GREP Operators

GREP is a UNIX search utility with a powerful and flexible syntax. GREP syntax can be used within search expressions to reduce the amount of hits returned by eliminating false positives. In addition, using GREP avoids the necessity of specifying many redundant instances of search terms. The use of GREP operators can greatly enhance raw search capabilities.

To explore the operators available within EnCase® software (EnCase), we will create a new case titled "GREP" and add a logical evidence file labeled "GREP Materials.L01." Contained within this LEF are two files: GREP Examples.txt, which is the file we will search, and Input GREP.txt, which contains the search terms we will use. Blue-check **GREP Examples.txt** as shown in Figure 24-1.

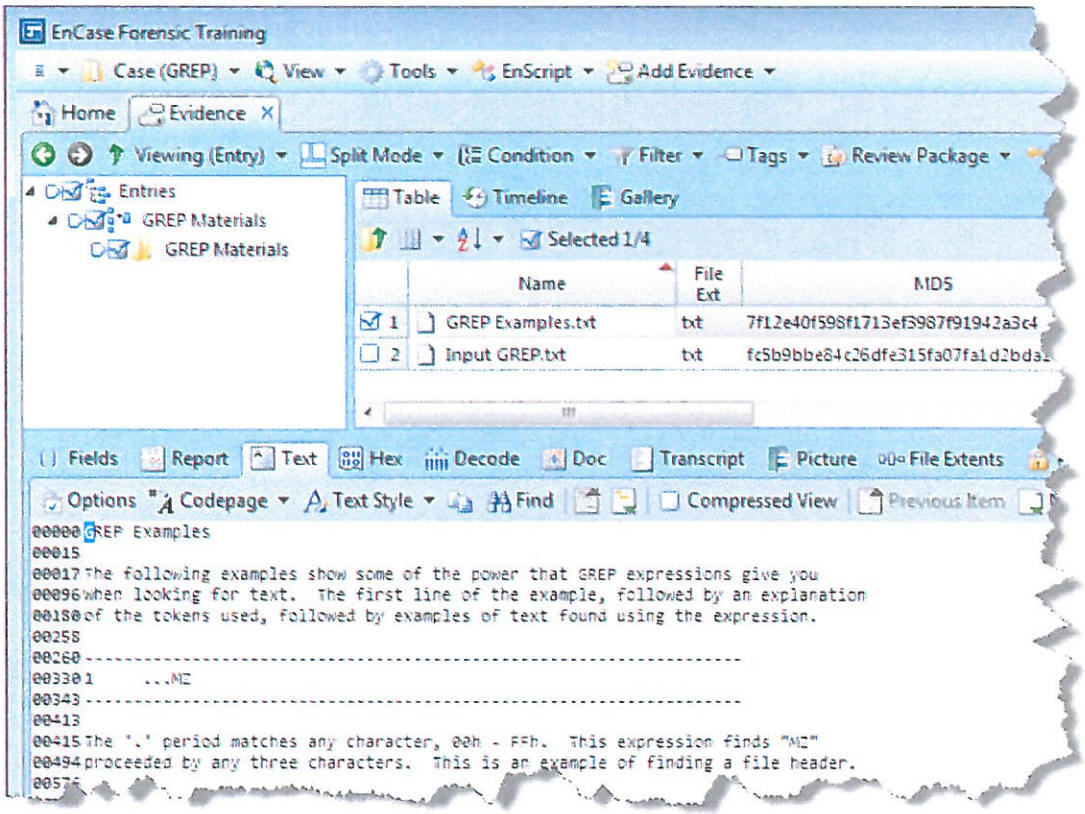

Figure 24-1 GREP Examples.txt selected

Highlight the remaining file, **Input GREP.txt**, and ensure that your active text style provides line breaks so that each term appears on an individual line. If your text style is set to **Fit to pane**, EnCase will interpret the entire block of text as a single term and generate an error due to the perceived excessive length.

In the lower view pane, highlight the content of this file; right-click and copy the content.

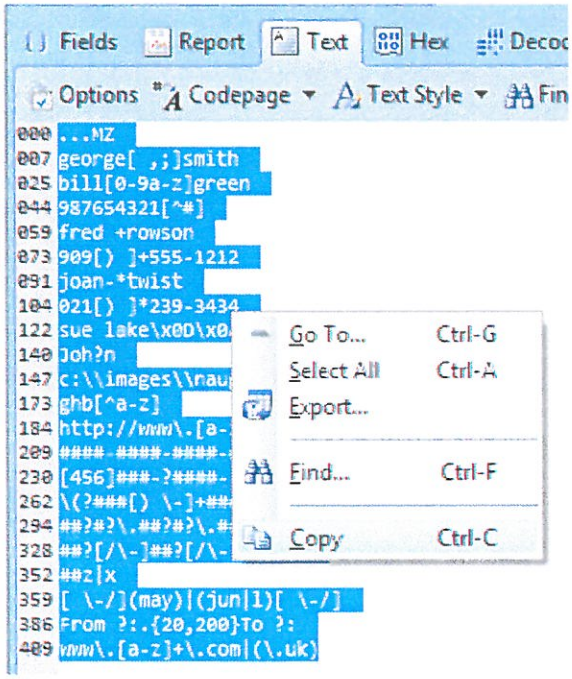

Figure 24-2 Copying the content of the file Input GREP.txt

Locate the Raw Search drop-down and select **New Raw Search**. This will open the New Raw Search dialog box within which we will paste the keywords. Select **Add Keyword List** to enter our search terms. Right-click and paste the data we copied previously and confirm that you have placed a check within the GREP checkbox. This selection informs EnCase to treat each keyword as a GREP statement rather than a literal value.

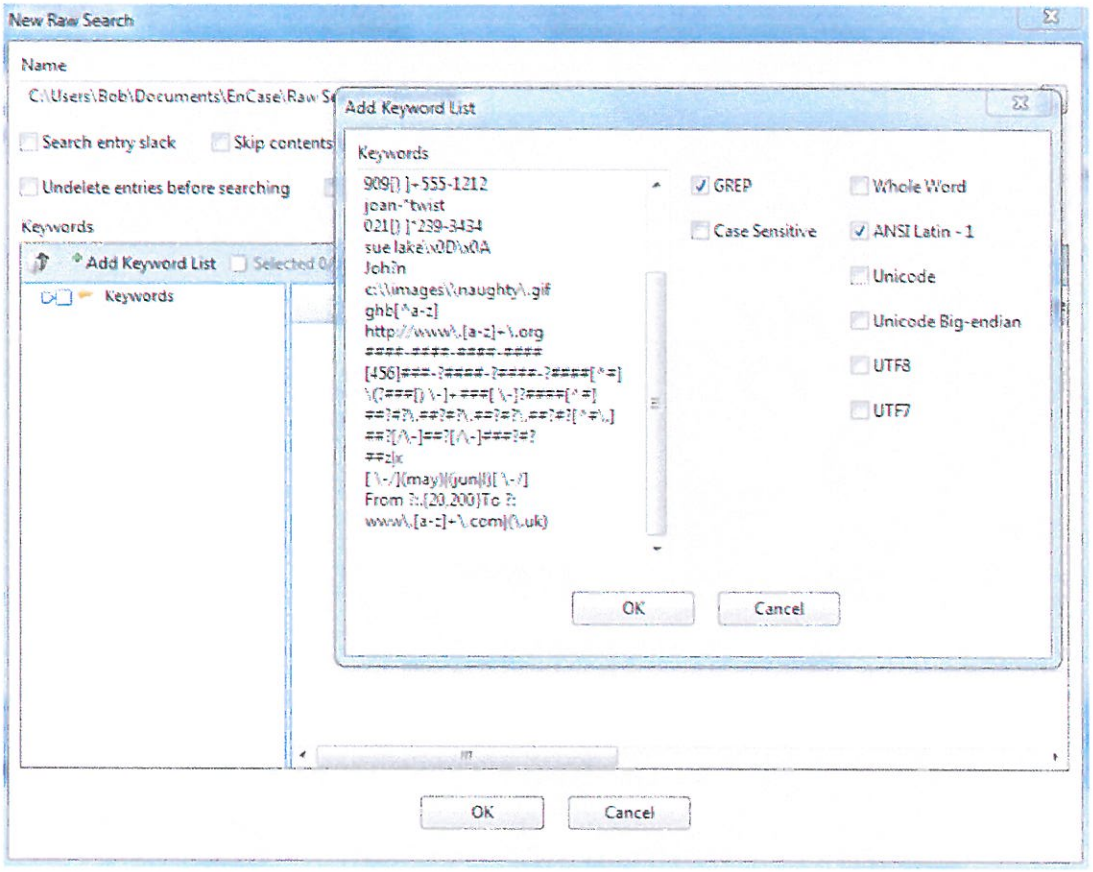

Figure 24-3 Search terms added as a list, with GREP selection active

Click **OK** and the twenty-two (22) search terms will be added to our search. After clicking **OK** on the final dialog box, our search is configured and executed by EnCase.

EnCase will transfer our active view to the Results tab where we can review the function of each GREP operator as well as the search hits obtained. We will spend the remainder of this lesson discussing each GREP operator supported by our search hits within the GREP Examples.txt file.

GREP Expressions

Symbol	Meaning
.	A period matches any single character or non-character byte.
#	A pound sign matches any single numeric character [0–9]. For example, ###-#### matches any number in the form 327-4323 (if looking for a US phone number for example).
[abc]	The square brackets hold the place of one character. That place can be any of the characters that are inside the brackets, and only those. For example, "smit[hy]" would match "smith" and "smity."
[^abc]	A circumflex at the start of the brackets changes the collection to represent any other character that is not inside the brackets. For example, "smit[^hy]" would NOT match "smith" and "smity," but it would match "smitt."
[a-z]	A dash within the brackets signifies a range of characters. For example, [a-e] matches any character from a through e, inclusive. This can be used for numbers and letters.
\x01	Specify a hex value. This can be an ASCII value that may not be able to be typed in (2 digits are required). For example, \x09 is a tab. \x0A is a line feed.
\w0123	Specify a unicode value. 4 integer code is required. See a unicode chart for mapping.
(abc)	Groups characters together.
?	A question mark after a character or group matches one or zero occurrences of that character or group. The character or group preceding the question is repeated zero or one time. For example, "##?/##?/####" would match "1/1/2008" or "01/01/2008," but would not match "2008/1/1."
*	An asterisk after a character or group matches any number of occurrences of that character or group, including zero times. The character or group preceding the asterisk is repeated zero, one or multiple times. For example, "john,*smith" would match "john,smith," "john,,smith," and "johnsmith."
+	A plus sign after a character or group matches any number of occurrences of that character or group at least one time. The character or group preceding the plus sign must be repeated at least once. For example, "john,+smith" would match "john,smith" or "john,,smith" but would not match "johnsmith."
abc{X,Y}	Curlyl braces repeat a character or group between X and Y times. Example: abc{3,7} would repeat the 'c' character three to seven times matching "abcccc" but not "abc." (abc){3,7} would match "abcabcabcabc" but not "abcabc."
\	A backslash before any of these special characters indicates that the character is to be treated literally and not as a GREP character. The GREP symbols are special characters, and if you were specifically looking for that character, you must use the backslash to indicate the character and not the GREP symbol is to be searched. For example, "one\+two" matches "one+two." A slash (\) must be placed in front of any GREP token (including a slash (\) itself) that you wish to be a literal part of the match.
a|b	The pipe is used for groups of keywords. For example: www\.whitehouse\.(com)|(gov)|(net) matches "www.whitehouse.com," "www.whitehouse.gov," and "www.whitehouse.net."

GREP EXAMPLES

The following examples show some of the power that GREP expressions give you when looking for text. The first line of the example is the GREP expression followed by:

1. An explanation of the tokens used
2. Examples of text either found or not found using the expression; the examples that are hits on this expression are highlighted in bold

Example 1

```
...MZ
```

The period (.) matches any character, 00h – FFh. This expression finds "MZ" preceded by any three characters. This is an example of finding a file header.

ADMZ
ZMZ
ASDDE**FJHMZ**KJYTRDF
02,MZ
,,-MZ
ABNMZLK
.!=MZ
MZ

Example 2

```
george[ ,;]smith
```

The square brackets form a set. The set, absent any other GREP operator, controls a single character location and defines within the square brackets what can exist in that single location. This expression finds "george" followed by a space, comma, or semi-colon followed by "smith."

george ,smith
george smith
georgesmith
george,smith
george;smith
george ,;smith
george smith

Example 3

```
bill[0-9a-z]green
```

The dash indicates a range of characters when inside a set. This expression finds "bill" followed by any character between "0" and "9" or "a" and "z" followed by "green."

bill green
bill0green
bill20green
bill1green
billzgreen
billa-zgreen
bill,green

Example 4

```
987654321[^#]
```

The caret (^) at the start of a set indicates any character other than those in the set. This expression finds "987654321" followed by any character other than '0'- '9.'

NOTE: The expression [^#] is identical to [^0-9].

YM**987654321A**S
1298765432145
987654321
1111111987654321000000
1111111**987654321A**00000
....**987654321.**

Example 5

```
fred +rowson
```

The plus sign (+) says repeat the preceding character (or set) any number of times (maximum of 255), but at least once. This expression finds "fred" followed by any number spaces followed by "rowson."

fredrowson
fred rowson
fred rowson
fred +rowson
Fred Rowson
FRED ROWSON

Example 6

```
909[) ]+555-1212
```

The + is most commonly used after a set in more complex statements. This expression finds "909" followed by a space or a parenthesis, or a space and a parenthesis.

909 555-1212
909 555-2121
(909)555-1212
(909) 555-1212
(909)) 555-1212
909-555-1212
(909))) 555-1212
909555-1212

Example 7

```
joan-*twist
```

The star (*) says repeat the preceding character (or set) any number of times (maximum of 255) including zero. This expression finds "joan" followed by any number of dashes followed by "twist." This operator is commonly used after a set.

Joan Twist
Joan-TWIST
joantwist
joan-twist
joan---twist

Example 8

```
021[) ]*239-3434
```

The * is most commonly used after a set in more complex statements. This expression finds 021 followed by 239, or by a parenthesis, a space, or both. This is the same as the + except it allows for zero characters.

021 239-3434
021)239-3434
021) 239-3434
021))))239-3434
021-239-3434
021239-3434
0212393434

Example 9

```
sue lake\x0D\x0A
```

The slash (\) followed by an 'x' indicates a two-digit hex number representation for a character. This expression finds "sue" followed by a space, followed by "lake," followed by a carriage-return linefeed sequence.

Sue LAKE
sue lake
sue lake.
sue,lake
sue lake
Sue Lake was a driving...

Example 10

```
Joh?n
```

The question mark (?) says repeat the preceding character (or set) one or zero times. This expression finds "Jon" or "John" with or without the "h."

John
JON
Jack
join

Example 11

```
c:\\images\\naughty\.gif
```

The slash (\) preceding any character (including \) indicates that this is a literal character and not a GREP symbol. Be careful when expressing file names and paths in GREP; slashes and dots should be preceded by a \.

```
C:\images\noughty.gif
c:\images\naughty.gif
c:\\images\\naughty.gif
c:\\images\\naughty\.gif
```

Example 12

```
ghb[^a-z]
```

This expression matches "ghb" followed by any non-alphabetic character. This ensures that short names and words are not found inside other words.

```
neighborhood
thighbone
ghb
highboard
Get some GHB !!!
5.GHB,
```

Example 13

```
http://www\.[a-z]+\.org
```

This expression matches "http://www." followed by any alphabetic characters followed by ".org." This is a good way to look for website references.

```
http://www.bozo.net
http://www.theonewiththeverylongsitename.org/index.htm
http://www.to-wong-foo.co.fr
http://microsoft.msdn.au.com
http://www.bozo.org
http://www.the_one_with_the_very_long_site_name.org
```

Example 14

```
####-####-####-####
```

The # character matches any number. This expression matches a credit card number with the numbers separated by dashes.

6666-6666-6666-6666-6666-6666 (**NOTE**: Three hits)
1234-3623-3410-2232
4534-2123-9866-651
4534212398666512
1233456780007654
456

Example 15

```
[456]###-?####-?####-?####[^#]
```

This expression matches a credit card number with the dashes between the numbers being optional and the first number being constrained to either 4, 5, or 6.

6234-3623-3410-2232
4534212398666512
6666666666666666
4444444444444444
1233456780007654
323345680007654

NOTE: There is now an EnScript® module that calculates the checksum to ensure hits could be legitimate card numbers, cutting down on the number of false positives.

Example 16

```
\(?###[) \-]+###[ \-]?####[^#]
```

This expression matches a U.S. phone number in one of several formats. The '\(?' expression means that the open paren '(' character can be present or not. The '[) \-]+' expression means that either a space or a close paren or a dash can be repeated any number of times except 0, and the '[\-]?' indicates the space or dash can be present or not.

(909) 875-4125
(123)-255 1700
204-725-2436
103 875 4344
345)))--4562134aDE/}
9098721344

Example 17

```
##?#?\.##?#?\.##?#?\.##?#?[^#\.]
```

This expression matches an IP number in regular form with 4 (up to 3 digits) numbers separated by periods.

123.235.23.1
255.255.255.255
0.0.0.0
512.79.789.367
234.1234.123.123
0.0.0.0.
12..12.34.2

The expressions below will screen out hits on 255.255.255.255

```
[^\.\x00]1##\.##?#?\.##?#?\.##?#?[^#\.]
[^#\.\x00][1-9]#?\.[12]#?#?\.[12]#?#?\.[12]#?#?[^#\.]
```

Example 18

```
##?[/\-]##?[/\-]###?#?
```

This expression matches a date in regular form with a 4-digit year and either one- or two-digit months and days separated by either forward slashes or dashes.

13/3/1999
13/03/1999
14-2-1998
14-02-2000
1-2-02
01/02/02
34/19/5079
73-49-3030
01-Feb-2002
3-Dec-2003

Example 19

```
##z|x
```

The vertical bar character (|) is used as a logical OR. This search expression will find any two digits (0 through 9) followed by the character 'z' or the character 'x.'

```
abc
cba
123zbc
321cba
432xbk
234gbn
```

Example 20

```
[ \-/](may)|(jun|l)[ \-/]
```

This expression looks for a literal space, hyphen, or slash followed by either "May" or, "Ju," followed by either an "n" or an "l," followed by another space.

```
3rd Dec 2002
20th May 2003
1-Jun-03
14 Jul 2003
11-Apr-2004
10 Dec 01
9/jun/01
```

Example 21

```
From ?:.{20,200}To ?:
```

This expression will search for e-mail headers by finding the word "From" followed by zero or one space character and then a colon, followed by between 20 and 200 of any character, followed by the word "To" followed by zero or one space character, and then a colon.

```
From : sharren_redmond@yahoo.co.uk
To : Larissa.moore@hotmail.com
Subject :Did I bother you?
Date :Wed, 26 Mar 2003 10:25:14 +0000
Received: from bellsouth.net ([194.232.78.135]) by mc6-
f18.law1.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600); Tue, 22 Apr
2003 02:33:13 -0700
Received: from [214.112.45.243] by bellsouth.net with esmtp; Mon, 28
May 2001 03:09:53 -0300
X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP
X-Mailer: Internet Mail Service (5.5.2650.21)
Return-Path: podqsupyqrdM@pacbell.net
Message-ID: <MC6-F18JMXehYSoOZzH0013a4a7@mc6-f18.law1.hotmail.com>
X-OriginalArrivalTime: 22 Apr 2003 09:33:13.0655 (UTC)
FILETIME=[323CC470:01C308B2]
From : bob.weitershausen@encase.com
To : Jamey.Tubbs@encase.com
Subject :Latest training materials
Date :Thu, 27 Mar 2003 18:37:11 +0000
MIME-Version: 1.0
Received: from 3com.com ([211.162.89.61]) by mc7-f37.law1.hotmail.com
with Microsoft SMTPSVC(5.0.2195.5600); Mon, 21 Apr 2003 19:50:19 -0700
X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP
Message-ID: <HMKPOGPOBBJEOKEPBELFEILLMIAB.wpeterson@att.com>
In-Reply-To: <25ce01c2ee96$210f2c95$39f4ccfd@zcc2iia>
X-MIMEOLE: Produced By Microsoft MimeOLE V6.00.2800.1106
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Return-Path: wpeterson@att.com
X-OriginalArrivalTime: 22 Apr 2003 02:50:19.0429 (UTC)
FILETIME=[E9469D50:01C30879]
From: bill.sutter@encase.com
To: bill.thompson@encase.com
Subject: Simon_Key Unemployed? Need a Job
Date: Mon, May 28 2001 00:54:37 +1100
MIME-Version: 1.0
Received: from aexeij ([65.69.110.33]) by mc9-f5.bay6.hotmail.com with
Microsoft SMTPSVC(5.0.2195.5600); Mon, 21 Apr 2003 09:54:44 -0700
X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP
Message-Id: <dfunysdebwviq@dmelkcs.bigfoot.com>
Return-Path: dtyiht@dmelkcs.bigfoot.com
X-OriginalArrivalTime: 21 Apr 2003 16:54:49.0038 (UTC)
FILETIME=[B84D9EE0:01C30826]
```

Example 22

```
www\.[a-z]+\.com|(\.uk)
```

This search expression searches for URLs starting with "www." followed by at least one other alpha character, followed by ".co," and then followed by either "m" or ".uk."

http://**www.encase.com**/

http://**www.torent.co.uk**/cgi-bin/honat.cgi?si=tr&st=1&ty=3+Bedroom&to=LIVERPOOL

http://**www.belvoirgroup.com**/sthelens.htm

http://www.haltabuse.org/help/headers/

http://www.planetdv.net/Frameset.asp?show=content&rID=818&RC=10

http://**www.aol.co.uk**/try/aod/aim