: **<9M>** - all values **nine months old**).

6. End notes.

Implementing such feature could be the killing one together with checkbox list (see: movie1 or movie 2
). TC might be the first file manager with such extended LiveFilter which would make file operations much easier to do.

PS. Auto-suggestion for last typed filters might be needed.

APPENDIX (9th of August, 2008)

Some time ago I've been thinking about another idea of filtering the files list. This is something which could be placed between Basic mode and the Advanced. I've already explained my idea on Locate32 forum board so I will allow myself to use post from there.

Lately I've been looking for the file which name was not well known for me. I knew what it was about but exact name was unknown. After some time I've managed to find it but it gave me few things to think about. I've realized that if the user doesn't know the exact name of the file he can always have a problem BUT it can be avoided giving more flexible search.

The most obvious might be RegExp but it can be troublesome for newbies. Sometimes it is needed to search files by similarities in their names.
My proposition is to implement algorithm which will compute Levenshtein distance between existing file names and the given one. With properly chosen (configurable) parameters search process could be more accurate.

I've made some tests with some dictionary looking for "cool" name with different "similarity" values and I've gained the results like below:

Similarity: 90%
Results: cool
Similarity: 80%
Results: codol, cohol, cool, cooly, corol, coyol, crool,
Similarity: 70%
Results: bool, chol, codol, cohol, coll, coof, cool, cooly, coom, coos, corol, coul, coyol, crool, dool, gool, mool, rool, sool,
Similarity: 60%

Results: bool, booly, brool, calool, ceorl, choel, choil, chol, chola, chold, choli, cholo, choop, cibol, cloof, cloop, cloot, coaly, coble, cobola, cocle, cocos, codol, cogon, cohol, coll, colla, colly, color, comal, comox, conal, conor, conoy, consol, cooba, coodle, cooee, cooer, coof, cooja, cool, coolen, cooler, coolie, coolly, coolth, cooly, coom, coomb, coomy, coony, coorg, coos, coost, copal, coroa, corol, coryl, coul, cowal, cowle, coxal, coyly, coyol, crood, crool, crowl, cumol, dool, dooli, dooly, gool, gools, hooly, iodol, jhool, joola, mool, mools, pooli, pooly, recool, rool, scoon, shool, sool, sotol, woold,

This is just an example and might look strange but giving the possibility to search files by "string similarities" might be worth to think about imho.

BTW. You can find sample C# app (with code and dict) here:
http://fenixproductions.republika.pl/tmp/Test_Levenshtein.zip

Because of that many spell-checkers use similarity algorithms, the best way is to take a look how they work in office suites. In matter of the fact: you've got a misspelled word (file name to look for) and the dictionary (filesystem database) to be searched.

In general the spell checking algorithms are the core of this idea. Levenshtein alghorithm is just an example because this is the simplest solution but it doesn't mean "the only one". Any variations might be good as long as they are still looking for strings similarity.

I have also wrote simple WDX plugin which can be found in this forgotten thread. It can show the basics of this way of filtering / searching.

Usage instructions:

1. download Similarity plugin,
2. install it in TC,
3. copy leven.ini file into TC's directory,
4. edit it's content in the way you like (the "Phrase" is the word which will be searched for similarity),
5. go to **Find files** dialog,
6. search for any file (***.***),
7. in **Plugins** tab choose **similarity**,
8. specify files names similarity using **simil1** field value (0 - nothing similar, 100 - exact word) and "greater than" sign.

The results results list will show the files with the names similar to Phrase (step 4) as much as you've specified in step 8.