Práctica 6: Programación en el shell Bash Parte II

Sistemas Operativos

Grado en Ingenieria en Informática
Departamento de Ingeniería Informática
Universidad de Cádiz

Curso 2015-16

Contenido

1. ¿Qué es un script?
2. ¿Qué son los parámetros en el shell bash?
3. Expansión básica de parámetros
4. Parámetros posicionales y especiales
5. Variables
6. Funciones
7. Ámbito de los parámetros
8. Manejo de la entrada estándar
9. Operaciones aritméticas

Scripts I

- ¿Por qué se programa el shell Bash?
  - Automatización de tareas (evita errores).
  - Ayuda para la programación de tareas para ejecutarlas de manera desatendida.
  - Permite hacer tareas repetitivas fácilmente (por ejemplo, crear 2.000 páginas web).
- Los programas creados en un lenguaje de shell como Bash se denominan scripts y suelen ser interpretados.

Scripts II

- ¿Cómo se crea un script?
  - Suelen llevar la extensión `.bash` o no llevar ninguna.
  - Primera línea: `#!/bin/bash`
  - Las demás líneas son órdenes que funcionen en la consola.
  - Los comentarios son líneas que empiezan por el símbolo `#`.

Scripts III

- Ejemplo de script:

```
#!/bin/bash
mkdir /tmp/d4
cp /etc/passwd /tmp/d4
cd /tmp/d4
#Realizo la compresión con gzip
gzip *
echo "Script terminado"
```

Scripts IV

Ejecución de scripts:

- Dos formas (la segunda requiere que el script tenga permiso de ejecución):
  - `bash nombre_script [ parámetro ... ]`
  - `nombre_script [ parámetro ... ]`
- La ejecución del script es la ejecución de las órdenes secuencialmente de igual modo que si se escribieran en una consola.
- Depuración:
  - `bash -xv nombre_script [ parámetro ... ]`

Parámetros en Bash I

Tipos de parámetros:

- Variables Ya conocidas.
- Parámetros posicionales Permiten comunicarse con el script en su invocación (sus nombres son números naturales: 1, 2, 3 . . .).
- Parámetros especiales Los establece la shell automáticamente (son `* @ \# ? - \$ !`).

Uso de parámetros:

- Anteponiendo el $, por ejemplo $PAR1 (se protegen con ${PAR1})

Parámetros en Bash II

- Si invoco:
  - `bash script1.bash casa a8a`
- Entonces:
  - $1 vale 'casa'
  - $2 vale 'a8a'
  - $# vale 2 (porque hay 2 parámetros)
  - $@ vale todos los parámetros
  - $* vale todos los parámetros separados por IFS

Parámetros en Bash III

Desplazamiento de parámetros con shift

- A veces queremos “consumir parámetros” para tratarlos.
- ¡Cuidado!, es irreversible.
- `shift` desplaza los parámetros hacia la izquierda perdiéndose los parámetros que están en las primeras posiciones.

Variables

- Nombre sensible a mayús. y minús. Tienen un débil tipado.
- Declaraciones:
  - Variable global: `VARIABLE=valor`
  - Variable local a una función: `local VARIABLE=valor`
  - `declare [+|- atributo] VARIABLE=valor`

Parámetros en Bash IV

Atributos de variables

- i, r, x, a

Vectores

- No hay límite máximo para el tamaño de un vector.
- El primer elemento es el de índice 0.
- `V[1]="Hola"`
- `asignaturas=([10]=SSOO [8]=MP [15]=BD)`

Parámetros en Bash V

Operaciones con variables

- Asignar el valor de otra variable: `VAR1=$VAR2`
- Concatenar dos variables: `VAR1=$VAR1"-"$VAR2`
- Calcular su longitud: `${#VAR1}`
- Asignar la salida de una orden (cuidado si tiene saltos de carro)
  - `VAR1=$(orden)`
  - `VAR1=`orden``
- Se puede eliminar su valor: `unset variable`

Funciones

- Similar a las funciones de otros lenguajes de programación.
- Para definir una función:

```
function nombre_func () {
órdenes
}
nombre_func () {
órdenes
}
```

- Sólo puede devolver enteros, que indican el status de salida de la función.
- El valor de salida de la función se puede utilizar en el programa que la llama con el parámetro $?.
- Se invocan como una orden cualquiera:

```
$ f1 arg1 arg2
```

Ámbito de los parámetros

- Las variables son globales por omisión.
- Dentro de una función se definen variables locales con `local` o `declare`.
- Los parámetros (de un script o función) son locales, al igual que $@, $* y $#:
  - con $@ cada parámetro es una cadena.
  - con $* todos los parámetros son una cadena y se separan con $IFS.

Leer de la entrada estándar

La orden `read`

- Lee una línea de la entrada estándar y la parte en palabras separadas por el valor de IFS, asignando cada palabra a una variable.
- Si hay más palabras que variables, se emparejan uno a uno menos el final, que va a la última:
  - `read A B C`
  - Entrada: 10 Manolo Juan 3 4
  - Reparto: $A=>10, $B=>Manolo , $C=>Juan 3 4

Operaciones aritméticas

- Sintaxis muy estricta:
  - `let "X=X + 1"`
  - `X=$((X+1))`
  - `X=X+1`
- También pueden usarse operadores lógicos (!, ==, ...) y operadores con asignación (+=, /=, ...)