## The YGrep Search Engine

The YGrep Search Engine (YGREP.DLL or YGREP32.DLL) is the name of a Dynamic Link Library built for MS-Windows by Yves Roumazeilles and which is able to provide two search functionalities:

Approximative Search

Regular Expression Search

Soundex Search

It can be found bundled with some applications like the shareware utility *ClusterView* or some other commercial ones. But, more to the point, you can use it to enhance your own applications.



The YGrep Search Engine was brought to life to by an effort to materialize the knowledge I acquired in text processing some years ago.

First, as I am sure you already infered, AGREP is the traditional name for Approximative Search. The notion has been brought to light - at least mine - by Ricardo Baeza-Yates and Gaston H.Gonnet under the name of Shift-Or (or Shift-Add) search method. To my knowledge, the name of **agrep** was originally adopted by Sun Wu and Udi Manber for a Unix utility implementing a similar algorithm.

Then, Regular Expression Search is based on old work by many researchers on automaton logic. Many implementations have already been found on Unix systems (under the application names of **ed**(1) and **grep**(1)) and others Operating Systems including VAX-VMS, DOS, CP/M, etc. They often differ from the original one on Unix, but most power users now recognize easily the common syntax and appreciate its powerful capabilities.

I added some extensions which were not in the original Unix version. In that I was following the interface specification of various authors including David Conroy (original author of the superb MicroEmacs editor which I advise you to use in its more recent version by Daniel Lawrence for your MS Windows editing tasks), Ozan S. Yigit and Karl Hoorsfish. To that I had to add my own salt and pepper to get a working interface. The original version of this Regular Expression Search was inserted in the YR-Emacs public domain text editor I wrote on the basis of the mentioned MicroEmacs editor.

Finally (up to now), Soundex Search is based upon a simple principle presented by Knuth, D.E. in his 1973 book "The art of computer programming - Volume 3: Sorting and searching" (Addison-Wesley Publishing Company, Reading, Mass. Page 392). I also kept an eye upon a public domain implementation of the basic transformation by N. Dean Pentcheff (University of California at Berkeley).

I apologize for omitting many of the other sources of my little knowledge in the field of programming and text programming.

## New features

Version 6.00 adds the new soundex set of functions.

Version 5.01 adds a minor but important improvement to the YGrep Search Engine: Delphi 2.0 interface!

Version 5.00 is the latest major improvement to the YGrep Search Engine in years. It includes a new set of functions for handling directly files (rather than only lines), a way to remove all dialog boxes (in order to help in internationalizing client applications), Boolean associations of search commands to build complex meta-searches. Also, some of the minor improvements may have a slightly good impact on overall performance (but it may be difficult to time it, though).

A completely new "test" application has been built for the file-handling operations. Since it can be of general use, I included it in the shareware package. Its source code comes in the Licensed package. It's a great and simple example for this set of functions.

The Full Developer Kit is now 100% built under Visual C++ v4.1 (and will leave under this status as long as possible since IMNSHO this is a rather nice development environment). In this move, it has been cleaned and documented some more. A few errors have been corrected.

Version 4.07 and v4.07a solve a pair of bugs discovered by Robert Schottland (thanks Robert!). The first one is a complete failure of character classes (those structures involving brackets like [a-z] to declare all lower-cased letters). The use of these was utterly random.

It also removed a bug in substitution functions when used with the sub-expression substitution patterns (when using the \0 to \9 tags in the substitution to insert parts of the original pattern). The functions had the very bad habit of inserting the tag (not only the pattern) into the substituted string.

Version 4.06 solves the problem of not getting a hit for regular expression ".*" when matching against an empty string.

Version 4.05 solves a no-match-case search problem in regular expressions.

Version 4.04 is the first version allowing full text substitution with the new functions:

AGrepSubstitute

RGrepSubstitute

SAGrepSubstitute

SRGrepSubstitute

Users will certainly prefer them to the old AGrepSubsBuild and RGrepSubsBuild which are still available for upward compatiblity.

Version 4.03 is adding improved documentation and some important new download instructions. Support on Compuserve has been added from our new address there (101233,1032).

Both are dwarfed by the apparition of the first 32-bit programming model version of the YGrep Search Engine. If you downloaded only the 16-bit version, you should look into the Windows 95 part of your BBS, FTP site or library.

Version 4.02 is adding the important new layer of functions with only scalar parameters. This allows to use more easily the YGrep Search Engine with Visual Basic and other programming languages which have difficulties calling too complex functions or functions with very complex parameters.

Version 4.01 is more or less a maintenance release with bug removal, performance improvements, documentation improvements, size reduction, help file improvements.

In version 4.00, the name was changed from AGrep Search Engine to YGrep Search Engine, in order to avoid confusion with existing Unix/Linux utility named *agrep*.

I also removed some bugs, optimized out a few bytes, removed some compilation warnings, and did a lot of field testing with the beta testers.

In version 3.02, the following functions were added:

AGrepSubsBuild

RGrepSubsBuild

InitWordCharTable

AddWordChar

RemoveWordChar

You may also have noticed that the documentation has been improved a lot. Many typos were removed, and many little errors were corrected (including - shame on me! - the removal of a subject page having nothing to do with the whole subject). A database of common questions and features has been added to this help file to improve the efficiency of your bug busting and your understanding of the operation of the whole YGrep Search Engine.

You may not have noticed, but the performance was improved, some bugs were found by the users (yes! it's you) and removed. Thanks for your help!

A new version of the YGrep Search Engine for MS-DOS is now available.

## Future developments

If you register conveniently, and keep on following the evolution of the YGrep Search Engine, you will see the following expected future developments real soon now:

New improved memory management strategies for the registered versions

Possible new ways to look for data

What you're asking for... I try to include it as long as it does not break the existing code and if it can be helpful to some

## FTP and Compuserve availability

The YGrep Search Engine is available through anonymous FTP (Internet file download). The reference storage is on the famous SimTel and CICA servers. Anyhow, to find the

ftp.winsite.com directory: pub/pc/win95/programr file: yg600w95.zip

ftp.coast.net directory: SimTel/win95/dll file: yg600w95.zip

Anyhow, if you want to get up-to-date availability information, you should have a look at **http://ourworld.compuserve.com/homepages/Yves_Roumazeilles/** which gives up-to-date information on how to upload the packages. Furthermore, it gives a few mirror sites in order for you to reduce download times by using less common sites.

However, SimTel and CICA are rather famous locations and people tend to overwhelm the capacity of the machines there. Here are some of my preferred mirror sites:

src.doc.ic.ac.uk (155.198.1.40)

nic.switch.ch (130.59.1.40)

ftp.unicamp.br 143.106.10.54)

plaza.aarnet.edu.au (139.130.23.2)

romulus.ucs.uoknor.edu (129.15.10.20)

micros.hensa.ac.uk (194.80.32.51)

ftp.jussieu.fr (134.157.0.130)

Of course, you can also make an Archie search if you have access to it. And you will easily find the most appropriate file server near to you. That is the way I look myself for the various versions of the YGrep Search Engine disseminated around the world.

You can also join me by email for help or updated information...

The YGrep Search Engine is also available from **Compuserve**. You should look into the library of the "Windows Shareware Forum" (GO WINSHARE). The YGrep Search Engine can be found (and downloaded) from the "[22] Programming/Dev. tools" section. If you cannot find it here, first try using the PC File Finder utility from WinCIM, then, reach me by Compuserve mail (at my 101233,1032 personal address).

## YGrep Search Engine's approximative search

The approximative search is allowing you to find a text without knowing the exact form of what you are looking for. For example, you can look for text without taking into account the case of the letters (without making a difference between uppercase and lowercase letters).

The operation is centered on the fact that in most cases you know a text string which is "approximately" what you are looking for in your files. Additionaly, you are able to say that you expect this text string to have a certain number of errors.

For example, the string '`East Germany`' is approximately identical to '`West Germany`', but there are 2 errors (the first two letters of the words) and 10 matches.

## YGrep Search Engine's regular expression search

The regular expression search is allowing you to find a text based on a description which will help you be more precise than with Approximative Search, but also more difficult to handle before you get used to it.

To give you a first look at what can be done (without going too far into the regular expressions syntax), here are a few of the possibilities.

You can search for pattern in the beginning (or in the end) of the lines, ignoring the similar patterns which appears in the middle. You can search for telephone numbers (XXX-X-XXX-XXXX), dates (XX/XX/XX), times (XX:XX), three-figure-numbers (XXX), or any other strictly formed field of numbers. but you can also look for patterns a little more evasive like: four letter-words containing no figures, but beginning with an S letter either in lower- or uppercase and in the end of a line (that is defined by the pattern '`\<[Ss][^0-9_][^0-9_][^0-9_]\>$`').

For an extended specification of the regular expression used, see also Regular Expressions

## YGrep Search Engine's Soundex search

The soundex search is based upon the idea that a in English, and more specifically when writing names of people, most of us do introduce typos and errors which are easily recognized if you speak the word up (say "hello!" and say "helow!") or that come from usual errors made when typing people names (the orthography of people's names has long been considered highly flexible). It is then useful to be able to compare two names using their internal simplest sound structure.

That way "Euler" and "Eller" are probably the same people, "Wockeck" and "Woikcek" are probably two different ways of writing the name of the same people. Soundex will make no difference between the names of these pairs.

## YGrep Search Engine regular expressions

The YGrep Search Engine regular expression routines support the full range of Unix regular expressions as defined in **ed(1)** and in **grep(1)**.

### Specification

^ A circumflex as the first character of the pattern forces matches to beginning of lines.

$ A dollar as the last character of the pattern forces matches to end of lines.

. A period anywhere in the string matches any single character.

* An expression followed by a asterisk matches zero or more occurrences of that expression.

+ An expression followed by a plus sign matches one or more occurrences of that expression.

- An expression followed by a minus sign optionally matches that expression.

[] A string enclosed in square brackets matches any character in that string, but no others. If the first character of the string is a circumflex the expression matches any character except the characters in the string. A range of characters may be specified by two characters separated by a -.

\< A backslash followed by an opening < matches the beginning of a word.

\> A backslash followed by a closing > matches the end of a word.

\( A backslash followed by an opening ( describes the beginning of a tagged sub-expression (see Substitution Expressions, it has no effect on search-only expressions).

\) A backslash followed by a closing ) describes the end of a tagged sub-expression (see Substitution Expressions, it has no effect on search-only expressions).

\ A backslash folowed by any other character quotes that character. This allows a search for a character that is usually a regular expression specifier.

### Examples

| | |
|---|---|
| ^Windows | matches all lines starting with *Windows* |
| Grep$ | matches all lines ending with *Grep* |
| H..p | matches all lines containing *Help*, *Hoop*, *Harp*, etc. |
| ^W.n | matches all lines starting with *Win*, *Won*, etc. |
| \$ | matches a dollar sign |
| fo* | matches *f*, *fo*, *foo*, etc. |
| fo+ | matches *fo*, *foo*, etc. |
| [xyz] | matches *x*, *y* and *z* |
| a[^xyz]c | matches *abc*, *arc* and *aXb* but not *axb* |
| ([0-9]) | matches *(0)*, *(1)*, *(2)*, *(3)*, *(4)*, *(5)*, *(6)*, *(7)*, *(8)* and *(9)* |
| ([0-9]*) | matches *()*, *(0)*, *(123)*, *(2512)*, etc. |
| \<[Aa].*\> | matches any non-empty word beginning with either *a* or *A* |

## YGrep Search Engine substitution expressions

The YGrep Search Engine regular expression substitution routines support a small set of expressions to define how the substitution will be performed.

### Specification

& An ampersand in the substituted string forces insertion of the full matched pattern.

\number A backslash followed by a number forces the insertion of the tag matched with the equivalent number in the pattern.

\& An escape sequence to allow the insertion of the & character (while removing its *matched pattern* meaning).

### Examples

| Patterns | Substitution | |
|---|---|---|
| Windows | MS-& | replaces all occurences of *Windows* with *MS-Windows* |
| \(dows\)\([Ww]in\) | \2\1 | allows to reorder the pattern *dowsWin* into the normal *Windows* regardless of the letter-casing of the *W* in the beginning of the word |

Note that \0 is equivalent to & and they both match the whole found string.

## YGrep Search Engine Limitations

The YGrep Search Engine is limited in both the length of the pattern it can manage and in the maximum number of errors it accepts.

The length of the pattern is limited to 512 letters maximum.

The number of errors is limited to 256 maximum.

Both of them are limited by the following formula:

length_of_pattern <= 512 / (log2(number_of_errors) + 1)

The simple meaning of this formula is that if you have a long pattern, you cannot have a large number of errors. For example, with a 256 character pattern you must limit yourself to 1 error only. In most cases, this is not too limiting, but it should be noticed.

To partly overcome that constraint it is also possible to use the non-cased text search functionality.

Should you need to have a limitation placed higher or lower, please, contact the author for a customized version (this is not much more expensive than a standard package) or a source code license.

**Single User Registration Fee**

Yes! We remind you the single user license fee is still only a mere 95FF or US$20.

**Full Developer Kit Registration Fee**

This highly efficient license has a fee of only 1410FF or US$295.

## Registering The YGrep Search Engine

The YGrep Search Engine is distributed as shareware. It is not free or public domain. This means you may copy and distribute it freely but should you find it useful and use it beyond an initial evaluation period of 30 days you are both legally and morally obliged to pay the registration fee or license fee.

Yes, I want to register now!

| | |
|---|---|
| Yves Roumazeilles - the author | France - money transfers / no credit cards |
| Public (Software) Library | US and worldwide - credit card orders |
| Public (Software) Library | Worldwide - on the Internet |
| Compuserve SWREG registration | Worldwide - CIS customers |

**What you get when registering**

This is the important question. Here is the list:

- updated and optimized full YGrep Search Engine with the license to use it on a single computer. This will include a complete registered MS-Windows DLL, and libraries for MS-DOS (small, compact, medium, and large memory models).

- user documentation on plain paper (more than 60 pages of code, reference data, advice and answers to questions)

- an immediate notice when a new release is ready on the market.

- rebate coupons for upgrading to new releases.

- source code of the this help file to allow you to easily build the help file for YOUR application. This will reduce your work when preparing your application to ship and you may find interesting ideas on how to build a nice help file for MS-Windows.

- source code for useful resources you can use in your application (dialogs, etc.) in relation with the YGrep Search Engine.

- sample files for different languages when available.

- a registration number to identify yourself when contacting us.

- support through fax, phone and Email.

- access to our database of bug reports between releases. No release is done while this database contains even a single bug. We do not ship products we know contain bugs. But after shipping, users may discover ugly things in our code, and we trace them while we hunt them.

**Commercial software and shareware developers**

You can get a complete Developer Kit for a flat rate including unlimited royalty-free right to distribute the registered Dynamic Link Library in your product. This allows you to include it in your nice universal text editor or encryption package.

Customized versions can also be obtained from the author (me, of course) upon request and after acceptance of a specific quotation (most customizations can be obtained for about twice the Developper Kit registration license fee).

Remember that you can also ask for source licenses which will include full C source code, with full resources, definition files (everything you need to rebuild the YGrep Search Engine from scratch). I cannot give more. Well! May be not give, because you pay for it. But, it's a bargain you could discuss with me if you need this.

**Acknowledgments**

I would like to thank the following people whose help has been invaluable during the development of the YGrep Search Engine:

Christian Lescuyer provided the original idea and a large amount of time for product testing (even in alpha state).

Anne Elisabeth Halpern proved that the concept was usable by exploiting it during years in the process of preparing her litterature thesis about Michaud (Belgian French-writing poet and painter). She provided permanent confidence even when her own work was overwhelming.

Martin Heller for his excellent book "Advanced Windows Programming" published by John Wiley & Sons. This is the most useful book about Windows programming I ever found. If you intend to do MS Windows programming, you NEED it.

The whole WIN3-L@UICVM.BITNET Internet mailing list who provided help when I was stopped in the development process. Not all the subscribers (more than 2000 currently) provided help, but a dozen of them are very proficient and helpful. I can remember and thank Walter Knopf, Yossi Oren, Yoav Chernobroda and Vance Gloster, and many others...

## Registration form

Yves Roumazeilles - author

Select "File-Print Topic" from the menu bar to print this form. You will then be able to fill it. You can accompany you order with check. Credit cards are NOT accepted. You can order by phone or fax.

Please use the form when ordering by mail.

NAME: _____

COMPANY: _____

ADDRESS: _____

_____

_____

TOWN: _____

CITY: _____

POSTCODE: _____ COUNTRY/STATE: _____

TELEPHONE: _____

FAX: _____

EMAIL: _____

YGrep Search Engine

Single user license ($20 or 95 French Francs) _____

Developer kit license ($295 or 1410 French Francs) _____

BitList Engine

Single user license ($20 or 95 French Francs) _____

Developer kit license ($295 or 1410 French Francs) _____

ClusterView Application

Single user license ($20 or 95 French Francs) _____

Shipping/Handling to Europe ($4 or 20 French Francs) _____

Shipping/Handling outside of Europe ($6 or 30 French Francs) _____

==========

Sub-Total.... _____

European residents, apply VAT/TVA 18.6% (on sub-total) _____

==========

Total..... _____

Here, shipping/handling costs are applied once even for multiple orders.

Make cheques payable to: Yves Roumazeilles

If you wish to pay in other currencies than French Francs or dollars, apply the normal change rate to French Francs and add a 6% increase to cover the costs my bank charges me. Thank you.

If you wish to pay in other currencies or if you want to use a fast cash payment, we can organize a Western Union | Money Transfer (*the fastest way to send money worldwide.*™). Contact me and I can provide the appropriate information to reach one of the 28000+ agent locations in the world (in more than

100 countries) and organize a money transfer which will "do it" in a few minutes.

If you want to pay with a credit card (MC, Visa, Amex or Discovery) please contact Public (Software) Library through mail, phone or fax.

Mail to:


Yves Roumazeilles
63 rue des Moines
75017 PARIS
(FRANCE)


This address is also the place where you can get information of various kinds (about the status of the shipment of the order, registration options, product details, technical support, volume discounts, dealer pricing, etc.)


Phone: +33-1-42.28.74.51
Fax: +33-1-30.38.37.07
Email: **YGrep@usa.net** (or 101233.1032@compuserve.com or Roumazeilles@sagem.fr)
Compuserve ID: 101233,1032


PLEASE, if you do not like the long distance calls involved with my address in France, do prefer sending a Fax (it won't hang your phone too long), sending an Internet Email (it won't cost too much, and Compuserve, BIX, AOL all have Internet gateways for your mail - I know you are paying for that, I won't overuse it, but any Email with the product name in its subject field gets a very high priority).

## Registration form

Public (Software) Library - registration service

Select "File-Print Topic" from the menu bar to print this form. You will then be able to fill it.

You can order with MC, Visa, Amex, or Discovery from the Public (Software) Library by calling. You can accompany you order with check. Credit cards are NOT accepted. You can order by phone or fax. But **THESE NUMBERS ARE FOR PLACING AN ORDER ONLY**!

Please use the form when ordering by mail (furthermore, filling it in **before** you call will help in getting the order reliably placed).

NAME: _____

COMPANY: _____

ADDRESS: _____

_____

_____

TOWN: _____

CITY: _____

POSTCODE: _____ COUNTRY/STATE: _____

TELEPHONE: _____

FAX: _____

EMAIL: _____ (please provide it in order to be informed)

YGrep Search Engine (item/product #11244)
Single user license ($20) _____
Developer kit license ($295) _____

BitList Engine (item/product #11245)
Single user license ($20) _____
Developer kit license ($295) _____

ClusterView Application (item/product #11246)
Single user license ($20) _____

Shipping/Handling to Europe ($4) per license _____

Shipping/Handling outside of Europe ($6) per license _____

==========

Sub-Total.... _____

==========

Total..... _____

Be very clear about which version of the software you wish to purchase (single license or Developer Kit license), since the same item number is used for both, but the packages and prices are quite different.

Contact one of the following:

by phone: **800-2424-PsL** or Intl+1+**713-524-6394**

or by FAX: Intl+1+**713-524-6398**

or by **CIS/Compuserve** Email to **71355,470**

or on the **Internet/web** at **http://206.109.101.6**

or send mail to:
**PsL**
**P.O. Box 35705**
**Houston, TX 77235-5705**
**(United States of America)**

Remember! **THE ABOVE NUMBERS ARE FOR ORDERS ONLY**. You cannot expect any kind of support through them. If you call there, not only will you be left without an appropriate answer, but I will have to pay for the useless call.

Any questions about the status of the shipment of the order, registration options, product details, technical support, volume discounts, dealer pricing, site licenses etc., must be directed to:

Yves Roumazeilles
63 rue des Moines
75017 PARIS
(FRANCE)
Phone: +33-1-42.28.74.51
Fax: +33-1-30.38.37.07
Email: YGrep@usa.net (or 101233.1032@compuserve.com or Roumazeilles@sagem.fr)
Compuserve ID: 101233,1032

To insure that you get the latest version, PsL will notify us the day of your order and we will ship the product directly to you.

## Registration instructions

Compuserve SWREG - registration service

This is the on-line registration service for Compuserve customers. It allows on-line, immediate, direct registration.

You should GO SWREG, to join the Shareware Registration Forum. You will then be asked the reference of the software product you want to get. Take it from the following list:

YGrep Search Engine v4.0 (item/product #)
- Single user license ($20) Id# 7811
- Developer kit license ($295) Id# 10443

ClusterView v2.0 Windows 3.1 Application (item/product #)
- Single user license ($20) Id# 7812

ClusterView v2.0 Win32/WinNT/Win95 Application (item/product #)
- Single user license ($20) Id# 7814

Shipping/Handling to Europe ($4) per license

Shipping/Handling outside of Europe ($6) per license

Any questions about the status of the shipment of the order, registration options, product details, technical support, volume discounts, dealer pricing, site licenses etc., must be directed to:

Yves Roumazeilles
63 rue des Moines
75017 PARIS
(FRANCE)
Phone: +33-1-42.28.74.51
Fax: +33-1-30.73.24.40
Internet Email: YGrep@usa.net (or 101233.1032@compuserve.com or Roumazeilles@sagem.fr)
Compuserve ID: **101233,1032**

To insure that you get the latest version, Compuserve will notify us the day of your order and we will ship the product directly to you.

## Application Copyright and User License

The YGrep Search Engine Dynamic Link Library (DLL) and its documentation files and manuals are copyrighted (C) 1992-93-94-95-96 by Yves Roumazeilles. Their use is subject to the acceptance of the User License terms.

All the names used here are trademarks and registered trademarks of their respective owners.

## Registration Fee

If you are using the YGrep Search Engine after the initial 30 day evaluation period, you must pay the license to continue using the package. This payment is named registration fee.

For use by corporations and other institutions, please contact the author for a licensing arrangement. Customizing and other special licensing are available upon request.

If you want to get the full source code of the library or of one of its components, please contact the author for a licensing arrangement.

## Shareware User License

The program files and the associated documentation (e.g. this documentation) are copyrighted by the author. The copyright owner hereby licenses you to use the software given these restrictions:

* The program shall be supplied in its original, unmodified form, which includes this documentation.
* For-profit use without a license is prohibited.
* The program may not be included - or bundled - with other goods or services. The licensed version is ready there for this purpose.
* No fee is charged. An exception is granted for not-for-profit user's group, which are permitted to charge a small fee (under $5) for materials, handling, postage and general costs. No other organization is permitted to charge any amount for distribution of copies of the software or documentation, or to include copies of the software or documentation with sales of their own products.

There is no warranty of any kind (either implied or not). The copyright owner may not be held liable for any damages, including any lost profits or other incidental or consequential damages arising out of or inability to use the software. By using the software, you agree to this.

## Common Questions and Features

If you have some problem while using the YGrep Search Engine it's the right place to look for a solution. The most common problems encountered by users (both end users and developpers) are stored in this unstructured database (in my mind unstructured means "with strictly no order").

## I cannot load the DLL, while I could some time ago.

The YGrep Search Engine shareware edition package is distributed for an evaluation period of 30 days. The software automagically detects the end of the period and stops accepting load requests. The expiration date of the shareware license has been reached... You should register now. I will send you immediately an up-to-date and unlimited package.

If you are using a registered package, you have certainly changed something in your computer/application configuration to stop the DLL loading process. It is a process which des not need many things and little memory, for sure.

### How do I initiliaze the expiration date?

You do not have to. The software automagically determines it and stores it appropriately.

## I am not sure what a [z-a] pattern would match.

While [a-z] matches any lower case letter, the [z-a] matches nothing at all. You have to be cautious with the order of the letter you put between the brackets when you define a set.

But if you give all the intermediate letters, the order is meaningless. [abcdefghij] is equivalent to [bcadefihgj] and to [a-j].

### My application seems to slow down while using the YGrep Search Engine and then stops because of lack of memory.

### After *some number* of searches, I cannot get anything other than "not enough memory" messages.

If you're an end-user.

The application calling the YGrep Search Engine has a bug in it. Call your product support and you can shorten the response time by stating that it could be related to "a memory leak in YGrep Search Engine, for not using **AGrepEmpty**". Then wait for a correction of the bug.

If you're a developer.

Did you remember to call the appropriate **AGrepEmpty** function after you no longer need the **AGREPINFO** structure? It is necessary to free memory used by data structures pointed to by elements of **AGREPINFO**, before freeing the structure itself.

This is a two step process, first release indirect memory by calling **AGrepEmpty**:

```
AGrepEmpty( aGI );
```

Then, free the structure by calling the traditional memory de-allocator free:

```
free( aGI );
```

## AGrepEmpty takes too much time in executing.

Some low-memory configuration may have difficulties in freeing memory with **AGrepEmpty**. This leads to long waiting time while running through this routine (up to 2 seconds in slow PC with low-memory condition).

You can either try to reduce the overall memory use of the application by unlocking as much global memory as possible before calling the AGrep family routines, or we can provide a customized YGrep Search Engine which would only use local memory instead. The difference may not appear great in most cases, but could specifically help your project.

You can also try to preserve an **AGREPINFO** structure after compiling a pattern to avoid repetitively calling **CompileAgrep**/**AgrepEmpty**.

### The YGrep Search Engine is great, but I would like to have it under MS-DOS.

That's not a problem. If you cannot find it on your usual BBS or Internet site, ask me. I have already available MS-DOS and Windows versions of the YGrep Search Engine.

Furthermore, when you register, you get both versions simultaneously. That's a bargain!

## In some cases, I get a General Protection Fault from either AGrep or RGrep.

It is always possible that you found a bug in the YGrep Search Engine. However, the most current cause of this kind of behavior appears to be a program compiling an expression and then providing the optimized data structure to the wrong function (e.g. calling AGrep with an RGREPINFO structure, or RGrep with an AGREPINFO structure). Both types of structures are positively incompatible with each other and should not be used with the wrong library function.

Another very common situation appears when a user forget that the Windows version **MUST** be compiled in the Large memory model. All others (Tiny, Small, Medium, or Compact) are not currently supported. In the future, the Medium memory model could become also available. But, there appears to be no hurry.

The MS-DOS version has not this limitation. You can use any of the models.

### CompileAGrep does not accept my large patterns or patterns with a large number of allowed errors.

### What are the limitations of the YGrep Search Engine?

That's normal. there are some limitations you will have to live with. If necessary, you can contact the author (me) in order to get a specifically customized version.

## I only want to match patterns with no regular expressions, no approximation.

That's quite normal! Look at the **AGrep** with a parameter of 0 errors (**k**=0). This will be detected and will call **strstr** with the only difference that you still are able to use the 'Match case' capability. However, if you look for sheer efficiency, you should be thinking of calling directly **strstr** from your compiler library. It has the advantage of avoiding you the optimization test at the entry of **AGrep** before calling **strstr**. This is a very small speed improvement, but sometimes...

## How do I know the version of a specific YGREP.DLL?

There are two ways. The simplest is to look at the date of the file YGREP.DLL. However, you could encounter files whose date have been modified by the BBS where you got the YGrep Search Engine (this does not occur when obtaining the original files from the author - when registering). In that case, you should use the **YGrepVersion** function to get the actual version number.

Additionally, if you got the appropriate tool, you can find the version number in the header of the YGREP.DLL file (in the library name field). But this may lead to some imprecision as the minor version number is not included in this message.

## In some cases, AGrep (or RGrep) will never return when called.

First, you may want to look at Note 007 and particularly to notice that you should compile your Windows application for the Large memory only.

In some cases, I observed that users where expecting a result from the calls in too short a time. It's rather uncommon, but AGrep (more than RGrep) can take some time in handling some very long patterns on the slowest machines.

Well! You may have found a bug, of course. Look for your registration number and reach me, please.

If you did not register, here comes the registration form and my address. Registration information

## I am not able to find explanations about how to use the AGrepInit function

That is normal. Even if this function is declared in the YGREP.H header file, it should not be used. It is reserved for debugging purposes (debugging of the library, not of your application).

Do not look for it.

## I cannot compile with the Small or Compact memory model

## I keep on getting stack overflows from the application I build

The problem is simply that these memory models only allow a single data segment of 64 kbytes. The problem is that the AGREPINFO is a very large data structure. Allocating from the stack or from the static data is a very large stress for such a small space. However, if you allocate it dynamically (with malloc() or calloc() as in the sample code provided) you may - at least - be able to know that allocation failed.

In some cases, this problem appears as a "stack overflow" at init time for your application (when the application allocates space for the initial stack).

## I do not find all the matches in the line. There are 12 of them and the returned value is less than that

If you check their description pages, RGrep and AGrep are limited to a certain number of matches. It depends on how many matches can be "documented" in the TagStart/TagEnd fields.

RGrep can only indicate one match (or none). AGrep can only indicate up to **MAXTAG** matches (or none). If the searched line contains more matches than that, they will come undetected and you will have to advance the search pointer in the line.

The sweeping search example in the reference manual is doing just that.

## I work with FoxPro (or another programming language) which cannot call the YGrep Search Engine functions

Look at the scalar-parameter-only functions [they have the same name as the original functions - **SAGrep()** for **AGrep()** ]. The main difference is that the scalar-parameter-only functions do use an internal **XGREPINFO** data structures and, therefore, do not need more than null-terminated strings and integers.

## I work with Delphi 2.0 which is said to be compatible with the YGrep Search Engine. How do I use it?

Nothing is more simple. Simply place 'YGrepInit' in the USES clause of your program and call the functions as they are defined in the YGREPINIT.PAS file.

See YGREPTEST.PAS for some sample code.

## The things that should be done when evaluating the YGrep Search Engine

Here are a few things that evalutors of the package should be well inspired of doing first.

- Put the DLL in the C:\WINDOWS or the C:\WINDOWS\SYSTEM directory

- First try with RGrep rather than AGrep (AGrep seems to be a little more difficult to handle for most people, and switching later from the RGrep family of functions to the AGrep family is a very simple switch where you're used to RGrep).

- Do not forget to call AGrepInit/AGrepEmpty (the single most common thing people forget about the AGrep family of functions...)

- Do not forget to call allocate memory for the data structures

## The things that should be done when evaluating the YGrep Search Engine

Note 19

## The things that should be done when evaluating the YGrep Search Engine

Note 20

## The things that should be done when evaluating the YGrep Search Engine

Note 21

## Alphabetical list of functions

AddWordChar

AGrep

AGrepEmpty

AGrepInit

AGrepSubsBuild (please prefer the following function)

AGrepSubstitute

CompileAGrep

CompileRGrep

CompileSGrep

InitWordCharTable

PushYInfo

RemoveWordChar

RGrep

RGrepSubsBuild (please prefer the following function)

RGrepSubstitute

SAGrep

SAGrepEmpty

SAGrepSubsBuild (please prefer the following function)

SAGrepSubstitute

SCompileAGrep

SCompileRGrep

SCompileSGrep

SFileAGrep

SFileClose

SFileOpen

SFileRGrep

SFileSetFlags

SFileSGrep

SGrep

SRGrep

SRGrepSubsBuild (please prefer the following function)

SRGrepSubstitute

SSGrep

SSGrepSubsBuild (please prefer the following function)

SSGrepSubstitute

YGrep

YGrepGetError

YGrepMessages

YGrepOptions

YGrepResetError

YGrepSetError

YGrepVersion

## Function groups

## Initialization functions

Functions used to initialize the YGrep Search Engine:

The initialization of the Dynamic Link Library (DLL) is done by the internal **LibMain** function and needs no specific documentation.

Functions used to initialize some ot the behaviour of the AGrep group of functions:

AGrepInit

Functions used to un-initialize (to finalize) some ot the behaviour of the AGrep group of functions:

AGrepEmpty

Functions used to initialize some ot the behaviour of the RGrep group of functions:

AddWordChar

InitWordCharTable

RemoveWordChar

Other functions:

YGrepMessages

## File-based functions

SFileAGrep

SFileClose

SFileOpen

SFileRGrep

SFileSGrep

SFileSetFlags

## Approximative Search functions

AGrep

AGrepEmpty

AGrepInit

AGrepSubsBuild (please prefer the following function)

AGrepSubstitute

CompileAGrep

SFileAGrep

## Regular Expression Search functions

CompileRGrep

RGrep

RGrepSubsBuild (please prefer the following function)

RGrepSubstitute

SFileRGrep

## Soundex Search functions

AGrep

AGrepEmpty

AGrepInit

AGrepSubsBuild (please prefer the following function)

AGrepSubstitute

CompileAGrep

SFileAGrep

## Substitution functions

AGrepSubsBuild (please prefer the following function)

AGrepSubstitute

RGrepSubsBuild (please prefer the following function)

RGrepSubstitute

SGrepSubsBuild (please prefer the following function)

SGrepSubstitute

## Scalar functions or functions with only scalar parameters

Some of the functions of the YGrep Search Engine are using a pointer on a complex structure to move around some important internal data between the different functions (AGREPINFO, RGREPINFO or SGREPINFO).

While being very flexible and memory efficient, these functions are difficult to call in some cases with development tools like FoxPro or Visual Basic. In order to reduce the burden on these tools, there is a set of equivalent functions which use an implicit (and therefore not user visible) parameter. These functions have exactly the same operation as there equivalent with extended parameters, a similar name (the scalar functions have a name with a prepended **S** - SRGrep is the scalar equivalent of RGrep).

Here is the list of these equivalent functions:

SAGrep

SAGrepEmpty

SAGrepInit

SAGrepSubsBuild (please prefer the following function)

SAGrepSubstitute

SCompileAGrep

SCompileRGrep

SRGrep

SRGrepSubsBuild (please prefer the following function)

SRGrepSubstitute

SCompileSGrep

SSGrep

SSGrepSubsBuild (please prefer the following function)

SSGrepSubstitute

SFileAGrep

SFileClose

SFileOpen

SFileRGrep

SFileSGrep

SFileSetFlags

## Other functions

PushYInfo

YGrep

YGrepGetError

YGrepMessages

YGrepOptions

YGrepResetError

YGrepSetError

YGrepVersion

Functions with only scalar parameters

Functions for handling files

## Structures

AGREPINFO structure

RGREPINFO structure

SGREPINFO structure

YGREPINFO structure

LPAGREPINFO type

LPRGREPINFO type

LPSGREPINFO type

LPYGREPINFO type

If you don't like twiddling with these types of parameters, you should have a look at the functions with only scalar parameters. They should simplify part of your work at the only expense of generality.

## AGREPINFO

```
typedef   struct   tagAGrepInfo   {          /* agi */
          int      iErrorCode;
          char     cPat[WORD_SIZE];
          LPSTR    TagStart[MAXTAG];
          LPSTR    TagEnd[MAXTAG];
          int      bMatchCase;
          int      bMessages;
          BLIST    uMask;
          BLIST    uOvMask;
          BLIST    uLimit;
          BLIST    uTable[MAXSYM];
          int      iBitsPerState;
          int      iWordSize;
          int      iType;
          char     cUPat[WORD_SIZE];
} AGREPINFO;
```

The AGREPINFO data structure is used between the various functions of the AGrep family (as opposed to RGrep family). Most of its fields are for internal use only and should not be modified.

| Parameter | Description |
|---|---|
| iErrorCode | See the appendix for possible values and the functions descriptions for meaning and circumstances of occurrence. In short, **CompileAGrep** returns a value which is copied here, **AGrep** returns a formal value indicating an error whose code must be found in this field.<br>Most functions verify that this field is set to AGERR_NO_ERROR before starting operation. |
| cPat | The original pattern provided to the **CompileAGrep** function. It is kept here unchanged and can be used by the user as secondary storage. |
| TagStart[0] | First character of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **AGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| TagEnd[0] | First character following end of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **AGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| bMatchCase | Internal only. Contains either **TRUE** or **FALSE** according to the value of this field in the call to **CompileRGrep**. |
| bMessage | In v5.0 and higher, contains either **TRUE** (normal messages) or **FALSE** (no messages, no message box). |
| uMask | Internal only. |
| uOvMask | Internal only. |
| uLimit | Internal only. |
| blState | Internal only. |
| blOverflow | Internal only. |
| blTemp | Internal only. |
| uTable | Internal only. |
| iBitsPerState | Internal only. |
| iWordSize | Internal only. Contains the actual word size of the BLIST structures used elsewhere in the **AGREPINFO** data structure. |
| iType | Internal only. |
| cUPat | Internal only. May temporarily contain a copy of cPat with all letters forced to upper case. |

**Comments**

The fields TagStart and TagEnd can be used in order to sweep from match to match in a single piece of

text (e.g. the whole text of a buffer). After compiling an expression with **CompileAGrep**, a call to **AGrep(**StringStart, pGI**)** will show the first match at pGI->TagStart[0] (with a length of pGI->TagEnd[0] - pGI->TagStart[0] ). Successive calls to **AGrep(**pGI->TagEnd[0], pGI**)** will allow to restart the search operation from the first character after each string match.

## LPAGREPINFO

typedef AGREPINFO FAR* LPAGREPINFO;

The **LPAGREPINFO** type is defined to provided a portable pointer to the **AGREPINFO** structure.

## RGREPINFO

```
typedef   struct   tagRGrepInfo   {          /* rgi */
          int      iErrorCode;
          char     cPat[WORD_SIZE];
          LPSTR    TagStart[MAXTAG];
          LPSTR    TagEnd[MAXTAG];
          int      bMatchCase;
          int      bMessages;
          int      iCircf;
          char     cDFA[MAXDFA];
} RGREPINFO;
```

The **RGREPINFO** data structure is used between the various functions of the RGrep family (as opposed to AGrep family). Some of its fields are for internal use only and should not be modified.

| **Parameter** | **Description** |
|---|---|
| iErrorCode | See the appendix for possible values and the functions descriptions for meaning and circumstances of occurrence. In short, **CompileRGrep** returns a value which is copied here, **RGrep** returns a formal value indicating an error whose code must be found in this field.<br>Most functions verify that this field is set to AGERR_NO_ERROR before starting operation. |
| cPat | The original uncompiled pattern provided to the **CompileRGrep** function. It is kept here unchanged and can be used by the user as secondary storage. |
| TagStart[0] | First character of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **RGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| TagEnd[0] | First character following end of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **RGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| TagStart[n] | First character of tagged string. See the description of regular expressions for substitution in order to have a description (and examples) of use of tags in patterns. |
| TagEnd[n] | First character following end of tagged string. See description of the regular expressions for substitution in order to have a description (and examples) of use of tags in patterns. |
| bMatchCase | Internal only. Contains either **TRUE** or **FALSE** according to the value of this field in the call to **CompileRGrep**. |
| bMessage | In v5.0 and higher, contains either **TRUE** (normal messages) or **FALSE** (no messages, no message box). |
| iCircf | Internal only. |
| cDFA | Internal only. This is the internal optimized search automaton. |

**Comments**

The fields TagStart and TagEnd can be used in order to sweep from match to match in a single piece of text (e.g. the whole text of a buffer). After compiling an expression with **CompileRGrep**, a call to **RGrep(**StringStart, pGI**)** will show the first match at pGI->TagStart[0] (with a length of pGI->TagEnd[0] - pGI->TagStart[0] ). Successive calls to **RGrep(**pGI->TagEnd[0], pGI**)** will allow to restart the search operation from the first character after each string match.

Please notice that this behavior is similar but not identical to the AGrep family of functions because the TagStart/TagEnd fields are mainly used (here, in the RGrep family) as storage for sub-expressions rather than as position of matches.

## LPRGREPINFO

typedef RGREPINFO FAR* LPRGREPINFO;

The **LPRGREPINFO** type is defined to provided a portable pointer to the **RGREPINFO** structure.

## SGREPINFO

```
typedef   struct   tagSGrepInfo   {          /* rgi */
          int      iErrorCode;
          char     cPat[WORD_SIZE];
          LPSTR    TagStart[MAXTAG];
          LPSTR    TagEnd[MAXTAG];
          int      bMatchCase;
          int      bMessages;
          char     cDFA[MAXDFA];
} SGREPINFO;
```

The **SGREPINFO** data structure is used between the various functions of the SGrep family (as opposed to AGrep family). Some of its fields are for internal use only and should not be modified.

| **Parameter** | **Description** |
|---|---|
| iErrorCode | See the appendix for possible values and the functions descriptions for meaning and circumstances of occurrence. In short, **CompileSGrep** returns a value which is copied here, **SGrep** returns a formal value indicating an error whose code must be found in this field.<br>Most functions verify that this field is set to AGERR_NO_ERROR before starting operation. |
| cPat | The original uncompiled pattern provided to the **CompileSGrep** function. It is kept here unchanged and can be used by the user as secondary storage. |
| TagStart[0] | First character of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **SGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| TagEnd[0] | First character following end of matched string. Only the first value of the table should be used. It is describing the position of the matched string in the text after call to **SGrep**. The additional locations are there for physical compatibility with the **RGREPINFO** data structure. |
| bMatchCase | Internal only. Contains either **TRUE** or **FALSE** according to the value of this field in the call to **CompileSGrep**. |
| bMessage | IContains either **TRUE** (normal messages) or **FALSE** (no messages, no message box). |
| cDFA | Internal only. This is the internal optimized search automaton. |

**Comments**

The fields TagStart and TagEnd can be used in order to sweep from match to match in a single piece of text (e.g. the whole text of a buffer). After compiling an expression with **CompileSGrep**, a call to **SGrep(**StringStart, pGI**)** will show the first match at pGI->TagStart[0] (with a length of pGI->TagEnd[0] - pGI->TagStart[0] ). Successive calls to **SGrep(**pGI->TagEnd[0], pGI**)** will allow to restart the search operation from the first character after each string match.

## LPSGREPINFO

typedef SGREPINFO FAR* LPSGREPINFO;

The **LPSGREPINFO** type is defined to provided a portable pointer to the **SGREPINFO** structure.

## YGREPINFO

```
typedef struct tagYGrepInfo {
        union {
                OPERINFO        opI;
                AGREPINFO       aGI;
                RGREPINFO       rGI;
                SGREPINFO       sGI;
        } info;
        struct tagYGrepInfo FAR*        pInfo;
        int     iTypeOfInfo;    /* One of IT_* values */
} YGREPINFO;

typedef YGREPINFO*          PYGREPINFO;
typedef YGREPINFO FAR*      LPYGREPINFO;

#define IT_NONE     -1
#define IT_OP       0
#define IT_AG       1
#define IT_RG       2
#define IT_SG       3
```

The **YGREPINFO** union is provided as a way to help the user define a common structure for both types of searches.

**Comments**

This union is not currently used by the YGrep Search Engine. However, when a similar union will be needed, **YGREPINFO** will be use. Consequently, you can see it as a premium proposed to the users.

## LPYGREPINFO

typedef YGREPINFO FAR* LPYGREPINFO;

The **LPYGREPINFO** type is defined to provided a portable pointer to the **YGREPINFO** structure.

## CompileAGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL CompileAGrep(***LPCSTR lpText, UINT k, BOOL bMatchCase, LPAGREPINFO pGI***)**

**int YGCALL SCompileAGrep(***LPCSTR lpText, UINT k, BOOL bMatchCase***)**

```
LPCSTR   lpPattern;     /* pattern string to look for          */
UINT     k;             /* number of errors                    */
BOOL     bMatchCase;    /* Match case in comparisons?          */
LPAGREPINFO  pGI;       /* pointer to search information block */
```

The **CompileAGrep** function processes the pattern in order to prepare approximative search.

| Parameter | Description |
|---|---|
| lpPattern | Specifies the pattern to look for. |
| k | Specifies the number of errors for approximative match. |
| bMatchCase | Specifies whether the search operation should be case sensitive. |

| Value | Meaning |
|---|---|
| **TRUE** | Force letter case checking |
| **FALSE** | Do not check letter casing |

| | |
|---|---|
| bMatchCase | Specifies whether the search operation should be case sensitive. |
| pGI | Pointer to an information block as will be used in a later to the **AGrep** function. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

Following are the possible returned values for **CompileAGrep**:

| Value | Meaning |
|---|---|
| AGERR_UNKNOWN_TYPE | YGrep Search Engine internal error (no available information on its origin).<br>This normally results from semi-automatic checks. This error should be expected but should trigger a default action like exiting the application. |
| AGERR_NO_PATTERN | A pattern was expected and not found as argument. |
| AGERR_TOO_LONG | AGrep expression is too complex to handle in the internal structures of the YGrep Search Engine. |
| AGERR_ALLOC_MEM | Not enough memory to build internal structures of the YGrep Search Engine. |
| AGERR_STATE | Reserved for future use. |

**Comments**

When setting the number of errors to **0**, the returned value is always AGERR_NO_ERROR (there can be no error).

This function must be called at least once before calling **AGrep**.

The user is advised that trying to search for a short pattern with a large number of errors may be useless (if more errors are allowed than there are characters in the pattern, the match will be trivial, and trivially detected in **AGrep**).

**See Also**

**AGrep**

## AGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL AGrep(***LPCSTR lpText, LPAGREPINFO pGI***)**

**int YGCALL SAGrep(***LPCSTR lpText***)**

```
LPCSTR    lpText;     /* text string to explore                */
LPAGREPINFO  pGI;     /* pointer to search information block    */
```

The **AGrep** function execute the approximative search with the Shift-Or method.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string to be explored (where to search for the pattern) |
| pGI | Pointer to an information block as built by a previous **CompileAGrep** call. |

**Returns**

The return value is the number of matches encountered in the explored text string.

If there is no match, the return value is **0**.

In case of error, the return value is negative.

When there is one or more matches, the **AGREPINFO** structure is filled with data describing the match(es). In particular, the user can use the **TagStart[ ]** and **TagEnd[ ]** fields.

If there are more than **MAXTAG** matches, the returned value is **MAXTAG**. This is caused by the size limitation of the **TagStart[ ]** and **TagEnd[ ]** fields. When this situation occurs, you should move to the last tagged position in order to search again for possibly remaining matches.

**Comments**

Following are the possible error values for **AGrep** when the returned value is negative.

| Value | Meaning |
|---|---|
| AGERR_ALLOC_MEM | Not enough memory to build internal structures of the YGrep Search Engine. |

Even though the structure of the **AGREPINFO** block is available, the programmer is advised not to try filling it with information without calling the **CompileAGrep** function.

If the number of matches is different from **0**, it is possible to find the position of the first occurence in the pGI structure. The first matching character is pointed by **pGI->TagStart[0]** and the first non-matching character is pointed by **pGI->TagEnd[0]**.

If the **AGREPINFO** structure contains an **iErrorCode** field with an error, the **AGrep** function will not execute anything and will return a negative value, leaving the **AGREPINFO** fields unchanged.

**See Also**

**CompileAGrep**

## AGrepInit

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL AGrepInit(***LPAGREPINFO pGI***)**

**int YGCALL SAGrepInit()**

**LPAGREPINFO** *pGI*;  /* pointer to search information block  */

The **AGrepInit** function should be called before any use of the pGI parameter in any other function of the AGrep family. It is used to initialize internal data structures in this data structure.

| Parameter | Description |
|---|---|
| pGI | Pointer to an information block just allocated and not yet used. |

**Returns**

The returned value is either **TRUE** in case of success, or **FALSE** in case of failure.

**Comments**

This function must be called at least once for each of the **AGREPINFO** structures which will be filled by the **CompileAGrep** function.

This function can will accept a pointer to an information block which must have been previously allocated (possibly with GlobalAlloc() under MS-Windows) and eventually locked in memory (with GlobalLock() under MS-Windows).

**See Also**

**AGrepEmpty**

## AGrepEmpty

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL AGrepEmpty(***LPAGREPINFO pGI***)**

**int YGCALL SAGrepEmpty()**

**LPAGREPINFO** *pGI*;  /* pointer to search information block  */

The **AGrepEmpty** function is used to clear the contents of the **AGREPINFO** structure before releasing memory.

| Parameter | Description |
|---|---|
| pGI | Pointer to an information block as built by the **CompileAGrep** function. |

**Returns**

The returned value is either **TRUE** in case of success, or **FALSE** in case of failure.

This function must be called at least once for each of the **AGREPINFO** structures filled by the **CompileAGrep** function. If not, when releasing memory for the **AGREPINFO** block, its contents are not cleared (mainly pointers in the **BLIST** fields) and memory leak occurs. The consequence is then a slowdown of Windows while your application consumes more and more memory, and in the end, out-of-memory condition for your application or one of its neighbors.

While programming with the YGrep Search Engine, it must be remembered that internal structures for that Dynamic Link Library are rather large and memory handling is an important part of any MS-Windows application.

**See Also**

**AGrepInit**

## AGrepSubsBuild

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL AGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize, LPAGREPINFO pGI***)**

**int YGCALL SAGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR    lpPattern;      /* pattern to replace matched strings                */
LPCSTR    lpDest;         /* destination buffer for building substitution string */
int       iSize;          /* size of the destination buffer                    */
LPAGREPINFO    pGI;       /* pointer to search information block               */
```

The **AGrepSubsBuild** function builds the replacement string for the previous match by **AGrep** based on the pattern argument. It does not operate the replacement in the original string (read comments at the end of this reference page).

To get a full substitution, you should use **AGrepSubstitute**.

| Parameter | Description |
|---|---|
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **AGrep**. |
| lpDest | Specifies the buffer which will receive the substitution string built from the pattern and the matched string. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileAGrep** call and used by a previous **AGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **AGrepSubsBuild**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **AGrep** was not called before, or **AGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **AGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **AGREPINFO** structure. |

Before calling this function, you must successively use the **CompileAGrep** (to initialize the *pGI* parameter) and **AGrep** (to perform the search operation preliminary to substituting a string to the match).

After calling **AGrepSubsBuild**, you are left with a "destination string" which contains the text to insert back into the original string. The insertion is not done by **AGrepSubsBuild**, because it could involve a large amount of memory management that the programmer/user could prefer doing by himself following the rules he need for his application. If you prefer allowing the YGrep Search Engine doing it by default, you should look for the **AGrepSubstitute** function.

For example, **CompileAGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text without any error), **AGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **AGrepSubsBuild** with the pattern "large &". It will return an *lpDest* string containing "large horse" you can use to substitute in the original text line (**AGrepSubsBuild** does not apply the actual substitution). You can then call again **AGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the *lpDest* string (here it grows from the 7 characters pattern - "large &" - to the final 11 characters *lpDest* - "large horse"). The user must provide a buffer large enough for building it.

**See Also**

**CompileAGrep**, **AGrep**, **AGrepSubstitute**

## AGrepSubstitute

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL AGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize, LPAGREPINFO pGI***)**

**int YGCALL SAGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR    lpText;       /* text previously search with AGrep                    */
LPCSTR    lpPattern;    /* pattern to replace matched strings                   */
LPCSTR    lpDest;       /* destination buffer for building substitution string */
int       iSize;        /* size of the destination buffer                       */
LPAGREPINFO  pGI;       /* pointer to search information block                  */
```

The **AGrepSubstitute** function substitutes the *lpPattern* in place of the matched string in the *lpText* used in the preceeding and successful search by **AGrep**.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string search with the previous call to **AGrep**. It will be used as the basis for the substitution of the matched portion detected by **AGrep**. |
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **AGrep**. |
| lpDest | Specifies the buffer which will receive the result of the substitution process. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileAGrep** call and used by a previous **AGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **AGrepSubstitute**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **AGrep** was not called before, or **AGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **AGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **AGREPINFO** structure. |

Before calling this function, you must successively use the **CompileAGrep** (to initialize the *pGI* parameter) and **AGrep** (to perform the search operation preliminary to substituting a string to the match).

For example, **CompileAGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text), **AGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **AGrepSubstitute** with the pattern "large &". It will return an *lpDest* string containing " A large horse! My kingdom for a horse! ".

Notice that only the firstmatch is substituted. It is the reponsibility of the programmer to repeat this process in order to cover all cases in a specific text. You can then call again **AGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the lpDest string (here it grows 6 characters). The user must provide a buffer large enough for building it.

**See Also**

**CompileAGrep**, **AGrep**

## CompileRGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL CompileRGrep(***LPCSTR lpText, BOOL bMatchCase, LPRGREPINFO pGI***)**

**int YGCALL SCompileRGrep(***LPCSTR lpText, BOOL bMatchCase***)**

```
LPCSTR   lpPattern;     /* pattern string to look for              */
BOOL     bMatchCase;    /* Match case in comparisons?              */
LPRGREPINFO  pGI;       /* pointer to search information block     */
```

The **CompileRGrep** function processes the pattern in order to prepare regular expression search.

| Parameter | Description |
|---|---|
| lpPattern | Specifies the text string describing the pattern to look for. |
| bMatchCase | Specifies whether the search operation should be case sensitive. |

| Value | Meaning |
|---|---|
| **TRUE** | Force letter case checking |
| **FALSE** | Do not check letter casing |

| | |
|---|---|
| bMatchCase | Specifies whether the search operation should be case sensitive. |
| pGI | Pointer to an information block as will be used in a later to the **RGrep** function. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine Regular Expression**.

Following are the possible returned values for **CompileRGrep**:

| Value | Meaning |
|---|---|
| AGERR_ALLOC_MEM | Insufficient memory to hold data structures for internal operation. |
| AGERR_STATE | Reserved for future use. |
| AGERR_NO_PATTERN | There was no pattern provided. **CompileRGrep** tried to use a previously proposed pattern. But this was the first call to the function. |
| RGERR_MUNGED_AUTO | Munged automaton. Internal error. Should be sign of memory corruption either by an YGrep Search Engine bug or another undetected program. |
| RGERR_MISS_BRACKET | Missing closing bracket ']' in expression. |
| RGERR_EMPTY_CL | Empty closure. Do not provide an expression starting with either **-**, **+** or ***** (closures at the beginning of an expression, or empty closures) |
| RGERR_ILLEGAL_CL | Illegal closure. Some characters are not allowed before a closure: **^$<>** cannot be followed by either **-**, **+** or ***** |
| RGERR_TOO_MANY_PAR | Too many parenthesis pairs in the expression. |
| RGERR_NULL_IN_PAR | Null expression inside parenthesis. |
| RGERR_UNMATCHED | Unmatched parenthesis. There is at least one more closing parenthesis than opening ones. |
| RGERR_NULL_IN_CRO | Null expression inside < >. |

| | |
|---|---|
| RGERR_CYCLICAL_REF | A reference is done do itself. |
| RGERR_UNDETERM_REF | A reference is done to an unknown sub-expression. |
| RGERR_UMATCHED_PAR | Unmatched parenthesis. There is at least one less closing parenthesis than opening ones. |

This function must be called at least once before calling **RGrep**.

**See Also**

**RGrep**

## RGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL RGrep(***LPCSTR lpText, LPRGREPINFO pGI***)**

**int YGCALL SRGrep(***LPCSTR lpText***)**

```
LPCSTR   lpText;     /* text string to explore                */
LPRGREPINFO  pGI;    /* pointer to search information block    */
```

The **RGrep** function execute the automaton-oriented search with regular expressions compatible with the Unix **ed(1)** editor.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string to be explored (where to search for the pattern) |
| pGI | Pointer to an information block as built by a previous **CompileRGrep** call. |

**Returns**

The return value is the number of matches encountered in the explored text string.

If there is no match, the return value is **0**.

In case of error, the return value is negative. The error code can be found in pGI (iErrorCode structure field).

If the number of matches is different from **0**, it is possible to find the position of the first occurence in the pGI structure. The first matching character is pointed by **pGI->TagStart[0]** and the first non-matching character is pointed by **pGI->TagEnd[0]**.

Because there can be no more than one match position stored in the **pGI->TagStart[0]** field, the returned value is never superior to 1 (only the first match is found, and additional calls to **RGrep** must be used to find the following ones).

**Comments**

Following are the possible error values for **RGrep** when the returned value is negative.

| Value | Meaning |
|---|---|
| RGERR_MUNGED_AUTO | The pGI structure contents have been modified, or never initialized by **CompileRGrep**.. |

Even though the structure of the **RGREPINFO** block is available, the programmer is advised not to try filling it with information without calling the **CompileRGrep** function.

If the **RGREPINFO** structure contains an **iErrorCode** field with an error, the **RGrep** function will not execute anything and will return a negative value, leaving the **RGREPINFO** fields unchanged.

**See Also**

**CompileRGrep**

## RGrepSubsBuild

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL RGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize, LPRGREPINFO pGI***)**

**int YGCALL SRGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR    lpPattern;      /* pattern to replace matched strings                */
LPCSTR    lpDest;         /* destination buffer for building substitution string */
int       iSize;          /* size of the destination buffer                    */
LPRGREPINFO  pGI;         /* pointer to search information block               */
```

The **RGrepSubsBuild** function builds the replacement string for the previous match by **RGrep** based on the pattern argument. It does not operate the replacement in the original string (read comments at the end of this reference page).

| Parameter | Description |
|---|---|
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **RGrep**. |
| lpDest | Specifies the buffer which will receive the substitution string built from the pattern and the matched string. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileRGrep** call and used by a previous **RGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **RGrepSubsBuild**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **RGrep** was not called before, or **RGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **RGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **RGREPINFO** structure. |

Before calling this function, you must successively use the **CompileRGrep** (to initialize the *pGI* parameter) and **RGrep** (to perform the search operation preliminary to substituting a string to the match).

After calling **RGrepSubsBuild**, you are left with a "destination string" which contains the text to insert back into the original string. The insertion is not done by the YGrep Search Engine, because it could involve a large amount of memory management that the programmer/user could prefer doing by himself following the rules he need for his application. The YGrep Search Engine could not follow these rules.

For example, **CompileRGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text), **RGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **RGrepSubsBuild**

with the pattern "large &". It will return an lpDest string containing "large horse" which you can use to substitute in the original text line (**RGrepSubsBuild** does not apply the actual substitution). You can then call again **RGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the lpDest string (here it grows from the 7 characters pattern - "large &" - to the final 11 characters lpDest - "large horse"). The user must provide a buffer large enough for building it.

**See Also**

**CompileRGrep**, **RGrep**

## RGrepSubstitute

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL RGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize, LPRGREPINFO pGI***)**

**int YGCALL RGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR    lpText;       /* text previously search with RGrep                  */
LPCSTR    lpPattern;    /* pattern to replace matched strings                 */
LPCSTR    lpDest;       /* destination buffer for building substitution string */
int       iSize;        /* size of the destination buffer                     */
LPRGREPINFO  pGI;       /* pointer to search information block                */
```

The **RGrepSubstitute** function substitutes the *lpPattern* in place of the matched string in the *lpText* used in the preceeding and successful search by **RGrep**.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string search with the previous call to **RGrep**. It will be used as the basis for the substitution of the matched portion detected by **RGrep**. |
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **RGrep**. |
| lpDest | Specifies the buffer which will receive the result of the substitution process. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileRGrep** call and used by a previous **RGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **RGrepSubstitute**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **RGrep** was not called before, or **RGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **RGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **RGREPINFO** structure. |

Before calling this function, you must successively use the **CompileRGrep** (to initialize the *pGI* parameter) and **RGrep** (to perform the search operation preliminary to substituting a string to the match).

For example, **CompileRGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text), **RGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **RGrepSubstitute** with the pattern "large &". It will return an *lpDest* string containing " A large horse! My kingdom for a horse! ".

Notice that only the firstmatch is substituted. It is the reponsibility of the programmer to repeat this process in order to cover all cases in a specific text. You can then call again **RGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the lpDest string (here it grows 6 characters). The user must provide a buffer large enough for building it.

**See Also**

**CompileRGrep**, **RGrep**

## InitWordCharTable

```
#include    <windows.h>
#include    <ygrep.h>
```

**void YGCALL InitWordCharTable()**

The **InitWordCharTable** function initializes the table containing the list of characters considered as word characters by the RGrep group of functions.

It takes no parameter.

**Returns**

No return value.

**Comments**

The initial characters considered as word characters are **0** to **9**, **A** to **Z**, **a** to **z** and the underscore character( **_** ). This list can be modified using the **AddWordChar** and **RemoveWordChar** functions. The mostly probable use of these modifications are to include accentuated characters for foreign language, or to remove the underscore characters which is not usually considered a text character out of the programming languages community.

**AddWordChar**, **RemoveWordChar** and **InitWordCharTable** should never be used while there are compiled **RGREPINFO** data structures around. The impact on the following searches using these **RGREPINFO** data structures is then unpredictable and depending on the version of the YGrep Search Engine you are using. After using one of these functions, you should rebuild all **RGREPINFO** data structures by calling **CompileRGrep** again before using them.

**See Also**

**AddWordChar**, **RemoveWordChar**.

## AddWordChar

```
#include        <windows.h>
#include        <ygrep.h>
void YGCALL AddWordChar(LPCSTR lpChars)

LPCSTR  lpChars;            /* text string containing the characters to add      */
```

The **AddWordChar** function adds more characters to the list of word characters for the RGrep group of functions.

| Parameter | Description |
|---|---|
| lpChars | Specifies all the characters to be added to the list of word characters |

**Returns**

No return value

**Comments**

The characters in the parameter string can be in any order and can be in the full range of the extended (8-bit) ASCII character set (excluding the null character, of course).

**AddWordChar**, **RemoveWordChar** and **InitWordCharTable** should never be used while there are compiled **RGREPINFO** data structures around. The impact on the following searches using these **RGREPINFO** data structures is then unpredictable and depending on the version of the YGrep Search Engine you are using. After using one of these functions, you should rebuild all **RGREPINFO** data structures by calling **CompileRGrep** again before using them.

**See Also**

**InitWordCharTable**, **RemoveWordChar**.

## RemoveWordChar

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL RemoveWordChar(***LPCSTR lpChars***)**

**LPCSTR** *lpChars*;  /* text string of characters to remove */

The **RemoveWordChar** function adds more characters to the list of word characters for the RGrep group of functions.

| Parameter | Description |
|---|---|
| lpChars | Specifies all the characters to be removed from the list of word characters |

**Returns**

No return value.

**Comments**

The characters in the parameter string can be in any order and can be in the full range of the extended (8-bit) ASCII character set (excluding the null character, of course).

**AddWordChar**, **RemoveWordChar** and **InitWordCharTable** should never be used while there are compiled **RGREPINFO** data structures around. The impact on the following searches using these **RGREPINFO** data structures is then unpredictable and depending on the version of the YGrep Search Engine you are using. After using one of these functions, you should rebuild all **RGREPINFO** data structures by calling **CompileRGrep** again before using them.

**See Also**

**InitWordCharTable**, **AddWordChar**

## CompileSGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL CompileSGrep(***LPCSTR lpText, BOOL bMatchCase, LPSGREPINFO pGI***)**

**int YGCALL SCompileSGrep(***LPCSTR lpText, BOOL bMatchCase***)**

```
LPCSTR    lpPattern;      /* pattern string to look for          */
BOOL      bMatchCase;     /* Match case in comparisons?          */
LPSGREPINFO  pGI;         /* pointer to search information block  */
```

The **CompileSGrep** function processes the pattern in order to prepare soundex search.

| Parameter | Description |
|---|---|
| lpPattern | Specifies the pattern to look for. |
| bMatchCase | not used. |
| pGI | Pointer to an information block as will be used in a later to the **SGrep** function. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

Following are the possible returned values for **CompileSGrep**:

| Value | Meaning |
|---|---|
| AGERR_UNKNOWN_TYPE | YGrep Search Engine internal error (no available information on its origin).<br>This normally results from semi-automatic checks. This error should be expected but should trigger a default action like exiting the application. |
| AGERR_NO_PATTERN | A pattern was expected and not found as argument. |
| AGERR_ALLOC_MEM | Not enough memory to build internal structures of the YGrep Search Engine. |
| AGERR_STATE | Reserved for future use. |

**Comments**

This function must be called at least once before calling **SGrep**.

**See Also**

**SGrep**

## SGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL SGrep(***LPCSTR lpText, LPSGREPINFO pGI***)**

**int YGCALL SSGrep(***LPCSTR lpText***)**

```
LPCSTR    lpText;     /* text string to explore                */
LPSGREPINFO  pGI;     /* pointer to search information block    */
```

The **SGrep** function execute searches based upon the soundex method.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string to be explored (where to search for the pattern) |
| pGI | Pointer to an information block as built by a previous **CompileSGrep** call. |

**Returns**

The return value is the number of matches encountered in the explored text string.

If there is no match, the return value is **0**.

In case of error, the return value is negative.

When there is one or more matches, the **SGREPINFO** structure is filled with data describing the match(es). In particular, the user can use the **TagStart[ ]** and **TagEnd[ ]** fields.

If there are more than **MAXTAG** matches, the returned value is **MAXTAG**. This is caused by the size limitation of the **TagStart[ ]** and **TagEnd[ ]** fields. When this situation occurs, you should move to the last tagged position in order to search again for possibly remaining matches.

**Comments**

Following are the possible error values for **SGrep** when the returned value is negative.

| Value | Meaning |
|---|---|
| AGERR_ALLOC_MEM | Not enough memory to build internal structures of the YGrep Search Engine. |
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **SGrep** was not called before, or **SGrep** was called but did not return success. |

Even though the structure of the **SGREPINFO** block is available, the programmer is advised not to try filling it with information without calling the **CompileSGrep** function.

If the number of matches is different from **0**, it is possible to find the position of the first occurence in the pGI structure. The first matching character is pointed by **pGI->TagStart[0]** and the first non-matching character is pointed by **pGI->TagEnd[0]**.

If the **SGREPINFO** structure contains an **iErrorCode** field with an error, the **SGrep** function will not execute anything and will return a negative value, leaving the **SGREPINFO** fields unchanged.

**See Also**

**CompileSGrep**

## SGrepSubsBuild

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL SGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize, LPSGREPINFO pGI***)**

**int YGCALL SSGrepSubsBuild(***LPCSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR   lpPattern;     /* pattern to replace matched strings              */
LPCSTR   lpDest;        /* destination buffer for building substitution string */
int      iSize;         /* size of the destination buffer                  */
LPSGREPINFO   pGI;      /* pointer to search information block             */
```

The **SGrepSubsBuild** function builds the replacement string for the previous match by **SGrep** based on the pattern argument. It does not operate the replacement in the original string (read comments at the end of this reference page).

To get a full substitution, you should use **SGrepSubstitute**.

| Parameter | Description |
|---|---|
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **SGrep**. |
| lpDest | Specifies the buffer which will receive the substitution string built from the pattern and the matched string. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileSGrep** call and used by a previous **SGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **SGrepSubsBuild**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **SGrep** was not called before, or **SGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **SGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **SGREPINFO** structure. |

Before calling this function, you must successively use the **CompileSGrep** (to initialize the *pGI* parameter) and **SGrep** (to perform the search operation preliminary to substituting a string to the match).

After calling **SGrepSubsBuild**, you are left with a "destination string" which contains the text to insert back into the original string. The insertion is not done by **SGrepSubsBuild**, because it could involve a large amount of memory management that the programmer/user could prefer doing by himself following the rules he need for his application. If you prefer allowing the YGrep Search Engine doing it by default, you should look for the **SGrepSubstitute** function.

For example, **CompileSGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text), **SGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **SGrepSubsBuild** with the pattern "large &". It will return an *lpDest* string containing "large horse" which you can use to substitute in the original text line (**SGrepSubsBuild** does not apply the actual substitution). You can then call again **SGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the *lpDest* string (here it grows from the 7 characters pattern - "large &" - to the final 11 characters *lpDest* - "large horse"). The user must provide a buffer large enough for building it.

**See Also**

**CompileSGrep**, **SGrep**, **SGrepSubstitute**

## SGrepSubstitute

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL SGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize, LPSGREPINFO pGI***)**

**int YGCALL SSGrepSubstitute(***LPSTR lpText, LPSTR lpPattern, LPCSTR lpDest, int iSize***)**

```
LPCSTR    lpText;       /* text previously search with SGrep                    */
LPCSTR    lpPattern;    /* pattern to replace matched strings                   */
LPCSTR    lpDest;       /* destination buffer for building substitution string */
int       iSize;        /* size of the destination buffer                       */
LPSGREPINFO  pGI;       /* pointer to search information block                  */
```

The **SGrepSubstitute** function substitutes the *lpPattern* in place of the matched string in the *lpText* used in the preceeding and successful search by **SGrep**.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string search with the previous call to **SGrep**. It will be used as the basis for the substitution of the matched portion detected by **SGrep**. |
| lpPattern | Specifies the replacement string to substitute for the previous match detected by **SGrep**. |
| lpDest | Specifies the buffer which will receive the result of the substitution process. |
| iSize | Size of the lpDest buffer. |
| pGI | Pointer to an information block as built by a previous **CompileSGrep** call and used by a previous **SGrep** call. |

**Returns**

The returned value is one of the AGERR_* error codes. In case of normal operation (no error), the returned value is AGERR_NO_ERROR.

**Comments**

The pattern uses a specific syntax to describe regular expressions. It is described under the title of **YGrep Search Engine substitution expressions**.

Following are the possible returned values for **SGrepSubstitute**:

| Value | Meaning |
|---|---|
| AGERR_NO_PREVIOUS | There was no previous pattern searched, or **SGrep** was not called before, or **SGrep** was called but did not return success. |
| AGERR_ NO_PATTERN | There was no pattern provided for substitution. |
| AGERR_TOO_SHORT | The substitution is building a destination string which is too large for the lpDest buffer as sized by the iSize argument. |
| AGERR_STATE | Can occur when badly constructed **SGREPINFO** is forwarded as parameter to this function. Most usually, it comes from forgetting to call the previous functions, or from erroneous **SGREPINFO** structure. |

Before calling this function, you must successively use the **CompileSGrep** (to initialize the *pGI* parameter) and **SGrep** (to perform the search operation preliminary to substituting a string to the match).

For example, **CompileSGrep** is used on the pattern "horse" (for the sake of simplicity we have choosen straight text), **SGrep** is used on the text line "A horse! My kingdom for a horse!". Match is observed on the third character (beginning of the first "horse" word). Then, for substitution you can call **SGrepSubstitute** with the pattern "large &". It will return an *lpDest* string containing " A large horse! My kingdom for a horse! ".

Notice that only the firstmatch is substituted. It is the reponsibility of the programmer to repeat this process in order to cover all cases in a specific text. You can then call again **SGrep** before substituting again.

The necessity of providing a size limit appears in this example since it is difficult to predict the final size of the lpDest string (here it grows 6 characters). The user must provide a buffer large enough for building it.

**See Also**

**CompileSGrep**, **SGrep**

## YGrepVersion

```
#include    <windows.h>
#include    <ygrep.h>
```

**WORD YGrepVersion()**

The **YGrepVersion** function provides the version number of the DLL.

| Parameter | Description |
|---|---|
| none | |

**Returns**

The return **WORD** value has the following format (when represented as an hexadecimal value):

Vrrr

Where V is the version number (major) and rrr is the release number (minor). For example, version 1.20d is coded as 0x1204 and version 4.06 is coded as 0x4060.

This value may be used to determine the capabilities/compatiblity of an already loaded version of the YGrep Dynamic Link Library and to insure that it is able to answer to specific calls.

**Comments**

The application may never call this function. But it can be used to check at run time the availability of certain functions in the Dynamic Link Library.

## SCompileAGrep

The **SCompileAGrep** function is the wrapper with only scalar parameters for the **CompileAGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SAGrep

The **SAGrep** function is the wrapper with only scalar parameters for the **AGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SAGrepInit

The **SAGrepInit** function is the wrapper with only scalar parameters for the **AGrepInit** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SAGrepEmpty

The **SAGrepEmpty** function is the wrapper with only scalar parameters for the **AGrepEmpty** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SAGrepSubsBuild

The **SAGrepSubsBuild** function is the wrapper with only scalar parameters for the **AGrepSubsBuild** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SAGrepSubstitute

The **SAGrepSubstitute** function is the wrapper with only scalar parameters for the **AGrepSubstitute** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SCompileRGrep

The **SCompileRGrep** function is the wrapper with only scalar parameters for the **CompileRGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SRGrep

The **SRGrep** function is the wrapper with only scalar parameters for the **RGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SRGrepSubsBuild

The **SRGrepSubsBuild** function is the wrapper with only scalar parameters for the **RGrepSubsBuild** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SRGrepSubstitute

The **SRGrepSubstitute** function is the wrapper with only scalar parameters for the **RGrepSubstitute** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SCompileSGrep

The **SCompileSGrep** function is the wrapper with only scalar parameters for the **CompileSGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SSGrep

The **SSGrep** function is the wrapper with only scalar parameters for the **SGrep** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SSGrepSubsBuild

The **SSGrepSubsBuild** function is the wrapper with only scalar parameters for the **SGrepSubsBuild** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## SSGrepSubstitute

The **SSGrepSubstitute** function is the wrapper with only scalar parameters for the **SGrepSubstitute** function.

It has the same operation but using a single internal (not visible *pGI* parameter).

## PushYInfo

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL PushYInfo(***LPYGREPINFO pGI***)**

**LPYGREPINFO** *pGI*;  /* pointer to search information block  */

The **PushYInfo** function allows to push onto the internal stack the search information block or the search boolean operator for later exploitation by the **YGrep** function.

| Parameter | Description |
|---|---|
| pGI | Pointer to an information block already prepared by either **CompileAGrep** or **CompileRGrep** function. |

**Returns**

The returned value is always AGERR_NO_ERROR.

**Comments**

Before building a list of commands, **PushYInfo** should always be called with a NULL (0) parameter to clear the internal data structures. After calling **YGrep**, it is always needed that the first call to **PushYInfo** is this clearing one.

See the description of the **YGrep** function for an explanation of how the Boolean oriented functions work.

**See Also**

**YGrep**

## YGrepSetError

See **YGrepGetError** function.

## YGrepResetError

See **YGrepGetError** function.

## YGrepGetError / YGrepResetError / YGrepSetError

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL YGrepGetError()**

**int YGCALL YGrepResetError(***int iError***)**

**int YGCALL YGrepSetError(***int iError***)**

**int** *iError*;  /* error value to set  */

The **YGrep** function sets its error state in a spearate variable in order to get it clean. The functions **YGrepGetError**, **YGrepResetError** and **YGrepSetError** are used to manipulate this error code.

**YGrepGetError** returns the value of the error code.

**YGrepResetError** sets the value of the error code to iError.

**YGrepSetError** sets the value of the error code to iError (but only if it was previously set to AGERR_NO_ERROR). This behavior allows to keep the first error encountered. This is used by the **YGrep** function.

| Parameter | Description |
|---|---|
| iError | Error value to set (for the functions **YGrepSetError** and **YGrepResetError**). |

**Returns**

The returned value is always the error value that was present before calling the function.

**See Also**

**YGrep**

## YGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**int YGCALL YGrep (***LPCSTR lpText***)**

**LPCSTR** *lpText*;  /* text string to explore */

The **YGrep** function executes the pushed stack of **AGREPINFO** and **RGREPINFO** commandsprepared by **PushYInfo** by appropriately calling the **AGrep** or **RGrep** functions.

| Parameter | Description |
|---|---|
| lpText | Specifies the text string to be explored (where to search for the pattern) |

**Returns**

The returned value is the number of matches matches encountered in the explored text string if there is a single search condition (no operator is used). But if there are several search conditions (in the most general case), the return value is built according to the table of operators (a few paragraphs later here).

If there is no match, the return value is **0**.

In case of error, the return value is negative.

**Comments**

Before starting execution, **YGrep** will call **YGrepResetError(** AGERR_NO_ERROR **)** in order to reset the error value. Following are the possible arror values for **YGrep** when the returned value is negative.

| Error value | Description |
|---|---|
| AGERR_ALLOC_MEM | Not enough memory to build internal structures of the YGrep Search Engine. |
| AGERR_STATE | Field iTypeOfInfo or iOperator was found to contain some value which was not recognized. |

As previously stated, these values can be obtained using a call to **YGrepGetError** and can be modified by the functions **YGrepSetError** and **YGrepResetError**.

You should be aware that the **YGrep** function is based upon a recursive descent analysis of the stacked commands. This has the consequence of limiting the capability of this function to what can be pushed onto the actual program stack. If you intend to use a complex boolean expression of searches, you should be prepared to increase stack size for this purpose. Anyway, in most cases, you should not see any difficulty since every possibility to reduce program stack consumption has been applied and this should not be a source of inconvenience.

The possible operators are described here:

| Value | Returned value |
|---|---|
| OP_NOP | Always **0**. |
| OP_NOT | The NOT of the matches encountered by the search operation. |
| OP_UCMP | The bit-wise unary complement of the matches encountered by the search operation. |
| OP_BAND | The bit-wise AND of the matches encountered by each of the search operations. |
| OP_BOR | The bit-wise OR of the matches encountered by each of the search operations. |
| OP_BXOR | The bit-wise exclusive-OR of the matches encountered by each of the search operations. |
| OP_BNAND | The bit-wise NAND (NOT AND) of the matches encountered by each of the search operations. |
| OP_BNOR | The bit-wise NOR (NOT OR) of the matches encountered by each of the search |

| | |
|---|---|
| | operations. |
| OP_AND | The AND of the matches encountered by each of the search operations. |
| OP_OR | The OR of the matches encountered by each of the search operations. |
| OP_NAND | The NAND (NOT AND) of the matches encountered by each of the search operations. |
| OP_NOR | The NOR (NOT OR) of the matches encountered by each of the search operations. |
| OP_ADD | The addition of the matches encountered by each of the search operations. |
| OP_SUB | The subtract of the matches encountered by each of the search operations. |
| Erroneous operators | Always returns a negative value. |

**See Also**

**PushYInfo**

## YGrepMessages

```
#include    <windows.h>
#include    <ygrep.h>
```

**void YGrepMessages(***BOOL bMessagesAllowed***)**

**BOOL** *bMessagesAllowed*;  /* are error messages allowed? */

The **YGrepMessages** function is the entry point to allow/prohibit error and warning messages coming out of the YGrep Search Engine in any other form than returned values. It provides a way to remove dialog boxes.

| Parameter | Description |
|---|---|
| bMessagesAllowed | **TRUE** means "messages are allowed",<br>**FALSE** means "no messages, please". |

**Returns**

None.

**Comments**

By default, the YGrep Search Engine displays a number of messages through dialog boxes. This function is a way to remove them. Keeping the default operation of "messages are allowed" is consistent with the need to keep upward compatibility from previous versions of the YGrep Search Engine (up to v4.x).

When no messages are provided by the YGrep Search Engine, the application must provide its own based on the returned values of the functions and the error codes in the data structures.

As this function is redundant with the **YGrepOptions** function, it may disappear in future releases. The users are advised to avoid using **YGrepMessages**.

**See Also**

**YGrepOptions**

## YGrepOptions

```
#include    <windows.h>
#include    <ygrep.h>
```

**void YGrepOptions(***long lOptions***)**

**long** *lOptions*;  /* option flags  */

The **YGrepOptions** function is used to set different behaviour flags that allow to change the standard, basic, or even historic only behaviour of the YGrep Search Engine.

| Parameter | Description |
|---|---|
| lOptions | is a long value containing all the option flags |

**Returns**

None.

**Comments**

All the data structures handled by the YGrep Search Engine should be considered useless (their content may have loosed any useful signification and may lead to uncommon errors) after calling this function. Therefore, it should not be used to change any behaviour between e.g. a call to **CompileRGrep()** and a call to **RGrep()** or between two **RGrep()** calls.

The possible flags to set in the lOptions parameter are described here:

| Value | Returned value |
|---|---|
| YO_EOLCR | see next comment |
| YO_EOLLF | see next comment |
| YO_EOLCRLF | see next comment |
| YO_ERRMSG | Setting this flag is equivalent to calling the function YGrepMessages() with a TRUE argument. |

The default value in the current version is: YO_ERRMSG. The user is advised to set his/her own value to avoid risks of future changes in the YGrep Search Engine behaviour.

By default, the YGrep Search Engine regular expression does not expect to find (and recognize) end-of-line characters in the character stream it is handling. Consequently, ^ and $ (the beginning-of-line and end-of-line markers are not used at all, and only match the 'beginning-of-lpText' and 'end-of-lpText').

But, depending on the option set with this function, the YGrep Search Engine behaviour can be modified to recognize as end-of-line the following characters or strings:

| Value | Returned value |
|---|---|
| YO_EOLCR | character CR or C-character '\r' or ASCII 8 (also known as Carriage Return) |
| YO_EOLLF | character LF or C-character '\n' or ASCII 9 (also known as Line Feed or New Line) |
| YO_EOLCRLF | any combination of CR and LF |

The lOptions flags are set as a whole with calls to **YGrepOptions** like

**YGrepOptions(***YO_EOLLF*|*YO_ERRMSG***);**

where all options are grouped by the '|' operator.

Note that **YGrepOptions** has no effect on end-of-line handling in the **SFileAGrep**, **SFileRGrep** and **SFileSGrep** functions.

**See Also**

**YGrepMessages**, **CompileRGrep**, **RGrep**

## SFileOpen

```
#include    <windows.h>
#include    <ygrep.h>
```

**BOOL SFileOpen(***LPCSTR file***)**

**LPCSTR** *file*;  /* name of the file to open  */

The **SFileOpen** function is used to open the file which will be exploited by the other functions of the **SFile** series.

| Parameter | Description |
|---|---|
| file | name of the file to open |

**Returns**

**TRUE** if the open operation succeeded. **FALSE** if the operation failed.

**Comments**

The function makes no difference between all the reasons the open operation may fail.

**See Also**

**SFileClose**, **SFileAGrep**, **SFileRGrep**, **SFileSGrep**.

## SFileAGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**LPSTR SFileAGrep()**

The **SFileAGrep** function activates the approximative search into the previously opened file (with the **SFileOpen** function).

| Parameter | Description |
|---|---|
| None | |

**Returns**

A line extracted from the searched file, or **NULL** if the end of the file was reached.

**Comments**

On the first call, **SFileAGrep** will return the first line with a match. Each time **SFileAGrep** is called again, it will return the next one. When it returns **NULL**, the end of the file was reached with no match.

Some flags can be set in order to slightly modify the operation of the **SFileAGrep**, **SFileRGrep** and **SFileSGrep** functions.

**See Also**

**SFileSetFlags**, **SFileRGrep**, **SFileSGrep**.

## SFileRGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**LPSTR SFileRGrep()**

The **SFileRGrep** function activates the regular expression search into the previously opened file (with the **SFileOpen** function).

| Parameter | Description |
|---|---|
| None | |

**Returns**

A line extracted from the searched file, or **NULL** if the end of the file was reached.

**Comments**

On the first call, **SFileRGrep** will return the first line with a match. Each time **SFileRGrep** is called again, it will return the next one. When it returns **NULL**, the end of the file was reached with no match.

Some flags can be set in order to slightly modify the operation of the **SFileAGrep**, **SFileRGrep** and **SFileSGrep** functions.

**See Also**

**SFileSetFlags**, **SFileAGrep**, **SFileSGrep**.

## SFileSGrep

```
#include    <windows.h>
#include    <ygrep.h>
```

**LPSTR SFileSGrep()**

The **SFileSGrep** function activates the approximative search into the previously opened file (with the **SFileOpen** function).

| Parameter | Description |
|---|---|
| None | |

**Returns**

A line extracted from the searched file, or **NULL** if the end of the file was reached.

**Comments**

On the first call, **SFileAGrep** will return the first line with a match. Each time **SFileSGrep** is called again, it will return the next one. When it returns **NULL**, the end of the file was reached with no match.

Some flags can be set in order to slightly modify the operation of the **SFileAGrep**, **SFileRGrep** and **SFileSGrep** functions.

**See Also**

**SFileSetFlags**, **SFileAGrep**, **SFileRGrep**, **SFileSGrep**.

## SFileClose

```
#include    <windows.h>
#include    <ygrep.h>
```

**long SFileClose()**

The **SFileClose** function is the entry point to allow/prohibit error and warning messages coming out of the YGrep Search Engine in any other form than returned values. It provides a way to remove dialog boxes.

| Parameter | Description |
|---|---|
| none | |

**Returns**

The number of encountered lines in the file (with one or more matches).

**See Also**

**SFileOpen**.

## SFileSetFlags

```
#include    <windows.h>
#include    <ygrep.h>
```

**void SFileSetFlags(***BOOL bPrintFileName, BOOL bPrintBlockNumber, BOOL bPrintLineNumber, BOOL bCountsOnly, BOOL bNonMatching***)**

```
BOOL bPrintFileName;        /* are error messages allowed?  */
BOOL bPrintBlockNumber;     /* are error messages allowed?  */
BOOL bPrintLineNumber;      /* are error messages allowed?  */
BOOL bCountsOnly;           /* are error messages allowed?  */
BOOL bNonMatching;   /* are error messages allowed?  */
```

The **SFileSetFlags** function is used to slightly modify the operation of the **SFileAGrep** and **SFileRGrep** functions by allowing specific operation modes.

| Parameter | Description |
|---|---|
| bPrintFileName | **TRUE** means "add file name at the beginning of the returned "matching" lines",<br>**FALSE** means "no, please". |
| bPrintBlockNumber | **TRUE** means "add block position in the file for the "matching" line at the beginning of the returned "matching" lines",<br>**FALSE** means "no, please". |
| bPrintLineNumber | **TRUE** means " add line number in the file for the "matching" line at the beginning of the returned "matching" lines ",<br>**FALSE** means "no, please". |
| bCountsOnly | **TRUE** means "do not output anything. We'll only collect the count of "matching" line when closing the file",<br>**FALSE** means "give me all lines, please". |
| bNonMatching | **TRUE** means "output non-matching lines rather than those matching",<br>**FALSE** means "normal operation, please". |

**Returns**

None.

**Comments**

By default, all parameters are set to **FALSE** when the DLL is loaded; And they are never reset by any mean other than calling this function. So, the user should always use it since it will not cost much performance.

**See Also**

**SFileAGrep**, **SFileRGrep**, **SFileSGrep**.

## Other packages of the Engine Series

The current package is part of a series of so-called *Engines* for power programmers and power users. They can be found in all good shareware libraries (Well! At least, they look good to me if they have my packages...).

The Engine Series include the following programmers tools:

YGrep Search Engine

BitList Engine

and an application:

ClusterView

## ClusterView Application

The ClusterView application is an MS-Windows file viewer able to handle multiple files grouped in a structure named a **cluster**. It is an efficient way to look at groups of files which are too large to be stored in main memory.

The main advantages of this application are:

file viewer for files larger than the memory size AND the swap file size.

file viewer for file groups (named **clusters**).

search capabilities including approximative search (or search for a pattern with a number of errors) and regular expression search (compatible with Unix GREP search).

This application is a must when you handle large files under MS-Windows and cannot afford large amounts of memory and/or large swap files and/or the performance penalty imposed by most other file viewers.

The ClusterView application uses and demonstrates the capabilities of the AGrep Search Engine in a real-life context.

## YGrep Search Engine

The YGrep Search Engine is a text search Dynamic Link Library (DLL) to be used with any kind of MS-Windows application. It has two possibilities:

approximative search based on Baeza-Yates algorithm to find a pattern which is only partly known (also known as search with erroneous patterns). For example, you can search for "pattern" with 1 error (at most) and it will match "pattern", "pittern" and "Pattern" while stepping over "lantern" (2 errors).

search modeled on the Unix utility named GREP. It is particularly useful for complex searching with the help of its specific search "language" to describe the pattern you look for. For example, you can search for "^pattern" to look for "pattern" at the beginning of a line; or for "[pl]a[nt]tern" to look for either "pattern" or "lantern". An extensive description of the language can be found on any Unix system, or in the help file accompanying the YGrep Search Engine shareware edition on your preferred BBS or Internet site.

Both are particularly useful to improve greatly the search capability of an existing tool such as a text editor, a data base search engine, etc.

Additional functionality allow to apply substitution after pattern recognition by the search functions.

## BitList Engine

The BitList Engine is a DLL designed to handle lists of bits (and to a small extent, big numbers). It was built because of the limitations of the ANSI-C bit fields which cannot be larger than an "unsigned long".

The BitList Engine allows you to build very large bit lists and to handle them with a set of functions covering a large range of needs (this is continuously expanding):

- constructors/copy-constructors/copy operators
- logical operators (AND, OR, NOT, etc.)
- arithmetic operations (ADD, SUB, etc.)
- shift operations (left and right)
- others...

This will be particularly useful to handle large sets (as belong to the programmer's bag of tools) and to work on encryption/compression code.

## Author Address

Registration fees can be sent to, and the author can be reached at the following address (Email and duly paid registration fee is the preferred interface if you want a prompt answer):


Yves Roumazeilles
63 rue des Moines
75017 PARIS (FRANCE)
Phone: +33 1-42.28.74.51
Fax: +33 1-30.38.37.07
CIS: 101233,1032
Email: 10233.1032@compuserve.com


You can also reach our web page at:

http://ourworld.compuserve.com/homepages/Yves_Roumazeilles

For comments, suggestions and bug reports, Email is also available at:


Yves_Roumazeilles@compuserve.com
Yves.Roumazeilles@sagem.fr
Roumazeilles@sagem.fr

## What is Shareware?

"Shareware" is a way to distribute software while retaining the best of all worlds. People are invited to freely make copies of the software for evaluation purposes (it's cheap distribution). You are both legally and morally obliged to pay the registration fee if you start using the software after an initial 30 day evaluation period (the author gets money from its work). This respects the rights of the author while avoiding burdening the users with high costs of traditional distribution channels.

Shareware is not free, Shareware is not public domain, but Shareware is not expensive (I actually cannot live from it...)

Remember! The fee is small because the distribution is simple, but the user (YOU) must honestly pay the registration fee. This will allow future releases to hit the market soon with many enhancements.