Shell Scripting: Metacharacters

Dr. Vimal Baghel, Assistant Professor, SCSET, BU

Metacharacters

- A metacharacter is a character ***that has a special meaning during pattern processing.***
- We use metacharacters in ***regular expressions*** to define the search criteria and any text manipulations.

\ \

["

✓ All characters enclosed between single quotation marks are considered quoted and are interpreted literally by the shell.

✓ The special meaning of metacharacters is retained if not quoted.

'*'

Metacharacters

Metacharacters

- ✓pipe (|),
- ✓ampersand (&),
- ✓semicolon (;),
- ✓less-than sign (<),
- ✓greater-than sign (>),
- ✓left parenthesis ((),
- ✓right parenthesis ()),

Metacharacters

- ✓dollar sign ($),
- ✓backquote (`),
- ✓backslash (\),
- ✓right quote (‘),
- ✓double quotation marks ("),
- ✓newline character \n,
- ✓space character , and tab character \t.

Types of Metacharacters

✓***Search string*** metacharacters

✓***Replacement string*** metacharacters.

Search string Metacharacters

Metacharacter	Action
^	*Beginning* of line
$	*End* of line
|	Or Not applicable to basic regular expressions.
[abc]	Match any character enclosed in the brackets
[^abc]	Match any character not enclosed in the brackets
[a-z]	Match the range of characters specified by the hyphen

a1 - aag

[abc

A A

Search string Metacharacters (classes)

[:*cclass*:]

- Use the character list that is specified by *cclass*:**alnum** = Uppercase and lowercase alphabetic characters and numbers: [A-Za-z0-9]
- **alpha** = Uppercase and lowercase alphabetic characters: [A-Za-z]
- **blank** = Whitespace and tab characters
- **cntrl** = Control characters
- **digit** = Numbers: [0-9]
- **lower** = Lowercase alphabetic characters: [a-z]
- **print** = Printable characters (the **graph** class plus whitespace)
- **punct** = Punctuation marks: !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
- **graph** = Visible characters (the **alnum** class plus the **punct** class)

Search string Metacharacters (classes)

- **space** = Whitespace characters: tab, newline, carriage-return, form-feed, and vertical-tab
- **upper** = Uppercase alphabetic characters: [A-Z]
- **xdigit** = Hexadecimal characters: [0-9a-fA-F]
- ***These classes are valid for single-byte character sets.***

[=*cname*=]	• Substitute the character name that is specified by *cname* with the corresponding character code.

Search string Metacharacters

.	Match any single character.
( )	Group the regular expression within the parentheses.
?	Match zero or one of the preceding expression. Not applicable to basic regular expressions.
*	Match zero, one, or many of the preceding expression.
+	Match one or many of the preceding expression.Not applicable to basic regular expressions.
\	Use the literal meaning of the metacharacter. For basic regular expressions, treat the next character as a metacharacter.

Replacement string metacharacters

Metacharacter	Action
&	• Reference the entire matched text for string substitution. • **For example**, the statement execute function regex_replace('abcdefg', '[af]', '.&.') replaces 'a' with '.a.' and 'f' with '.f.' to return: '.a.bcde.f.g'.

Replacement string metacharacters

\n

- Reference the subgroup *n* within the matched text, where *n* is an integer 0-9.
- \0 and & have identical actions.
- \1 - \9 substitute the corresponding subgroup.

Replacement string metacharacters

\	Use the literal meaning of the metacharacter, **for example**, \& escapes the Ampersand symbol and \\ escapes the backslash. For basic regular expressions, treat the next character as a metacharacter.

Metacharacters

- Most commonly used: *
- Search the current directory for file names in which any strings occurs in the position of *

```
% echo *   # same effect as
% ls *
```

- To protect metacharacters from being interpreted: enclose them in single quotes.

```
% echo ‘***’
***
```

Metacharacters (cont.)

- Or to put a backslash \ in front of each character:

```
% echo \*\*\*
***
```

- Double quotes can also be used to protect metacharacters, but …
- The shell will interpret **$,** \ and `…` inside the double quotes.
- So don’t use double quotes unless you intend some processing of the quoted string (to be discussed later).

Quotes

- Quotes do not have to surround the whole argument.

```
% echo x’*’y    # same as echo ‘x*y’
x*y
```

- What’s the difference between

```
% ls  x*y
% ls ‘x*y’
```

Thanks

Q & A