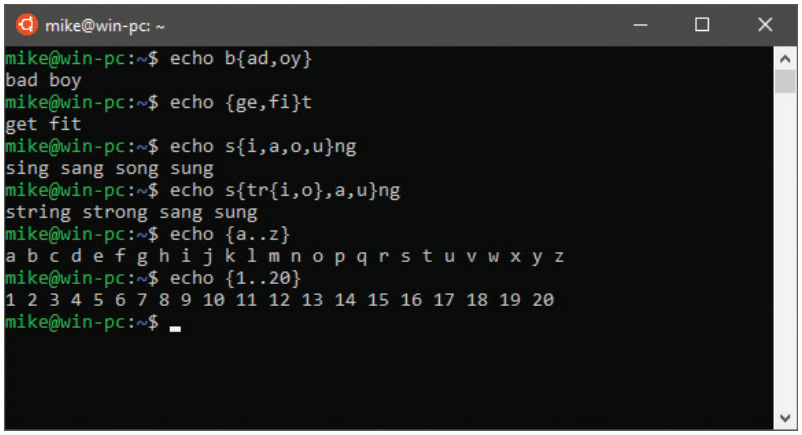
Recognizing Metacharacters

Just as the special wildcard characters **? * [ ]** can be used to perform pathname expansion, plain strings can be expanded using **{ }** brace characters. These may contain a comma-separated list of substrings that can be appended to a specified prefix or prepended to a specified suffix, or both, to generate a list of expanded strings. The brace expansions can also be nested for complex expansion. Additionally, brace expansion can produce a sequence of letters or numbers by specifying a range separated by .. between the braces:

1. At a prompt, type **echo b{ad,oy}** then hit **Return** to see two expanded strings – appended to the specified prefix
2. Next, enter **echo {ge,fi}t** to see two expanded strings – prepended to the specified suffix
3. Now, enter **echo s{i,a,o,u}ng** to see four expanded strings – both appended and prepended
4. Enter **echo s{tr{i,o},a,u}ng** to see four complex expanded strings – appended and prepended
5. Next, enter **echo {a..z}** to see an expanded letter sequence of the lowercase alphabet
6. Finally, enter **echo {1..20}** to see an expanded numeric sequence from 1 to 20

Beware

There must be no spaces within the braces, or between the braces and each specified prefix and suffix.



```
mike@win-pc:~$ echo b{ad,oy}
bad boy
mike@win-pc:~$ echo {ge,fi}t
get fit
mike@win-pc:~$ echo s{i,a,o,u}ng
sing sang song sung
mike@win-pc:~$ echo s{tr{i,o},a,u}ng
string strong sang sung
mike@win-pc:~$ echo {a..z}
a b c d e f g h i j k l m n o p q r s t u v w x y z
mike@win-pc:~$ echo {1..20}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
mike@win-pc:~$
```

NEW

Bash version 4 introduced zero-padded brace expansion so that **echo {001..3}** produces 001 002 003.

...cont'd

The wildcards ? * [ ] and braces { } are just some examples of "metacharacters" that have special meaning to the Bash shell. The table below lists all metacharacters that have the special meaning described when used in commands at a shell prompt only – the same characters can have other meanings when used in other situations, such as in arithmetical expressions.

Metacharacter:	Meaning:
~	Home directory
`	Command substitution (old style)
#	Comment
$	Variable expression
&	Background job
*	String wildcard
(	Start of subshell
)	End of subshell
\	Escape next character
|	Pipe
[	Start of wildcard set
]	End of wildcard set
{	Start of command block
}	End of command block
;	Pipeline command separator
'	Quote mark (strong)
"	Quote mark (weak)
<	Redirect input
>	Redirect output
/	Pathname address separator
?	Single-character wildcard
!	Pipeline logical NOT

Don't forget

Some of the metacharacters in this table have been introduced already but others are described later in this book.

Hot tip

Notice that the semi-colon **;** character allows two commands to be issued on the same line. For example, type **echo {a..z} ; echo {1..9}** then hit Return.