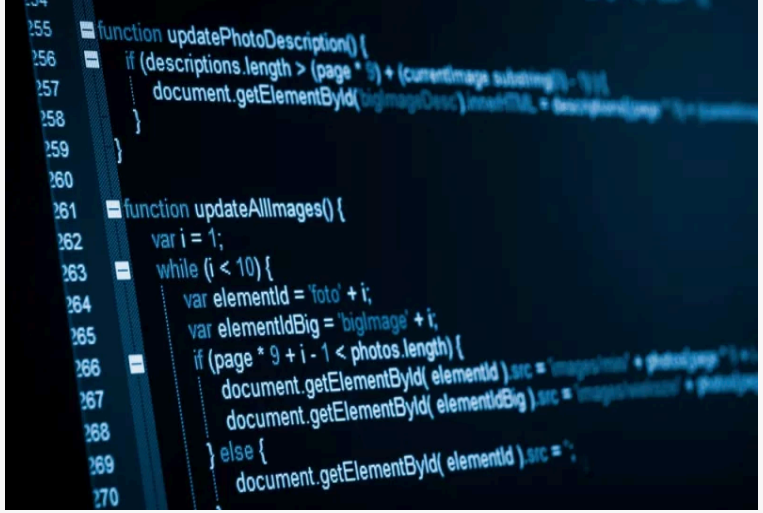
PEDECIBA

FACULTAD DE CIENCIAS
UDELAR | fcien.edu.uy

FACULTAD DE MEDICINA MONTEVIDEO 1875

Introducción a la línea de comandos y a la programación para análisis bioinformáticos.



Teórico 8

Programación en el Shell II:
Loops (for, while)

FOR

- Es una declaración en programación que permite que un **código sea ejecutado de forma reiterada**.
- Permite la **iteración** de un proceso.
  - correr un comando de Unix N veces.
  - Leer y escribir diferentes archivos.
- En general se itera sobre una serie de palabras en una lista.
- Puede correrse en la línea o en el script.

Sintaxis:

```
for X in 1 2 3 4 … N
do
  command1 X
  command2
done
```

Ejemplo I: rango numérico

1. **#!/*bin/bash***
2. **for** i **in** 1 2 3 4 5
3. **do**
4. **echo** "Welcome $i times"
5. **done**
6. **exit**

- En la 2da línea **se declara la variable** que tomara diferentes valores en las iteraciones.
- La 4ta línea puede extenderse, incluso pueden agregarse acciones en el cuerpo del *loop*.
- El do (3ra línea) y el done (5ta línea) limitan el *loop for*.

Ejemplo I: rango numérico

```
#!/bin/bash
for i in 1 2 3 4 5
do
    #saludar
    echo "Welcome $i times"
done
exit
```

El **indentado** y los **comentarios** no son necesarios en *bash*, pero es una buena costumbre para facilitar el **debug** (depuración) del programa.

Ejemplo II: recorrer lista

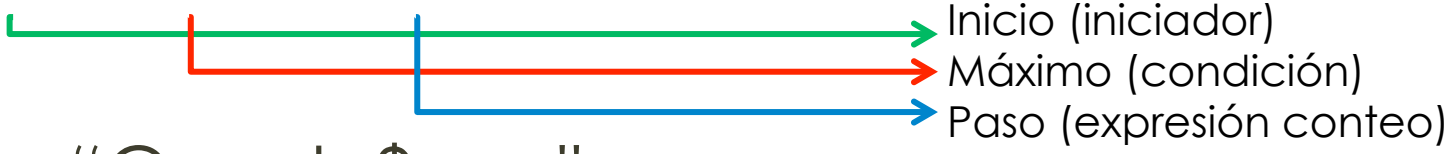
```
#!/bin/bash
lista_fasta=$(ls *.fasta | sed 's/.fasta//g')
for fas in $lista_fasta
do
    echo "$fas"
    transeq "$fas".fasta "$fas".aa
    infoseq "$fas".aa >> información.out
done
exit
```

Ejemplo III: "*loop for*" anidados

```
#!/bin/bash
lista_fasta_files=$(ls *.fasta | sed 's/.fasta//g')
for fas in $lista_fasta_files
do
    echo "$fas"
    lista_encabezados=$(egrep ">" "$fas".fasta |sed 's/>//g')
    for head in $lista_encabezados
    do
        echo "$head"
        echo "$head" | egrep "hypothetical" | sed 's/>/&NA/g' >> H.out
    done
done
exit
```

Ejemplo IV: "*loop for*" con conteo

```
#!/bin/bash
for (( c=1; c<=5; c++ ))
do
   echo "Count, $c ..."
done
exit
```



WHILE

- Es una declaración en programación que se utiliza para **controlar el flujo de procesos**.

- Permite que ciertos códigos o comandos sean **repetidos hasta que se cumple con una condición** dada.

Sintaxis:

```
while [ condition ]
do
   command1
   command2
   command3
done
```

Ejemplo I: contador

```
1. #!/bin/bash
2. x=1
3. while [ $x -le 5 ]
4. do
5.     echo "Contando ..." "$x"
6.     let x=x+1
7. done
8. exit
```

Ejemplo I: contador

```
1. #!/bin/bash
2.    x=1
3.    while [ $x -le 5 ]
4.    do
5.        echo "Iteracion numero: " "$x"
6.        let x=x+1
7.    done
8.    exit
```

Para Valores:
l = lower
e = equal
g = greater
-lt, le, eq, ge, gt

Ejemplo I: responde al usuario

```
1. #!/bin/bash
2. x=$(echo "yes")
3. while [ $x == "yes" ]
4. do
5.     echo "tres tristes tigres en un trigal"
6.     echo "Quiere seguir repitiendo (yes / no)"
7.     read x
8. done
9. exit
```

Para *strings*:
== equal
!= different

Break& Exit

```
while [ conditionRR ]
do
    statements1
    statements2
    if (conditionMM)
    then
        break #Abandona el while.
    fi
    statements3
done
```

#Ejecutados mientras la condición(RR) sea verdadera y hasta que se de una condicion(MM), si es que ocurre.

#Ejecutado mientras la condicion (RR) sera verdadera y mientras no ocurra condición (MM).