SENTENCIAS CONDICIONALES

- Las sentencias condicionales son estructuras de control que nos permiten ejecutar lineas de código solamente si se cumplen las condiciones que hemos especificado

- IF

```
if CONDICION; then
    comandos
fi
```

SE EJECUTAN SI LA CONDICIÓN ES CIERTA

```
if CONDICION; then
    comandos
else
    otros comandos
fi
```

SE EJECUTAN SI LA CONDICIÓN ES FALSA

SCRIPT NOTAS.SH

- Recibirá un número del 0 al 10 y responderá con la calificación según la tabla:

0	Suspenso
1	Suspenso
2	Suspenso
3	Suspenso
4	Suspenso
5	Aprobado
6	Bien
7	Notable
8	Notable
9	Sobresaliente
10	Sobresaliente

SENTENCIAS CONDICIONALES

- La sentencia **case** ejecutará unas instrucciones u otras en función del valor que encuentre en una variable o expresión

- CASE

```
case valor in
        exp1)
            ...
            ultimaorden1;;
        exp2)
            ...
            ultimaorden2;;
        expN)
            ...
           ultimaordenN;;
esac
```

EN LAS EXPRESIONES PODEMOS USAR LOS MISMOS COMODINES QUE EN LOS NOMBRES DE LOS FICHEROS:
? * [..]

LA ULTIMA ORDEN TERMINA CON DOBLE PUNTO Y COMA PARA INDICAR QUE TERMINA EL "CASE"

SENTENCIAS CONDICIONALES

- Sustitución o expansión de comandos: Permite ejecutar un comando e insertar en valor devuelto por éste, en cualquier lugar de otra instrucción.

- Para ello se introduce la orden entre $() o entre comillas invertidas ``

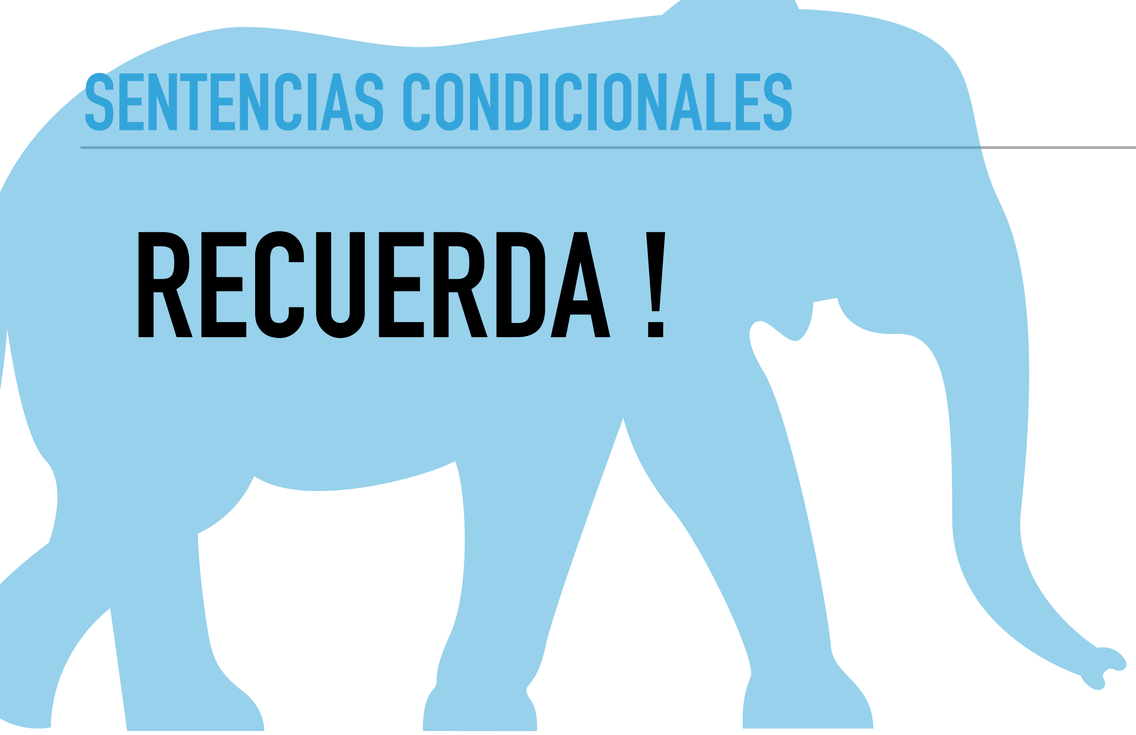
Ejemplos

```
echo Hoy es: $(date)
```

```
=> Mostrará por pantalla la fecha actual
```

```
echo El directorio actual ocupa: `du -sh .`
```

```
=> Mostrará un mensaje indicando cuanto ocupa el
directorio actual
```



RECUERDA !

*La variable **$?** contiene el resultado del último comando ejecutado*

Si el comando se ha ejecutado con éxito tendrá el valor 0,
en caso contrario, un valor distinto de cero.

```
ls -d /etc/
echo $?      => imprimirá 0 por la pantalla
ls -d /noexiste/
echo $?      => imprimirá 2 por la pantalla
```

SENTENCIAS CONDICIONALES

- Una forma abreviada de ejecutar un comando dependiendo del resultado de otro, es utilizar las operaciones lógicas AND y OR, expresadas con && y ||

Se pondrían dos comandos unidos por alguno de estos operadores, por ejemplo:

```
test $1 = "suma"  &&  resultado=$(($v1+$v2))
```

Usando && el segundo comando sólo se ejecuta si el primero se cumple, pues si es falso no necesita consultar el resultado del segundo comando para saber que el resultado de la unión de ambos será falso.

En el caso de ||, sólo ejecuta el segundo comando si el primero es falso.

```
id ana &> /dev/null || echo "El usuario ana NO existe"
```