DE O A 100 CON BASH SHELL SCRIPTING Y AWK

ADOLFO SANZ DE DIEGO

COMMIT-CONF 2018

2 AUTOR

2.1 ADOLFO SANZ DE DIEGO

Asesor. Desarrollador. Profesor. Formador.

- Blog: asanzdiego.com
- Correo: asanzdiego@gmail.com
- GitHub: github.com/asanzdiego
- Twitter: twitter.com/asanzdiego
- LinkedIn: in/asanzdiego
- SlideShare: slideshare.net/asanzdiego

2.2 DISCLAIMER

- He intentado montar una charla útil para principiantes, pero con tips para gente con más conocimientos.
- Espero haberlo conseguido :-)

3 SHELL SCRIPT

3.1 INTRODUCCIÓN

- Un shell script es un fichero de texto con comandos.
- Para ejecutarlo no hay ni que compilar, ni tener nada instalado, y puedes utilizar todos los comandos del sistema.
- Usalo para automatizar tareas del sistema y/o procesar datos de forma rápida.
- Usalo para hacer pequeños scripts, no grandes programas, para eso tienes otros lenguajes.

3.2 HOLA MUNDO

```
#! /bin/bash
# script showing a "Hello world!"
echo "Hello world!"
```

3.3 PERMISOS

• Antes de ejecutar hay que darle permisos, pero recuerda, un gran poder conlleva una gran responsabilidad :-)

```
$ chmod +x 01_hello_world.sh
```

3.4 EJECUCIÓN

- Para ejecutar un script:
 - si está en el \$PATH, el nombre directamente
 - si no, desde la carpeta, ./nombre.sh

\$./01_hello_world.sh

examples/01_hello_word.sh

3.5 NOMBRES

• Estas son mis reglas de estilo (en realidad de Google), si no te gustan tengo otras. :-)

ficheros_shell_scripts.sh
VARIABLES_DE_ENTORNO
variables_locales
nombres_de_funciones

3.6 INICIO

• Es una buena práctica empezar los shells scripts así:

```
#! /bin/bash

# Short description of the script

set -o errexit # the script ends if a command fails
set -o pipefail # the script ends if a command fails in a pipe
set -o nounset # the script ends if it uses an undeclared variable
# set -o xtrace # if you want to debug
```

3.7 EXIT

- Es una buena práctica terminar los shells scripts con un un código de retorno:
 - mayor de 0 si ha habido un error
 - igual a 0 si termina correctamente (si no pones 'exit')

```
#! /bin/bash
num_params=$#

if [ $num_params -lt 1 ]]; then
    echo "At least one parameter must be introduced."
    exit 1 # error and exits with a return code > 0
fi
echo "All ok" # ok and exits with a return code = 0
```

3.8 PARCIALES

• Podemos guardar el resultado de la ejecución de comandos en variables con \$(codigo):

```
date=$(date +'%Y-%m-%d %H:%M:%S')
```

3.9 FUNCIONES

• Es una buena práctica asignar los parámetros de la función al principio ya sean como variables locales o, si es necesario, globales.

```
my_function() {
  local function_param_1="$1"  # 1st param assigned as local
  global_param_2=${2:-default}  # 2nd param assigned as global (default)
  local function_num_params=$#  # numbers of params assigned as local
  local all_function_params=($@) # all params assigned as a local
}
```

```
my_function function_param_1 function_param_2 ... function_param_N
```

3.10 MAIN

• Es una buena práctica estructurar el código en funciones y tener una función main que llamamos al final del script con todos los parámetros.

```
# Main function
main() {
  check "$@"
  params "$@"
  print
}
main "$@" # call the main function with all the parameters
```

3.11 PARÁMETROS

• Los parámetros los cogemos de la linea de comandos cuando ejecutamos.

```
# Default values
default_2="Commit Conf"

param_1=$1  # the first script param
param_2=${2:-${default_2}} # the second script param (with default value)
num_params=$#  # the numbers of script params
all_params=($@)  # all params assigned as an array
$ ./02_parameters.sh param_1 param_2 ... param_N
```

examples/02_parameters.sh

3.12 TEMPLATE

asanzdiego / commit-conf-charla-shell-script-y-awk	Unwatch ▼ 1	★ Star 1	∛ Fork 0
♦ Code ① Issues 0	Settings		
Branch: master ▼ commit-conf-charla-shell-script-y-awk / template.sh		Find file	Copy path
asanzdiego casi terminado		c0903d5	13 hours ago
1 contributor			
Executable File 105 lines (89 sloc) 1.73 KB	Raw	Blame History	/ 1
1 #! /bin/bash			
3 # small template for my bash shell scripts.			
5 set -o errexit # the script ends if a command fails			
6 set -o pipefail # the script ends if a command fails in a pipe			
7 set -o nounset # the script ends if it uses an undeclared variable			
8 # set -o xtrace # if you want to debug			
8 # set -o xtrace # if you want to debug			
8 # set -o xtrace # if you want to debug 9 10 # Defaults values			
<pre># set -o xtrace # if you want to debug # Defaults values logLevel=4 #4-debug;3-info;2-success;1-warning;0-error</pre>			
8 # set -o xtrace # if you want to debug 9 10 # Defaults values			

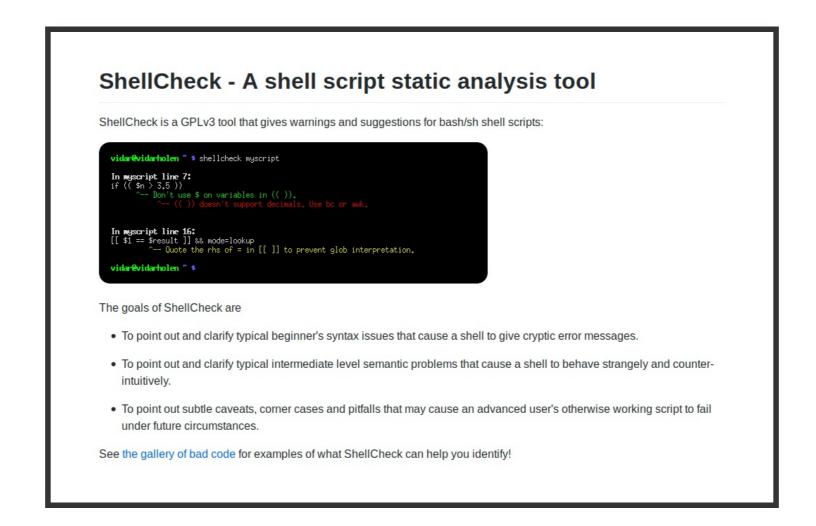
resources/template.sh

3.13 CHULETA

				ENTRECOMILL ADD		ESTRUCTURAS DE CONTROL		
REDIRECCIONES		OPERADORES		#1 RUTA	# ruta al interprete (/bin/bash)	if expresion1; then	# conficional	
output (salida estándar)		operadores aritméticos		\carácter	# valor literal del carácter	bloque1 elif expresion2; them	# si expresión1 entonces # bloque1	
tee fichero	# output a fichero y a pantalla	*	# suns	lineal \ linea2	# para escribir en varias lineas	bloque2	# sino y expresión2 entonces	
> fichero	# output a fáchero	-	# resta	"cadera"	# valor literal cadena	bloque3	# bloque2 # si minguna entonces	
>> fichero	# output al final del fichero	-	# multiplicación	"cadera"	# valor literal cadena, excepto 8 ' \	fi	a bloque2	
> /dev/null	# descarta output	/	# division	EXPANSIÓN		case VARIABLE in	# selectiva	
error		× .	# resto	[prefijo]{cadi,[,],cadN}[sufijo]	# = precadisuf precadisuf	patronii patroniN) bloquei::	W si VARIABLE coincide con patronesi W entonces bloquei	
241	# error a output	**	# incremento	S(VARIABLE:-valor)	W SI WARIABLE nula, retorna valor	patron21[[patron2N]	# si VARIABLE coincide con patrones2	
2> fichero	# error a fichero		# decremento	\$[VARIABLE:=valor]	V SI VARIABLE nula, asigna valor	bloque2;;	# entonces bloque2 # si ningune	
2>> fichero	# error al final del fichero	operadores comparaciones numéric		\$[VARIABLE:?mensaje]	W si WARIABLE nula, mensaje error y fin	bloqueDefecto ;;	# entonces bloqueDefecto	
2> /dev/null	# descarta error	numerol -eq numero2	# numerol distinto our numerol	<pre>\${VARIABLE:inscio}</pre>	W recorta desde inicio hasta el final	esac		
output y error		numero1 -me numero2	# numerol distinto que numero2	S[VARIABLE: inicio:longitud]	W recorta desde inicio hasta longitud	for VARIABLE in LISTA; do bloque	# iterativa con lista # ejecuta bloque sustituyendo	
2-41 tee fichero	# ambos a fichero y a pantalla	numero1 -1t numero2 numero1 -1e numero2	# numero1 menor que numero2	S[Iprefijo*]	W nombres de variables con prefijo	done	# VARIABLE por cada elemento de LISTA	
⇒ fichero	# ambos a fichero	numero1 -1e numero2 numero1 -gt numero2	# numero1 menor o igual que numero2 # numero1 mayor que numero2	\$(#VARIABLE)	W número de caracteres de VARIABLE	for ({epr1; expr2; expr3; }); do	# Iterativa con contador	
&⇒ fichero	# ambos al final del fichero	numero1 -ge numero2	# numeroi mayor o igual que numero2	\$(masser(*))	V elementos de ARRAY	done	# primero se evalúa expl # luego mientras expl sea cierta	
VARIABLES variables de entorno			a make or make, a store die unselen	\$[VARIABLE#patron]	V elimina minimo patrôn desde inicio		# se ejecutan el bloque y expr3	
variables de entorno	# directorio de trabajo actual	operadores lógicos	# NOT	\$[VARIABLEWpatron]	W elimina māximo patrôn desde inicio	while expresion; do	# bucle "mientras"	
50.0 PMD	# directorio de trabajo actual	66 8	# AND	\$(WRIMBLEMpatron)	W elimina minimo patrôn desde fin	bloque	# se ejecuta bloque # mientras expresión sea cierta	
9910	# identificador del proceso padre	112	# OR	S [VARIABLENKpatron]	W elimina māximo patrôn desde fin	until expresion; do	# bucle "hasta"	
SHOS THANK	# nombre del ordenador	operadores de ficheros		\$[VARIABLE/patron/reemplaze]	W reemplaza primera coincidencia	expression	# se ejecuta bloque	
SUSE R	# nombre del usuario	-e fichero	# existe	S[VARIABLE//patron/reomplaze]	V reemplaza todas las coincidencias	done	# hasta que expresión sea cierta	
SHORE	# directorio del usuario	-s fichero	# no està vacio	S((expression))	W sustituye expresión por su valor	[function] expression () { [return [valor]]	# función # se invoca con	
SATE	il rutas hisqueda de conandos	-f fichero	# normal	S[expresion]	# sustituye expresión por su valor) ((())	# nombreFunción [parami parami]	
SLANG	# idioma para los mensajes	-d fichero	# directorio	EJECUCIÓN		INTERACTIVIDAD		
SFUNCMARE	# nombre función en ejecución	-h fichero	# enlace simbolico	./comando	# ejecuta desde directorio actual	read [-p cadena] [variable1]	# imput	
SLINENO	# número de linea actual (del script)	-r fichero	# permiso de lectura	SRUT A/comando	# ejecuta desde cualquier sitio		# lee teclado y asigna a variables # nuede mostrarse un mensale antes	
SRANDOM	# número aleatorio	-w fichero	# permiso de escritura	comando	# ejecuta si està en el SPATH		# si minguna variable, REPLY = todo	
variables especiales	- 11361 0 616610130	-x fichero	# permiso de ejecución	. script	# ejecuta exportando variables	echo cadena	# output	
50	# nombre del script	-0 fichero	# propietario	\$(comando parant parant)	W ejecuta de forma literal	-m no hace salto de linea -e interpreta caracteres con \	# manda el valor de la cadena # a la salida estándar	
800	g parametro N	-6 fichero	# pertenece al grupo	'comando parant parant'	# ejecuta sustituyendo variables	print f	# output formateado (squal que C)	
95	# identificador del proceso actual	f1 -ef f2	# f1 y f2 enlaces mismo archivo	C1 C2	# ejecuta en segundo plano	CONTROL DE PROCESOS	(41)	
\$1	# identificador del último proceso	f1 -et f2	# f1 más nuevo que f2	a ; a	# redirige salida c1 a entrada c2 # ejecuta c1 y luego c2	commod &	V e secuta en secundo plano	
Ser .	# todos los parámetros recibidos	f1 -et f2	# fi más antiguo que f2	C1 44 C2		bg númeroProceso	# continúa ejecución en segundo plano	
Sir	# número de parâmetros recibidos	operadores de cadenas	•		W ejecuta c2 si c1 termina sin errores	fig númeroProceso	# continua ejecución en primer plano	
\$7 # (G:normal, >Grerror)	# código de retorno del último comando	-n cadena	# no vacia	ARGUMENTOS DE LÍNEA DE COMANDOS	p cyclore ca as ca contract con errores	jobs	# muest ra procesos en ejecución	
shift	# \$1=\$2, \$2=\$3, \$(N-1)=\$(N)	-z cadena	# vacia	while getopts "hs:" option ; do	# getops + "opciones disponibles"	kill sefal PIDIjnümeroProcesol	# mata proceso(s) indicado(s)	
ARRAYS		cadena1 = cadena2	# cadenal igual a cadena2	case "Soption" in	W mientras haya argumentos	exit código	# salir con còdigo de retorno	
declare -a ARRAY	# declaración array	cadenal == cadena2	# cadena1 igual a cadena2	h) DO_MELP=1 ;; s) argument=90PTARG ; DO_SEARCH=1 ;;	# selectionamos # -h sin optiones		# (6=normal, >6=error)	
ARRAYm[valor1 valorN]	# asignación compuesta	cadena1 != cadena2	# cadenai distinta a cadena2	") echo "Invalid" ; return ;;	# -s con opciones en SOPTARG	trap [comando] [código1]	# ejecuta comando cuando señal(es)	
ARRAY[8] =valorN	# asignación simple			esac	a error	wait [PID1]nümeroProceso1]	# espera hasta fin proceso(s) hijo(s)	
ASSAY=[[N]=valorN valorM [P]=valorP]	# asigna celdas N, M y P					mice -n prioridad comando remice -n prioridad comando	# ejecuta comando con prioridad [-29/19] # modifica prioridad comando [-20/19]	
S(ARRAY[N])	# valor celds N					remare on privo 1030 Continuo	# -20 māxima prioridad y 19 minima	
S(ASSAY[*]) Autor: Adalfo Sano De Chago (<u>superfinos</u> - <u>Minoj (LTA)</u>	# todos los valores (Limetir/District) Limetir (C.St.6)	Autori Adolfo Sens De Giogo (<u>propriémo</u> - <u>Elori</u>)	Hardwarfellinger father times of P. S.	Antoni Adulfo Sano De Stega (<u>more Stega - Stegi Stitled</u>)	.okedb(Llob-Dar-(hu-ther) Linewise (CD-1)	Autor: Adulfo Serv De Diago (<u>averdiago</u> - <u>Singl Sto</u>	habilidadis(Hilabhar (hilter) Liowite <u>6.0</u>	

resources/chuleta-shell-script.pdf

3.14 SHELLCHECK



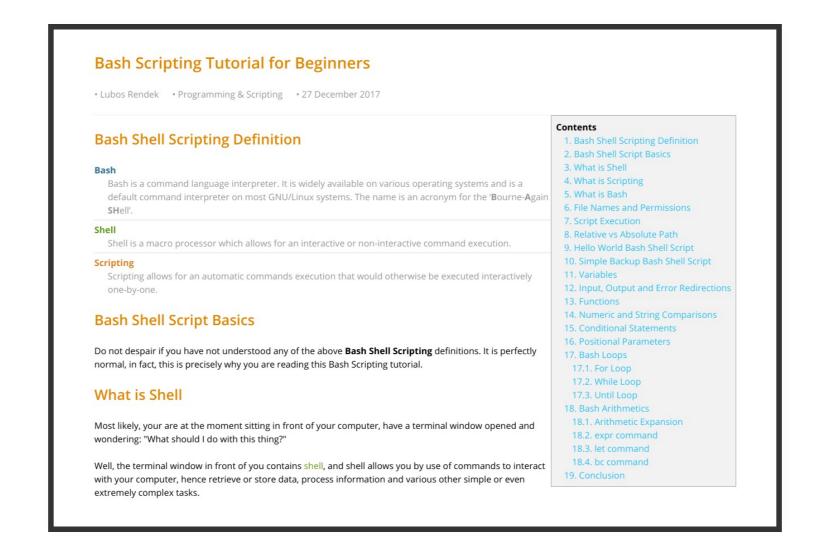
https://github.com/koalaman/shellcheck

3.15 GOOGLE GUIDE

	Shell Style Guide
	Revision 1.26
	Paul Armstrong Too many more to mention
	Each style point has a summary for which additional information is available by toggling the accompanying arrow button that looks this way: ▶ . You may toggle all summaries with the big arrow button:
	▶ Toggle all summaries
Shell Files and Interpreter	File Extensions SUID/SGID
Invocation Environment	STDOUT vs STDERR
Comments	File Header Function Comments Implementation Comments TODO Comments
Farmatting.	Indentation Line Length and Long Strings Pipelines Loops Case statement Variable expansion Quoting
Formatting	Command Substitution Test, [and [Testing Strings Wildcard Expansion of Filenames Eval Pipes to While
Features and Bugs	
	Function Names Variable Names Constants and Environment Variable Names Source Filenames Read-only Variables Use Local Variables Function Location main
Features and Bugs	

https://google.github.io/styleguide/shell.xml

3.16 TUTORIAL



https://linuxconfig.org/bash-scripting-tutorial-for-beginners

4 AWK

4.1 INTRODUCCIÓN

- Es una hoja de cálculo por línea de comandos.
- Tiene su propio penguaje que es muy parecido a C.
- Muy útil para procesar datos dentro de un shell script.
- Muy útil para hacer cosas raras con datos y que con una hoja de cálculo normal es dificil de hacer.
- Muy útil cuando hay muchos datos y una hoja de cálculo se queda colgada.

4.2 EJECUCIÓN

```
awk 'awk_program' data_file
```

awk -f 'awk_file' data_file

4.3 GRADES

Sacar las medias de los alumnos:

```
Pepito    4.4    3.1    5.7
Fulanito    4.2    6.5    8.8
Menganito    5.6    5.0    5.3
awk '{ print $1"="($2+$3+$4)/3 }' 03_grades.csv
```

examples/04_grades.sh

4.4 ROLES

• Agrupar por rol:

```
Pepito:Jefe,Sistemas
Fulanito:Jefe,Desarrollo
```

Menganito:Operario,Sistemas,Desarrollo

Sistemas -> Menganito Pepito
Operario -> Menganito
Jefe -> Fulanito Pepito
Desarrollo -> Menganito Fulanito

4.5 SIN AWK

```
roles_file=./05_roles.csv

roles=$(cut -d : -f 2 $roles_file | sed 's/,/\n/g' | sort | uniq)

for rol in $roles; do
   echo -n "${rol} -> "
   echo $(grep -E "${rol}" "${roles_file}" | cut -d : -f 1)
   done
```

examples/06_roles_sin_awk.sh

4.6 CON AWK

```
# this will run only once at first
BEGIN { FS = ",|:" }
# this will be executed for each of the lines in the file
{
   name=$1
   for (i=2; i<=NF; i++) {
      roles[$i]=roles[$i]" "old_names
   }
}
# This will only run once at the end
END {
   for (rol in roles) {
      print rol" -> " roles[rol]
   }
}
```

examples/08_roles.awk

4.7 TUTORIAL

Grymoire Navigation	<u>Awk</u>
Navigation	
Unix/Linux	
Quotes	
Bourne Shell	
C Shell	
File Permissions	Last modified: Thu Apr 23 16:37:47 EDT 2015
Regular Expressions	
awk worth	Part of the Unix tutorials And then there's My blog
sed UPDATED	
find	Table of Contents
tar	
inodes	Why learn AWK?
Security	Basic Structure
IPv6	Executing an AWK script
Wireless	Which shell to use with AWK?
Hardware	Dynamic Variables
spam	The Essential Syntax of AWK
Deception	Arithmetic Expressions Summary of AWK Commands
PostScript Halftones	Summary or Awc Commands AWK Built-in Variables
Privacy	Associative Arrays
Bill of Rights	Picture Perfect PRINTF Output
References	Flow Control with next and exit
Top 10 reasons to avoid CSH	AWK Numerical Functions
sed Chart PDF	String Functions
awk Reference HTML	User Defined Functions
Magic	AWK patterns
Search	Formatting AWK programs
About	Environment Variables AWK, NAWK, GAWK, or PERL
Donate KW	AWK, NAWK, GAWK, OF PERL
Google+: Bruce Barnett	Copyright 1994,1995 Bruce Barnett and General Electric Company
Twitter: @grymoire	Copyright 2001, 2004, 2013, 2014 Bruce Barnett
Blog: Wordpress Blog	All rights reserved

http://www.grymoire.com/Unix/Awk.html

5 ACERCA DE

5.1 LICENCIA

- Creative Commons Reconocimiento-Compartir Igual 3.0
 - http://creativecommons.org/licenses/by-sa/3.0/es/

5.2 FUENTES

github.com/asanzdiego/commit-conf-charla-shell-script-y-awk

5.3 SLIDES

- Las slides están hechas con MarkdownSlides
 - https://github.com/markdownslides/markdownslides

6 PREGUNTAS

7 GRACIAS