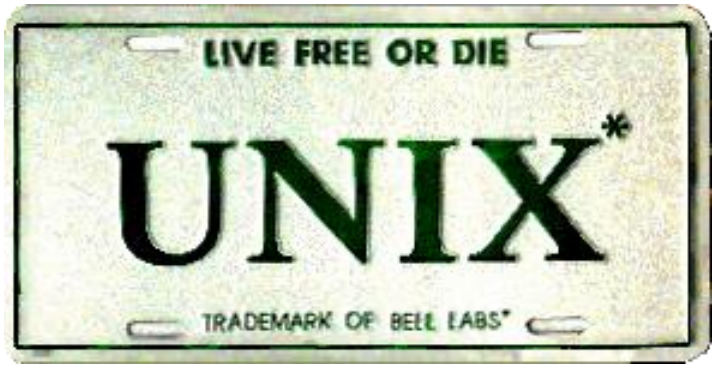

UNIX for Programmers and Users

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

- **Displaying Information : echo**

The built-in echo command displays its arguments to standard output and works like this:

> Shell Command: **echo** arg
>
> echo is a built-in shell command that displays all of its arguments to standard output.
> By default, it appends a new line to the output.

- **Changing Directories : cd**

The built-in cd command changes the current working directory of the shell to a new location.

- **METACHARACTERS**

Some characters are processed specially by a shell and are known as metacharacters.

All four shells share a core set of common metacharacters, whose meanings are described in next two slides.

- **METACHARACTERS**

Symbol	Meaning
>	Output redirection; writes standard output to a file.
>>	Output redirection; appends standard output to a file.
<	Input redirection; reads standard input from a file.
<<	Input redirection; End of file (EoF) customization.
*	File-substitution wildcard; matches zero or more characters.
?	File-substitution wildcard; matches any single character.
[...]	File-substitution wildcard; matches any character between the brackets.

- **METACHARACTERS**

Symbol	Meaning
|	Pipe symbol; sends the output of one process to the input of another.
||	Conditional execution; executes a command if the previous one fails.
&&	Conditional execution; executes a command if the previous one succeeds.
&	Runs a command in the background.
$	Expands the value of a variable.
\	Prevents special interpretation of the next character.

- **METACHARACTERS**

- When you enter a command,
  the shell scans it for metacharacters and processes them specially.

  When all metacharacters have been processed,
  the command is finally executed.

- **Backslash(\)**

To turn off the special meaning of a metacharacter,
precede it by a backslash(\) character.

Here's an example:

```
$ echo  hi > file   --->  store output of echo in "file".

$ cat  file          --->  look at the contents of "file".
  hi

$ echo hi \> file   --->  inhibit > metacharacter.
  hi > file            --->   > is treated like other characters.

$ _
```

- **Redirection**

The shell redirection facility allows to:

1) store the output of a process to a file (output redirection)
2) use the contents of a file as input to a process (input redirection)

Output redirection

To redirect output, use either the “>” or “>>” metacharacters.

The sequence

```
$ command > fileName
```

sends the standard output of command to the file with name fileName.

The shell creates the file with name fileName if it doesn’t already exist or overwrites its previous contents if it does already exist.

- If the file already exists but doesn't have write permission, an error occurs.

```
$ cat > alice.txt              ---> creates a text file.
In my dreams that fill the night,
I see your eyes,
^D                              ---> end of input.

$ cat  alice.txt
In my dreams that fill the night, ---> look at its contents.
I see your eyes,
$ _
```

- The sequence

    $ command >> fileName

appends the standard output of command to the file with name fileName.

$ cat >> alice.txt ---> append to the file.
And I fall into them,
Like Alice fell into Wonderland.
^D ---> end of input.

$ cat alice.txt ---> look at the new contents.
In my dreams that fill the night,
I see your eyes,
And I fall into them,
Like Alice fell into Wonderland.
$ _

- The Bash, C and Korn shells also provide protection against accidental overwriting of a file due to output redirection.

In Bash:

```
$ set -o noclobber

$ echo text > test

$ echo text > test
bash: test: cannot overwrite existing file

$ echo text >| test
$ _
```

set +o noclobber → to revert the effect

- **Input Redirection**

To redirect input, use either the `<` or `<<` metacharacters.

The sequence

```
$ command < fileName
```

executes command using the contents of the file fileName as its standard input.

If the file doesn't exist or doesn't have read permission, an error occurs.

- When the shell encounters a sequence of the form

$ command << word

- it copies its standard input up to, but not including,
the line starting with word into a buffer and then executes command using the contents of the buffer as its standard input.

- that allows shell programs(scripts) to supply the standard input to other commands as in-line text,

```
$ cat << eof
> line 1
> line 2
> line 3
> eof
line 1
line 2
line 3
$ _
```