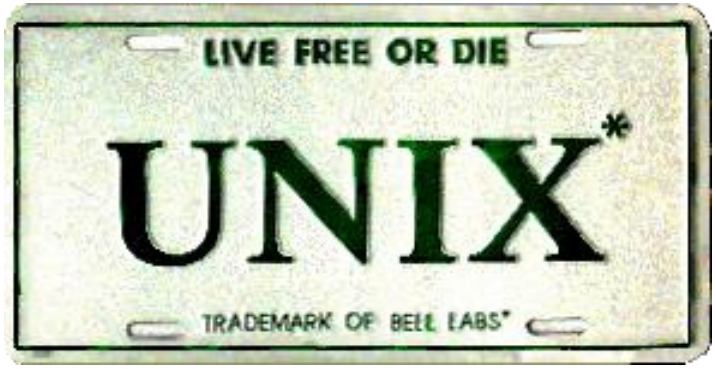
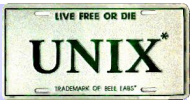


Shell Scripting



Programming or Scripting ?

- Shell scripting allows us to automate a lot of tasks that would otherwise require a lot of commands.

  - Programming languages:
    - more powerful and a lot faster than scripting languages
    - compiled into an executable
    - executable is not easily ported into different operating systems

  - Scripting languages:
    - generally slower than compiled programs
    - interpreter reads the instructions and execute
    - source files easily portable to any operating system

The first bash script

```
$ cat > hello.bash
#!/bin/bash
echo "Hello World!!!"
^D

$ chmod 700 hello.bash

$ hello.bash
Hello World
```

SHELL PROGRAMS: SCRIPTS

- The system decides which shell the script is written for by examining the first line of the script.

- Here are the rules that it uses to make this decision:

1) If the first line of the script is just a pound sign(#),
   then the script is interpreted by the shell from which you executed this script as a command.

2) If the first line of the script is of the form #! path name,
   then the executable program *pathName* is used to interpret the script.

3) If neither rule1 nor rule2 applies,
   then the script is interpreted by a Bourne shell (sh).

SHELL PROGRAMS: SCRIPTS

- Here is an example that illustrates the construction and execution of two scripts, one for the Bash shell and the other for the Korn shell.

```
$ cat  >  script1          ---> create  the bash script.
#!/bin/bash
# This is a sample bash script.
echo  -n  "the date today is "    # in bash,  -n omits new line
date  # output today's date.
^D                                  ---> end of input.
```

Bash Shell

- several common built-in variables that have special meanings:

Name	Meaning
$$	The process ID of the shell.
$#	The number of parameters passed.
$0	The name of the shell script (if applicable).
$1..$9	$n refers to the nth command line argument (if applicable).
$*	A list of all the command-line arguments.
$@	Array of words containing all the parameters passed to the script

Bash Shell

Example:

```
$ cat  script2          ---> list the script.
 #!/bin/bash
 echo the name of this script is $0
 echo the first argument is $1
 echo a list of all the arguments is $*
 echo this script places the date into a temporary file called $1.$$
 date > $1.$$      # redirect the output of date.
 ls $1.$$          # list the file.

$ script2  paul  ringo  george  john   ---> execute the script.
 the name of this script is script3
 the first argument is paul
 a list of all the arguments is paul ringo george john
 this script places the date into a temporary file called paul.24321
 paul.24321
 $ _
```

Read command

- The read command allows to prompt for input and store it in a variable.

- Example:

```
$ cat exp1
#!/bin/bash
echo -n "Enter name of file to delete: "
read file
echo "Type 'y' to remove it, 'n' to change your mind ... "
rm -i $file
echo "That was YOUR decision! "
```

Arithmetic Evaluation

- The let statement can be used to do mathematical functions:

```
$ let X=10+2*7
$ echo $X
24
$ let Y=X+2*4
$ echo $Y
32
```

- An arithmetic expression can be evaluated by $((expression)) or $[expression]

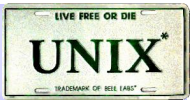
```
$ echo $((123+20))
143
$ abc=$[123+20]
$ echo $[123*abc]
17589
```

Arithmetic Evaluation

- Available operators: +, -, /, *, %

- Example

```
$ cat exp2
#!/bin/bash
echo -n "Enter the first number: "; read x
echo -n "Enter the second number: "; read y
add=$[x+y]
sub=$[x-y]
mul=$[x*y]
div=$[x/y]
mod=$[x%y]
# print out the answers:
echo "Sum: $add"
echo "Difference: $sub"
echo "Product: $mul"
echo "Quotient: $div"
echo "Remainder: $mod"
```



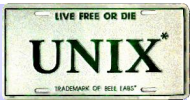
Conditional Statements

- Conditionals let us decide whether to perform an action or not, this decision is taken by evaluating an expression. The most basic form is:

```
if [ expression ];
then
        statements
elif [ expression ];
then
        statements
else
        statements
fi
```

- the elif (else if) and else sections are optional

- Put spaces after [and before], and around the operators and operands.



Expressions

- An expression can be: String comparison, Numeric comparison, File operators and Logical operators and it is represented by [expression]:

- String Comparisons:

=	compare if two strings are equal
!=	compare if two strings are not equal

- Examples:

[$s1 = $s2]	(true if s1 same as s2, else false)
[$s1 != $s2]	(true if s1 not same as s2, else false)
[$s1]	(true if s1 is not empty, else false)

Expressions

- Examples: String Comparisons:

```
$ cat exp3
#!/bin/bash
s1="ITWS1";s2="UG1";s3="UG1";s4=" ";
if [ $s1 = $s2 ]; then echo True ; else echo false ; fi
if [ $s2 = $s3 ]; then echo True ; else echo false ; fi
if [ $s1 != $s2 ]; then echo True ; else echo false ; fi
if [ $s2 != $s3 ]; then echo True ; else echo false ; fi
if [ $s3 ]; then echo True ; else echo false ; fi
if [ $s4 ]; then echo True ; else echo false ; fi
```

Expressions

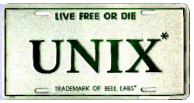
- Examples: String Comparisons:

Make errors

```
s1="ITWS1"; s2="UG1";s3="UG1";

if[ $s2 = $s3 ]; then echo True ; else echo false ; fi

if [$s2 = $s3 ]; then echo True ; else echo false ; fi
if [$s2 = $s3]; then echo True ; else echo false ; fi

if [ $s2= $s3 ]; then echo True ; else echo false ; fi
if [ $s2=$s3 ]; then echo True ; else echo false ; fi
if [ $s1=$s3 ]; then echo True ; else echo false ; fi

if [ $s2 -eq $s3 ]; then echo True ; else echo false ; fi
if ( $s2 = $s3 ); then echo True ; else echo false ; fi
```

Examples – Check the login name

```
$ cat exp4
#!/bin/bash
 echo -n "Enter your login name: "
 read name
  if [ $name ]; then
     if [ $name = $USER ]; then
          echo "Hello, $name. How are you today?"
     else
          echo "You are not the actual user, so who are you?"
     fi
  else
      echo "Username can not be empty!"
  fi
```



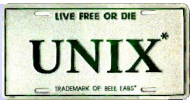
Expressions

- Number Comparisons:

-eq	compare if two numbers are equal
-ge	compare if one number is greater than or equal to a number
-le	compare if one number is less than or equal to a number
-ne	compare if two numbers are not equal
-gt	compare if one number is greater than another number
-lt	compare if one number is less than another number

- Examples:

[n1 -eq n2]	(true if n1 same as n2, else false)
[n1 -ge n2]	(true if n1 greater than or equal to n2, else false)
[n1 -le n2]	(true if n1 less than or equal to n2, else false)
[n1 -ne n2]	(true if n1 is not same as n2, else false)
[n1 -gt n2]	(true if n1 greater than n2, else false)
[n1 -lt n2]	(true if n1 less than n2, else false)



Expressions

•Examples: Number Comparisons:

```
num1=10; num2=25;

if [ $num1 -eq $num2 ]; then echo True ; else echo false ; fi
if [ $num1 = $num2 ]; then echo True ; else echo false ; fi

if [ $num1 -ne $num2 ]; then echo True ; else echo false ; fi
if [ $num1 != $num2 ]; then echo True ; else echo false ; fi

if [ $num1 -gt $num2 ]; then echo True ; else echo false ; fi
if [ $num1 -lt $num2 ]; then echo True ; else echo false ; fi
```

Make errors

```
if[ $num1 -lt $num2 ]; then echo True ; else echo false ; fi
if [$num1 -lt $num2 ]; then echo True ; else echo false ; fi
if [ $num1-lt $num2 ]; then echo True ; else echo false ; fi
```

Examples – Compute square of numbers [0, 10]

```
$ cat exp5
#!/bin/bash
 echo -n "Enter a number 0 <= x <= 10: "
 read num
  if [ ! $num ]; then
          echo "You have not entered any number!"
 elif [ $num -le 10 ]; then
          if [ $num -ge 0 ]; then
                  echo "$num*$num=$[num*num]"
          else
                  echo "Wrong insertion! You have entered x < 0"
                  exit 1
          fi
 else
          echo "Wrong insertion! You have entered x > 10"
          exit 1
 fi
```

Expressions

- Files operators:

-d	check if path given is a directory
-f	check if path given is a file
-e	check if file name exists
-s	check if a file has a length greater than 0
-r	check if read permission is set for file or directory
-w	check if write permission is set for a file or directory
-x	check if execute permission is set for a file or directory

- Examples:

[-d fname]	(true if fname is a directory, otherwise false)
[-f fname]	(true if fname is a file, otherwise false)
[-e fname]	(true if fname exists, otherwise false)
[-s fname]	(true if fname length is greater then 0, else false)
[-r fname]	(true if fname has the read permission, else false)
[-w fname]	(true if fname has the write permission, else false)
[-x fname]	(true if fname has the execute permission, else false)

Expressions

$ cat exp6 Examples: File operators

```
#!/bin/bash
touch xz1;mkdir xz2
if [ -d xz1 ]; then echo True; else echo False; fi
if [ -d xz2 ]; then echo True; else echo False; fi
if [ -f xz1 ]; then echo True; else echo False; fi
if [ -f xz2 ]; then echo True; else echo False; fi
if [ -e xz1 ]; then echo True; else echo False; fi
if [ -e xz2 ]; then echo True; else echo False; fi
if [ -e xz3 ]; then echo True; else echo False; fi
if [ -s xz1 ]; then echo True; else echo False; fi
echo do not give up > xz1
if [ -s xz1 ]; then echo True; else echo False; fi
if [ -r xz1 ]; then echo True; else echo False; fi
if [ -w xz1 ]; then echo True; else echo False; fi
if [ -x xz1 ]; then echo True; else echo False; fi
```

Example

$ cat exp7 (Copy a file to a directory)

```
#!/bin/bash
echo -n "Enter the file name: "; read f1
echo -n "Enter the directory name: "; read d1
if [ -f $f1 ]; then
    if [ ! -d $d1 ]; then
        echo "The $d1 directory does not exist, creating it."
        mkdir $d1
    fi
    cp $f1 $d1
    echo Done.
 else
     echo "This file does not exist."
     exit 1
 fi
```

Expressions

- Logical operators:

!	negate (NOT) a logical expression
-a	logically AND two logical expressions
-o	logically OR two logical expressions

Example:

$ cat exp8 (Compute the square of any number from 1 to 10)

```
#!/bin/bash
 echo -n "Enter a number 1 <= x <= 10: "; read num
 if [ $num -ge 1 -a $num -le 10 ];
 then
        echo "$num*$num=$[num*num]"
 else
        echo "Wrong insertion!"
        exit 1
 fi
```

Expressions

- Logical operators:

&& logically AND two logical expressions
|| logically OR two logical expressions

Example:

$ cat exp9 (Compute the square of any number from 1 to 10)

```
#!/bin/bash
 echo -n "Enter a number 1 <= x <= 10: "; read num
 if [ $num -ge 1 ] && [ $num -le 10 ];
 then
        echo "$num*$num=$[num*num]"
 else
        echo "Wrong insertion!"
        exit 1
 fi
```

Case Statement

- Used to execute statements based on specific values. Often used in place of an if statement if there are a large number of conditions.

- Value used can be an expression
- each set of statements must be ended by a pair of semicolons;
- a *) is used to accept any value not matched with list of values

```
case $var in
 val1)
        statements;;
 val2)
        statements;;
 *)
        statements;;
 esac
```

Example (case)

```
$ cat exp10
#!/bin/bash
 echo -n "Enter a number 0 < x < 10:"
  read x
 case $x in
          1) echo "Value of x is 1.";;
          2) echo "Value of x is 2.";;
          3) echo "Value of x is 3.";;
          4) echo "Value of x is 4.";;
          5) echo "Value of x is 5.";;
          6) echo "Value of x is 6.";;
          7) echo "Value of x is 7.";;
          8) echo "Value of x is 8.";;
          9) echo "Value of x is 9.";;
          10 | 0) echo "wrong number.";;
          *) echo "Unrecognized value.";;
 esac
```