18.2. Globbing

Bash itself cannot recognize Regular Expressions. Inside scripts, it is commands and utilities -- such as sed and awk -- that interpret RE's.

Bash *does* carry out *filename expansion* [1] -- a process known as *globbing* -- but this does *not* use the standard RE set. Instead, globbing recognizes and expands *wild cards*. Globbing interprets the standard wild card characters [2] -- * and ?, character lists in square brackets, and certain other special characters (such as ^ for negating the sense of a match). There are important limitations on wild card characters in globbing, however. Strings containing `*` will not match filenames that start with a dot, as, for example, `.bashrc`. [3] Likewise, the `?` has a different meaning in globbing than as part of an RE.

```
bash$ ls -l
total 2
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
 -rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh
 -rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt

bash$ ls -l t?.sh
-rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh

bash$ ls -l [ab]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1

bash$ ls -l [a-c]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1

bash$ ls -l [^ab]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
 -rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh
 -rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt

bash$ ls -l {b*,c*,*est*}
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
 -rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
 -rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt
```

Bash performs filename expansion on unquoted command-line arguments. The echo command demonstrates this.

```
bash$ echo *
a.1 b.1 c.1 t2.sh test1.txt

bash$ echo t*
t2.sh test1.txt

bash$ echo t?.sh
t2.sh
```

It is possible to modify the way Bash interprets special characters in globbing. A **set -f** command disables globbing, and the `nocaseglob` and `nullglob` options to shopt change globbing behavior.

See also Example 11-5.

Filenames with embedded whitespace can cause *globbing* to choke. David Wheeler shows how to avoid many such pitfalls.

```
IFS="$(printf '\n\t')"   # Remove space.

#  Correct glob use:
#  Always use for-loop, prefix glob, check if exists file.
for file in ./* ; do         # Use ./* ... NEVER bare *
  if [ -e "$file" ] ; then   # Check whether file exists.
     COMMAND ... "$file" ...
  fi
done

# This example taken from David Wheeler's site, with permission.
```

Notes

[1] *Filename expansion* means expanding filename patterns or templates containing special characters. For example, `example.???` might expand to `example.001` and/or `example.txt`.

[2] A *wild card* character, analogous to a wild card in poker, can represent (almost) any other character.

[3] Filename expansion *can* match dotfiles, but only if the pattern explicitly includes the dot as a literal character.

```
~/[.]bashrc    #  Will not expand to ~/.bashrc
~/?bashrc      #  Neither will this.
               #  Wild cards and metacharacters will NOT
               #+ expand to a dot in globbing.

~/.[b]ashrc    #  Will expand to ~/.bashrc
~/.ba?hrc      #  Likewise.
~/.bashr*      #  Likewise.

# Setting the "dotglob" option turns this off.

# Thanks, S.C.
```