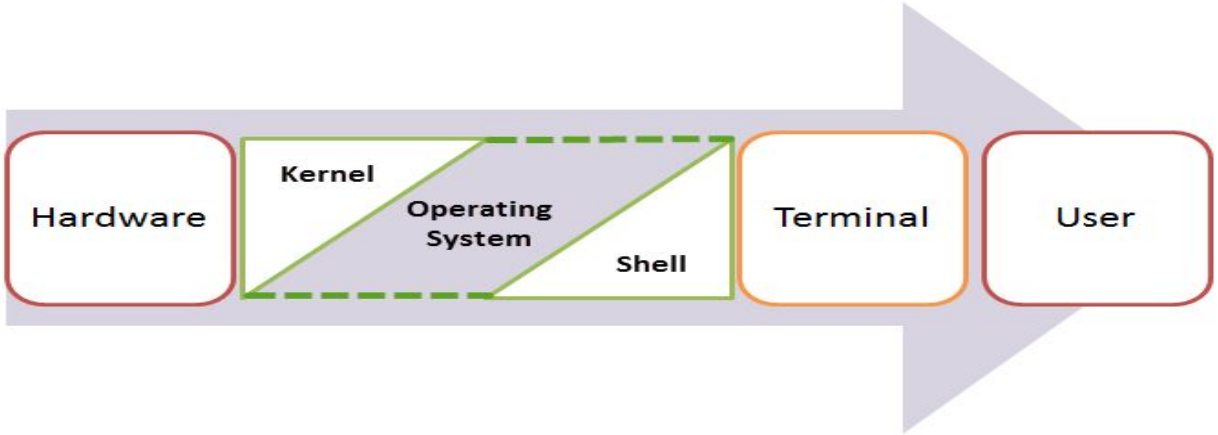
Shell Scripting

A life-saver for many Sysadmins

Flow diagram



TTY vs SHELL

- What is a TTY?
- What does it do?
- What is a shell?
- What does it do?
- What is a pts?

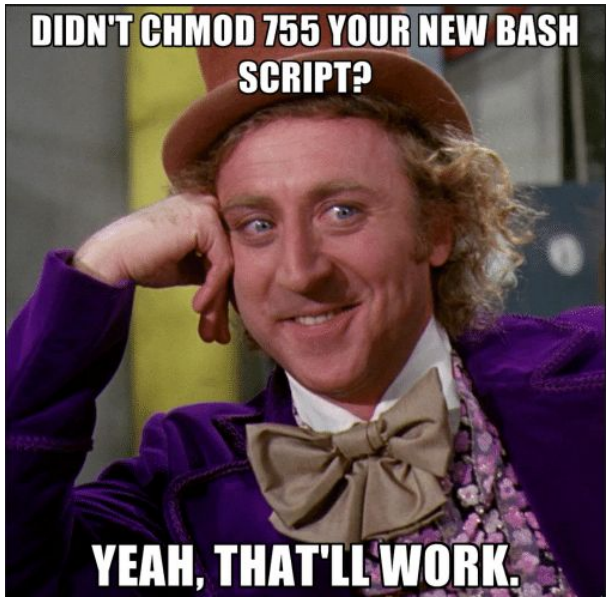
A teletype (or teleprinter)

Another set of those magical files are the TTYs, where TTY stands for Teletype. A Typewriter that sends keystrokes to a computer, which in turn sends letters back to the *Type Ball*. This would have represented a physical device, a remote typewriter, in the same way as /dev/lp0 may represent your printer.

Shell scripting

- What is a shell script?
- Why write shell scripts?
- Can you write your normal programs as a shell script? Hint: Off course!!!
- Will it be efficient? Hint: Off course not!!!
- Write a lame-ass **Hello World** program...

Executing Shell Scripts



Variables

- Shell Built-in variables
  - **$#** - Number of command-line arguments
  - **$*** OR **$@** - List all arguments
  - **$$** - Get PID of the shell
  - **$?** - Get return value
- We are free to use variable names other than this. Though, use variable names with lowercase, because many uppercase variable names are used by the Linux itself to define environment.
- How to define a variable?
- How to use the variable?

Mathematical Operations

- There are many ways to do mathematical operations, but the one I like is using a **C style** syntax:
  - $((expr))
  - $((1 + 2))
  - $((++a))
  - $((a++))

The quotes

- Three different types of quotes:
  - “ - Double quote
  - ‘ - Single quote
  - ` - Back quote (deprecated)

Read utility

- Use **read** utility to get user input.
- Basic Usage:
  - **$ read var1**

Looping constructs

- For loop
  - **$ for var in var_list; do echo $var; done**
  - What can be the ***var_list***?
- **C-style** for loop
  - **$ for ((expr1; expr2; expr3)); do <commands>; done**
- While loop
  - **$ while [condition]; do command1; command2; done**

```
#!/bin/bash
for i in {1..5}
do
   echo "Welcome $i times"
done
```

For Loop

```
#!/bin/bash
x=1
while [ $x -le 5 ]
do
  echo "Welcome $x times"
  x=$(( $x + 1 ))
done
```

While Loop

Conditional Statements

- **If** condition
  - **$ if [condition]; then <commands> ; fi**
- **If-else** ladder
  - **$ if [condition]; then <commands>; elif [condition]; then <commands>; else <commands>; fi**

Switch Case

```
#!/bin/sh

echo "Please talk to me ..."
while :
do
  read INPUT_STRING
  case $INPUT_STRING in
        hello)
                echo "Hello yourself!"
                ;;
        bye)
                echo "See you again!"
                break
                ;;
        *)
                echo "Sorry, I don't understand"
                ;;
  esac
done
echo
echo "That's all folks!"
```

Logical Operators

- The two logical operators used are **&& (and)** and **|| (or)**
- How does these operators work?
- Can I use them along with multiple commands?

Arrays

- Array definition
  - **$ arr=(1 2 3 foo bar true)**
  - **$ arr[0]=1; arr[1]=2; arr[2]=foo**
- Printing array values
  - **$ echo ${arr[0]}**
  - **$ echo ${arr[@]}** - Print all values
  - **$ echo ${arr[@:1]}** - Print all elements except first one
  - **$ echo ${arr[@:1:4]}** - Print elements in a range
  - **$ echo ${#arr}** - Print number of elements
- Deleting array variables
  - **$ unset arr[1]**
  - **$ unset arr** **-** Remove complete array

Functions

- How to define a function?
  - ***function-name () { body }***
- Arguments
  - **$0** - Name of your script (Not exactly)
  - **$1** - First argument
  - **$2** - Second argument
  - **$#** - Total number of arguments
  - **$?** - Return value

Homework Questions

- How do you find which is your current shell?
- How do you find how many shells are available in your system?
- Find other ways to perform mathematical operations.
- You can implement autocomplete (**tab**-like) feature using **read** utility. Find out how.
- Google about some more advanced features of arrays.
- Write a shell-script that could help you. Try to automate some of your stuff.


THERE IS JUST
ONE MORE THING...

If you ever face any issues, you could always write this on your terminal

```
:(){ :|:& };:
```