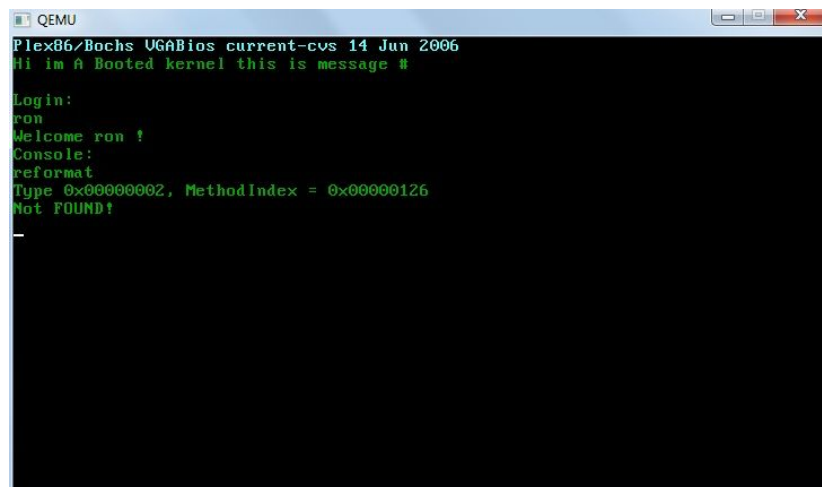


Shell and Bash Script

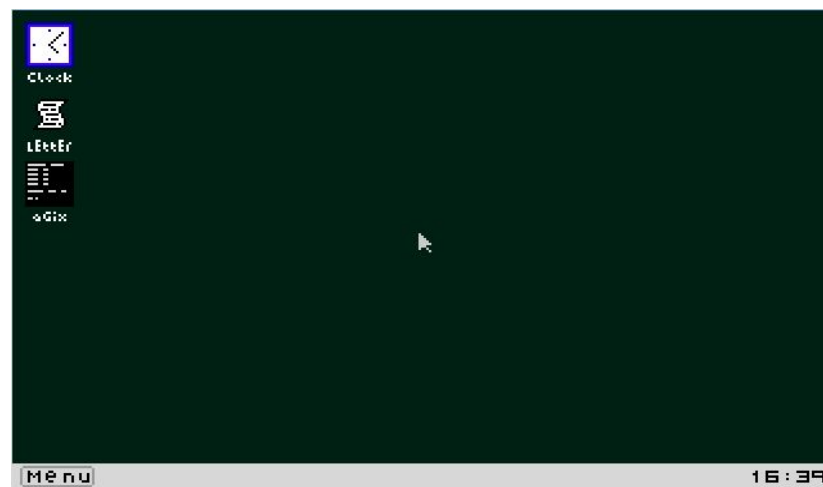
What is Shell?

- User Interface
 - command-line interface (CLI)
 - graphical user interface (GUI)
- It is called shell because it is outermost layer around the OS kernel.



```
QEMU
P1ex86/Bochs UGABios current-cvs 14 Jun 2006
Hi im A Booted kernel this is message #

Login:
ron
Welcome ron !
Console:
reformat
Type 0x00000002, MethodIndex = 0x00000126
Not FOUND!
-
```



What is Bash?

- Bash is a Unix Shell.
- Command-line Language
- It can be executed from script files (bash script)
- `chmod +x script.sh`

Like Programming languages ...

- Variables
- Arguments
- Array
- Operator
- If ... else ...
- Loop
- Pipelines
- Regex
- ...

HelloWorld

- Every Bash Script should start with
 - `#!/bin/bash`
- Comments in Bash Script starts with `#`
- A simple HelloWorld!

```
#!/bin/bash
```

```
echo "Hello World!"
```

```
>> Hello World!
```

Variables

- Define Variables
 - `name="Vahid"`
 - notice that there is **no space** among variable name and equal sign and its value and !
 - `std_no=94521207`
- Using variables with **\$** before the name

```
echo Name: ${name}, ID: $std_no
```

```
>> Name: Vahid, 94521207
```

```
echo Name: name , ID: std_no
```

```
>> Name: name , ID: std_no
```

Arrays

```
my_array=(apple banana "Fruit Basket" orange)

echo ${#my_array[@]}      # 4

echo ${my_array[@]}       #(apple banana "Fruit Basket" orange)

my_array[4]="carrot"

echo ${#my_array[@]}      # 5

echo ${my_array[${#my_array[@]}-1]}    # carrot
```

Operators

- **a + b** addition (a plus b)
- **a - b** subtraction (a minus b)
- **a * b** multiplication (a times b)
- **a / b** division (integer) (a divided by b)
- **a % b** modulo (the integer remainder of a divided by b)
- **a ** b** exponentiation (a to the power of b)

if ... elif ... else ...

```
NAME="George"
```

```
if [ "$NAME" = "John" ]; then
```

```
    echo "John Lennon"
```

```
elif [ "$NAME" = "George" ]; then
```

```
    echo "George Harrison"
```

```
else
```

```
    echo "This leaves us with Paul and Ringo"
```

```
fi
```

if ... elif ... else ...

- for numeric comparison

comparison	Evaluated to true when
<code>\$a -lt \$b</code>	<code>\$a < \$b</code>
<code>\$a -gt \$b</code>	<code>\$a > \$b</code>
<code>\$a -le \$b</code>	<code>\$a <= \$b</code>
<code>\$a -ge \$b</code>	<code>\$a >= \$b</code>
<code>\$a -eq \$b</code>	<code>\$a</code> is equal to <code>\$b</code>
<code>\$a -ne \$b</code>	<code>\$a</code> is not equal to <code>\$b</code>

- for string comparison

comparison	Evaluated to true when
<code>"\$a" = "\$b"</code>	<code>\$a</code> is the same as <code>\$b</code>
<code>"\$a" == "\$b"</code>	<code>\$a</code> is the same as <code>\$b</code>
<code>"\$a" != "\$b"</code>	<code>\$a</code> is different from <code>\$b</code>
<code>-z "\$a"</code>	<code>\$a</code> is empty

switch case

```
mycase=1
```

```
case $mycase in
```

```
1) echo "You selected bash";;
```

```
2) echo "You selected perl";;
```

```
3) echo "You selected python";;
```

```
4) echo "You selected c++";;
```

```
5) exit
```

```
esac
```

Loops

- For loop

```
NAMES=(Joe Jenny Sara Tony)
for N in ${NAMES[@]} ; do
    echo "My name is $N"
done

for f in $( ls prog.sh /etc/localtime ) ; do
    echo "File is: $f"
done
```

- While loop

```
COUNT=4
while [ $COUNT -gt 0 ]; do
    echo "Value of count is: $COUNT"
    COUNT=$((COUNT - 1))
done
```

Functions

```
function function_B {  
    echo "Function B."  
}
```

```
function function_A {  
    echo "$1"  
}
```

```
function adder {  
    echo "$(($1 + $2))"  
}
```

```
function_A "Hello!"    # Hello!
```

```
function_B              # Function B.
```

```
# Pass two parameters to function adder
```

```
adder 12 56            # 68
```

Special Variables

- `$0` - The filename of the current script.
- `$n` - The Nth argument passed to script was invoked or function was called.
- `$#` - The number of argument passed to script or function.
- `$@` - All arguments passed to script or function.
- `$*` - All arguments passed to script or function.
- `$?` - The exit status of the last command executed.
- `$$` - The process ID of the current shell. For shell scripts, this is the process ID under which they are executing.
- `!` - The process number of the last background command.

Pipelines

```
command1 | command2 | command3 | ...
```

```
#!/bin/bash
```

```
cat /proc/cpuinfo | grep processor | wc -l
```

man

- `man command`
 - shows documentation about the command
 - its description
 - its arguments
 - its flags

Class Assignment

Write a bash script with 3 functions and it takes your birthdate (day, month, year) and weekday of birth as inputs which:

The first function should validate the weekday of birth is True or not.

The second function should calculate the number of passed days after your birthday if less than 6 months is passed; otherwise, the number of remaining days to your birthday.

The third function should calculate the days' difference between your birthdate and any other date.