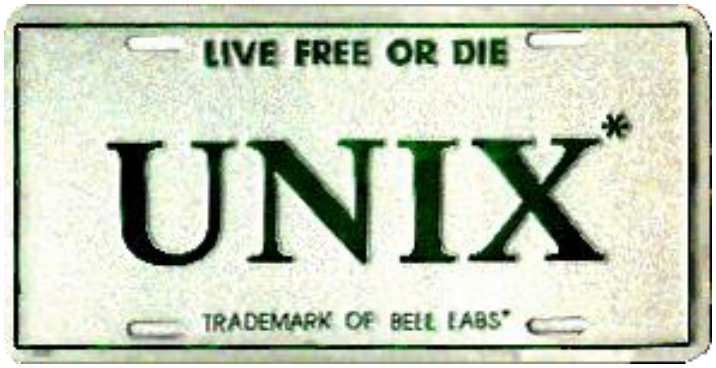


# UNIX for Programmers and Users

**“UNIX for Programmers and Users”**
**Third Edition, Prentice-Hall, GRAHAM GLASS, KING ABLES**

- **FILENAME SUBSTITUTION( WILDCARDS )**

- All shells support a wildcard facility that allows you to select files that satisfy a particular name pattern from the file system.

- The wildcards and their meanings are as follows:

| Wildcard | Meaning |
|---|---|
| * | Matches any string, including the empty string. |
| ? | Matches any single character. |
| [..] | Matches any one of the characters between the brackets. A range of characters may be specified by separating a pair of characters by a hyphen. |

- Prevent the shell from processing the wildcards in a string by surrounding the string with single quotes(apostrophes) or double quotes.

- A backslash(/) character in a filename must be matched explicitly.

$ ls -FR ---> recursively list the current directory.

```
a.c        b.c        cc.c          dir1/              dir2/

dir1:
d.c        e.e

dir2:
f.d        g.c
```

$ ls *.c ---> list any text ending in ".c".

```
a.c        b.c        cc.c
```

$ ls ?.c ---> list text for which one character is followed by ".c".

```
a.c        b.c
```

```
$ ls [ac]*          ---> list any string beginning with "a" or "c".
a.c        cc.c

$ ls [A-Za-z]*   ---> list any string beginning with a letter.
a.c        b.c          cc.c

$ ls dir*/*.c      ---> list all files ending with ".c" in "dir*" directories
                   ---> ( that is, in any directories beginning with "dir" ).
dir1/d.c     dir2/g.c


$ ls */*.c          ---> list all files ending in ".c" in any subdirectory.
dir1/d.c     dir2/g.c

$ ls *2/?.? ?.?  ---> list all files with extensions in "2*" directories
                                  and current directory.
a.c      b.c      dir2/f.d        dir2/g.c
$ _
```

- **PIPES**

- Shells allow you to use the standard output of one process as the standard input of another process by connecting the processes together using the pipe | metacharacter.

- The sequence

```
$ command1 | command2
```

causes the standard output of command1 to "flow through" to the standard input of command2.

- Any number of commands may be connected by pipes.

- A list of commands in this way is called a *pipeline*.

- Based on one of the basic UNIX philosophies: large problems can often be solved by a chain of smaller processes

- Example, pipe the output of the **ls** utility to the input of the **wc** utility in order to count the number of files in the current directory.

```
$ ls            ---> list the current directory.
a.c   b.c    cc.c   dir1   dir2

$ ls | wc -w
    5

$ ls -l | awk '{ print $1 }' | sort          ---> example
```

- **COMMAND SUBSTITUTION**

A command surrounded by grave accents (`) - back quote - is executed, and its standard output is inserted in the command's place in the entire command line.

Any new lines in the output are replaced by spaces.

For example:

```
$ echo  the date today is `date`
the date today is Wednesday August 24 11:40:55 2016
$ _
```

- By piping the output of who to the wc utility,
  it's possible to count the number of users on the system:

```
$ who   ---> look at the output of who.
posey       ttyp0        Jan   22  15:31  ( blackfoot:0.0 )
glass       ttyp3        Feb    3  00:41  ( bridge05.utdalla )
huynh       ttyp5        Jan   10  10:39  ( atlas.utdallas.e )

$ echo  there  are  `who  |  wc -l`   users on the system
there are 3 users on the system
$ _
```