Shell II

1. Shell II

1.1 Introduction

1.2 more bash shell basics

more bash shell basics

- Wildcards
- Regular expressions
- Quoting & escaping

1.3 Command Line Behavior

Command Line Behavior

- Special characters have special meaning:

$	variable reference
=	assignment
!	event number
;	command sequence
`	command substitution
> < |	i/o redirect
&	background

* ? + [ ] { }
- wildcards
- regular expressions

‘ “ \
- quoting & escaping

1.4 Wildcards: * ? [ ] { }

Wildcards: * ? [ ] { }

A pattern of special characters used to match file names on the command line

***** zero or more characters

```
% rm *
% ls *.txt
% wc -l assign1.*
% cp a*.txt docs
```

? exactly one character

```
% ls assign?.cc
% wc assign?.??
% rm junk.???
```

1.5 Wildcards: [ ] { }

Wildcards: [ ] { }

[...] matches any of the enclosed characters

[a-z] matches any character in the range a to z

- if the first character after the **[** is a **!** or **^** then any character that is not enclosed is matched
- within [], [:class:] matches anything from a specific class: alnum, alpha, blank, digit, lower, upper, punct

{word1,word2,word3} matches any entire word

1.6 Wildcards: [ ] { } examples

Wildcards: [ ] { } examples

```
% wc -l assign[123].cc
% ls csci[2-6]30
% cp [A-Z]* dir2
% rm *[!cehg]
% echo [[:upper:]]*
% cp {*.doc,*.pdf} ~
```

1.7 Regular Expression

Regular Expression

- Pattern of special characters to match strings
- Typically made up from special characters called meta-characters: `. * + ? [ ] { } ( )`
- Regular expressions are used throughout UNIX:
  - utilities: grep, awk, sed, ...
- 2 types of regular expressions: basic vs. extended

1.8 Metacharacters

Metacharacters

.	Any one character, except new line
[a-z]	Any one of the enclosed characters (e.g. a-z)
*	Zero or more of preceding character
? also: \?	Zero or one of the preceding characters
+ also: \+	One or more of the preceding characters
^ or $	Beginning or end of line
\< or \>	Beginning or end of word
( ) also: \(\)	Groups matched characters to be used later
| also: \\|	Alternate
x{m,n} also: x \{m,n\}	Repetition of character x between m and n times

1.9 Basic vs. Extended

Basic vs. Extended

- Extended regular expressions use these meta-characters:

  ? + { } | ( )

- Basic regular expressions use these meta-characters:

  \? \+ \{ \} \| \(\)

1.10 The grep Utility

The grep Utility

- searches for text in file(s)

Syntax:

```
grep "search-text" file(s)
```

- search-text is a basic regular expression

```
grep -E "search-text" file(s)
```

- search-text is an extended regular expression

1.11 The grep Utility

The grep Utility

Examples:

```
% grep "root" /var/log/auth.log

% grep "r..t" /var/log/auth.log

% grep "ro*t" /var/log/auth.log

% grep "error" /var/log/*.log
```

Caveat: watch out for shell wild cards if not using “”

1.12 Regular Expression

Regular Expression

- consists of atoms and operators
- an atom specifies what text is to be matched and where it is to be found
- an operator combines regular expression atoms

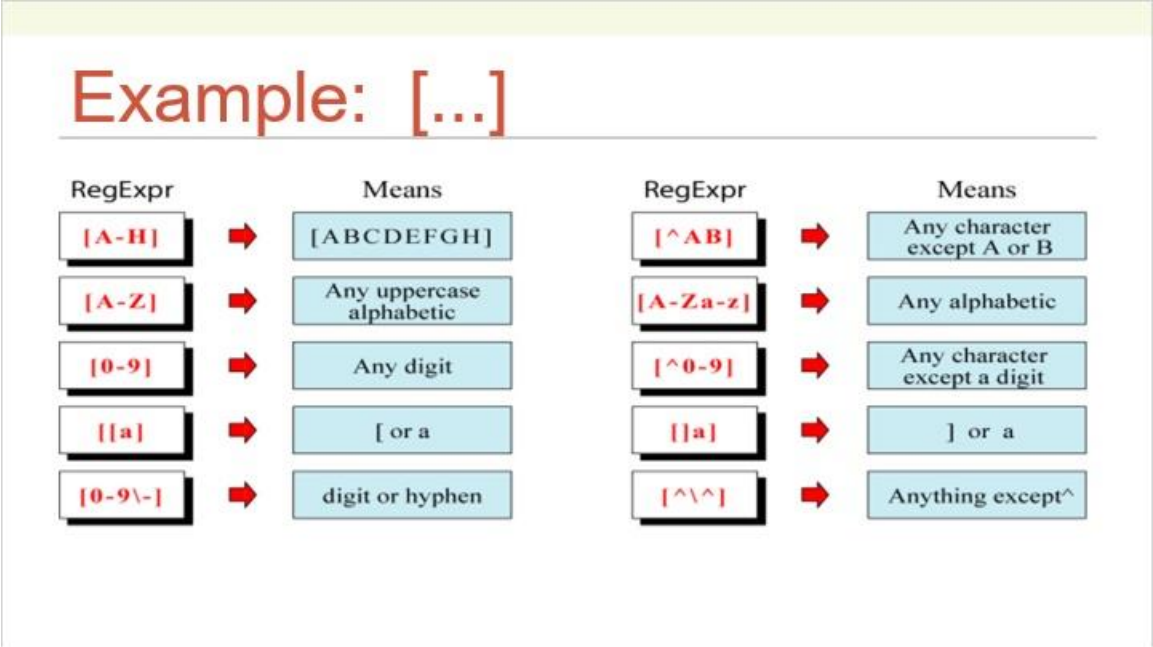
1.13 Atoms

Atoms

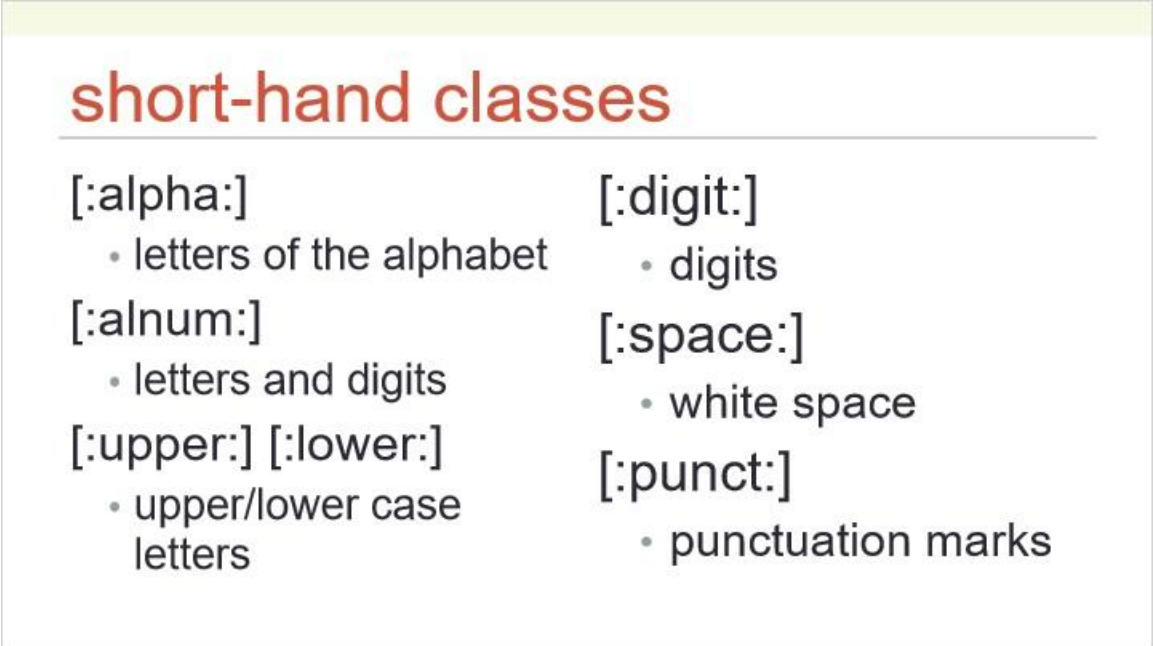
any character (not a meta-character) matches itself

`.`	matches any single character
`[...]`	matches any of the enclosed characters
`^ $ \< \>`	anchor: beginning or end of line or word
`\1 \2 \3`	... back reference

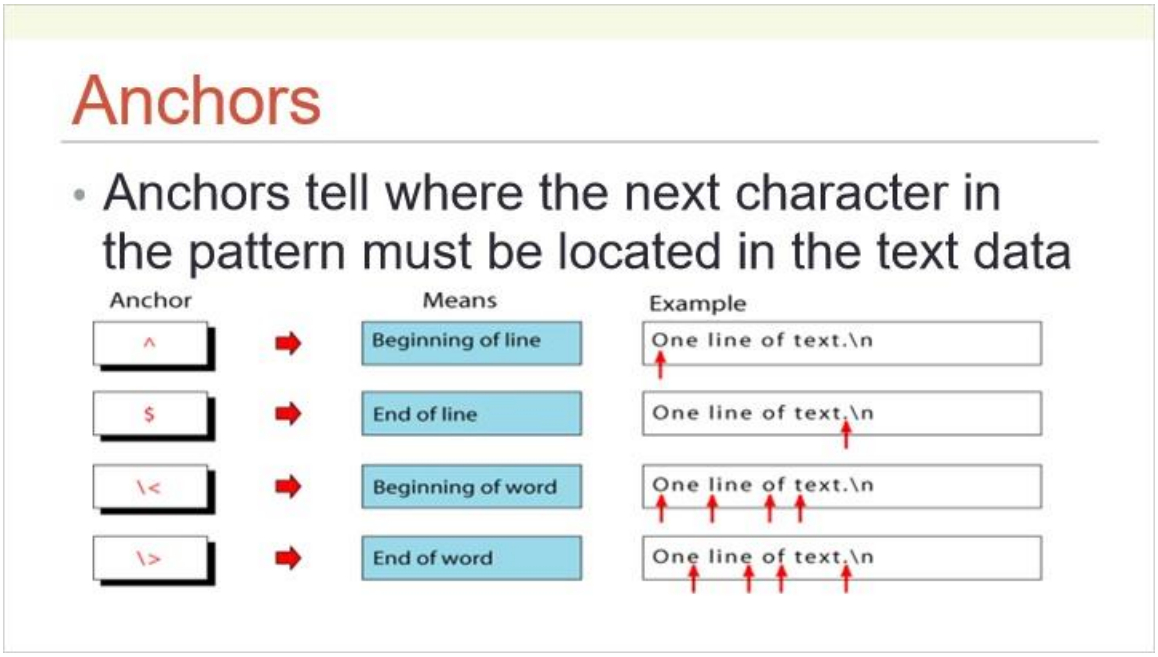
1.14 Example: [...]



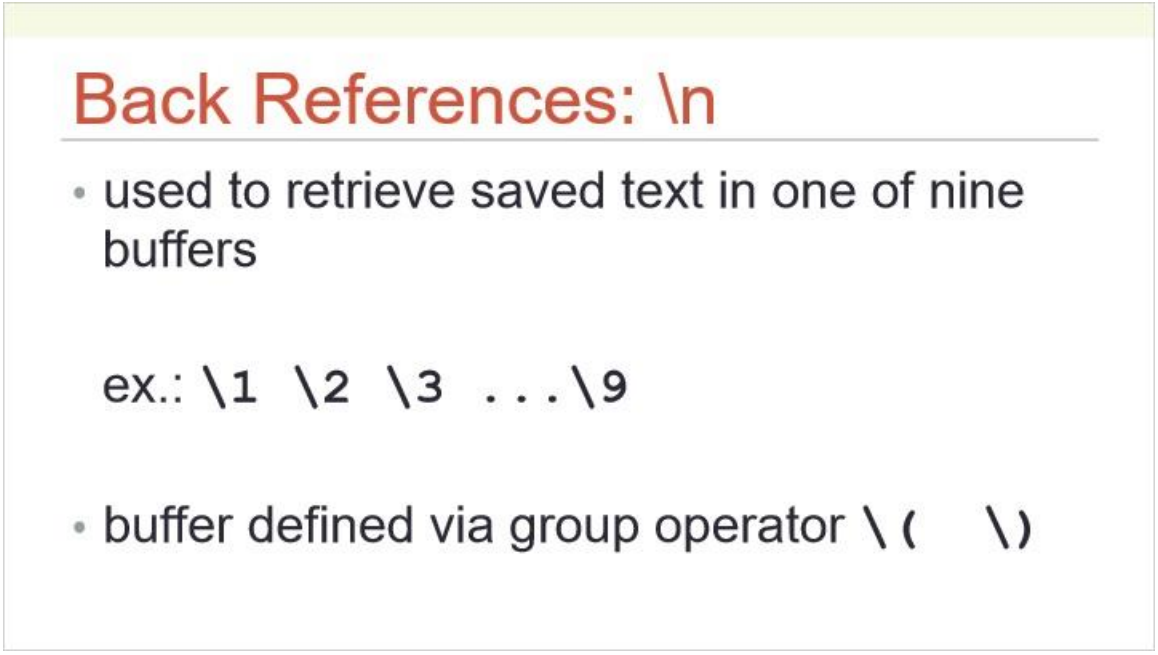
1.15 short-hand classes



1.16 Anchors



1.17 Back References: \n



1.18 Operators

Operators

	sequence
	or |	alternate
{n,m} or \{n,m\}	repetition
( ) or \(\)	group & save

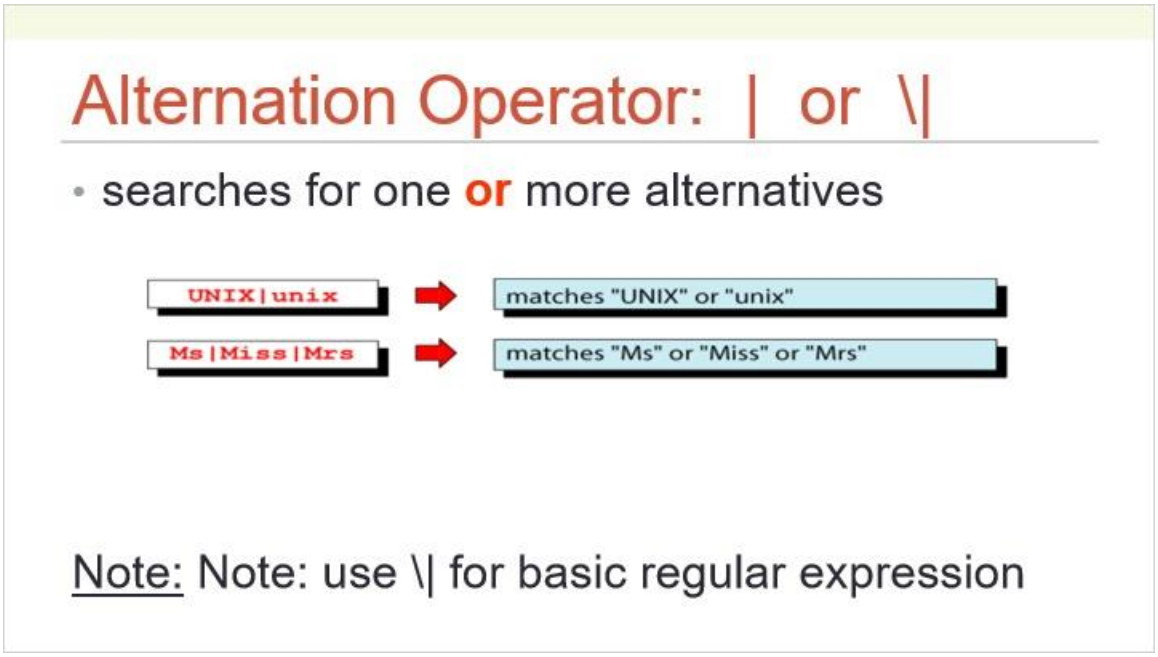
1.19 Sequence Operator

Sequence Operator

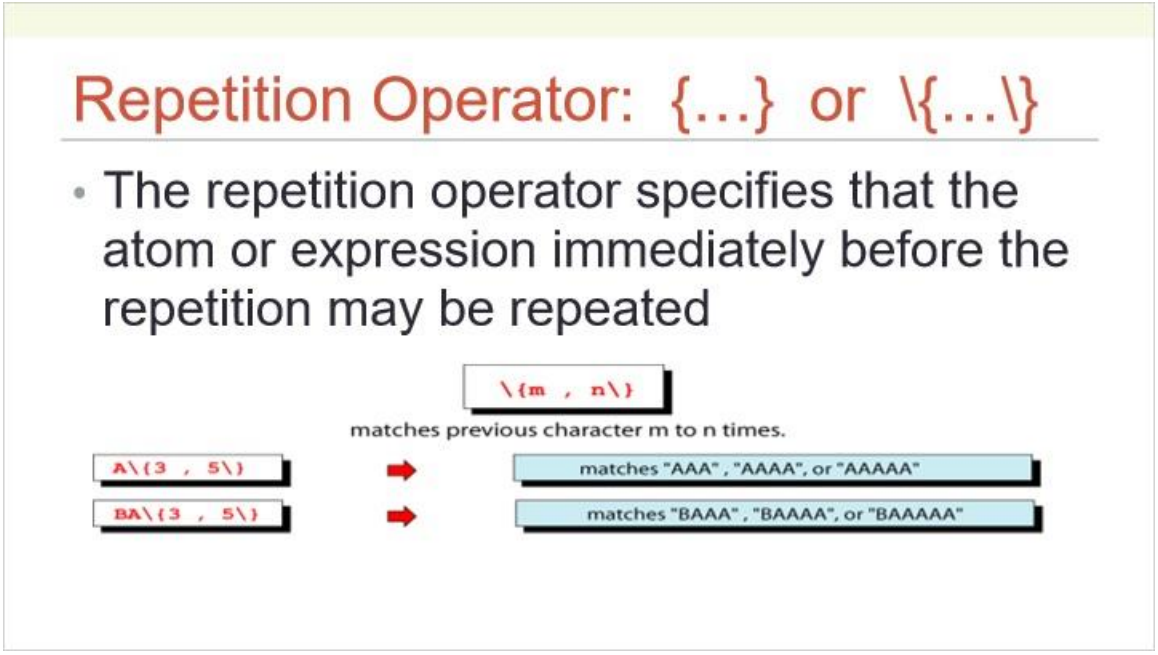
- In a sequence operator, if a series of atoms are shown in a regular expression, there is no operator between them

dog	matches the pattern "dog"
a..b	matches "a", any two characters, and "b"
[2-4][0-9]	matches a number between 20 and 49
[0-9][0-9]	matches any two digits
^$	matches a blank line
^.$	matches a one-character line
[0-9]-[0-9]	matches two digits separated by a "-"

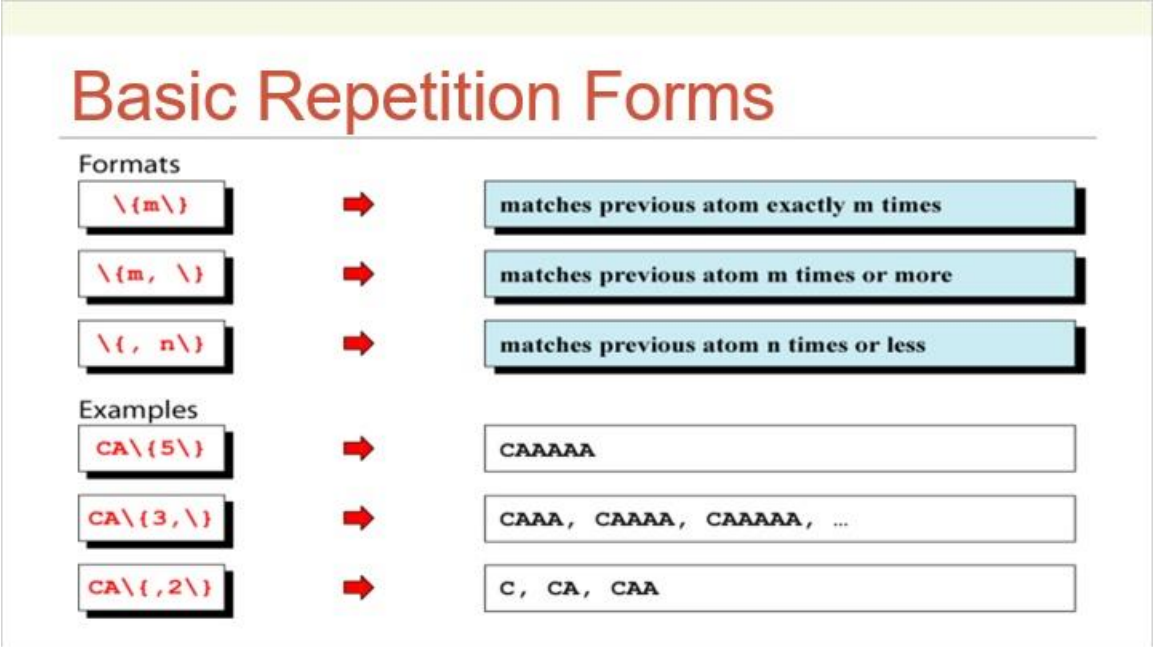
1.20 Alternation Operator: | or \|



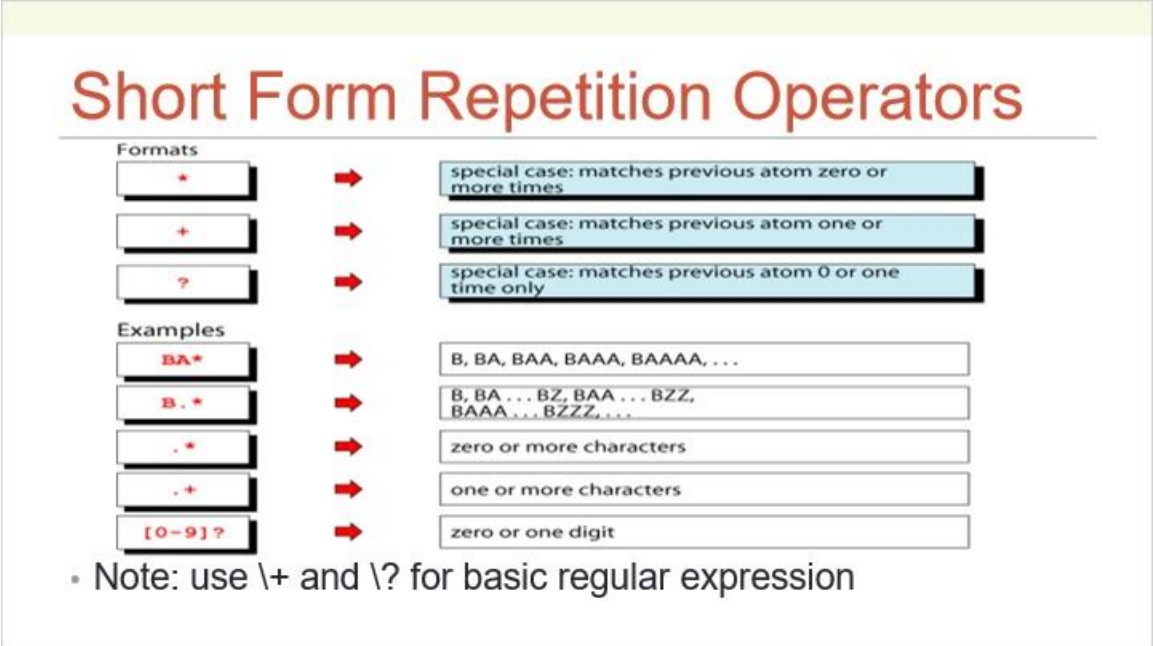
1.21 Repetition Operator: {…} or \{…\}



1.22 Basic Repetition Forms



1.23 Short Form Repetition Operators



1.24 Group Operator: ( ) or \(\)

Group Operator: ( ) or \(\)

- when a sequence of atoms is enclosed in parentheses, the next operator applies to the whole group, not only the previous characters

Regexp		Matches
A(BC)\{3\}	➡	ABCBCBC
(F(BC)\{2\}G)\{2\}	➡	FBCBCGFBCBCG

- groups define numbered buffers, can be recalled via back reference: \1, \2, \3, ...

1.25 Summary: Regular Expressions

Summary: Regular Expressions

.	Any one character, except new line
[a-z]	Any one of the enclosed characters (e.g. a-z)
*	Zero or more of preceding character
? also: \?	Zero or one of the preceding characters
+ also: \+	One or more of the preceding characters
^ or $	Beginning or end of line
\< or \>	Beginning or end of word
( ) also: \(\)	Groups matched characters to be used later
| also: \\\|	Alternate
x{m,n} also: x \{m,n\}	Repetition of character x between m and n times

1.26 Quoting & Escaping

Quoting & Escaping

- allows to distinguish between the literal value of a symbol and the symbols used as meta-characters
- done via the following symbols:
  - Backslash (\)
  - Single quote (')
  - Double quote (")

1.27 Backslash (\)

Backslash (\)

- also called the escape character
- preserve the character immediately following it
- For example:
  to create a file named “tools>”, enter:
  `% touch tools\>`

1.28 Single Quote (')

Single Quote (')

- protects the literal meaning of meta-characters
  - protects all characters within pair of single quotes
- exception: it cannot protect itself

Examples:

```
% echo 'Joe said "Have fun *@!"'
Joe said "Have fun *@!"
% echo 'Joe said 'Have fun''
Joe said Have fun
```

1.29 Double Quote (")

Double Quote (")

- protects all characters within pair of double quotes, except for:

$ (dollar sign)	! (event number)
` (back quote)	\ (backslash)

Examples:

```
% echo "I've gone fishing"
I've gone fishing
% echo "your home directory is $HOME"
your home directory is /home/student
```

1.30 Quoting Examples

Quoting Examples

```
% echo "Hello Ray []^?+*{}<>"
Hello Ray []^?+*{}<>
% echo "Hello $USER"
Hello student
% echo "It is now `date`"
It is now Mon Feb 25 10:24:08 CST 2012
% echo "you owe me \$500"
you owe me $500
```

1.31 Summary

Summary

- wildcards
- regular expressions
  - grep
- quoting