Presentación <Scripting>
Práctica 3

Arquitectura de Computadores

3º de grado en Ingeniería Informática y

3º de doble grado en Ing. Informática y Matemáticas

Bash Scripting

```
#!/bin/bash

# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Indicar qué intérprete debe usarse para ejecutar los comandos que siguen.
- Aparte de bash, funciona también con Python, PHP, Perl, etc.

Bash Scripting

```
#!/bin/bash


# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Comentarios de una sola línea.
- Comentarios multi-linea no soportados.

Bash Scripting

```
#!/bin/bash


# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Declaración de variables
  - sin espacio entre el nombre de variable y el igual
  - sin espacio entre igual y valor
- Acceso a variables
  - Signo $ antes del nombre de variable

Bash Scripting

```
#!/bin/bash


# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Se pueden ejecutar comandos de la línea de comandos directamente y rellenar los parámetros con variables que declaremos.

Bash Scripting

```
#!/bin/bash

# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Se ejecutar comandos de la línea de comandos y usar su resultado.
- En este caso, devolverá una secuencia de números desde 1 hasta 1234 (valor de la variable P) en saltos de 16.

Bash Scripting

```
#!/bin/bash


# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Se pueden usar estructuras de control tipo for, while, select, etc.
- En este ejemplo, el for iterará por todos los valores que ha retornado “seq”.

Bash Scripting

```
#!/bin/bash

# comentario
P=1234
file="cache.dat"
rm -f $file
echo "header" > $file
for i in $(seq 1 16 $P); do
    echo "$i" >> $file
done
```

- Se puede redireccionar la salida de los programas a ficheros.
  - “>”: empezar a escribir desde el principio del fichero borrando su contenido anterior.
  - “>>”: añadir al final del fichero.
  - En todo caso añade un retorno de línea.

Ejecución de Scripts Bash

> chmod u+x bash_example.sh

Dar permisos de ejecución al script creado

> ./bash_example.sh

Ejecutar el script.

Alternativa:

> source bash_example.sh

Otros comandos útiles

- **cat**: volcar un fichero por pantalla

  cat data.txt — retornar el contenido del fichero data.txt

- **tee**: volcar los datos de entrada a uno o varios ficheros y por pantalla

  echo "datos" | tee d1.txt d2.txt — pasar “hola” al comando tee como entrada y que este lo escriba en los ficheros d1.txt y d2.txt

- **head/tail**: mostrar las primeras/últimas líneas de un fichero

  head -n 5 data.txt — retornar las 5 primeras líneas de data.txt

  tail -n 5 hola1.txt — retornar las 5 últimas líneas de data.txt

- **sort**: ordenar las líneas de un fichero

  sort -k 2,2 data.txt — ordenar data.txt usando la columna 2 como clave

- **grep**: filtrar las líneas de un fichero según un criterio dado

  grep "time" data.txt — retornar las líneas de data.txt que contengan “time”

- **awk**: parsear y/o modificar los datos de entrada

  awk '{print $2 "\t" $3}' data.txt — retornar las columnas 2 y 3 del fichero data.txt

  awk 'length($0) > 18' data.txt — retornar líneas con más de 18 caracteres

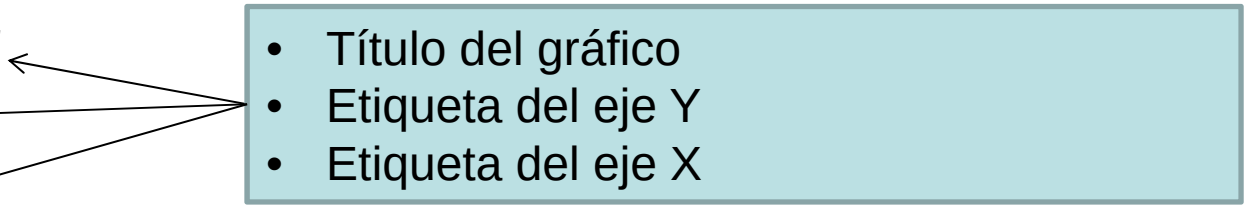
Tutoriales Bash Scripting

- **Bash Scripting tutorial.**
  - https://linuxconfig.org/bash-scripting-tutorial

- **Introduction to bash shell redirections.**
  - https://linuxconfig.org/introduction-to-bash-shell-redirections

GNUplot scripting

```
set title "Time vs Size"
set ylabel "Time"
set xlabel "Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
    "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```



GNUplot scripting

```
set title "Time vs Size"
set ylabel "Time"
set xlabel "Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
    "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```

- Mostrar leyenda abajo a la derecha
- Mostrar la cuadrícula del gráfico

GNUplot scripting

```
set title "Time vs Size"
set ylabel "Time"
set xlabel "Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
    "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```

- Configurar el formato de salida: fichero de imagen PNG
- Definir el nombre del fichero de salida: slow_fast_time.png

GNUplot scripting

```
set title "Time vs Size"
set ylabel "Time"
set xlabel "Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
     "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```

- Definir las series de datos a pintar, ambas del fichero slow_fast_time.csv
  - Serie “slow”: X => columna 1, Y => columna 2. Usar línea de grosor 2.
  - Serie “fast”: X => columna 1, Y => columna 3. Usar línea de grosor 2.
- Dejar a GNUplot que escoja el color de cada serie.

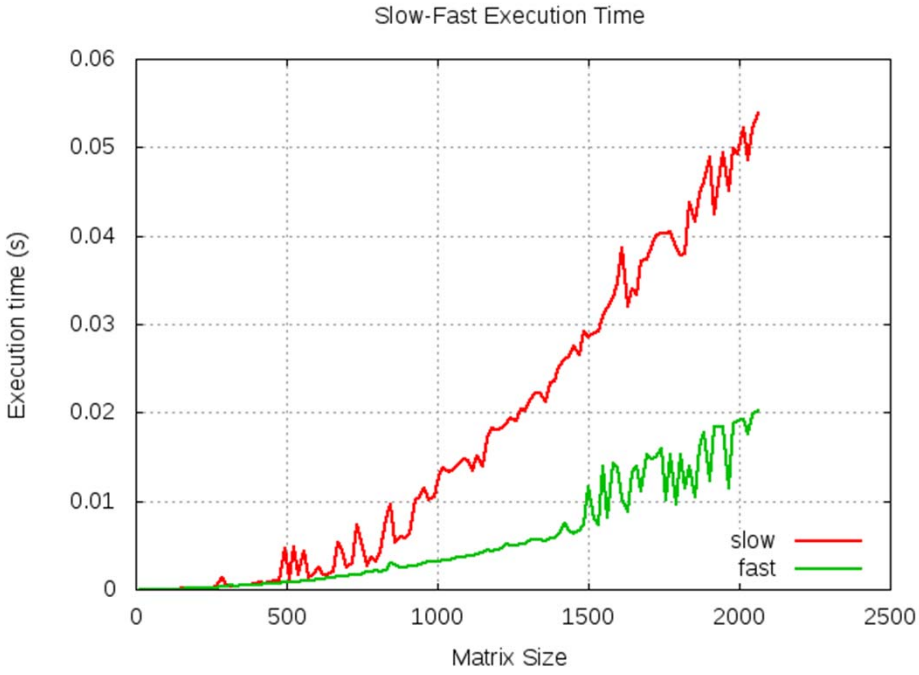
GNUplot scripting

```
set title "Time vs Size"
set ylabel "Time"
set xlabel "Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
    "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```

- Repintar para que genere el fichero
- Finalizar script y salir de GNUplot

GNUplot scripting

```
set title “Slow-Fast Execution Time"
set ylabel “Execution Time (s)"
set xlabel “Matrix Size"
set key right bottom
set grid
set term png
set output "slow_fast_time.png"
plot "slow_fast_time.csv" using 1:2 with lines lw 2 title "slow", \
    "slow_fast_time.csv" using 1:3 with lines lw 2 title "fast"
replot
quit
```



Ejecución de Scripts GNUplot

> gnuplot script.gnuplot

Ejecutar script

Modo avanzado: incrustar el script de GNUplot en Bash:

```
#!/bin/bash
# script to generate data.csv
gnuplot << EOF
set term png
set output “data.png"
plot " data.csv " using 1:2 with lines lw 2 title "slow“
replot
quit
EOF
```

- Ejecutar gnuplot y pasarle como entrada todo lo que viene después de “<< EOF”
- Dejar de pasarle comandos cuando se encuentre la etiqueta indicada al ejecutar el comando (“EOF” en este caso).

Tutoriales GNUplot Scripting

- **GNUPLOT 4.2 - A Brief Manual and Tutorial**
  - http://people.duke.edu/~hpgavin/gnuplot.html