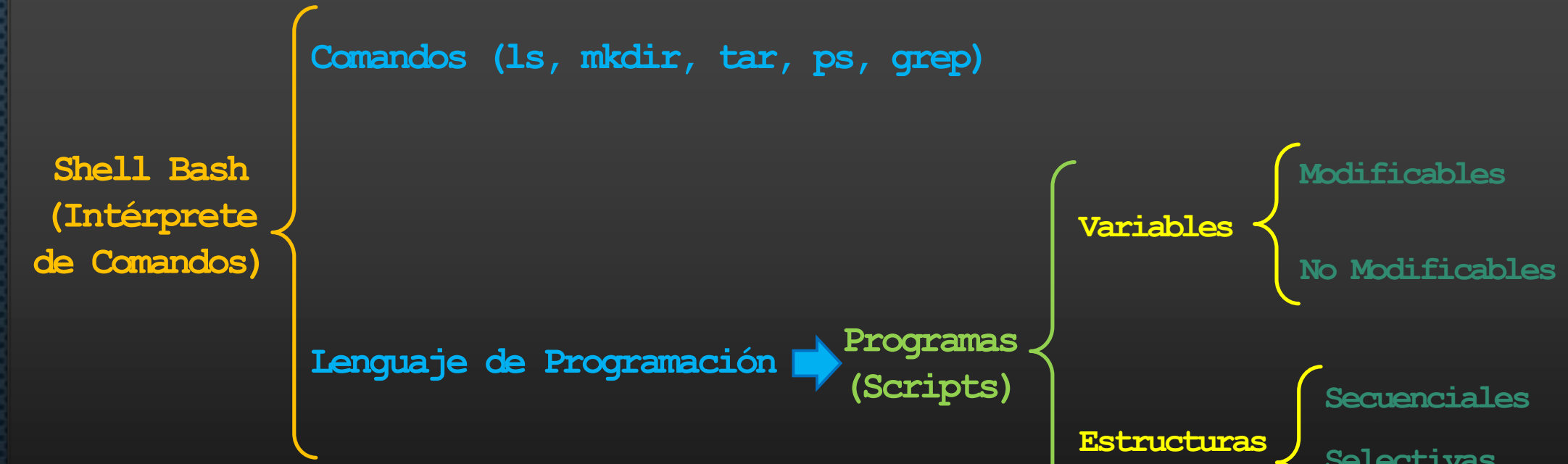
Facultad de Ingeniería
Universidad Nacional de Jujuy

Laboratorio de Sistemas Operativos I

Trabajo Práctico N° 9

Scripts (parte 1)

PROGRAMACIÓN DE LA SHELL



VARIABLES DE LA SHELL

- Las variables de entorno o variables de la Shell permiten configurar el sistema definiendo valores para la ejecución de los distintos procesos.
- Ejemplo: PATH, HOME, LANG, USER, PWD
- Variables no modificables
  - $0 (nombre del comando o programa que se ejecuta)
  - $1 - $9 (parámetros de un comando o programa)
  - $*, $@ (todos los parámetros de un comando o programa)
  - $# (cantidad de parámetros de un comando o programa)
  - $$ (id del proceso actual)
  - $? (resultado del último comando, correcto o incorrecto)
  - $! (id del proceso en segundo plano más reciente)

OPERACIONES SOBRE VARIABLES

- Crear de variables

```
nombre_variable=valor
```

- Listar variables

```
env (sólo variables de entorno)
set, typeset, declare (todas)
```

- Eliminar variables

```
unset nombre_variable
```

- Acceder al contenido de variables

```
$nombre_variable
```

DECLARACIÓN DE VARIABLES

- Por defecto, las variables de entorno son de tipo cadena de caracteres. No obstante, pueden definirse variables de tipo entero.

```
typeset o declare usuario
typeset o declare directorio="/home/alumno"
typeset o declare -i num1 num2 suma=0
```

- Las variables pueden definirse como de sólo lectura

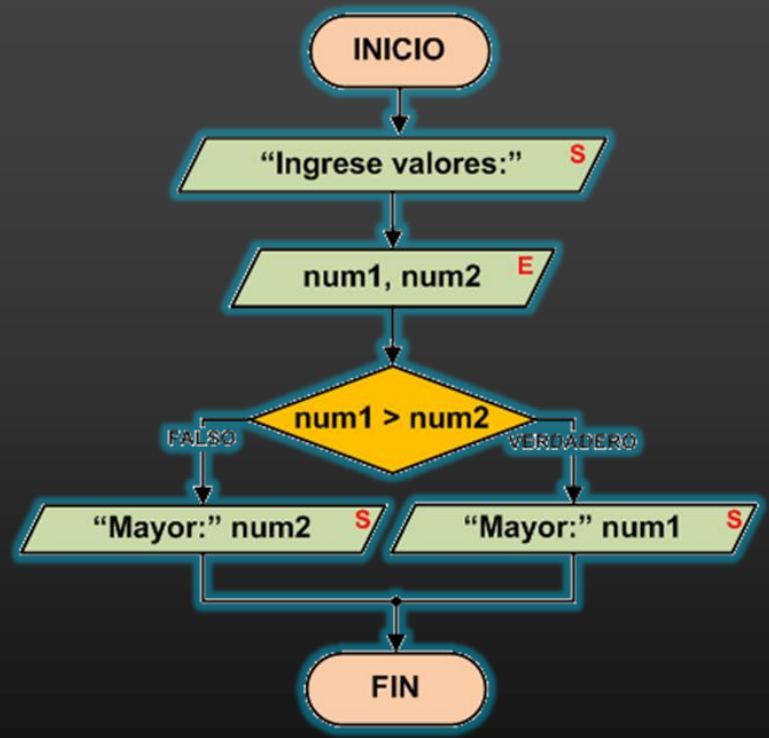
```
typeset o declare -r tp="tp9"
```

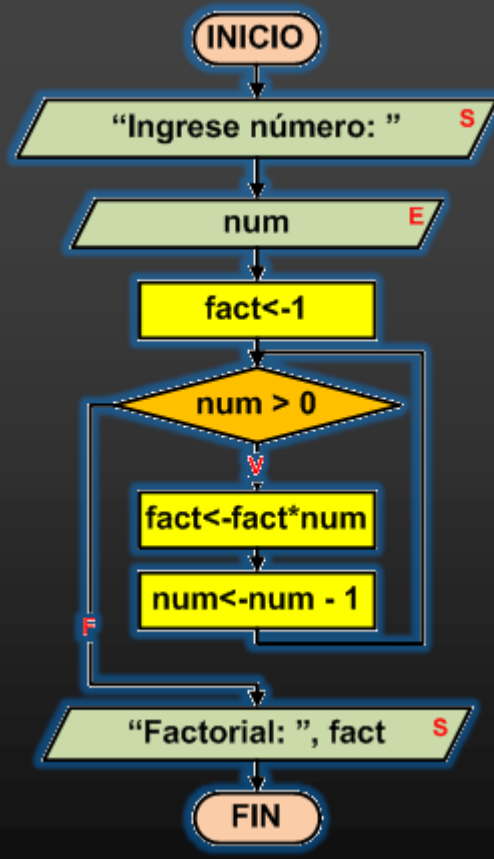
- Las variables pueden exportarse para ser usadas por subprocesos

```
typeset o declare -x materia="lsoi"
```

ESTRUCTURAS DE CONTROL

- Secuenciales
  - echo
  - read
- Selectivas
  - if
  - case
- Repetitivas
  - for
  - while
  - until





SECUENCIALES

- El comando **echo** permite presentar en pantalla una cadena de caracteres o el contenido de una variable.

```
echo [-opciones] [texto,variable]
```

- El comando **read** permite capturar la entrada del usuario.

```
read [-opciones] nombre_variable
```

CÓMO CREAR Y EJECUTAR UN SCRIPT

- Editar el archivo

```
alumno@debian:~$ nano ejemplo1.sh
```

- Agregar permisos de ejecución

```
alumno@debian:~$ chmod u+x ejemplo1.sh
```

- Ejecutar el script

```
alumno@debian:~$ ./ejemplo1.sh
```

EJEMPLO 1

- Codifique un script que muestre por pantalla el mensaje "APU 2008 – LSOI 2022".

```
#!/bin/bash
echo "APU 2008 – LSO I"
```

- Codifique un script que muestre una cadena ingresada por el usuario.

```
#!/bin/bash
echo -n "Ingrese cadena: "
read frase
echo "Frase ingresada: " $frase
```

EJEMPLO 2

- Codifique un script que sume 2 valores ingresados por el usuario.

```
#!/bin/bash
declare -i num1 num2 suma
echo "Ingrese numero: "
read num1
read -p "Ingrese numero: " num2
suma=$num1+$num2
echo "suma: " $suma
```

PARÁMETROS POSICIONALES

- Los parámetros posicionales permiten identificar los argumentos que acompañan a un comando o un script al ejecutarse.

```
alumno@apu2008:~$ cp /etc/passwd /home/alumno/usuario.txt
                  $0      $1                $2
```

```
$0      cp (nombre del programa)
$1-$9   /etc/passwd ($1), /home/alumno/usuario.txt ($2)
$*,$@   /etc/passwd /home/alumno/usuario.txt (todos los argumentos)
$#      cantidad de argumentos: 2
```

EJEMPLO 3

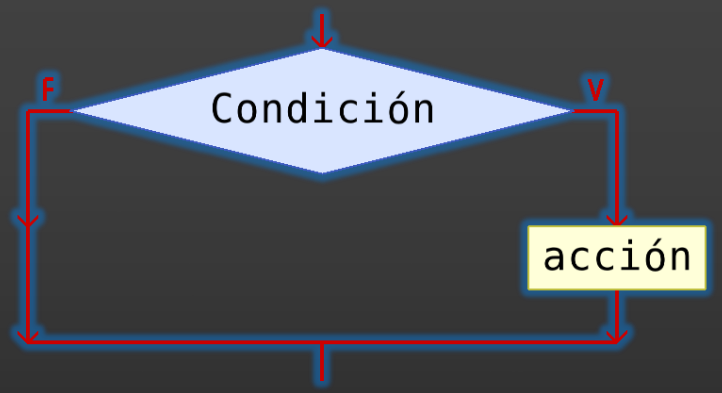
- Modifique el ejemplo 2 de modo que la suma se realice usando parámetros.

```
#!/bin/bash
declare -i suma
suma=$1+$2
echo "suma: " $suma
```

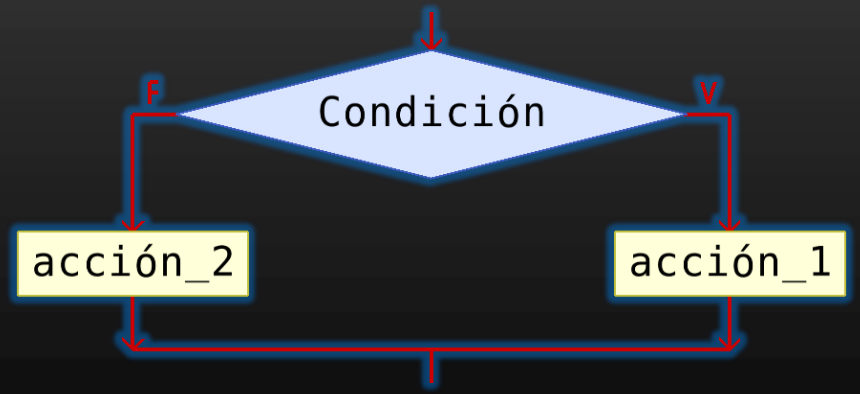
SELECTIVAS SIMPLES Y DOBLES

- El comando if permite verificar una condición elegir entre 2 caminos alternativos de acción.

```
if [ condición ]; then
     acciones
fi
```



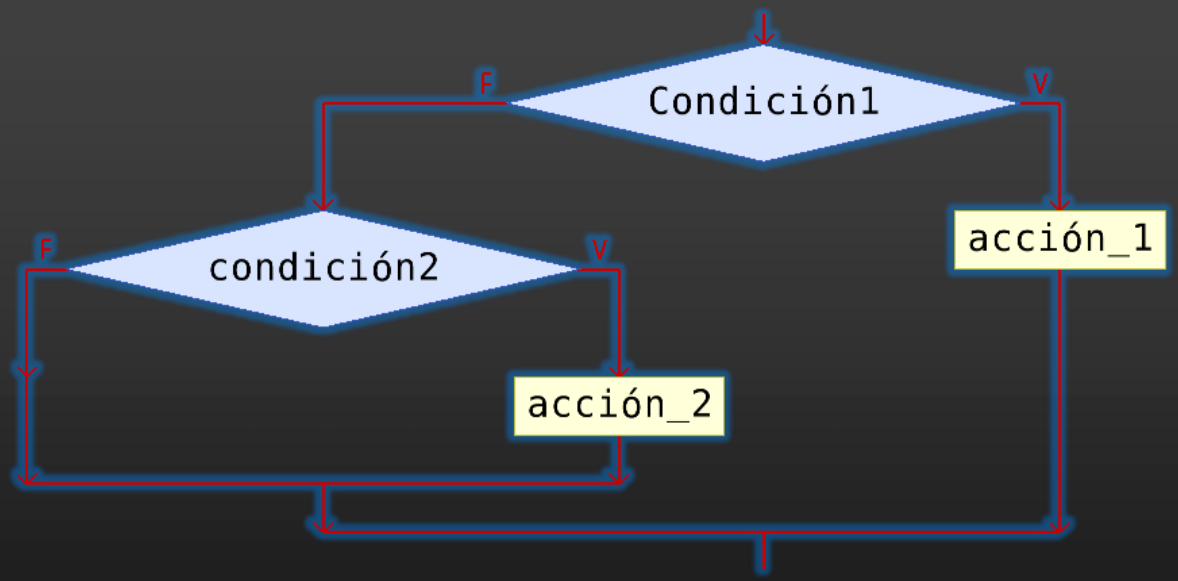

```
if [ condición ]; then
     acciones
else
     acciones
fi
```



ANIDAMIENTO DE SELECTIVAS

- Las estructuras selectivas pueden anidarse.

```
if [ condición_1 ]; then
      acción_1
else
    if [ condición_2 ]; then
        acción_2
    fi
fi
```




```
if [ condición_1 ]; then
      acción_1
elif [ condición_2 ]; then
      acción_2
fi
```

OPERADORES (1)

- Operadores lógicos
  - Negación (not): !
  - Conjunción (and): -a, &&
  - Disyunción (or): -o, ||

```
if [ cond1 -a cond2 ];
if [ cond1 -o cond2 ];
if [ cond1 ] && [ cond2 ];
if [ cond1 ] || [ cond2 ];
```

OPERADORES (2)

- Operadores numéricos
  - -eq (igual)
  - -ne (distinto)
  - -lt (menor que)
  - -le (menor o igual)
  - -gt (mayor que)
  - -ge (mayor o igual)
- Operadores de cadenas
  - = (igual)
  - != (distinto)
  - -n (cadena no nula)
  - -z (cadena nula)
- Operadores de archivos
  - -e (existe)
  - -d (directorio)
  - -f (archivo ordinario)
  - -r (tiene permiso lectura)
  - -w (tiene permiso escritura)
  - -x (tiene permiso ejecución)
  - -s (tamaño mayor que cero)
  - -L (archivo simbólico)

EJEMPLO 4

- Modifique el ejemplo 3 de modo que se verifique que se cuenta con suficientes parámetros para realizar la suma.

```
#!/bin/bash
declare -i suma
if [ $# -eq 2 ]; then
    suma=$1+$2
    echo "suma: " $suma
else
    echo "error de parámetros"
fi
```

EJEMPLO 5

- Codifique un script que compare 2 valores, pasados por parámetros, y muestre el mayor. ¿Qué ocurre si los valores son iguales?

```
#!/bin/bash
if [ $1 -gt $2 ]; then
  echo "el mayor es " $1
else
  echo "el mayor es " $2
fi
```

EJEMPLO 6

- Codifique un script que determine si un parámetro es un archivo ordinario y si además tiene tamaño mayor que cero.

```
#!/bin/bash
if [ -f $1 ]; then
  if [ -s $1 ]; then
     echo $1 "archivo ordinario, tamaño mayor que
cero"
  else
     echo $1 "archivo ordinario y vacío"
  fi
else
  echo $1 "no es un archivo ordinario"
fi
```

EJEMPLO 7

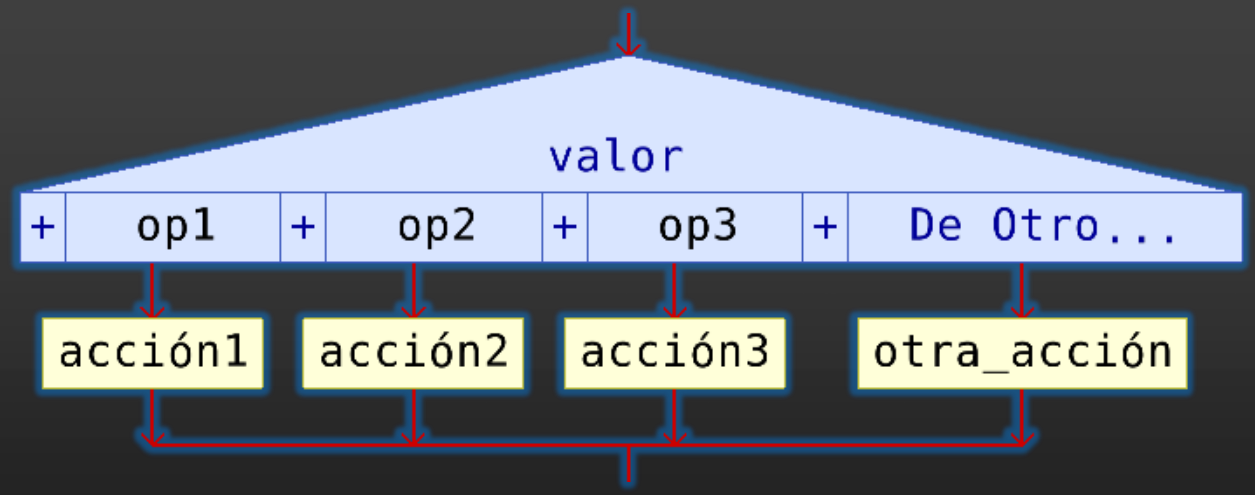
- Codifique un script que busque una cadena en un archivo, considere que ambos pasados por parámetros. Si la búsqueda resulta exitosa muestre el mensaje "Cadena encontrada", caso contrario presente el mensaje "Deberás seguir buscando".

```
#!/bin/bash
grep -i $1 $2 > /dev/null
if [ $? -eq 0 ]; then
  echo "Cadena encontrada"
else
  echo "Debes seguir buscando"
fi
```

SELECTIVAS MÚLTIPLES

- El comando case permite elegir entre n caminos alternativos de acción.

```
case $valor_variable in
  op_1) acciones_1
        ;;
  op_2) acciones_2
        ;;
  op_3) acciones_3
        ;;
  *) otra_acción
     ;;
esac
```



EJEMPLO 8

- Codifique un script que presente un menú con las siguientes opciones: a) Mostrar el nombre del usuario, b) Mostrar el tamaño del directorio del usuario, c) Mostrar los procesos de la terminal actual (formato largo).

```
#!/bin/bash
echo "a) Mostrar usuario"
echo "b) Mostrar tamaño de directorio personal"
echo "c) Mostrar procesos del usuario"
read -p "Ingrese opción: " op
case $op in
 a|A) whoami
      ;;
 b|B) du -sh ~
      ;;
 c|C) ps -l
      ;;
 *) echo "error de opcion"
      ;;
esac
```