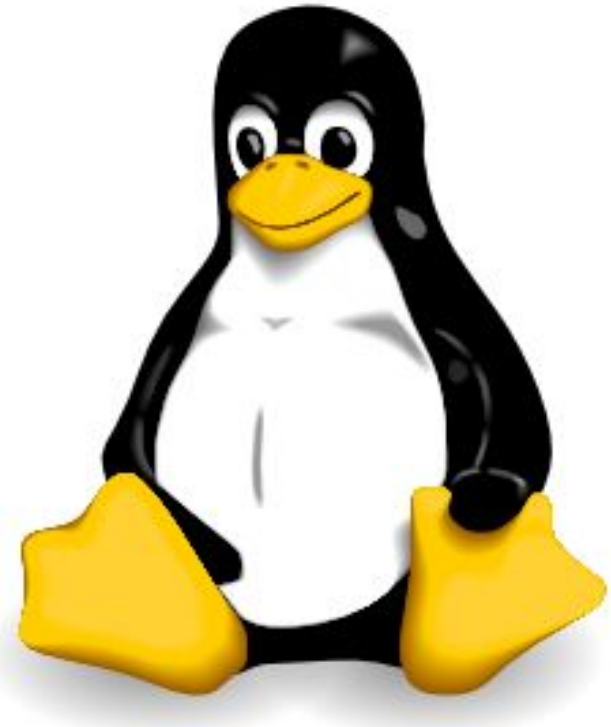
ARQUITECTURA Y SISTEMAS OPERATIVOS

Programación Bash



GNU/Linux

INTERPRETES DE COMANDOS

* La programación bash es un interprete de comando que nos permite Interpretar nuestros comandos.

* Lo que vamos a programas en mediante un shell que en este caso sera bash, hay otros interpretes de comando como ksh, sh, etc.

INTERPRETE DE COMANDO BASH

Dentro del interprete de comandos **bash,** tenemos combinaciones de teclas que nos permite realizar ciertas tareas en forma más rápida, como por ejemplo ejecutar comandos que ya hemos ejecutado.

Este interprete de comandos guarda todos los comandos que ejecutamos en un archivo dentro de nuestro **home** el cual se llama **.bash_history**.

Donde con las teclas de **flecha para arriba** podemos ir recorriendo los comandos que escribimos desde el final al principio de la lista, y con la **flecha para abajo** recorremos la lista desde donde estemos parada hacia abajo.

GRILLA DE TECLAS DE BASH

Ctrl+a	Lleva el cursor al inicio de la línea de comandos.
Ctrl+e	Lleva el cursor al final de la línea de comandos.
Ctrl+l	Limpia la terminal, similar a la que hace el comando **clear**.
Ctrl+u	Limpia desde la posición del cursor hasta el inicio de la línea. Si se está al final limpia la línea entera.
Ctrl+k	Limpia desde la posición del cursor hasta el final de la línea. Si se está al inicio limpia la línea entera.
Ctrl+h	Hace lo mismo que la tecla **backspace**, borra el carácter inmediatamente anterior a la posición del cursos.
Ctrl+w	Borra la palabra inmediatamente antes del cursor.

GRILLA DE TECLAS DE BASH

Alt+d o Esc+d	Borra la palabra siguiente después del cursor.
Ctrl+p	Establece la línea de comandos con el último comando introducido.
Ctrl+r	Inicia la búsqueda de comandos usados anteriormente, tecleando parte de un comando usos anteriores que hayamos realizado incluyendo las opciones y parámetros. Hecha una búsqueda pulsando de nuevo la combinación de teclas encontraremos coincidencias anteriores.
Ctrl+c	Termina el proceso que se esté ejecutando, útil para recuperar el control del sistema.
Ctrl+d	Sale de la terminal, similar al comando **exit**.
Ctrl+z	Suspende la ejecución del proceso que se está ejecutando y lo pone en segundo plano, con el Comando **fg** podremos volver a continuar su ejecución.

GRILLA DE TECLAS DE BASH

Ctrl+t	Intercambia la posición de los dos caracteres antes del cursor, útil para corregir malos tecleos.
Esc+r	Intercambia la posición de las dos palabras antes del cursor, útil para corregir malos tecleos.
Alt+f	Mueve el cursor al inicio de la palabra siguiente de la línea, lo mismo que **Ctrl+right** en la terminal de GNOME.
Alt+b	Mueve el cursor al inicio de la anterior de la línea, lo mismo que **Ctrl+left** en la terminal de GNOME.
Tab	Autocompleta comandos o rutas de directorios o archivos.

PROGRAMACION BASH

$ nano prog01.sh

```
#!/bin/bash

echo “Hola Mundo”
```

$ chmod +x prog01.sh

./prog01.sh

PROGRAMACION BASH

Variables

```
$ env
$ printenv
```

Mostrar variables de entorno:

```
$ echo $HOME
```

Asignar variable:

```
$ MIVAR='HOLA'
$ echo $MIVAR
```

PROGRAMACION BASH

$ nano prog02.sh

```
#!/bin/bash

echo –n “Introduzca su nombre: “
      read NOMBRE

      echo “Su nombre es: “ $NOMBRE
```

$ chmod +x prog02.sh

$ ./prog02.sh

PROGRAMACION BASH – CONDICION (IF)

```
    if [ .... ]; then
    else
    fi

if [ ... ]; then
      elif [ ... ]; then
      fi
    fi
```

PROGRAMACION BASH – CONDICIONES

[-a FILE] [-e FILE]	Verdadero si existe el archivo.
[- b FILE]	Verdadero si es de bloque especial.
[-c FILE]	Verdadero si es de carácter especial.
[-d FILE]	Verdadero si es un directorio.
[-b FILE]	Verdadero si un bloque especial.
[-x FILE]	Verdadero si el archivo es de ejecución.
[-r FILE]	Verdadero si el archivo es de lectura.
[-w FILE]	Verdadero si el archivo es de escritura.
[-L FILE]	Verdadero si el archivo es un link.
[-Z FILE]	Verdadero si el es de 0 bytes.

PROGRAMACION BASH – CONDICIONES NUMÉRICAS

[EXP1 –eq EXP2]	EXP1 es igual a EXP2.
[EXP1 –ne EXP2]	EXP1 es distinto de EXP2.
[EXP1 –ge EXP2]	EXP1 es mayor o igual que EXP2.
[EXP1 –gt EXP2]	EXP1 es mayor que EXP2.
[EXP1 –le EXP2]	EXP1 es menor o igual que EXP2.
[EXP1 –lt EXP2]	EXP1 es menor que EXP2.

PROGRAMACION BASH – CONDICION (IF)

$ nano prog03.sh

```
#!/bin/bash
echo –n “Introduzca su nombre: “
read NOMBRE

if [ “$NOMBRE” == “PABLO” ]; then
   echo “- SU NOMBRE ES : PABLO”
        echo “”
      else
         echo “- SU NOMBRE ES : “ $NOMBRE
     fi
```

$ chmod +x prog03.sh

$ ./prog03.sh

PROGRAMACION BASH – BUCLE (FOR)

```
for variable in 1 2 3 4…N; do
        comando 1
        comando 2
        ….
    comando N
     done
```

PROGRAMACION BASH – BUCLE (FOR)

```
for variable in {1..N}; do
        comando 1
        comando 2
        ....
   comando N
    done
```

PROGRAMACION BASH – BUCLE (FOR)

$ nano prog04.sh

```
#!/bin/bash

for NUMEROS in {1..5}; do
        echo “- Valor: “ $NUMEROS
    done
```

$ chmod +x prog04.sh

$ ./prog04.sh

PROGRAMACION BASH – BUCLE (FOR)

$ nano prog05.sh

```
#!/bin/bash

for ARCHIVOS in /etc/*; do
        echo “- Nombre del archivo : “ $ARCHIVOS
    done
```

$ chmod +x prog05.sh

$ ./prog05.sh

PROGRAMACION BASH – BUCLE (FOR)

$ nano prog06.sh

```
#!/bin/bash

for ARCHIVOS in /etc/*; do
        if [ ${ARCHIVOS} == '/etc/passwd' ]; then
           echo "- El archivo es : " ${ARCHIVOS}
           break
        fi
     done
```

$ chmod +x prog06.sh

$ ./prog06.sh

PROGRAMACION BASH – BUCLE (WHILE)

```
while true; do
        comando 1
        comando 2
        ....
    comando N
     done
```

PROGRAMACION BASH – BUCLE (WHILE)

```
while read LINEA; do
        comando 1
        comando 2
        ....
    comando N
     done < ARCHIVO
```

PROGRAMACION BASH – BUCLE (WHILE)

$ nano prog07.sh

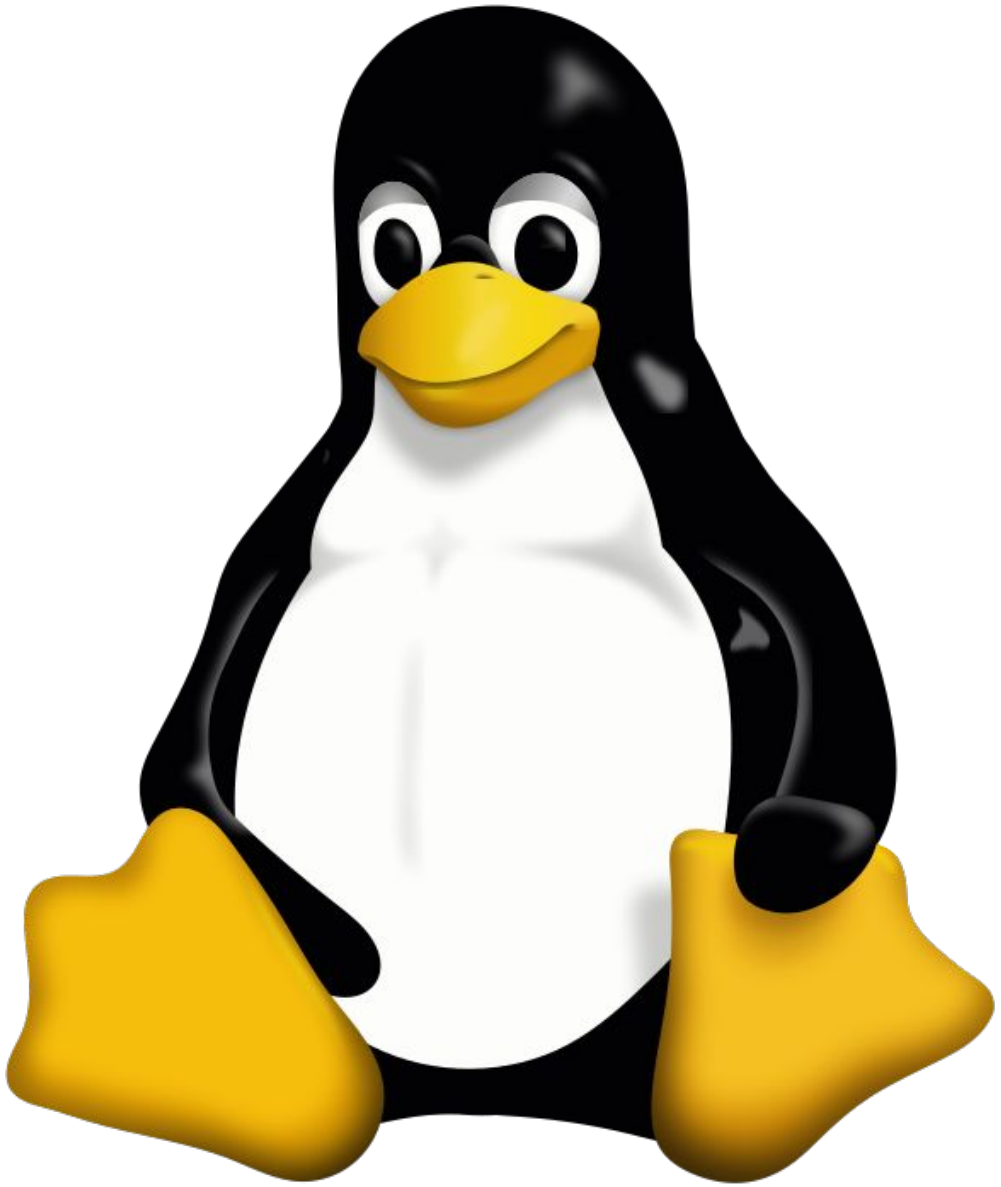
```
#!/bin/bash

while read LINEA; do
        USUARIO=`echo ${LINEA} | cut –d ‘:’ –f1`
        echo “- Nombre del usuario : “ ${USUARIO}
 done < /etc/passwd
```

$ chmod +x prog07.sh

$ ./prog07.sh

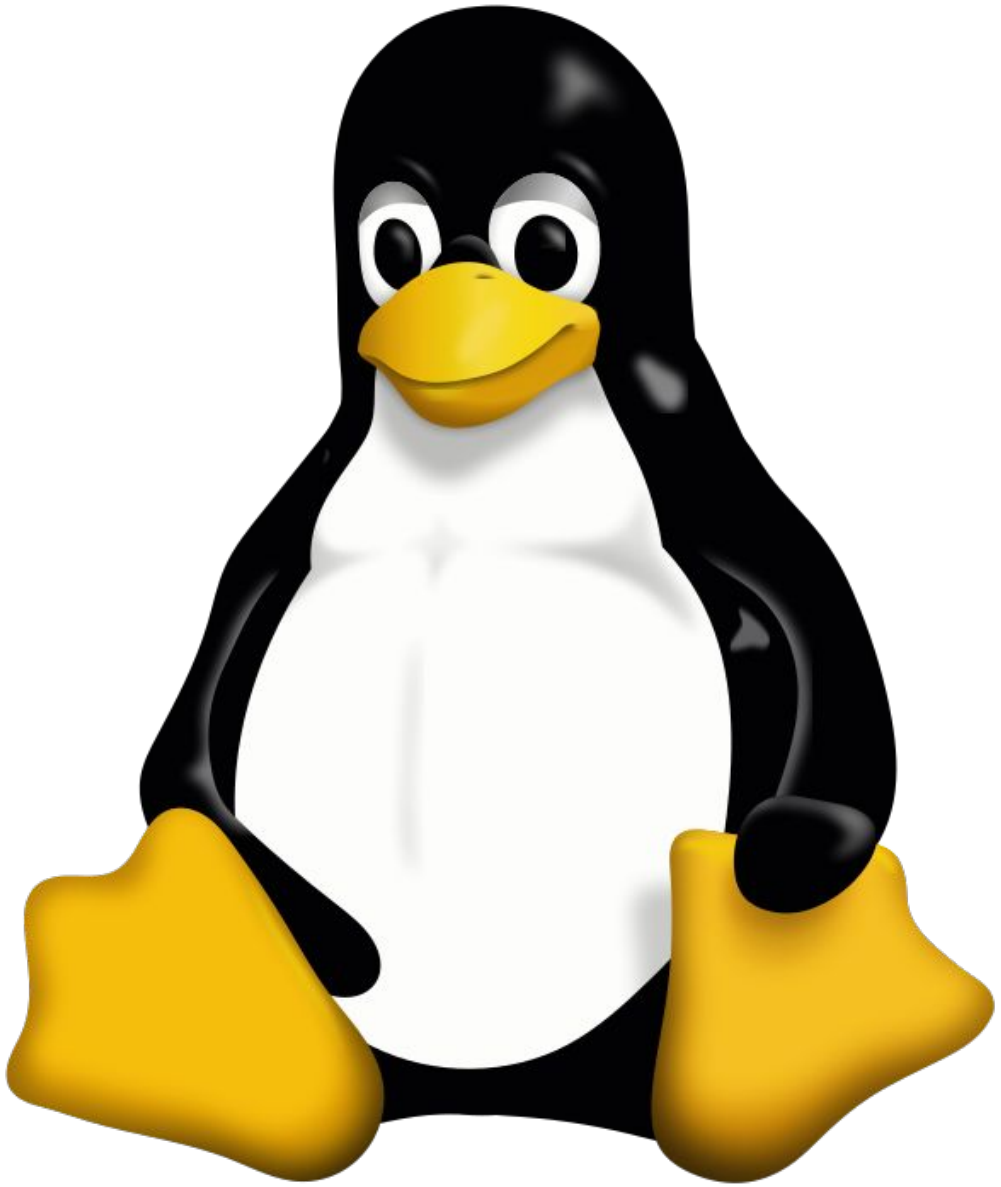
EJERCICIO PARTE 1



EJERCICIO 1

- Listar del archivo /etc/passwd el usuario y su descripción.

EJERCICIO PARTE 2



EJERCICIO 2

- Crear el siguiente archivo

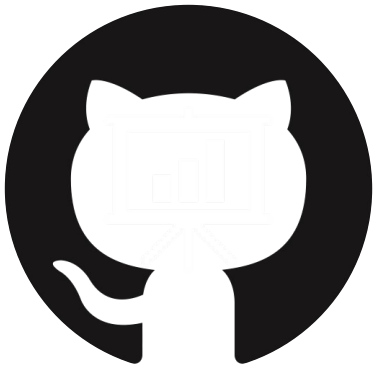
nano usuarios.txt

```
usuario1|Usuario de prueba 1|/bin/bash
usuario2|Usuario de prueba 2|/bin/bash
usuario3|Usuario de prueba 3|/bin/false
```

Tomar esa lista de usuarios y mediante un script generar los usuarios y asignarle como password el nombre del usuario.

RECURSOS

SO-UTNFRA.SLACK.COM



GITHUB.COM/MARTIN919191/ARQUITECTURAYSISTEMASOPERATIVOS

SO.UTNFRA@GMAIL.COM