

The YGrep Search Engine

The YGrep Search Engine (YGREP.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](#)

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

[New features](#)

[Alphabetical list of functions](#)

[Function groups](#)

[Structures](#)

[Registration information](#)

[Common questions and features](#)

[Future developments](#)

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

On one hand, as I am sure you already inferred, 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.

On the other hand, [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 including Vax-VMS, DOS, CP/M, etc. They often differ from the original one from 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 afore mentionned MicroEmacs editor.

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

New features

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

Dialog boxes and functions similar to those found in the COMMDLG Dynamic Link Library

Possible new ways to look for data

YGrep Search Engine 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. Additionally, 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.

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_]\>$'`).

For an extended specification of the regular expression used, see also [Regular Expressions](#)

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 an 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 followed 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 occurrences of <i>Windows</i> with <i>MS-Windows</i>
---------	------	---

\(dows\)([Ww]in\)	\2\1	allows to reorder the pattern <i>dowsWin</i> into the normal <i>Windows</i> regardless of the letter-casing of the <i>W</i> 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 1024 letters maximum.

The number of errors is limited to 512 maximum.

Both of them are limited by the following formula:

$$\text{length_of_pattern} \leq 1024 / (\log_2(\text{number_of_errors}) + 1)$$

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 512 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 only a mere 130FF or US\$20.

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 - No credit cards

Public (Software) Library

US - credit card orders

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 Developer 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).

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 130 French Francs) _____

Developer kit license (\$295 or 1920 French Francs) _____

BitList Engine

Single user license (\$20 or 130 French Francs) _____

Developer kit license (\$295 or 1920 French Francs) _____

ClusterView Application

Single user license (\$20 or 130 French Francs) _____

Shipping/Handling to Europe (\$4 or 25 French Francs) _____

Shipping/Handling outside of Europe (\$6 or 40 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.

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: +33-1-34.30.53.34
Email: Roumazeilles@sagem.fr

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.

NAME: _____

COMPANY: _____

ADDRESS: _____

TOWN: _____

CITY: _____

POSTCODE: _____ COUNTRY/STATE: _____

TELEPHONE: _____

FAX: _____

EMAIL: _____

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.....

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** Email to **71355,470**

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. 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).

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: +33-1-34.30.53.34

Email: Roumazeilles@sagem.fr

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.

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 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.

Alphabetical list of functions

AddWordChar

AGrep

AGrepEmpty

AGrepSubsBuild

CompileAGrep

CompileRGrep

InitWordCharTable

RemoveWordChar

RGrep

RGrepSubsBuild

YGrepVersion

Function groups

Initialization function

Approximative Search functions

Regular Expression Search functions

other functions

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 of the behaviour of the **AGrep** group of functions:

None.

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

AddWordChar

InitWordCharTable

RemoveWordChar

Approximative Search functions

AGrep

AGrepEmpty

AGrepSubsBuild

CompileAGrep

Regular Expression Search functions

[CompileRGrep](#)

[RGrep](#)

[RGrepSubsBuild](#)

Other functions

YGrepVersion

Structures

AGREPINFO structure

RGREPINFO structure

XGREPINFO union

LPAGREPINFO type

LPRGREPINFO type

AGREPINFO

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

The **AGREPINFO** structure contains information about the approximative search to be executed by the **AGrep** function.

Parameter	Description
iErrorCode	propagated error code
cPat	propagated pattern text string
TagStart	table of tags start
TagEnd	table of tags end
uMask	internal data
uOvMask	internal data
uLimit	internal data
uTable	internal data (characteristic vectors table)
iBitsPerState	internal data
iWordSize	internal data (actual size in bits of BLIST data structures)
iType	MATCH or MISMATCH
cUPat	internal data

Comments

Applications should use **CompileAGrep** to fill this data structure.

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     iCircf;
    char    cDFA[MAXDFA];
} RGREPINFO;
```

The **RGREPINFO** structure contains information about the approximative search to be executed by the **RGrep** function.

Parameter	Description
iErrorCode	propagated error code
cPat	propagated pattern text string
TagStart	table of tags start
TagEnd	table of tags end
bMatchCase	propagated match information
iCircf	match at beginning of line?
cDFA	automaton

Comments

Applications should use **CompileRGrep** to fill this data structure.

LPRGREPINFO

```
typedef RGREPINFO FAR* LPRGREPINFO;
```

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

YGREPINFO

```
typedef struct tagXGrepInfo {      /* ygi */
    union {
        AGREPINFO  aGI;
        RGREPINFO  rGI;
    } y;
    int    iTypeofInfo;    /* 0: empty, 1:AGREP, 2:RGREP */
} XGREPINFO;
```

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.

CompileAGrep

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

```
int FAR PASCAL CompileAGrep(LPCSTR lpText, UINT k, BOOL bMatchCase, AGREPINFO FAR* pGI)
```

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

The **CompileAGrep** function reprocesses 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.

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

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). 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.

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 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 FAR PASCAL AGrep(LPCSTR lpText, AGREPINFO FAR* pGI)
```

```
LPCSTR lpText;          /* text string to explore          */
AGREPINFO FAR* 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.

Comments

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 occurrence 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]**.

See Also

CompileAGrep

AGrepInit

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

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

```
int FAR PASCAL AGrepInit(AGREPINFO FAR* pGI)
```

```
AGREPINFO FAR* pGI; /* pointer to search information block */
```

The **AGrepInit** function should not be used. It is present only for library debugging.

AGrepEmpty

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

```
int FAR PASCAL AGrepEmpty(AGREPINFO FAR* pGI)
```

```
AGREPINFO FAR* 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 <u>CompileAGrep</u> 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 neighbours.

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.

AGrepSubsBuild

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

```
int FAR PASCAL AGrepSubsBuild(LPCSTR lpPattern, LPCSTR lpDest, int iSize, AGREPINFO FAR*
pGI)
```

```
LPCSTR lpPattern;          /* pattern to replace matched strings          */
LPCSTR lpDest;            /* destination buffer for building substitution string */
int iSize;                /* size of the destination buffer              */
AGREPINFO FAR* 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).

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 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, **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 **AGrepSubsBuild**

with the pattern "large &". It will return an lpDest string containing "large horse" which 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**

CompileRGrep

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

```
int FAR PASCAL CompileRGrep(LPCSTR lpText, BOOL bMatchCase, RGREPINFO FAR* pGI)
```

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

The **CompileRGrep** function reprocesses 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_ENCL	Empty closure. Do not provide an expression containing only [] (i.e. an empty closure).
RGERR_ILLEGAL_ENCL	Illegal closure. Some characters are not allowed in a closure: ^\$<>
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 FAR PASCAL RGrep(LPCSTR lpText, RGREPINFO FAR* pGI)
```

```
LPCSTR lpText;          /* text string to explore          */
RGREPINFO FAR* 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 occurrence 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]**.

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.

See Also

CompileRGrep

RGrepSubsBuild

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

```
int FAR PASCAL RGrepSubsBuild(LPCSTR lpPattern, LPCSTR lpDest, int iSize, RGREPINFO FAR*
pGI)
```

```
LPCSTR lpPattern;          /* pattern to replace matched strings          */
LPCSTR lpDest;            /* destination buffer for building substitution string */
int iSize;                /* size of the destination buffer              */
RGREPINFO FAR* 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 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 **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**

InitWordCharTable

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

void FAR PASCAL 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

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

```
void FAR PASCAL 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).

See Also

InitWordCharTable, **RemoveWordChar**.

RemoveWordChar

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

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

```
int FAR PASCAL RemoveWordChar(LPCSTR lpChars)
```

```
LPCSTR lpSrc;          /* text string to explore          */
```

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).

See Also

InitWordCharTable, **AddWordChar**

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.

This value may be used to determine the capabilities/compatibility 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.

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

The Engine Series and their documentation files and manuals are copyrighted (C) 1993-94-95 by Yves Roumazeilles.

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 modelled 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.

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 1-34.30.50.28
Email: Yves.Roumazeilles@sagem.fr

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

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.

