## Consideraciones Iniciales

Para empezar, todo esto viene del archivo Shell-Primeros-Pasos de platea Link al PDF en la web de platea: P07-shell-primeros-pasos Link al PDF en mi carpeta (Esto es solo para mi) [[P07-shell-primeros-pasos.pdf]] Inicio de cualquier **script de bash**

*#! /bin/bash*

*# Autor: Manuel Sánchez Salazar*
*# Descripción: Hace tal tal*

**echo** codigo

Todos los shell scripts deberían tener las siguientes partes:

- **Cabecera (hash-bang)**: La primera línea de un shell script es un tipo especial de comentario.
  - Comienza por los caracteres #! y va seguido del nombre de un intérprete de órdenes o shell.
  - IMPORTANTE: ==Los caracteres #! deben aparecer justo al comienzo del archivo==, es decir, deben ser los dos primeros bytes del archivo de texto para que el sistema los pueda reconocer.
  - Puedes encontrar tu path de la shell de bash (el cual podría variar del de arriba) usando el comando: which bash
- **Comentarios** Se indican con el carácter # al comienzo de una línea o palabra.
  - Todos los shell scripts ==deberían incluir como mínimo quién es su autor y una breve descripción== de para qué sirve el script.
  - El intérprete ignora el comentario desde el signo # hasta el final de la línea.
  - **IMPORTANTE**: Un carácter #, sólo se interpreta como comentario, si va ==justo al principio de una línea o de una palabra de un shell script==. Si va en medio o al final de una palabra de un shell script, no se interpreta como comentario.
- **Órdenes**: Las órdenes que componen el shell script.
  - ==Se escriben de la misma forma que se escriben en la consola== cuando se ejecutan a mano
- No es necesario, pero los ==scripts deberían tener una extensión==, como por ejemplo ==.sh== , que ayudan a reconocer que archivos son scripts.
- Para ejecutarlos ==SIEMPRE HA DE DARSE EL PERMISO DE EJECUCIÓN==

## Escribiendo nuestro primer script

Script del típico "Hello World", vamos a hacer **los pasos a seguir de CUALQUIER SCRIPT,** esto son los pasos indispensables para cualquier script que hagamos.

### Elegir el directorio y abrir el script

Vamos al directorio en el que queramos guardar el script. En este caso imaginamos que la carpeta Scripts está ya hecha en nuestro directorio de usuario

**cd** ~/Scripts

abrimos el editor de textos usando el nombre del script que queramos y usando el editor de texto que queramos (recomiendo gedit para no tener que escribir en la misma consola)

gedit helloworld.sh &

==el símbolo & al final del comando hace que se ejecute de fondo==, por asi decirlo, lo que hace que no te secuestre la terminal

Se pueden usar otros en el examen, el mismo Manuel Carlos me ha dicho que se puede usar VSCode PERO no lo recomienda porque si alguien de un examen anterior se ha dejado instalada alguna extensión rara, puede suspenderte el examen (tipo integración con ChatGPT o ayudas a bash) así que la mas recomendable es gedit. ### Escribiendo y guardando el codigo Una vez abierto el explorador veremos algo de este estilo (OBVIAMENTE EL ARCHIVO ESTARA VACIO) ![[gedit guia.png]] dejo el código debajo (adaptado para mi porque me sale de los huevos)

*#! /bin/bash*

*#:Autor: Manuel sánchez Salazar*
*#:Descripción: Imprime por pantalla el mensajito*

**echo** ¡Vivan los toros!

### Ejecutando el código

Después le daremos al botón guardar (se que eres capaz de hacerlo solo/a) y le daremos los ==permisos de ejecución al script==

chmod +x holamundo.sh

Después ==ejecutaremos el script de la siguiente manera== (a estas alturas no deberías ni necesitarlo)

./holamundo.sh

### Ejecutando código sin el ./ (OPCIONAL NO NECESARIO PARA EXAMEN)

En Ubuntu Linux (desde la versión Ubuntu 20.04 LTS), la ruta ~/bin o $HOME/bin se añade automáticamente a la variable de entorno PATH (para ver su valor ejecute echo $PATH), con lo que los programas que se coloquen en esa carpeta se pueden ejecutar directamente aunque no estemos situados en ese directorio.

Para crear la carpeta ~/bin escribimos: mkdir ~/bin o mkdir ${HOME}/bin

Cerramos la sesión y volvemos a entrar, a partir de ese momento, para ejecutar los programas que hayamos copiado en la carpeta bin sólo hay que escribir su nombre (sin ./), y el intérprete los encontrará, da igual el directorio en donde estemos.

También se puede poner el comando bash holamundo.sh

## Script para imprimir el contenido de una carpeta proporcionada por el usuario

*#!/bin/bash*
*# Autora: Lina García Cabrera*
*# Descripción: Imprime la fecha y el contenido de una # carpeta que lee de la entrada*

**echo** "Hoy␣es␣" `date`
**echo**−e "\nEscribe␣la␣ruta␣al␣directorio" **read** la_ruta

**echo**−e "\n␣tu␣ruta␣tiene␣los␣siguientes␣archivos␣y␣carpetas:␣"
ls $la_ruta

(Este script en los apuntes esta mal hecho xd)

- Línea #1: El shebang (#!/bin/bash) apunta hacia la ruta de la shell de bash.
- Línea #2: Autor
- Línea #3: Descripción
- Línea #5: El comando echo muestra la fecha actual y el tiempo en la shell. Nota que date está con las comillas invertidas
- Línea #6: el comando read lee la entrada y lo almacena en la variable la_ruta
- Línea #8: ls toma la variable de la ruta y muestra los archivos y carpetas del directorio almacenado en la variable.
  - El símbolo $ hace que utilice la variable

## Parámetros

- Parámetros posicionales: argumentos presentes en la línea que invoca a la orden
  - Por ejemplo, en la orden ls /bin, hay un parámetro posicional, /bin, que ocupa la posición 1.
- Parámetros especiales: Su contenido lo rellena el intérprete de órdenes para guardar información sobre su estado actual, como por ejemplo el número de parámetros posicionales de la orden que se está ejecutando. Se referencian mediante caracteres no alfanuméricos, como por ejemplo * o #.
- Variables: Las usa el programador del shell script para almacenar cualquier información que necesite.
  - Se referencian mediante un nombre.
  - El nombre de una variable puede ser cualquier combinación de letras, números o el guión bajo (_) y siempre comienza por una letra o el guión bajo
  - Por ejemplo, nombres válidos de variables son: resultado, res_1, o res, pero no 1res ### ¿Por qué NO debo usar mayúsculas en los nombres de las variables? Porque los nombres de variables en mayúsculas están reservados para las variables internas del shell, y se corre el riesgo de sobrescribirlas. Los scripts pueden dejar de funcionar o tener un mal comportamiento. Por tanto, ==NUNCA uses MAYÚSCULAS en los nombres de tus variables.==

### Parámetros posicionales

**Accediendo al valor de los parámetros** Para acceder al valor de un parámetro se usa $

Por ejemplo, para acceder al contenido del parámetro posicional que ocupa la segunda posición se escribe ==2 == $Para acceder al contenido del parmetro especial #, se escribe ==#== Para acceder al contenido de la variable resultado, se escribe ==$resultado==

También se puede encerrar entre llaves el identificador del parámetro. Esto permite delimitar bien cual es ese identificador y es el método preferible3 aunque sea más difícil de escribir. Así, los ejemplos anteriores también se pueden escribir de la siguiente forma:

${2}
${#}
${resultado}

**Estableciendo el valor de una variable** Igual que en casi todos los lenguajes de programación, se hace con un igual:

nombre=lafosforito

Para eliminar una variable se utiliza unset

**unset** nombre

**espacios en variables NO** no debe haber espacios en esa asignación o el sistema dará un error.

nombre = lafosforito

Esto dará error.

**Ejemplo de uso de Parámetros posicionales** Vamos a usar de ejemplo el holamundo de antes

*#! /bin/bash*

*#:Autor: Manuel sánchez Salazar*
*#:Descripción: Imprime por pantalla el mensajito*

**echo** ${1}

En este caso, si nosotros invocamos al script usando el siguiente comando:

./holamundo mamahuevo chupapinga

lo que imprimirá en pantalla el script será mamahuevo ya que es el primero en las posiciones.

Esto de aquí son los argumentos, si quieres asociarlo a algo.

**Parámetros especiales**

![[parametrosespeciales.png]] Lo pongo como foto porque no me deja copiarlo y me da pereza

**Diferencia entre comillas dobles y comillas simples**

**Comillas Dobles (" ")** Permiten la expansión de variables y la interpretación de algunos caracteres especiales * Expansión de variables: Si usas comillas dobles, el shell expande las variables (como $var$), $loscaracteresdeescape($
$), yloscomandosentrecomillasinversas(). * CarcterdeEscape($
$) : puedesusarelcarcterdeescape($
$)paraevitarquealgunoscaracteresespecialesseexpandan. *$"*$"$: todos los argumentos se tratan como una sola cadena, unidos por el primer carácter del IFS (Internal Field Separator, que por defecto es el espacio). *"$@"cada argumento se trata como un elemento individual, respetando las comillas originales de los argumentos ##### Ejemplos

var="Mundo"
**echo** "Hola␣$var"

Resultado:
Hola Mundo

Aquí, el contenido de la variable $var se expandió dentro de las comillas dobles. Eso lo dicen en los apuntes, en cristiano significa que ==como el $ está dentro de las comillas dobles, se coge la variable==. esto puede resolverse poniendo delante la barra , por ejemplo:

**echo** "Me␣debes␣$10"

Esto intentara imprimir algo que este en la variable 1, la forma correcta sería

**echo** "Me␣debes␣\$10"

Esto devolverá la cadena tal y como se ve, pero la barra invertida NO se mostrará. Por si acaso preguntasen (no lo creo) a esto se le llama que un valor "escape"

**Comillas Simples ( ‘ ‘ )** Las comillas simples deshabilitan toda expansión de variables y la interpretación de caracteres especiales. Todo lo que se coloque entre comillas simples se interpreta literalmente.

No tiene mucho mas. No hay casos como los anteriores, lo que se meta entre comillas es cadena, y todo es todo, incluido la barra invertida

**echo** "Me␣debes␣\$10"

Devolverá: Me debes \$10

**Ejemplos de uso de Parámetros Especiales** Vamos a escribir un shell script que haciendo uso de los parámetros especiales, nos muestre información relevante sobre la ejecución de ese shell script. Llamaremos a ese shell script muestrainfo.sh.

*#! /bin/bash*

*#Autor: Manuel Sánchez Salazar*
*#Descripción: Muestra información relevante sobre la ejecución de este shell*
*#  script*

**echo** la ruta del script es: ${0}
**echo** El número de parámetros posicionales: ${#}
**echo** Los parámetros son: ${*}
**echo** el PID del proceso actual: ${$}

![[muestrainfolinux.png]] *LO SIENTO* si se ve borroso, ajo y agua Podemos ver que se usan los siguientes parámetros: * 0 -> Se sustituye por la ruta hacia el script * # -> Se sustituye por el número de parámetros (se puede ver en la foto) * * ->Se sustituye por el valor de todos los parámetros (una cadena de texto normalmente) * $ -> Se sustituye por el identificador del proceso actual

**Ejemplos de uso de variables** ![[usavariableslinux.png]] Si os tengo que explicar esto salíos de la carrera (Es broma pero se entiende fácil, preguntadme si lo necesitáis)

==IMPORTANTE== Al final del documento de primeros pasos de shell (este de aquí maquina) al final de la pagina 15 y en adelante son ejercicios propuestos y ejercicios que ponen los profes para probar. hacedlos TODOS, no solo leerlos, los hacéis y veis que cambian, os dejo como deberes que me resolváis los ejercicios y me los enseñéis hechos (si tenéis dudas preguntadme, es literalmente la gracia) y intentad tenerlos para el domingo 10 de noviembre como muy tarde para que podamos corregirlos o ese mismo domingo o el lunes, porfi. Este examen lo sacamos entre todos.

Figure 1: Alt Text