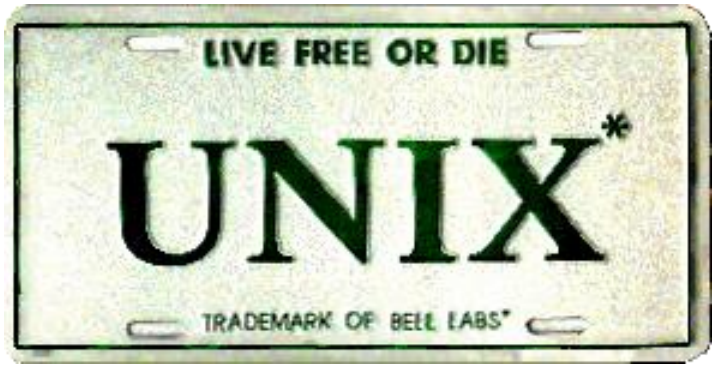

UNIX for Programmers and Users

- **SEQUENCES**

If you enter a series of simple commands or pipelines separated by semicolons, the shell will execute them in sequence, from left to right.

This facility is useful for type-ahead(and think-ahead) addicts who like to specify an entire sequence of actions at once.

Here's an example:

```
$ date;  pwd;   ls    ---> execute three commands in sequence.
Wednesday August 24 11:40:55 2016
/home/glass/wild
a.c    b.c    cc.c       dir1       dir2
$ _
```

- Each command in a sequence may be individually I/O redirected as well:

```
$ date > date.txt;  ls;   pwd  > pwd.txt
a.c       b.c       cc.c       date.txt       dir1       dir2

$ cat  date.txt
Wednesday August 24 11:40:55 2016

$ cat   pwd.txt       ---> look at output of pwd.
/home/glass
$ _
```

- **Conditional Sequences**

- Every UNIX process terminates with an exit value.
  an exit value of 0 --> process completed successfully
  a nonzero exit value --> failure

- All built-in shell commands return a value of 1 if they fail.

1) Commands are separated by “&&” tokens

2) Commands are separated by “||” tokens

- For example,
  if the C compiler cc compiles a program without fatal errors,
  it creates an executable program called "a.out" and returns an exit code of 0;
  otherwise, it returns a nonzero exit code.

```
$ cc  myprog.c  &&  ./a.out
```

- The following conditional sequence compiles a program called "myprog.c" and displays an error message if the compilation fails:

```
$ cc  myprog.c  ||  echo  compilation  failed.
```

- **GROUPING COMMANDS**

- Commands may be grouped by placing them between parentheses.

- The group of commands shares the same standard input, standard output, and standard error channels and may be redirected and piped as if it were a simple command.

- Here are some examples:

```
$ date; ls; pwd > out.txt            ---> execute a sequence.
Wednesday August 24 11:40:55 2016    ---> output from date.
a.c          b.c                     ---> output from ls.
$ cat  out.txt                       ---> only pwd was redirected.
 /home/glass

$ ( date; ls; pwd ) > out.txt        ---> group and then redirect.
$ cat out.txt                        ---> all output was redirected.
Wednesday August 24 11:40:55 2016
a.c                b.c
/home/glass
$ _
```

- TERMINATION AND EXIT CODES

In the Bash, Bourne and Korn shells, the special shell variable $? always contains the value of the previous command's exit code.

In the C shell, the $status variable holds the exit code.

-In the following example, the date utility succeeded, whereas the cc utility failed:

```
$ date              ---> date succeeds.
Mon  29  8  22:13:38  2016
$ echo $?           ---> display its exit value.
0                   ---> indicates success.

$ cc prog.c         ---> compile a nonexistent program.
cpp: Unable to open source file 'prog.c'.
$ echo $?           ---> display its exit value.
1                   ---> indicates failure.
$ _
```