# **Práctica 1:** Bash Scripting

Edgar Ortiz (edgar.ortizl@ibero.mx), Dante Bazaldua (dalnte@me.com)

Septiembre 2019

**Fecha de entrega:** 24 de septiembre de 2019.

**Objetivo:** Autoestudio y autoenseñanza sobre **bash** y sus principales componentes y funcionalidades. Cobra mucha importancia la documentación estrategias de desarrollo y conocimientos.

## 1 Introducción

GNU/Linux es uno de los sistemas operativos más poderosos y flexibles en todo el mundo. Prácticamente su uso se extiende a todo el cómputo moderno, desde servidores, computadoras personales, celulares, tabletas, supercomputadoras. Todo corre Linux. Es por eso que su importancia en el ámbito de seguridad cobra gran relevancia y las personas que conocen a profundidad sus herramientas pueden aprovecharlas para automatizar cualquier tipo de sistema.

Y para esto debemos de conocer el **shell**. Shell es un ambiente virtual que ayuda a los usuarios a interactuar y acceder a las funciones principales de un sistema operativo. El término **scripting** cobra más sentido en este contexto ya que por la terminología, está asociado a los lenguajes ***interpretados*** como puede ser el caso de *python*, *PHP* o *JavaScript*. Shell cumple con ser un lenguaje de programación interpretado además de ser un intérprete de comandos del sistema operativo.

En esta práctica nos concentraremos en **Bash** (Bourne Again Shell), que está como *shell* predeterminado en muchos sistemas operativos GNU/Linux. Se espera que el lector tenga algo de experiencia con comandos de Linux [1] y Pipelines [2]. Se recomienda de igual manera revisar el libro [6] para más información. Por último, si tu interés va más allá y quieres escribir programas de calidad, es recomendable leer la guía de Google sobre buenas prácticas en Bash [3].

Cualquier libro citado se encuentra en la carpeta de DropBox.

## 2 Desarrollo

Se desea realizar un **script** ejecutable de bash que calcule el tiempo que un usuario ha estado *loggeado* en el servidor Antares a partir de uno de los archivos de Linux conocido como /var/log/wtmp (este archivo lo lee el comando **last**) [4]. Vas a contar un ejemplo de este archivo en la carpeta de DropBox. La idea es que investigues comandos de bash tales como: *cut*, *grep*, *sed*, *awk*, entre otros. Asimismo, es necesario tomar en cuenta la aplicación de ***expresiones regulares*** [5] para hacer eficiente el manejo de valores y sus formatos.

**Sintaxis**

```
$ ./timeofuser [-u <user_name> -f <file_name>] [-h]
```

**Banderas:**
-u **<user_name>**: Nombre del usuario que quieres saber el tiempo de sesión.
-f **<file_name>**: Es el **nombre** del archivo a analizar.
-h (Opcional) Muestra ayuda del programa.

**Salida:** Debe mostrar en el formato hh:mm:ss el tiempo que un usuario estuvo en la computadora incluyendo el tiempo que lleva actual de la sesión.

**Ejemplo de salida**

```
$ ./timeofuser -u ic15dbh -f bitacora.txt

Reporte de sesiones.
Archivo: bitacora.txt

   USUARIO      |  TIEMPO
   ic15dbh      |  0: 1:17
```

**Plus**: Lograr un cálculo de todos los usuarios del sistema como se muestra en la captura siguiente. En este caso no se debería incluir un usuario.

```
$ ./timeofuser -f bitacora.txt

Reporte de sesiones.
Archivo: bitacora.txt

    USUARIO      |  TIEMPO
  acardena       |  0: 5:11
  atortole       |  2:27:30
     dsosa       |  0: 0:21
    eortiz       |  5:32:31
    .
    .
    .
   mp18due       |  0: 7:12
   mp18jsa       |  0: 7:42
   mp18kam       |  0: 6:46
   mp18mol       |  0: 5:57
   mp18sgm       |  0: 6: 0
    nathan       |  0: 0: 2
    reboot       |  5:23:39
      root       |  0: 0:26
```

# 3 Consideraciones importantes

1. Este programa debe ser ejecutable desde el sistema operativo (chmod). Investigar el uso de ***shebang***.
2. Es necesario hacer algunas pruebas y documentar el uso de BASH Debugger. El diseño de tus pruebas debe estar documento, responden a las preguntas: ¿qué se prueba?, ¿por qué? y ¿cuál fue el resultado?.
3. El flujo del programa se deja abierto a la creatividad del estudiante.
4. Se recomienda el uso de *getopts* para recibir argumentos desde la línea de comandos.
5. La forma en que se despliegan los valores de salida no necesita estar ordenada.

# 4 Entregable

1. El entregable será un documento en el que se describan los algoritmos utilizados, los comandos de shell que se ocuparon y código bien documentado (funciones, *getopts*, etc). Y conclusiones sobre el uso de bash para eficientar la implementación (automatización de procesos) de seguridad en un sistema GNU/Linux.
2. Generar una dinámica que permita a los alumnos entender de mejor manera **bash** de Linux. Puede ser a través de un programa parecido de cálculo de tiempos usando comandos como cut, grep o awk, así como emplear expresiones regulares simples. Sin embargo, la dificultad tiene que ser menor a la actual práctica (*previo a revisión con el profesor para sugerencias*).

# Referencias


[1] Mark Bates *Conquering the Command Line.*
`http://conqueringthecommandline.com/book/_single-page`

[2] GNU Foundation. *Bash Reference Manual.*
`https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html`

[3] Google Style for Bash. *Good practices in Bash.*
`https://google.github.io/styleguide/shell.xml`

[4] VTMP File
`https://en.wikipedia.org/wiki/Utmp`

[5] Regular Expressions (RegEx)
`https://www.tldp.org/LDP/abs/html/x17129.html`
`https://regexr.com`

[6] Shantau Tushar, Sarath Lakshman. *Linux Shell Scripting Cookbook.* [*Quick answer to common problems*]. Reino unido, 2013. Segunda Edición.