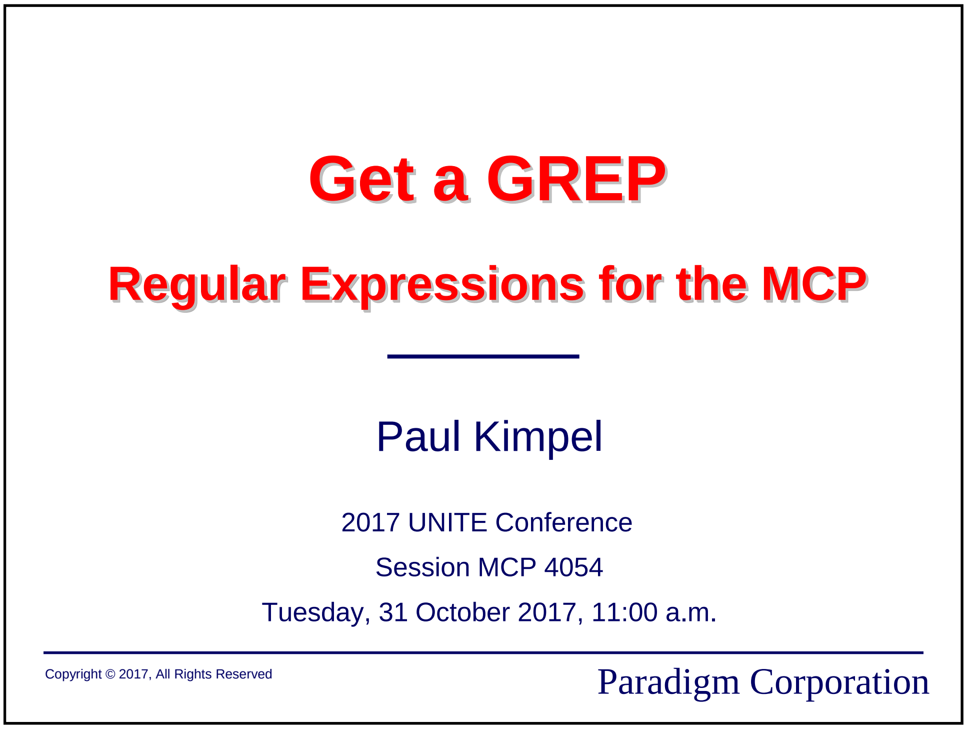


Get a GREP

Regular Expressions for the MCP

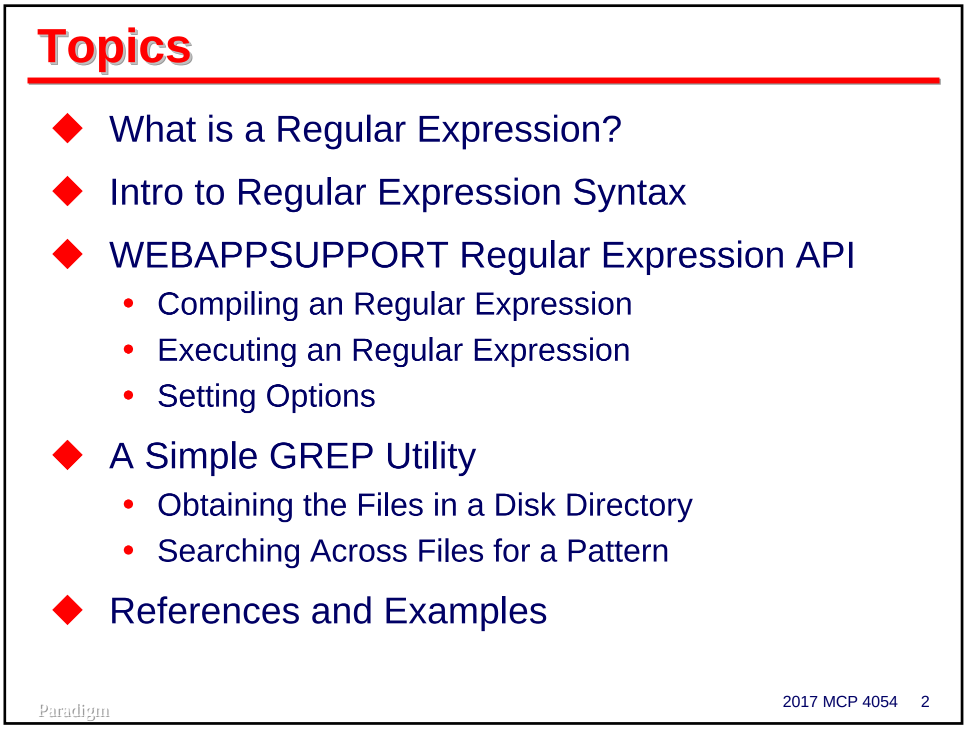
Paul Kimpel

2017 UNITE Conference

Session MCP 4054

Tuesday, 31 October 2017, 11:00 a.m.

Paradigm Corporation

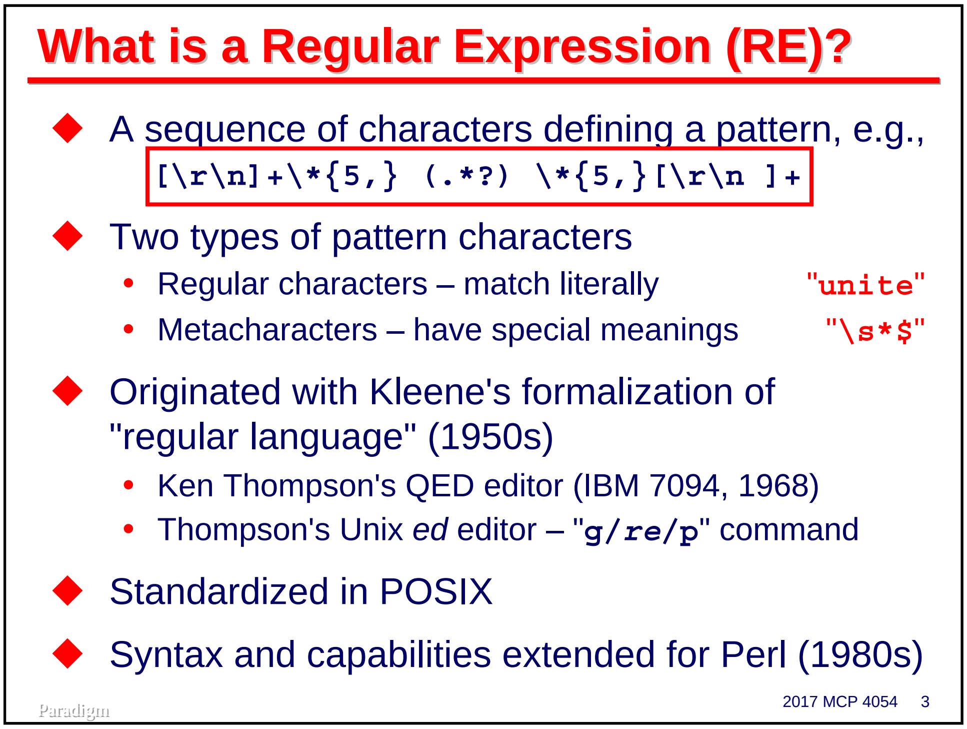


Topics

- What is a Regular Expression?
- Intro to Regular Expression Syntax
- WEBAPPSUPPORT Regular Expression API
  - Compiling an Regular Expression
  - Executing an Regular Expression
  - Setting Options
- A Simple GREP Utility
  - Obtaining the Files in a Disk Directory
  - Searching Across Files for a Pattern
- References and Examples

Paradigm

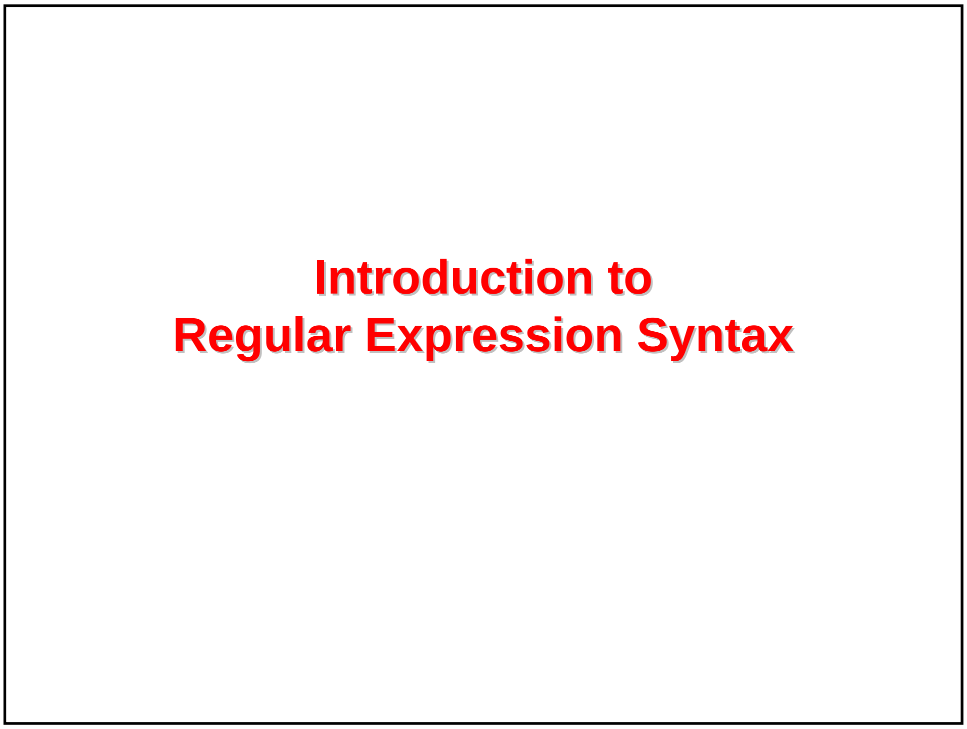
2017 MCP 4054 2



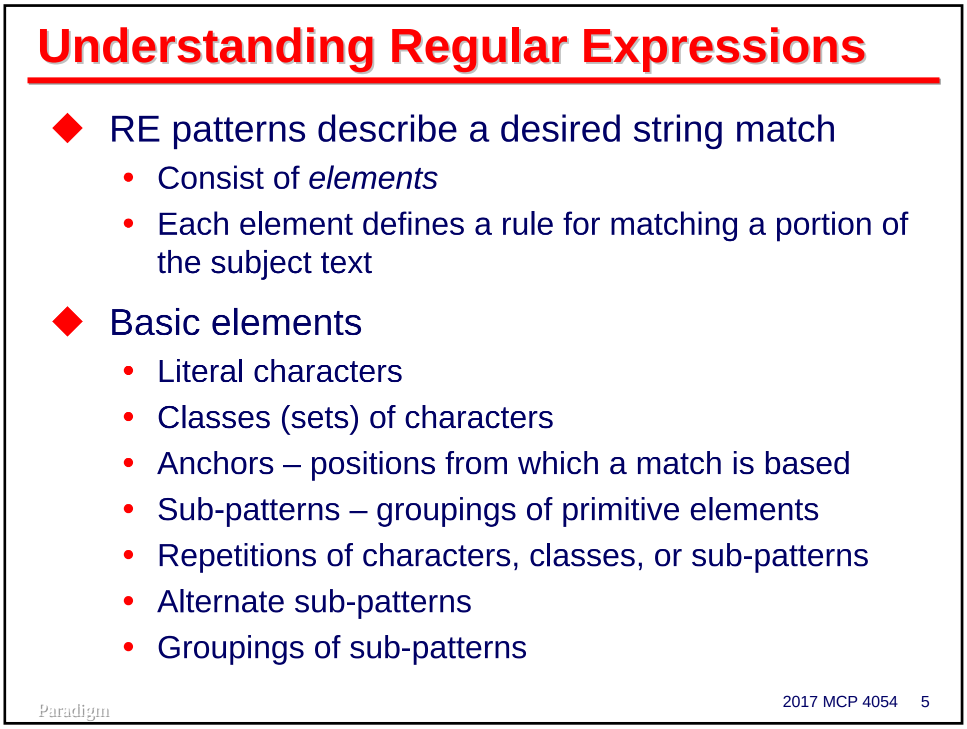
What is a Regular Expression (RE)?

- A sequence of characters defining a pattern, e.g.,

  `[\r\n]+\*{5,} (.*?) \*{5,}[\r\n ]+`
- Two types of pattern characters
  - Regular characters – match literally "`unite`"
  - Metacharacters – have special meanings "`\s*$`"
- Originated with Kleene's formalization of "regular language" (1950s)
  - Ken Thompson's QED editor (IBM 7094, 1968)
  - Thompson's Unix *ed* editor – "**g/*re*/p**" command
- Standardized in POSIX
- Syntax and capabilities extended for Perl (1980s)



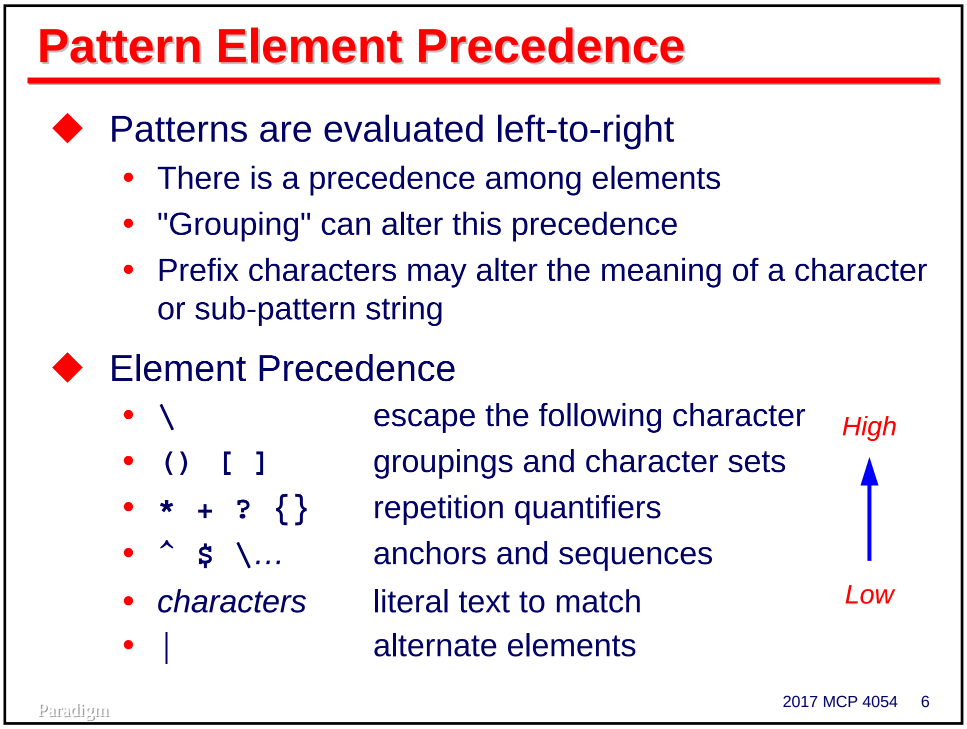
Introduction to Regular Expression Syntax



Understanding Regular Expressions

- RE patterns describe a desired string match
  - Consist of *elements*
  - Each element defines a rule for matching a portion of the subject text
- Basic elements
  - Literal characters
  - Classes (sets) of characters
  - Anchors – positions from which a match is based
  - Sub-patterns – groupings of primitive elements
  - Repetitions of characters, classes, or sub-patterns
  - Alternate sub-patterns
  - Groupings of sub-patterns

Paradigm 2017 MCP 4054 5



Pattern Element Precedence

- Patterns are evaluated left-to-right
  - There is a precedence among elements
  - "Grouping" can alter this precedence
  - Prefix characters may alter the meaning of a character or sub-pattern string
- Element Precedence

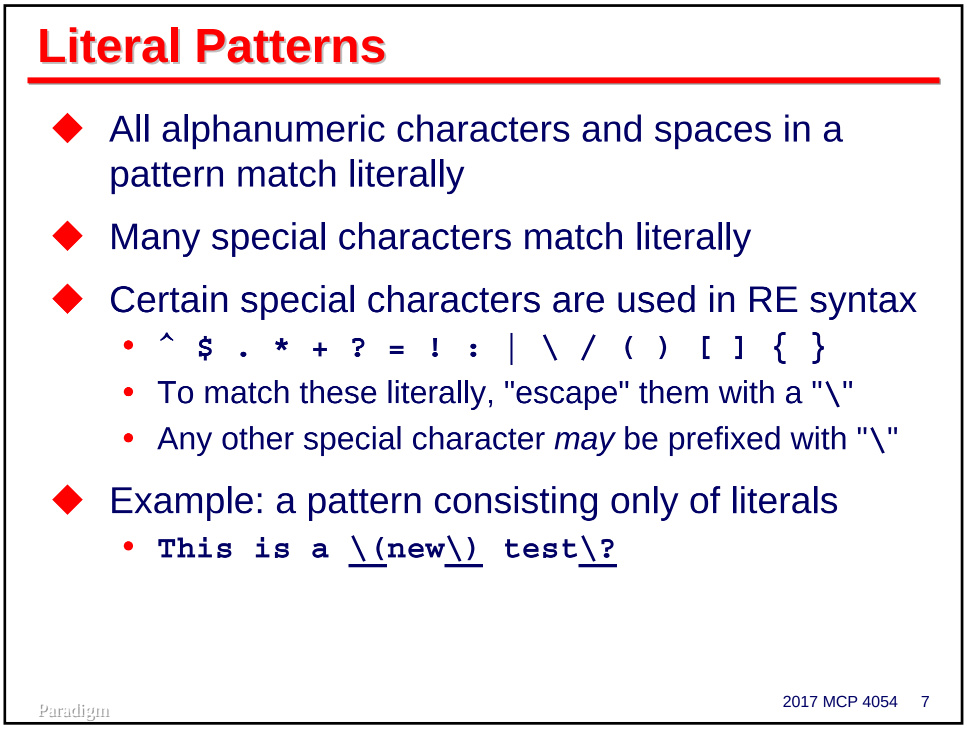
`\`	escape the following character
`() [ ]`	groupings and character sets
`* + ? {}`	repetition quantifiers
`^ $ \…`	anchors and sequences
characters	literal text to match
|	alternate elements

High

↑

Low

Paradigm 2017 MCP 4054 6

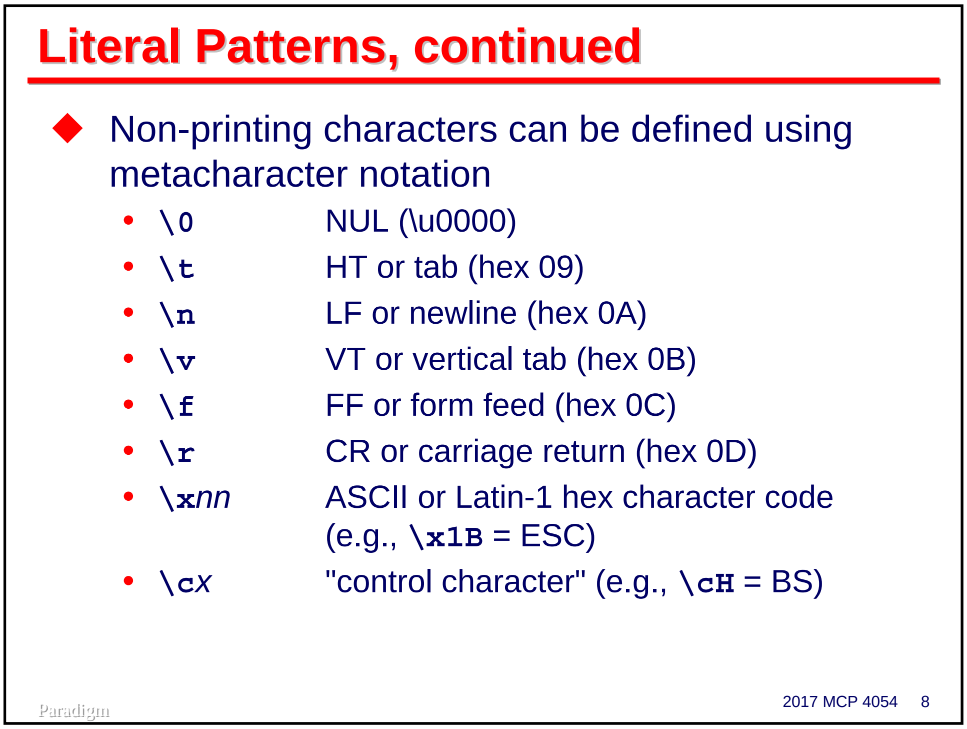


Literal Patterns

- All alphanumeric characters and spaces in a pattern match literally
- Many special characters match literally
- Certain special characters are used in RE syntax
  - `^ $ . * + ? = ! : | \ / ( ) [ ] { }`
  - To match these literally, "escape" them with a "\"
  - Any other special character *may* be prefixed with "\"
- Example: a pattern consisting only of literals
  - `This is a \(new\) test\?`

Paradigm

2017 MCP 4054 7



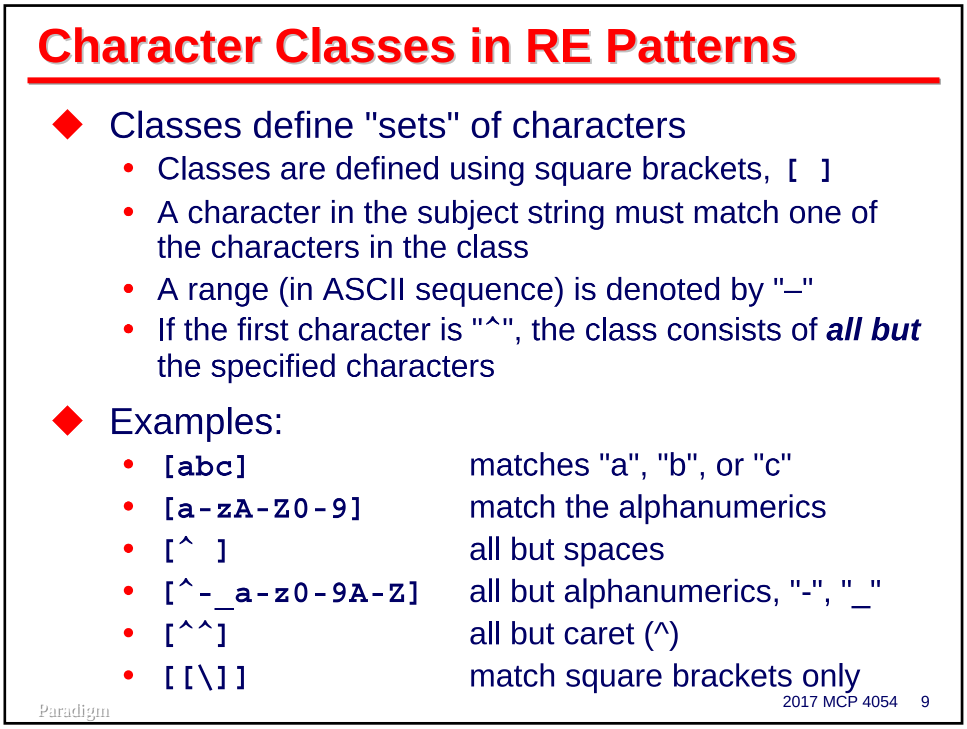
Literal Patterns, continued

- Non-printing characters can be defined using metacharacter notation

`\0`	NUL (\u0000)
`\t`	HT or tab (hex 09)
`\n`	LF or newline (hex 0A)
`\v`	VT or vertical tab (hex 0B)
`\f`	FF or form feed (hex 0C)
`\r`	CR or carriage return (hex 0D)
`\x`*nn*	ASCII or Latin-1 hex character code (e.g., `\x1B` = ESC)
`\c`*x*	"control character" (e.g., `\cH` = BS)

Paradigm

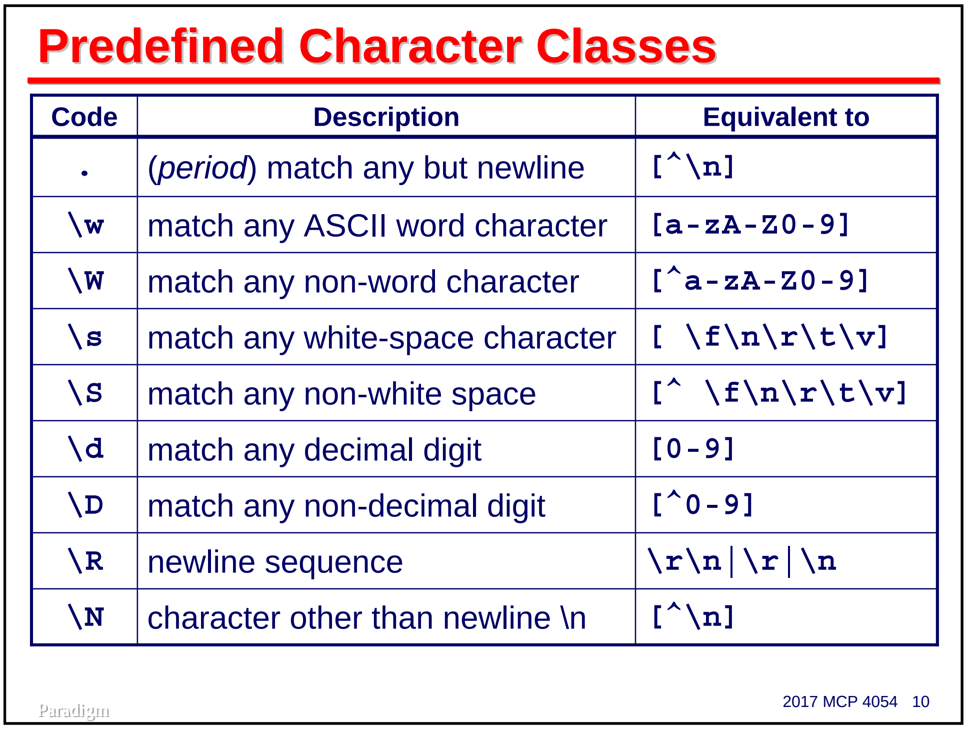
2017 MCP 4054 8



Character Classes in RE Patterns

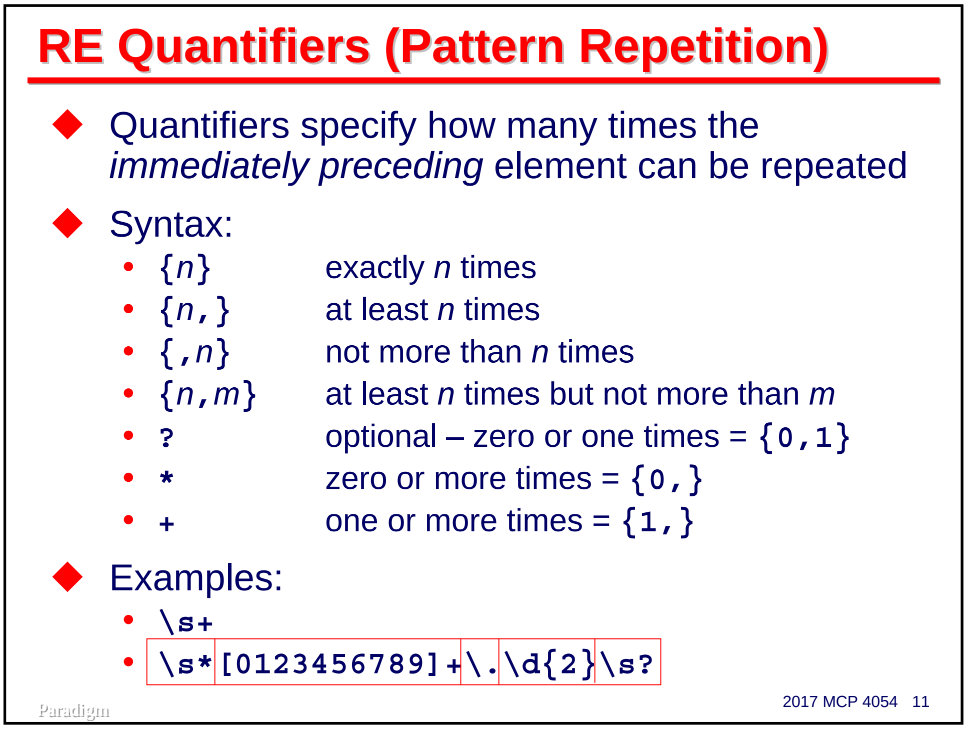
- Classes define "sets" of characters
  - Classes are defined using square brackets, `[ ]`
  - A character in the subject string must match one of the characters in the class
  - A range (in ASCII sequence) is denoted by "–"
  - If the first character is "^", the class consists of ***all but*** the specified characters
- Examples:

`[abc]`	matches "a", "b", or "c"
`[a-zA-Z0-9]`	match the alphanumerics
`[^ ]`	all but spaces
`[^-_a-z0-9A-Z]`	all but alphanumerics, "-", "_"
`[^^]`	all but caret (^)
`[[\]]`	match square brackets only



Predefined Character Classes

Code	Description	Equivalent to
`.`	(*period*) match any but newline	`[^\n]`
`\w`	match any ASCII word character	`[a-zA-Z0-9]`
`\W`	match any non-word character	`[^a-zA-Z0-9]`
`\s`	match any white-space character	`[ \f\n\r\t\v]`
`\S`	match any non-white space	`[^ \f\n\r\t\v]`
`\d`	match any decimal digit	`[0-9]`
`\D`	match any non-decimal digit	`[^0-9]`
`\R`	newline sequence	`\r\n|\r|\n`
`\N`	character other than newline \n	`[^\n]`



RE Quantifiers (Pattern Repetition)

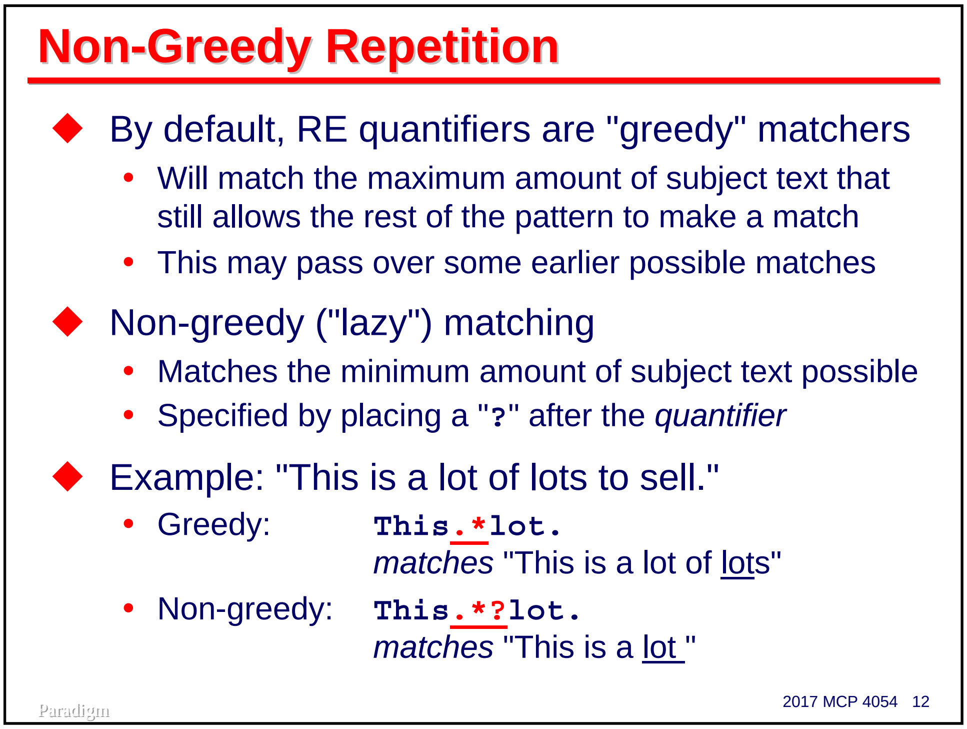
- Quantifiers specify how many times the *immediately preceding* element can be repeated
- Syntax:

`{`*n*`}`	exactly *n* times
`{`*n*`,}`	at least *n* times
`{,`*n*`}`	not more than *n* times
`{`*n*`,`*m*`}`	at least *n* times but not more than *m*
`?`	optional – zero or one times = `{0,1}`
`*`	zero or more times = `{0,}`
`+`	one or more times = `{1,}`

- Examples:
  - `\s+`
  - `\s*[0123456789]+\.\d{2}\s?`

Paradigm

2017 MCP 4054 11



Non-Greedy Repetition

- By default, RE quantifiers are "greedy" matchers
  - Will match the maximum amount of subject text that still allows the rest of the pattern to make a match
  - This may pass over some earlier possible matches
- Non-greedy ("lazy") matching
  - Matches the minimum amount of subject text possible
  - Specified by placing a "`?`" after the *quantifier*
- Example: "This is a lot of lots to sell."
  - Greedy: `This.*lot.`
    matches "This is a lot of lots"
  - Non-greedy: `This.*?lot.`
    matches "This is a lot "

Paradigm

2017 MCP 4054 12

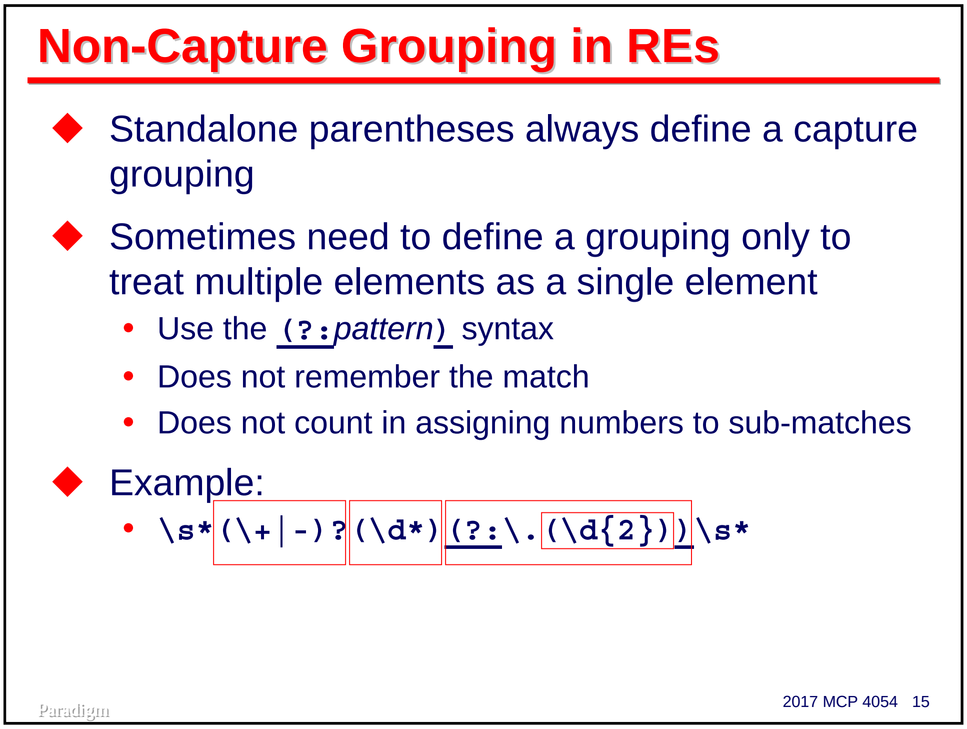
Alternation Matching in REs

- Alternation allows a match on one of a list of sub-patterns
  - Uses the "`|`" operator (vertical bar)
  - Has the lowest precedence of all pattern operators
  - Evaluated left-to-right
  - If the left sub-pattern matches, the right is ignored
  - Very powerful when combined with "grouping"
- Examples:
  - **`A|B|C`** same as **`[ABC]`**
  - **`this|that|something else`**
  - **`\s*\+?|-\d+\s*`**

Standard (Capture) Grouping in REs

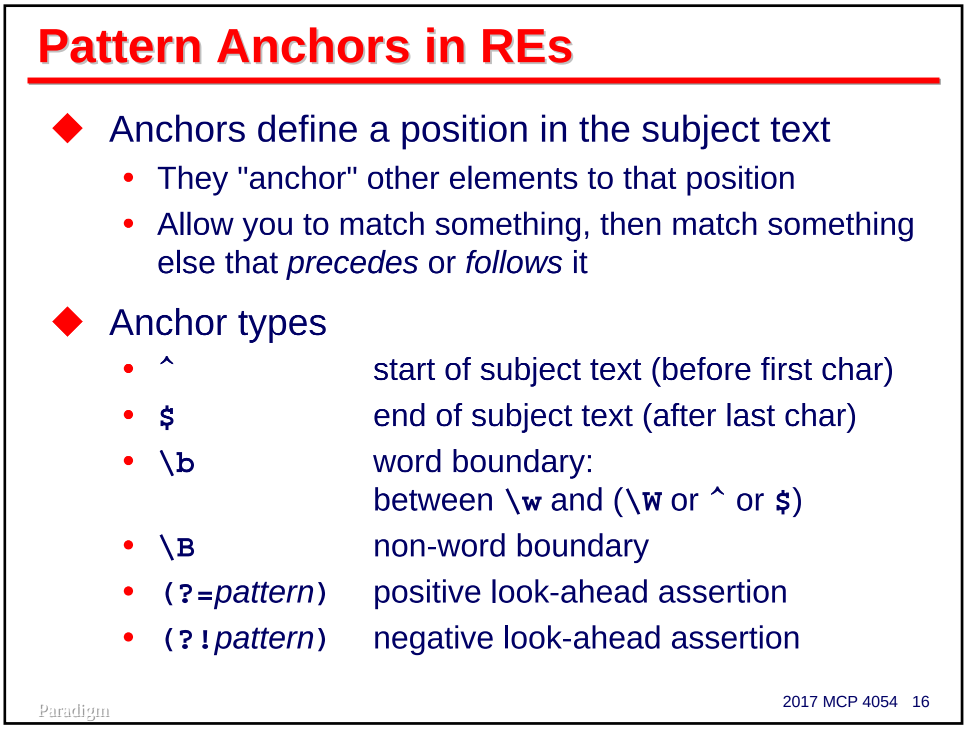
- Parentheses "`()`" in a RE perform two functions:
  - Group elements so they can be treated as a unit
    - Repetition
    - Alternation, etc.
  - Define "sub-matches" that are remembered
    - Can be retrieved after the match is complete
    - Can also be used as "back references" in the same RE: **`\1`**, **`\2`**, etc.
    - Numbers are assigned by counting the "`(`"s
    - Example: **`(['"])([^'"])+(\1)`**
- Parentheses can be nested

`\1` `\2` `\3`



Non-Capture Grouping in REs

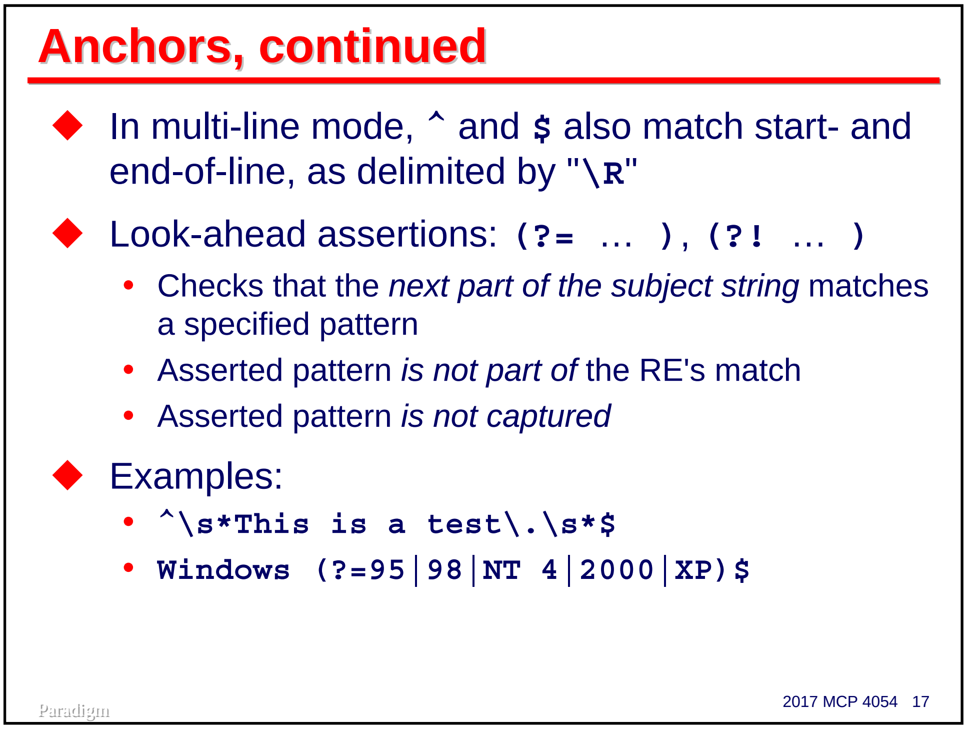
- Standalone parentheses always define a capture grouping
- Sometimes need to define a grouping only to treat multiple elements as a single element
  - Use the `(?:`*pattern*`)` syntax
  - Does not remember the match
  - Does not count in assigning numbers to sub-matches
- Example:
  - `\s*(\+|-)?(\d*)(?:\.(\d{2}))\s*`



Pattern Anchors in REs

- Anchors define a position in the subject text
  - They "anchor" other elements to that position
  - Allow you to match something, then match something else that *precedes* or *follows* it
- Anchor types

`^`	start of subject text (before first char)
`$`	end of subject text (after last char)
`\b`	word boundary: between `\w` and (`\W` or `^` or `$`)
`\B`	non-word boundary
`(?=`*pattern*`)`	positive look-ahead assertion
`(?!`*pattern*`)`	negative look-ahead assertion

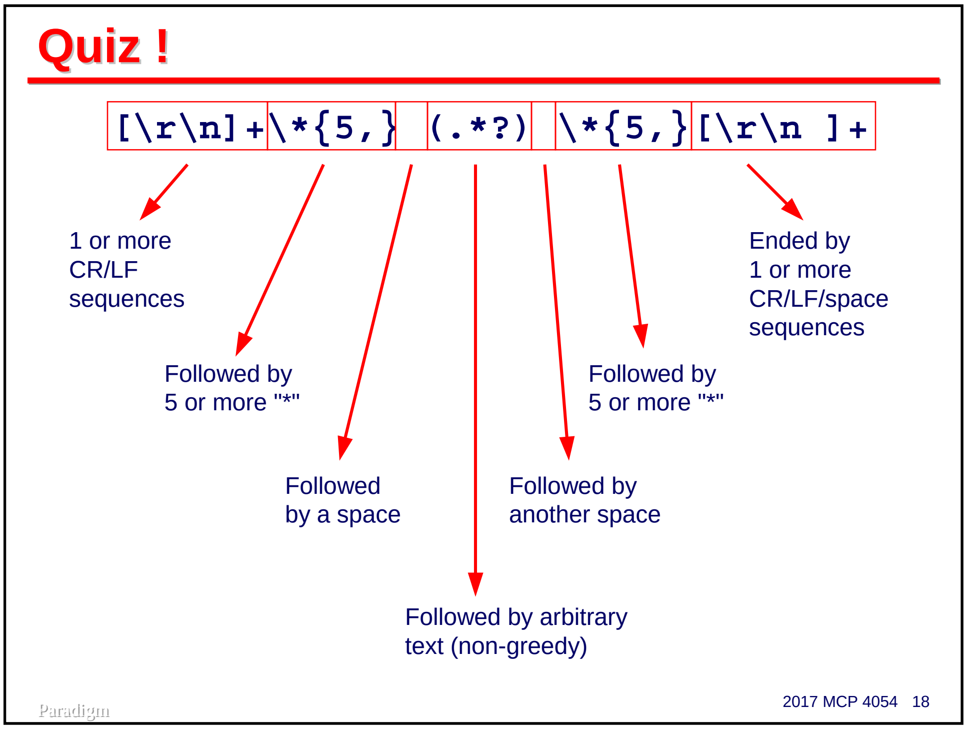


Anchors, continued

- In multi-line mode, **^** and **$** also match start- and end-of-line, as delimited by "**\R**"
- Look-ahead assertions: `(?= … )`, `(?! … )`
  - Checks that the *next part of the subject string* matches a specified pattern
  - Asserted pattern *is not part of* the RE's match
  - Asserted pattern *is not captured*
- Examples:
  - `^\s*This is a test\.\s*$`
  - `Windows (?=95|98|NT 4|2000|XP)$`

Paradigm

2017 MCP 4054 17



Quiz !

```
[\r\n]+\*{5,} (.*?) \*{5,}[\r\n ]+
```

1 or more CR/LF sequences

Followed by 5 or more "*"

Followed by a space

Followed by arbitrary text (non-greedy)

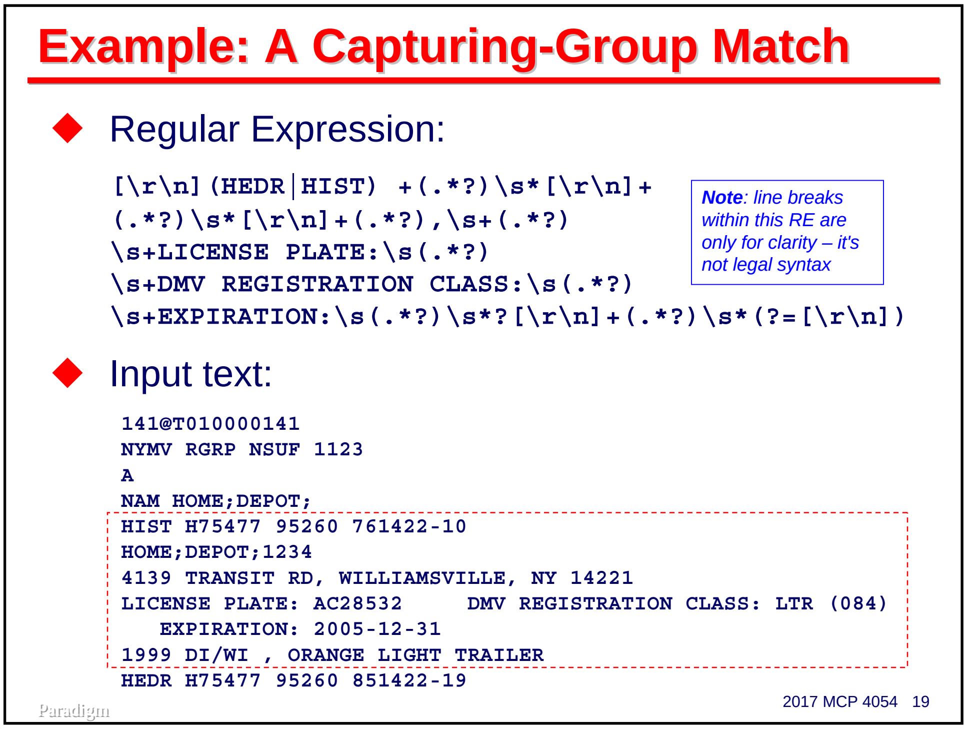
Followed by another space

Followed by 5 or more "*"

Ended by 1 or more CR/LF/space sequences

Paradigm

2017 MCP 4054 18



Example: A Capturing-Group Match

- Regular Expression:

```
[\r\n](HEDR|HIST) +(.*?)\s*[\r\n]+
(.*?)\s*[\r\n]+(.*?),\s+(.*?)
\s+LICENSE PLATE:\s(.*?)
\s+DMV REGISTRATION CLASS:\s(.*?)
\s+EXPIRATION:\s(.*?)\s*?[\r\n]+(.*?)\s*(?=[\r\n])
```

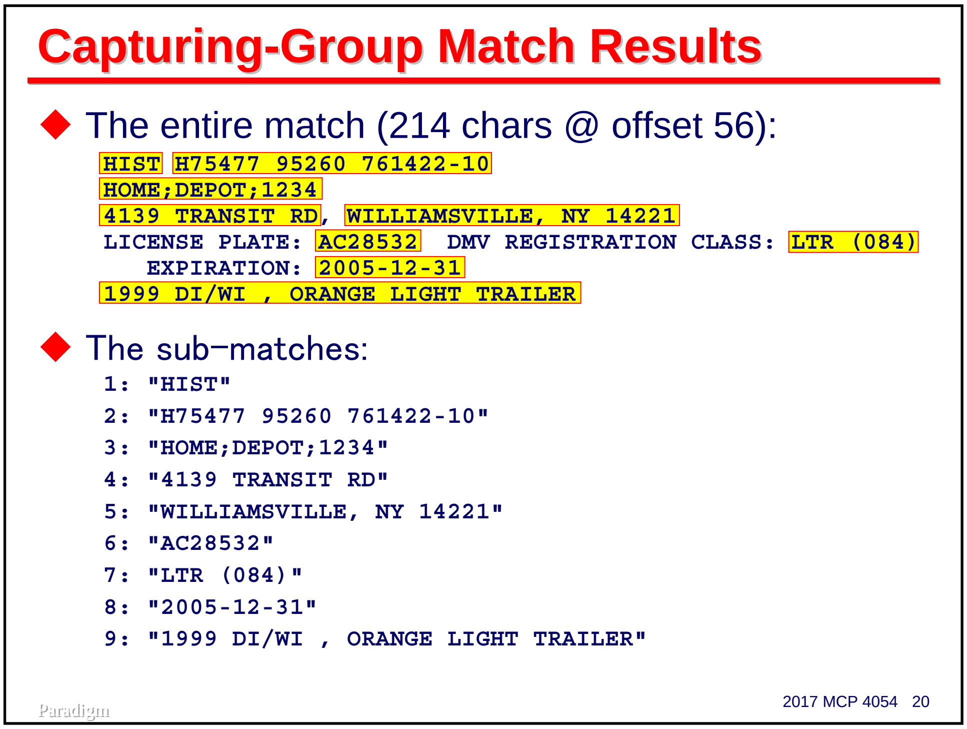
***Note**: line breaks within this RE are only for clarity – it's not legal syntax*

- Input text:

```
141@T010000141
NYMV RGRP NSUF 1123
A
NAM HOME;DEPOT;
HIST H75477 95260 761422-10
HOME;DEPOT;1234
4139 TRANSIT RD, WILLIAMSVILLE, NY 14221
LICENSE PLATE: AC28532     DMV REGISTRATION CLASS: LTR (084)
   EXPIRATION: 2005-12-31
1999 DI/WI , ORANGE LIGHT TRAILER
HEDR H75477 95260 851422-19
```

Paradigm

2017 MCP 4054 19



Capturing-Group Match Results

- The entire match (214 chars @ offset 56):

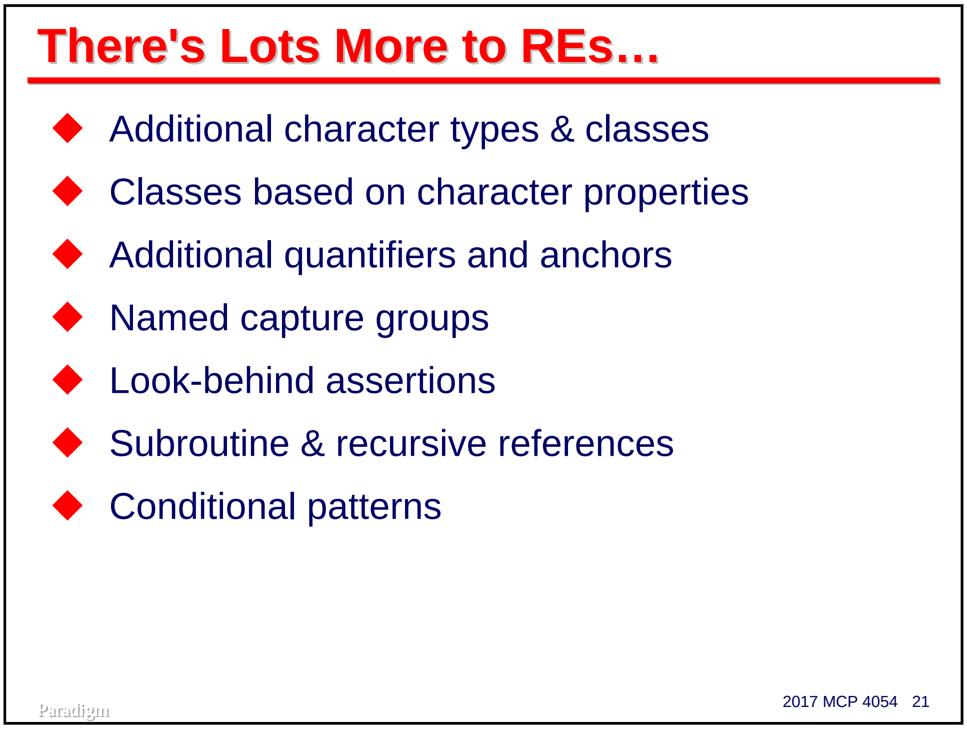
```
HIST H75477 95260 761422-10
HOME;DEPOT;1234
4139 TRANSIT RD, WILLIAMSVILLE, NY 14221
LICENSE PLATE: AC28532  DMV REGISTRATION CLASS: LTR (084)
   EXPIRATION: 2005-12-31
1999 DI/WI , ORANGE LIGHT TRAILER
```

- The sub-matches:

```
1: "HIST"
2: "H75477 95260 761422-10"
3: "HOME;DEPOT;1234"
4: "4139 TRANSIT RD"
5: "WILLIAMSVILLE, NY 14221"
6: "AC28532"
7: "LTR (084)"
8: "2005-12-31"
9: "1999 DI/WI , ORANGE LIGHT TRAILER"
```

Paradigm

2017 MCP 4054 20

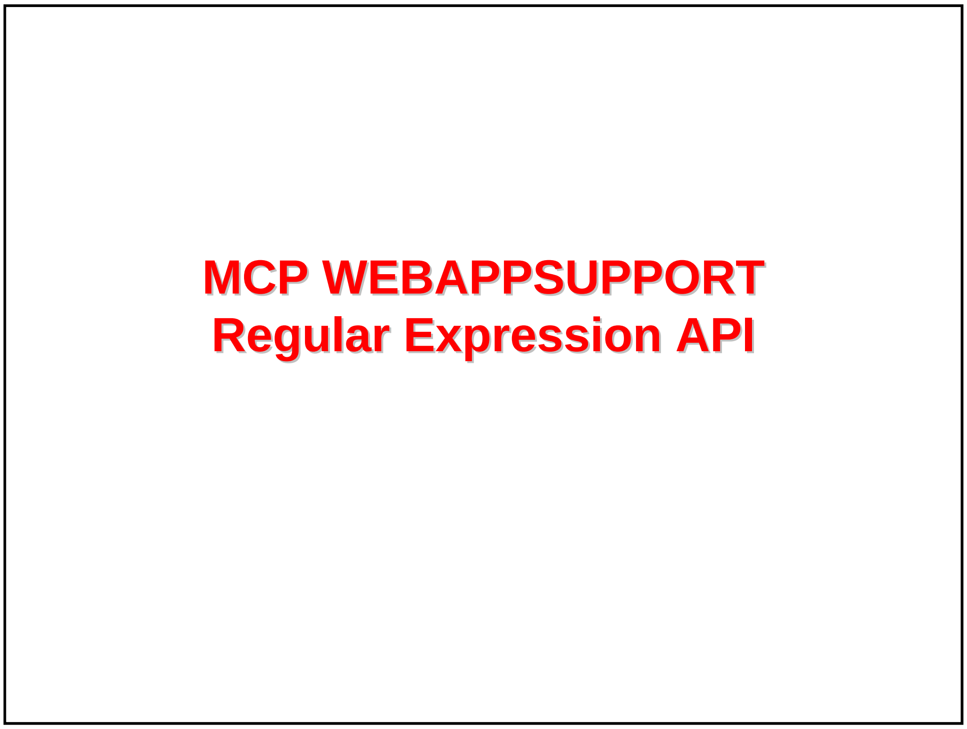


There's Lots More to REs…

- Additional character types & classes
- Classes based on character properties
- Additional quantifiers and anchors
- Named capture groups
- Look-behind assertions
- Subroutine & recursive references
- Conditional patterns

Paradigm

2017 MCP 4054 21



MCP WEBAPPSUPPORT Regular Expression API

Regular Expressions for MCP

- Available starting in MCP 13.0
- Based on **P**erl **C**ompatible **R**egular **E**xpressions
  - http://www.pcre.org/
  - Free, open-source C library
  - MCP uses original PCRE, not PCRE2
- MCP implementation limitations
  - Maximum length of subject string = 15.5 MB
  - Maximum length of pattern = 31 KB
  - PCRE "callouts" not supported
- Two sets of WEBAPPSUPPORT procedures:
  - ALGOL-friendly parameters
  - COBOL-friendly parameters

Paradigm

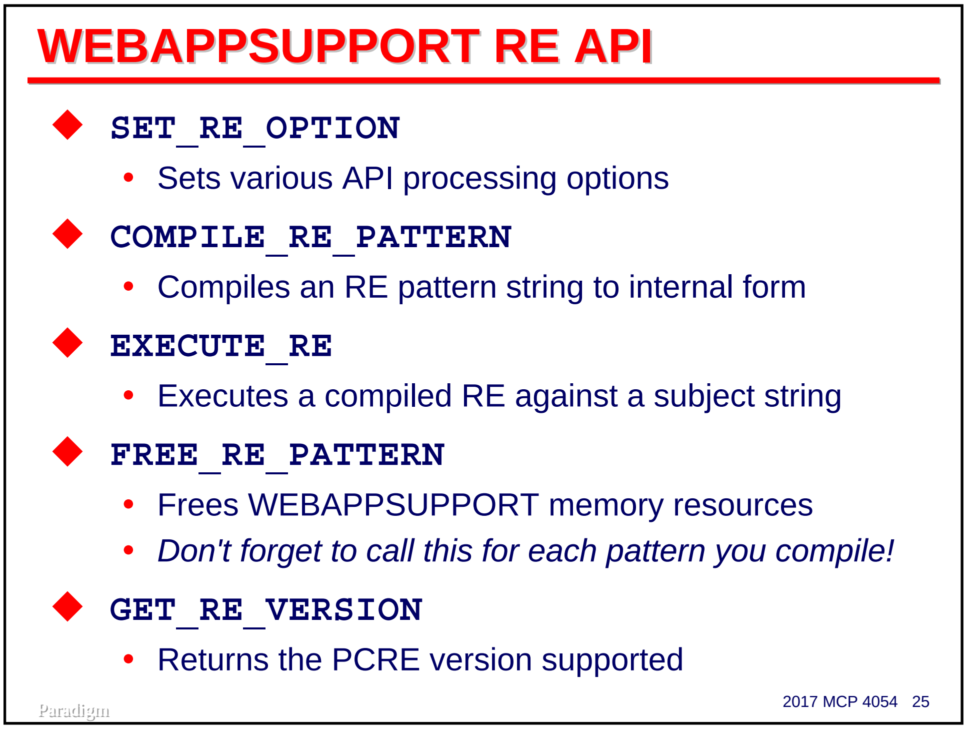
2017 MCP 4054 23

Character Set Handling

- REs are processed using the "application character set"
  - Character set used by the app calling the RE API
  - Default for non-WEBPCM apps is ASERIESEBCDIC
- Subject and pattern strings must be:
  - Unicode (UTF-8, UCS2) ***or —***
  - Any character set MCP MLS can translate to ASCII or UCS2 (includes ASERIESEBCDIC, ASCII, Latin1ISO)
- Application character set can be changed:
  - WEBAPPSUPPORT `SET_TRANSLATION` procedure
  - Takes an MLS CCS number

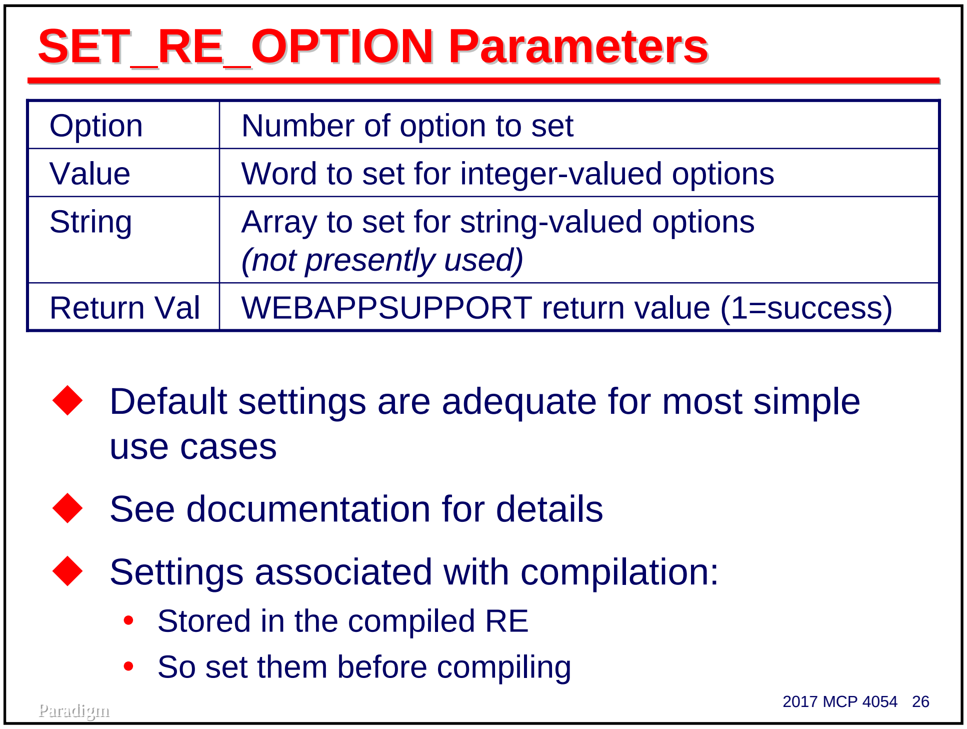
Paradigm

2017 MCP 4054 24



WEBAPPSUPPORT RE API

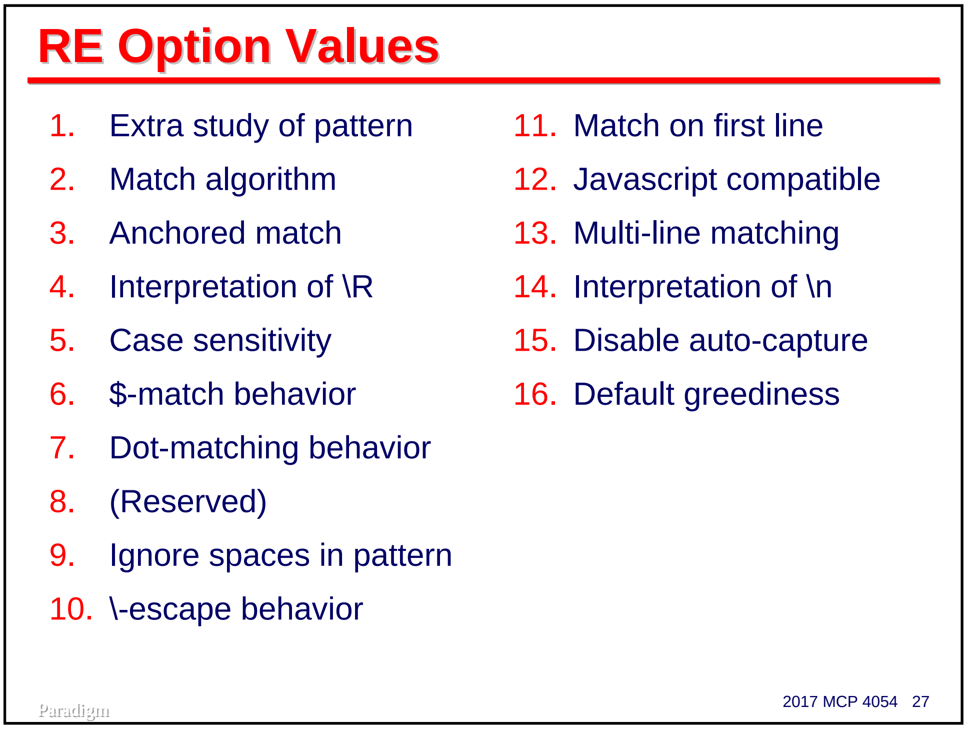
- **`SET_RE_OPTION`**
  - Sets various API processing options
- **`COMPILE_RE_PATTERN`**
  - Compiles an RE pattern string to internal form
- **`EXECUTE_RE`**
  - Executes a compiled RE against a subject string
- **`FREE_RE_PATTERN`**
  - Frees WEBAPPSUPPORT memory resources
  - *Don't forget to call this for each pattern you compile!*
- **`GET_RE_VERSION`**
  - Returns the PCRE version supported



SET_RE_OPTION Parameters

Option	Number of option to set
Value	Word to set for integer-valued options
String	Array to set for string-valued options *(not presently used)*
Return Val	WEBAPPSUPPORT return value (1=success)

- Default settings are adequate for most simple use cases
- See documentation for details
- Settings associated with compilation:
  - Stored in the compiled RE
  - So set them before compiling

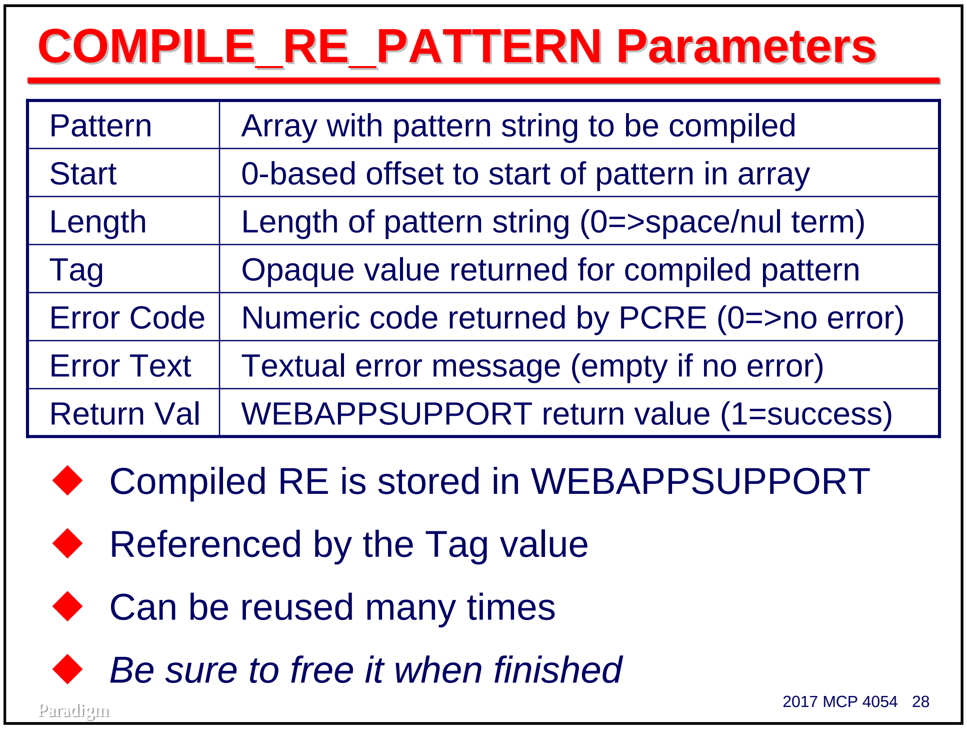


RE Option Values

1. Extra study of pattern
2. Match algorithm
3. Anchored match
4. Interpretation of \R
5. Case sensitivity
6. $-match behavior
7. Dot-matching behavior
8. (Reserved)
9. Ignore spaces in pattern
10. \-escape behavior
11. Match on first line
12. Javascript compatible
13. Multi-line matching
14. Interpretation of \n
15. Disable auto-capture
16. Default greediness

Paradigm

2017 MCP 4054 27



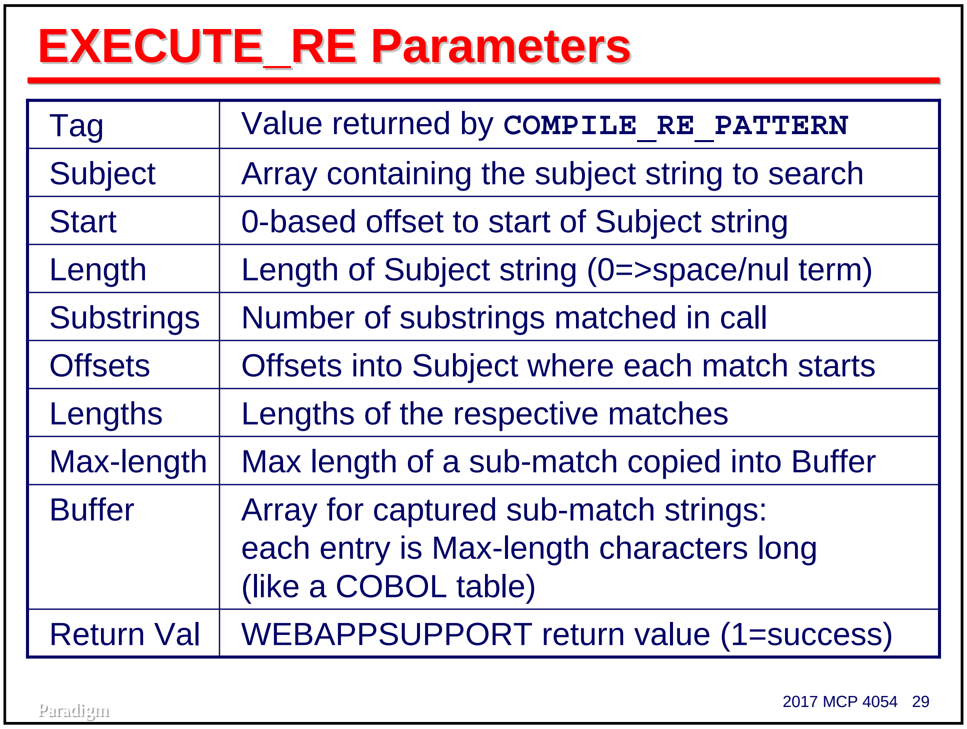
COMPILE_RE_PATTERN Parameters

Pattern	Array with pattern string to be compiled
Start	0-based offset to start of pattern in array
Length	Length of pattern string (0=>space/nul term)
Tag	Opaque value returned for compiled pattern
Error Code	Numeric code returned by PCRE (0=>no error)
Error Text	Textual error message (empty if no error)
Return Val	WEBAPPSUPPORT return value (1=success)

- Compiled RE is stored in WEBAPPSUPPORT
- Referenced by the Tag value
- Can be reused many times
- *Be sure to free it when finished*

Paradigm

2017 MCP 4054 28

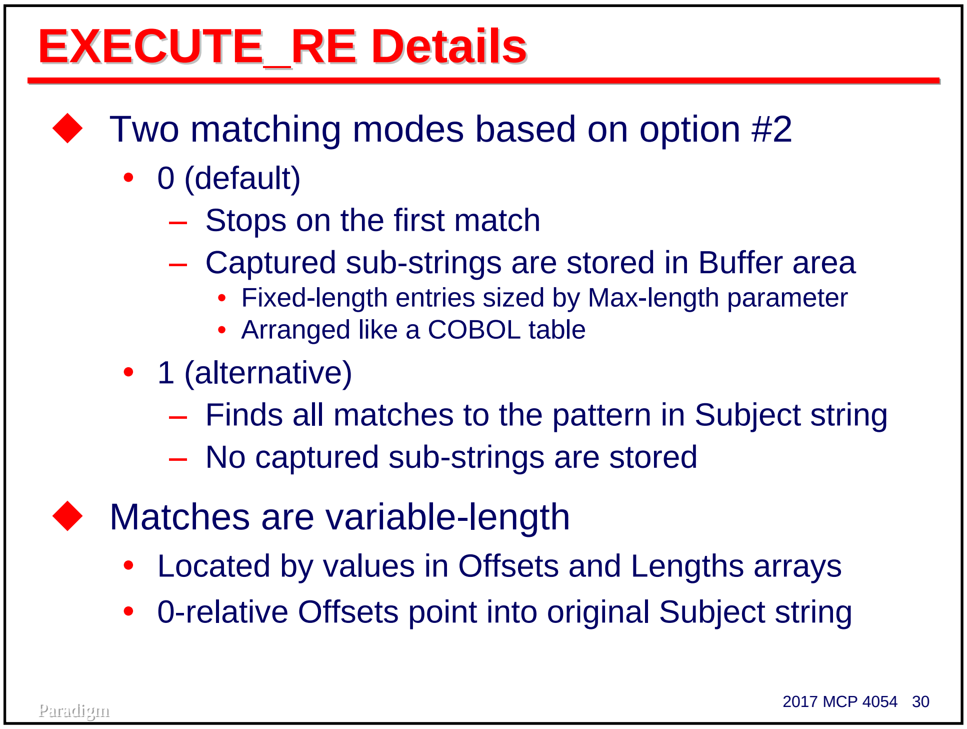


EXECUTE_RE Parameters

Tag	Value returned by **`COMPILE_RE_PATTERN`**
Subject	Array containing the subject string to search
Start	0-based offset to start of Subject string
Length	Length of Subject string (0=>space/nul term)
Substrings	Number of substrings matched in call
Offsets	Offsets into Subject where each match starts
Lengths	Lengths of the respective matches
Max-length	Max length of a sub-match copied into Buffer
Buffer	Array for captured sub-match strings: each entry is Max-length characters long (like a COBOL table)
Return Val	WEBAPPSUPPORT return value (1=success)

Paradigm

2017 MCP 4054 29

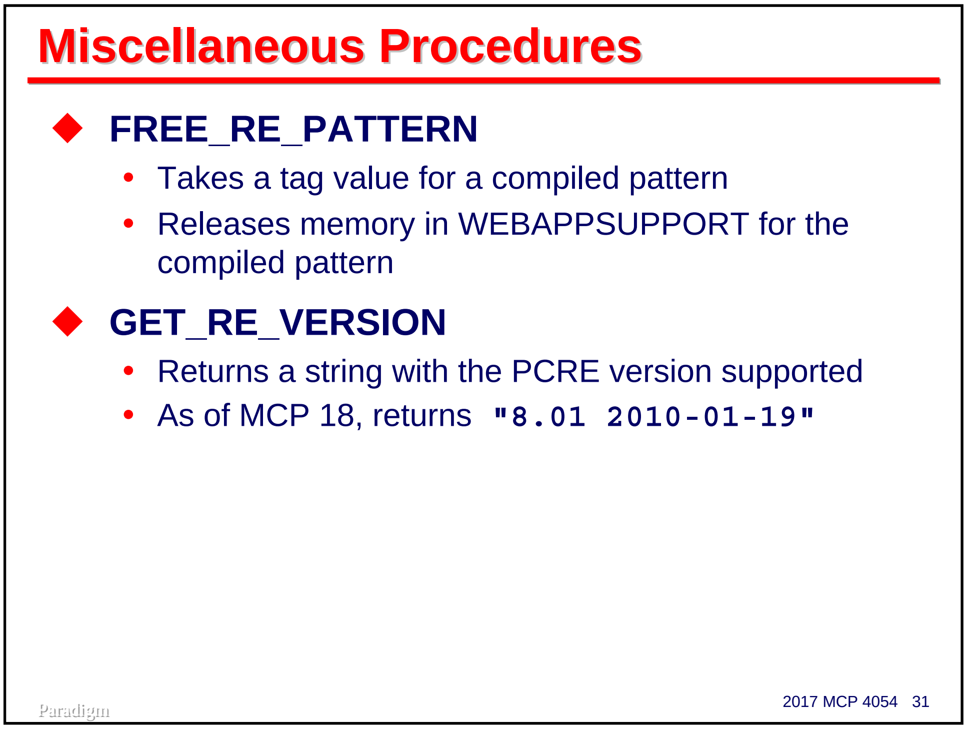


EXECUTE_RE Details

- Two matching modes based on option #2
  - 0 (default)
    - Stops on the first match
    - Captured sub-strings are stored in Buffer area
      - Fixed-length entries sized by Max-length parameter
      - Arranged like a COBOL table
  - 1 (alternative)
    - Finds all matches to the pattern in Subject string
    - No captured sub-strings are stored
- Matches are variable-length
  - Located by values in Offsets and Lengths arrays
  - 0-relative Offsets point into original Subject string

Paradigm

2017 MCP 4054 30

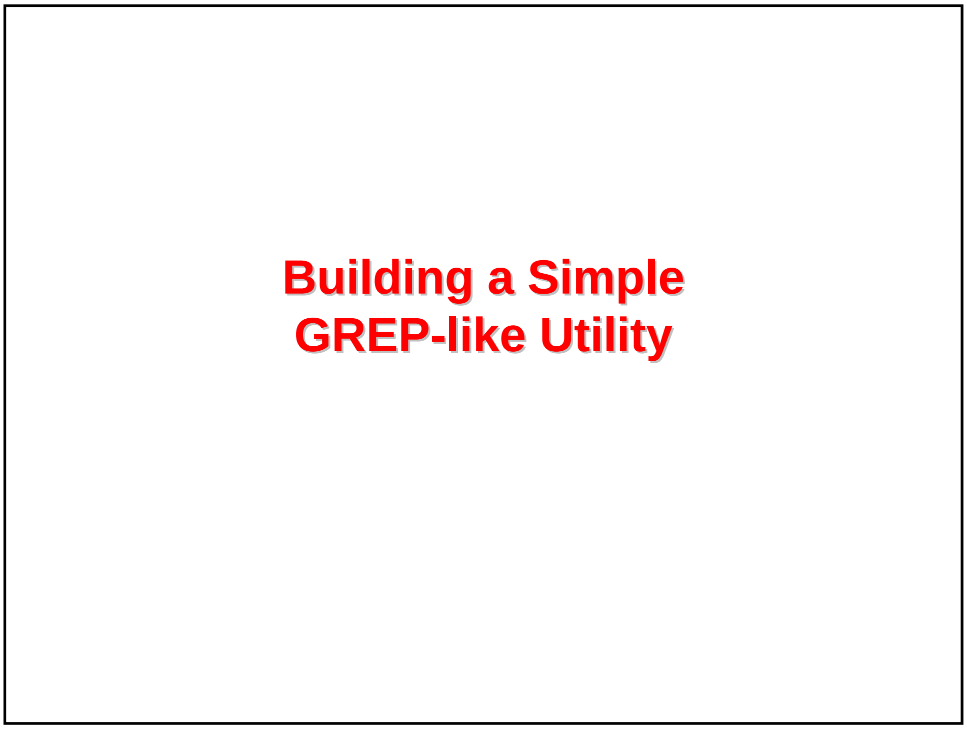


Miscellaneous Procedures

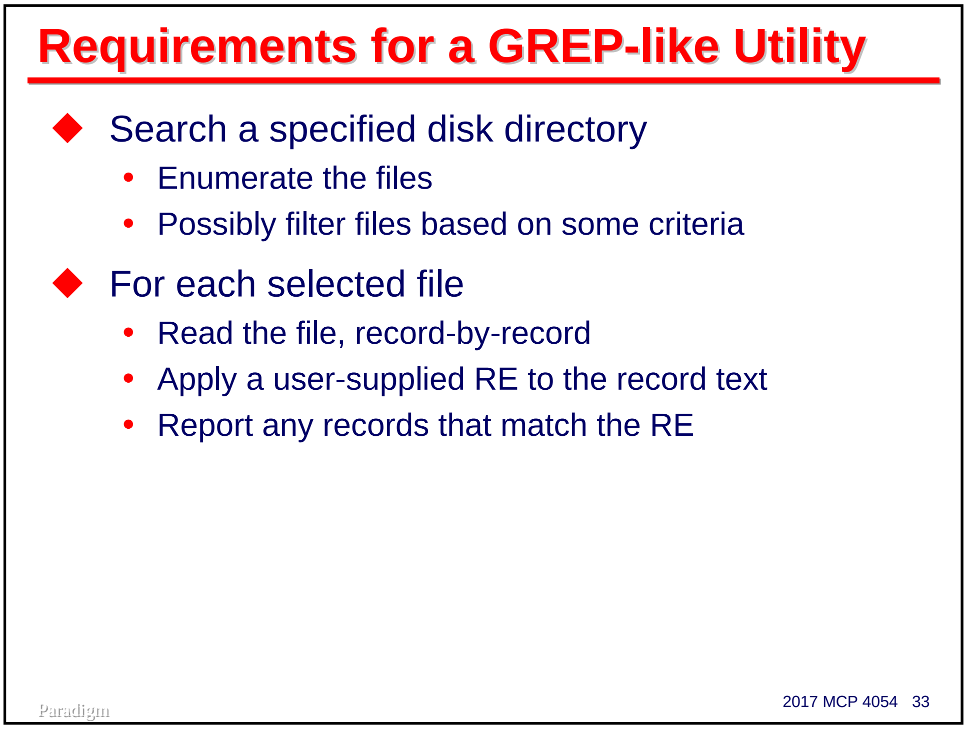
- **FREE_RE_PATTERN**
  - Takes a tag value for a compiled pattern
  - Releases memory in WEBAPPSUPPORT for the compiled pattern
- **GET_RE_VERSION**
  - Returns a string with the PCRE version supported
  - As of MCP 18, returns `"8.01 2010-01-19"`

Paradigm

2017 MCP 4054 31



Building a Simple GREP-like Utility

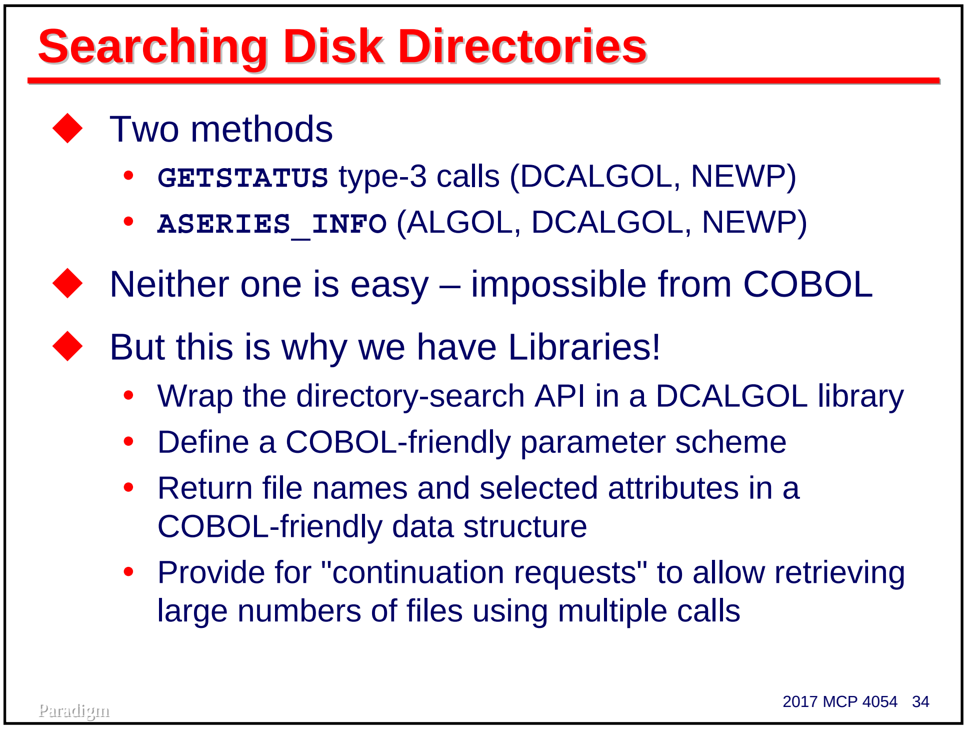


Requirements for a GREP-like Utility

- Search a specified disk directory
  - Enumerate the files
  - Possibly filter files based on some criteria
- For each selected file
  - Read the file, record-by-record
  - Apply a user-supplied RE to the record text
  - Report any records that match the RE

Paradigm

2017 MCP 4054 33

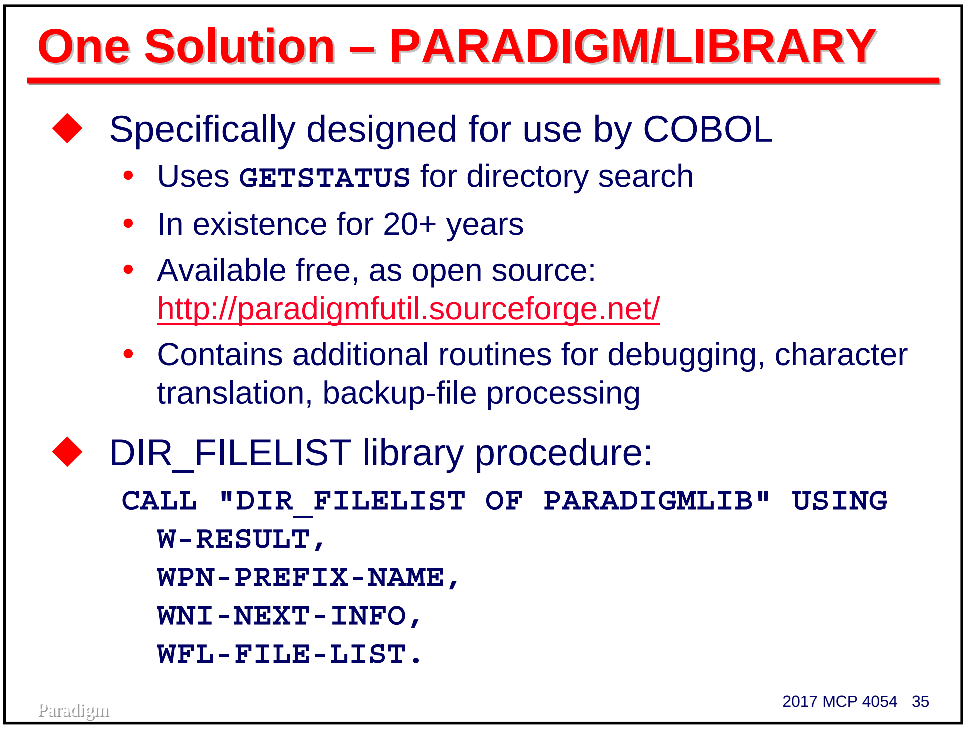


Searching Disk Directories

- Two methods
  - **`GETSTATUS`** type-3 calls (DCALGOL, NEWP)
  - **`ASERIES_INFO`** (ALGOL, DCALGOL, NEWP)
- Neither one is easy – impossible from COBOL
- But this is why we have Libraries!
  - Wrap the directory-search API in a DCALGOL library
  - Define a COBOL-friendly parameter scheme
  - Return file names and selected attributes in a COBOL-friendly data structure
  - Provide for "continuation requests" to allow retrieving large numbers of files using multiple calls

Paradigm

2017 MCP 4054 34



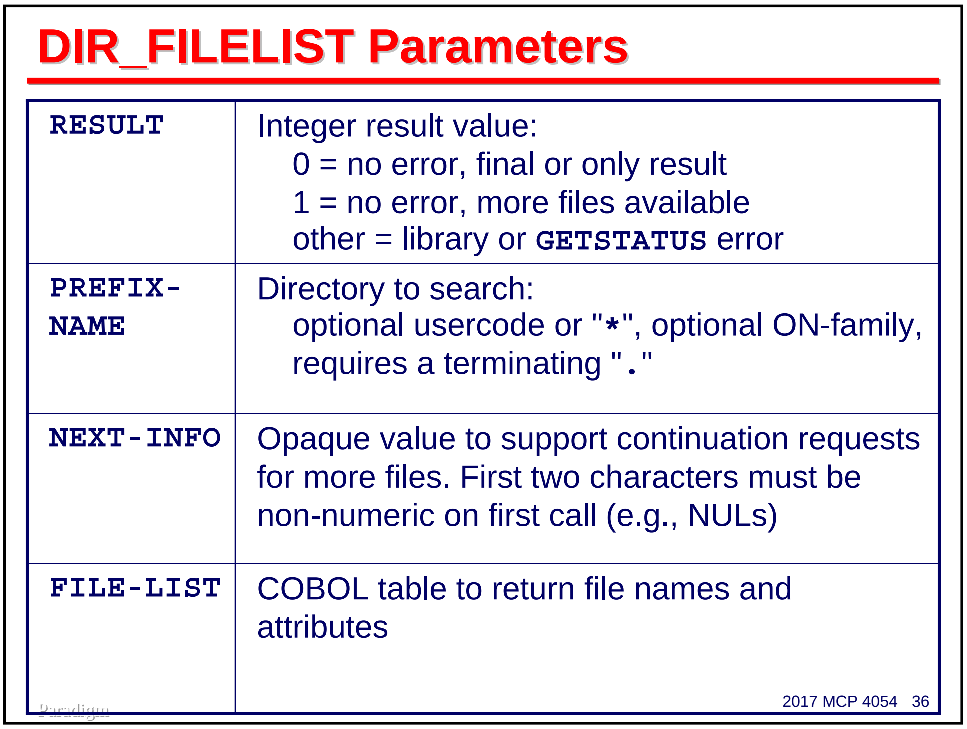
One Solution – PARADIGM/LIBRARY

- Specifically designed for use by COBOL
  - Uses **`GETSTATUS`** for directory search
  - In existence for 20+ years
  - Available free, as open source: http://paradigmfutil.sourceforge.net/
  - Contains additional routines for debugging, character translation, backup-file processing
- DIR_FILELIST library procedure:

```
CALL "DIR_FILELIST OF PARADIGMLIB" USING
  W-RESULT,
  WPN-PREFIX-NAME,
  WNI-NEXT-INFO,
  WFL-FILE-LIST.
```

Paradigm

2017 MCP 4054 35

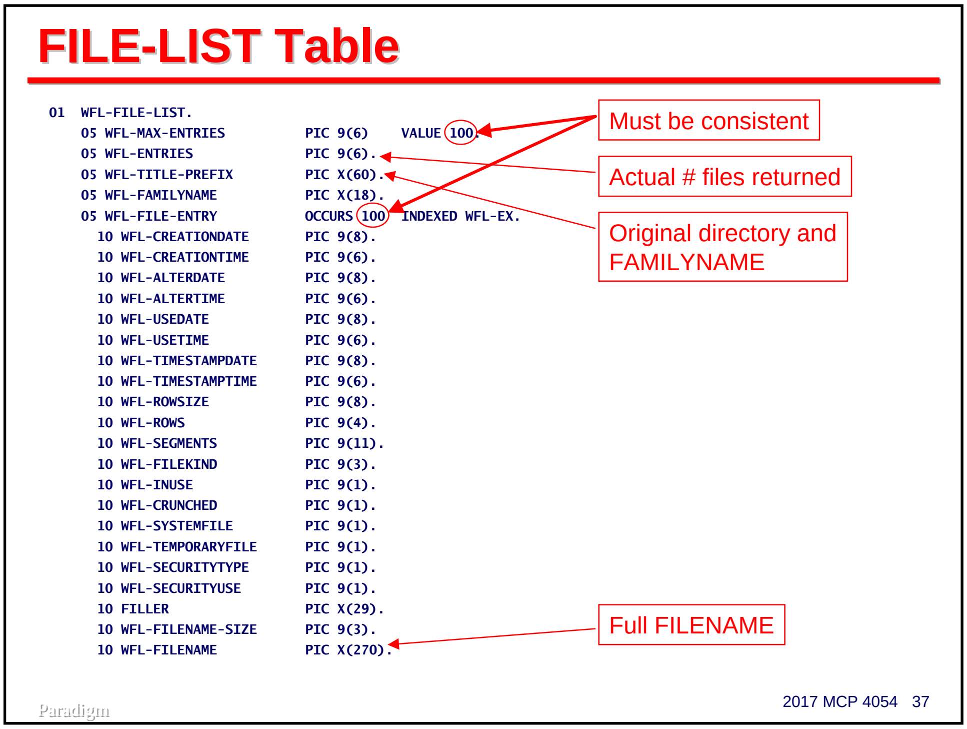


DIR_FILELIST Parameters

RESULT	Integer result value: 0 = no error, final or only result 1 = no error, more files available other = library or **GETSTATUS** error
PREFIX-NAME	Directory to search: optional usercode or "*****", optional ON-family, requires a terminating "**.**"
NEXT-INFO	Opaque value to support continuation requests for more files. First two characters must be non-numeric on first call (e.g., NULs)
FILE-LIST	COBOL table to return file names and attributes

Paradigm

2017 MCP 4054 36



FILE-LIST Table

```
01  WFL-FILE-LIST.
    05 WFL-MAX-ENTRIES            PIC 9(6)    VALUE 100.
    05 WFL-ENTRIES                PIC 9(6).
    05 WFL-TITLE-PREFIX           PIC X(60).
    05 WFL-FAMILYNAME             PIC X(18).
    05 WFL-FILE-ENTRY             OCCURS 100  INDEXED WFL-EX.
      10 WFL-CREATIONDATE         PIC 9(8).
      10 WFL-CREATIONTIME         PIC 9(6).
      10 WFL-ALTERDATE            PIC 9(8).
      10 WFL-ALTERTIME            PIC 9(6).
      10 WFL-USEDATE              PIC 9(8).
      10 WFL-USETIME              PIC 9(6).
      10 WFL-TIMESTAMPDATE        PIC 9(8).
      10 WFL-TIMESTAMPTIME        PIC 9(6).
      10 WFL-ROWSIZE              PIC 9(8).
      10 WFL-ROWS                 PIC 9(4).
      10 WFL-SEGMENTS             PIC 9(11).
      10 WFL-FILEKIND             PIC 9(3).
      10 WFL-INUSE                PIC 9(1).
      10 WFL-CRUNCHED             PIC 9(1).
      10 WFL-SYSTEMFILE           PIC 9(1).
      10 WFL-TEMPORARYFILE        PIC 9(1).
      10 WFL-SECURITYTYPE         PIC 9(1).
      10 WFL-SECURITYUSE          PIC 9(1).
      10 FILLER                   PIC X(29).
      10 WFL-FILENAME-SIZE        PIC 9(3).
      10 WFL-FILENAME             PIC X(270).
```

Must be consistent

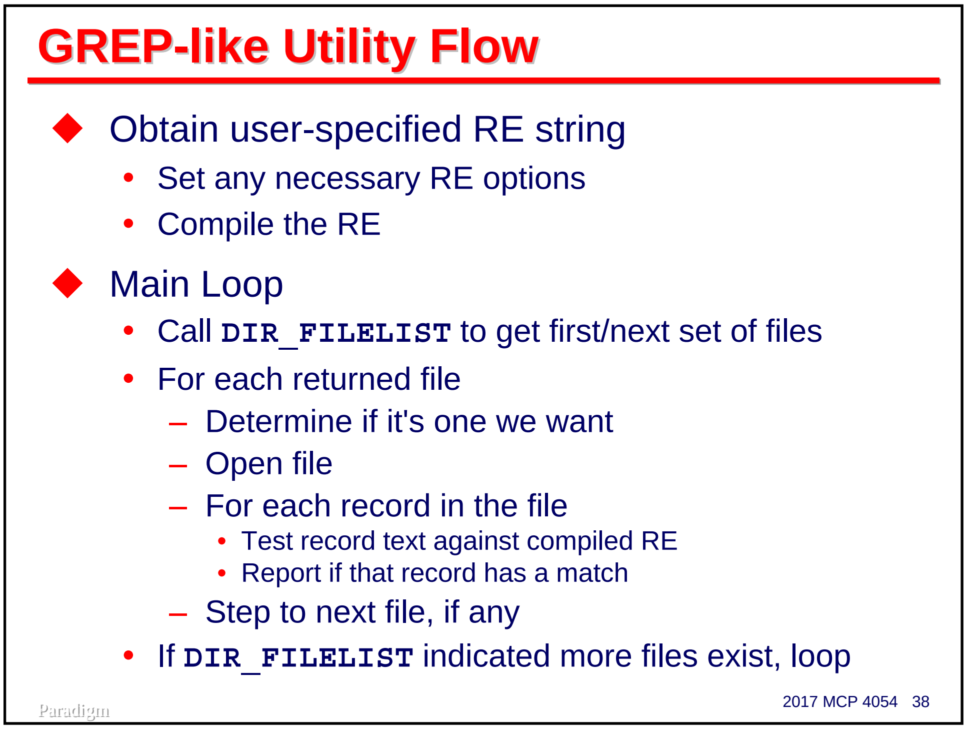
Actual # files returned

Original directory and FAMILYNAME

Full FILENAME

Paradigm

2017 MCP 4054   37

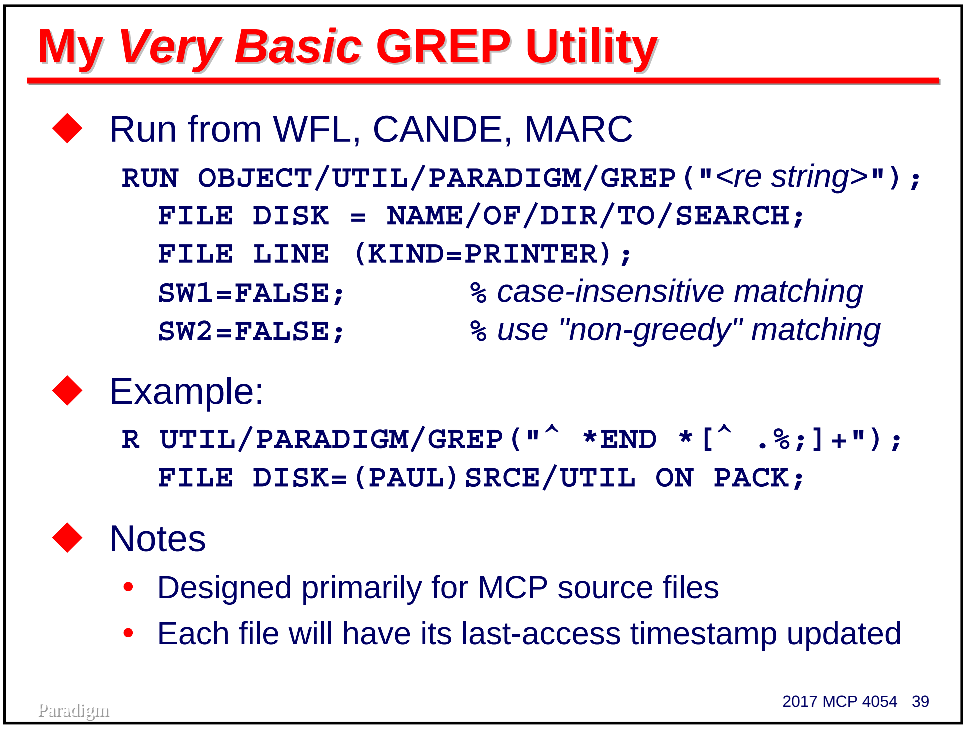


GREP-like Utility Flow

- Obtain user-specified RE string
  - Set any necessary RE options
  - Compile the RE
- Main Loop
  - Call **`DIR_FILELIST`** to get first/next set of files
  - For each returned file
    - Determine if it's one we want
    - Open file
    - For each record in the file
      - Test record text against compiled RE
      - Report if that record has a match
    - Step to next file, if any
  - If **`DIR_FILELIST`** indicated more files exist, loop

Paradigm

2017 MCP 4054   38



My *Very Basic* GREP Utility

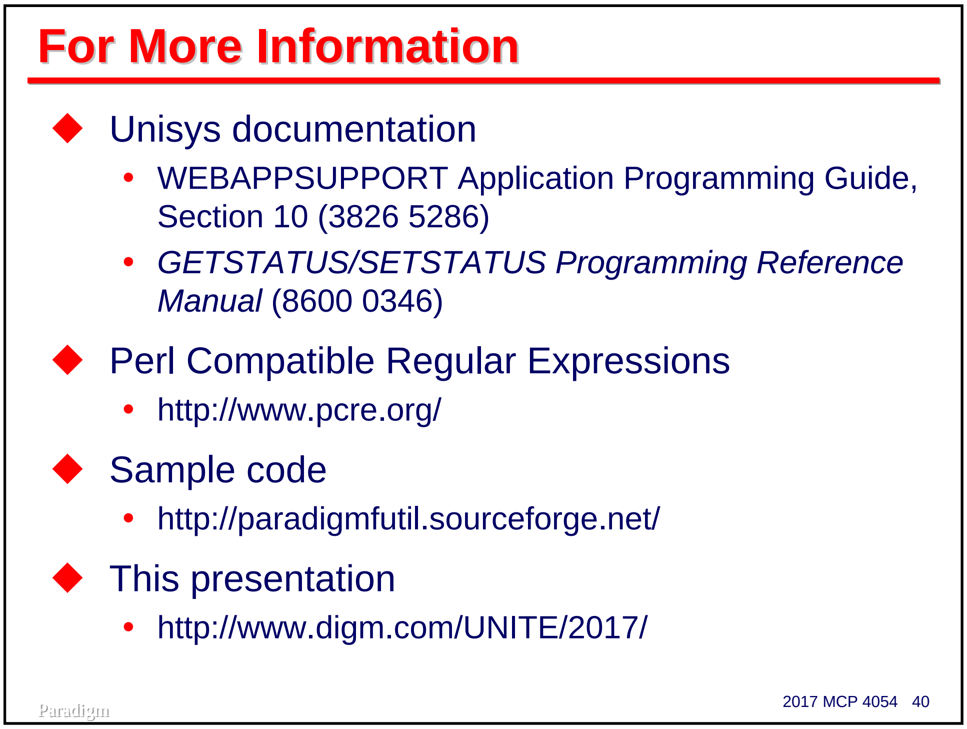
- Run from WFL, CANDE, MARC

```
RUN OBJECT/UTIL/PARADIGM/GREP("<re string>");
  FILE DISK = NAME/OF/DIR/TO/SEARCH;
  FILE LINE (KIND=PRINTER);
  SW1=FALSE;        % case-insensitive matching
  SW2=FALSE;        % use "non-greedy" matching
```

- Example:

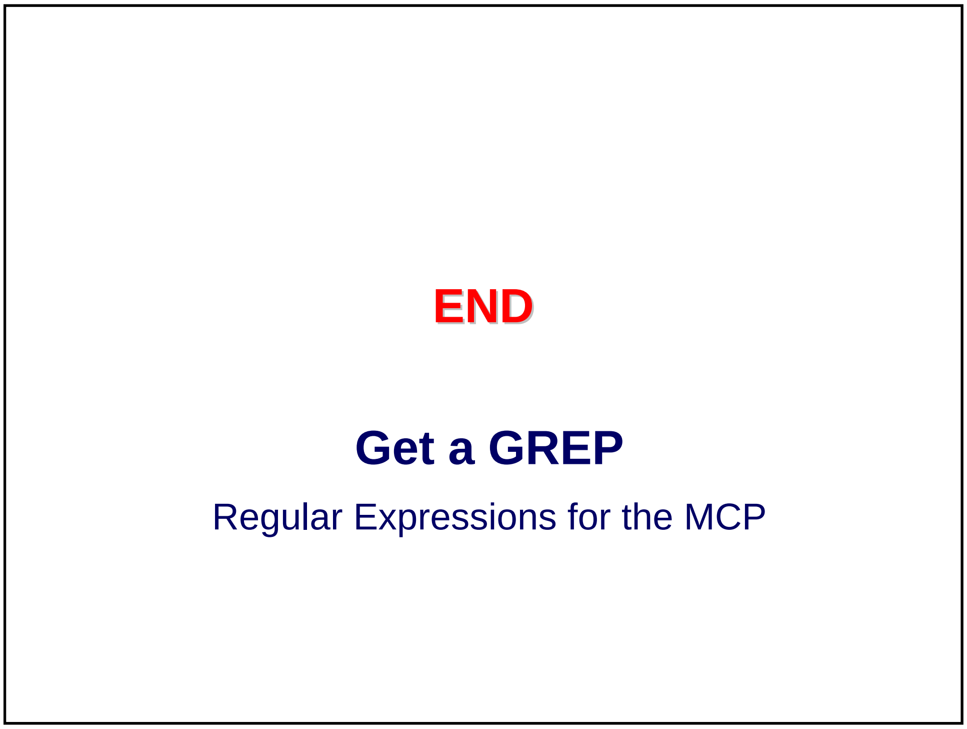
```
R UTIL/PARADIGM/GREP("^ *END *[^ .%;]+");
  FILE DISK=(PAUL)SRCE/UTIL ON PACK;
```

- Notes
  - Designed primarily for MCP source files
  - Each file will have its last-access timestamp updated



For More Information

- Unisys documentation
  - WEBAPPSUPPORT Application Programming Guide, Section 10 (3826 5286)
  - *GETSTATUS/SETSTATUS Programming Reference Manual* (8600 0346)
- Perl Compatible Regular Expressions
  - http://www.pcre.org/
- Sample code
  - http://paradigmfutil.sourceforge.net/
- This presentation
  - http://www.digm.com/UNITE/2017/



END

Get a GREP

Regular Expressions for the MCP