Programación

Bourne Shell

Ramón Manjavacas Ortiz

··· Web: http://www.inf-cr.uclm.es/www/rmanjava ·
··· Email: Ramon.Manjavacas@uclm.es ·

Tabla de Contenidos I

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Tabla de Contenidos II

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

¿qué es un shell?

- Entorno proporcionado para interactuar con el sistema
- Intérprete de lenguaje de comandos que ejecuta comandos leídos del dispositivo de entrada estándar o de un archivo.
- No forma parte del núcleo del sistema pero lo utiliza para ejecutar comandos...
- Tipos: bash(Bourne-Again Shell), csh(C Shell), tcsh, ksk(Korn Shell)
- Shells disponibles: cat /etc/shells
- Shell actual: echo $SHELL

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructura de la línea de órdenes

- **Separador de órdenes**
  - *newline*
  - ;
- **Redirección de la E/S**
  - >, >>, >&2
- **Interconexión de órdenes**
  - |
- **Modificación de la precedencia**
  - (date; who) | wc
- **Ejecución en segundo plano**
  - &

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Metacaracteres I

>	'prog > fichero' dirige la salida estándar hacia *fichero*
>>	'prog >> fichero' agrega la salida estándar a *fichero*
<	'prog < fichero' extrae de *fichero* la entrada estándar
|	'p_1 | p_2' conecta la salida estándar de p_1 a la entrada estándar de p_2
<< str	sigue la entrada estándar hasta el siguiente *str* en el renglón
*	concondar cualquier cadena de cero o más caracteres con nombres de ficheros
?	concondar cualquier carácter individual con nombres de ficheros
[ccc]	concondar cualquier carácter individual de *ccc* con nombres de ficheros; son legales los intervalos como 0-9 o a-z
;	terminador de orden: 'p_1;p_2' hace p_1, y luego p_2
&	igual a ; pero no espera a que termine p_1. Ejecución en *background*
`...`	ejecutar orden(es) en ...; la salida estándar de la orden sustituye a `...`

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Metacaracteres II

(...)	ejecutar orden(es) en ... en un *subshell*
{...}	ejecutar orden(es) en ... el *shell* actual (de poco uso)
$1,$2 etc.	$0...$9 reemplaza por los argumentos en el guión
$var	valor de la variable de *shell var*
${var}	valor de *var*; evita confusión cuando está concatenada con el texto
\	'\c' tomar literalmente el carácter *c*; \newline desechado
...	tomar ... literalmente
"..."	tomar ... literalmente después de interpretar $, `...` y \
#	si una línea comienza por # se considera un comentario
var=valor	asignar a la variable *var* el valor *valor*
p_1 && p_2	ejecutar p_1; si se logra; ejecutar p_2
$p_1 \| p_2$	ejecutar p_1; si no se logra; ejecutar p_2

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Scripts I

- **Serie de órdenes repetidas con frecuencia escritos en texto plano**
  - **Igual a los archivos por lotes de MS-DOS pero más potentes**
  - Contar el número de usuarios conectados: who | wc -l
- **Creación del script**
  - echo 'who | wc -l' > un
- **Ejecución del script**
  - sh < un
  - sh nu
  - un (*con permisos de ejecución*)
- **La primera línea del script especifica el shell que debe ejecutar el archivo de órdenes**
  - #!/bin/sh

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Scripts II

- ¿por qué utilizar "shell scripts"?
  - pueden recuperar, almacenar y mostrar información
  - útiles para crear tus propios comandos
  - ahorro de tiempo
  - automatizan parte de las tareas cotidianas
  - facilitan tareas relacionadas con la Administración de Sistemas

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Scripts III

- **Pasos para escribir un script:**
  - editarlo
  - establecer permiso de ejecución. Ejemplos:

    ```
    $ chmod +x nombre-script
    $ chmod 755 nombre-script
    ```

  - ejecutarlo. Ejemplos:

    ```
    bash nombre-script
    sh nombre-script
    ./nombre-script
    ```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Argumentos y parámetros

- **Parámetros posicionales**
  - $1, $2, $3, $4, $5, $6, $7, $8, $9
  - Todos: $*
  - echo $1
- **El resultado de la ejecución de un programa como argumento**
  - echo “la hora es `date`”

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Variables en el Shell I

- **Definición**
  - nombre → nombre de la variable
  - $nombre → valor de la variable

- **Reglas de nombrado:**
  - deben comenzar con carácter alfanumérico
  - al asignar valor no dejar espacios:

    ✗n = 10 ✗n= 10 ✗n =10 ✓n=10
  - sensible a mayúsculas y minúsculas
  - variables nulas: n= ó n=""

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Variables en el Shell II

- **Tipos**
  - Parámetros posicionales
  - De entorno
    - Globales
    - Los procesos hijo las heredan de su padre
    - Se visualizan con **setenv** ó **env**
    - Ejemplos: PATH, HOME
      - Por convención se escriben en mayúsculas
  - Locales
    - Sólo tienen vigencia en el Shell donde se han definido
    - Se visualizan con **set**

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Variables del Shell III

Variables Intrínsecas de shell	
$#	número de argumentos
$*	todos los argumentos de shell
$@	semejante a la variable anterior
$-	opciones suministradas al shell
$?	Devolver valor de la última orden ejecutada
$$	identificación de proceso del shell
$!	identificación del proceso de la última orden que comenzó con &
$HOME	argumento por defecto de la orden cd
$IFS	lista de caracteres que separa las palabras en los argumentos
$PATH	lista de directorios para buscar programas ejecutables
$PS1	cadena de símbolo de espera, $ por defecto
$PS2	cadena de símbolo de espera para la línea que siguen a las órdenes, > por defecto

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Sentencias del Shell I

- **Comentarios**
  - # texto
- **Expresiones**
  - test *expresión*
  - [expresion]
  - [-x construir -a "$proceder"=cons]
- **Expresiones de cadenas**

$s_1 = s_2$	Verdad si las cadenas s_1 y s_2 son idénticas
s_1 != s_2	Verdad si las cadenas s_1 y s_2 son diferentes
-n s_1	Verdad si laa cadena s_1 tiene al menos un carácter
-z s_1	Verdad si laa cadena s_1 no tiene caracteres

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Sentencias del Shell II

- Expresiones numéricas

n_1 -eq n_2	Verdad si $n_1 = n_2$
n_1 -ne n_2	Verdad si $n_1 \neq n_2$
n_1 -lt n_2	Verdad si $n_1 < n_2$
n_1 -le n_2	Verdad si $n_1 \leq n_2$
n_1 -gt n_2	Verdad si $n_1 > n_2$
n_1 -ge n_2	Verdad si $n_1 \geq n_2$

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Sentencias del Shell III

- Estado de archivos

ESTADO	DESCRIPCIÓN
-s	El fichero existe y su tamaño es mayor que cero
-r	El fichero existe y es legible
-w	El fichero existe y se puede escribir
-x	El fichero existe y es ejecutable
-e	El fichero existe
-o	El fichero existe y eres propietario
-z	El fichero existe y tiene tamaño cero
-f	El fichero existe y es un fichero ordinario
-d	El fichero existe y es un directorio

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Sentencias del Shell IV

- **Combinación de expresiones**

! *expr*	Verdad si *expr* es falsa
$expr_1$ -a $expr_2$	Verdad si $expr_1$ y $expr_2$ son verdaderas
$expr_1$ -o $expr_2$	Verdad si $expr_1$ o $expr_2$ es verdadera
(*expr*)	Verdad si *expr* es verdaera

- **Fin de la ejecución**
  - exit *expresion*
  - El valor resultado de evaluar la *expresión* se almacena en la variable **$?**

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Sentencias del Shell V

- **Expresiones artimétcias**

  expr op1 math-operator op2

- **Ejemplos:**

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo `expr 6 + 3`
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control I

- Alternativas *if-then-else*

```
if [ expresión ]
then
        ordenes1
else
        ordenes2
fi
```

NB: si la condición es verdadera o si “exit status” de la condición es cero.

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control II

- Alternativas *case*

```
case variable in
        patron)     ordenes;;
        patron)     ordenes;;
        ....
esac
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control III

- Tabla de patrones

Reglas de reconocimiento de patrones en shell	
*	reconoce cualquier cadena, incluso la nula
?	reconoce cualquier carácter individual
[ccc]	reconoce cualquier de los caracteres en ccc
"..."	reconoce ... exactamente; funciona igual '...'
\c	reconoce c literalmente
a | b	en la instrucción case reconoce a o b

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control IV

- Ejemplo de alternativas

```
# calendario: mejor interfaz para la orden cal
# sintaxis: calendario mes año

mes=$1; anio=$2 ;;              # 2 args: mes y año

case $mes in
     ene*|Ene*)     mes=1 ;;
     feb*|Feb*)     mes=2 ;;
     mar*|Mar*)     mes=3 ;;
     abr*|Abr*)     mes=4 ;;
     may*|May*)     mes=5 ;;
     jun*|Jun*)     mes=6 ;;
     jul*|Jul*)     mes=7 ;;
     ago*|Ago*)     mes=8 ;;
     sep*|Sep*)     mes=9 ;;
     oct*|Oct*)     mes=10 ;;
     nov*|Nov*)     mes=11 ;;
     dec*|Dec*)     mes=12 ;;
     [1-9]|10|11|12) ;;  # mes numerico
     *)   anio=$mes ; mes="" ;;    # año simple
esac
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control V

- Bucles *for*

for i **in** *lista-de-palabras*
do
    cuerpo-del-bucle
done

for i
do
    cuerpo-del-bucle
done

- Ejemplo:

```
#Sintaxis: mover fich1, fich2, ..., a /tmp
for file
do
   mv $file /tmp/$file.old
done
cd $file
ls *.old
echo "movidos a /tmp"
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control VI

- Bucles *while / until*

while *orden*
do
    cuerpo que se ejecuta a condición de que la orden devuelva verdadero
done

until orden
do
    cuerpo que se ejecuta a condición de que la orden devuelva falso
done

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Estructuras de control VII

- Lectura de las entradas del usuario

```
$ read saludo
Hola, que tal!
$ echo $saludo
Hola, que tal!
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Funciones I

- **Sintaxis**

```
function function-name( )
{
        statement1
        statement2
        statementN
}
```

- **Paso se argumentos**

```
function-name arg1 arg2 arg3 argN
```

- **Variables locales:** `local nombre_variable`

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Funciones II

- **Recuperar argumentos: $0, $1, $2,...**

```
function function-name( )
{
      x=$1
      y=$2
         ...
}
```

- **Retorno de valores: sólo numéricos**

```
function function-name( )
{
      ...
      return $variable_numérica
}

...
x=$?
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo I

```
#!/bin/sh
#
# Script imprime informacion del usuario, quien mantiene
# sesison, fecha y dia actual
clear
echo "Hola $USER"
echo "Hoy es: `date`"
echo "Numero de usuarios con sesion abierta: `who | wc -l`"
echo "Calendario"
cal
exit 0
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo II

```
#
#!/bin/sh
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ]
then
  echo "$1 is negative"
elif [ $1 -eq 0 ]
then
  echo "$1 is zero"
else
  echo "Opps! $1 is not number, give number"
fi
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo III

```
#!/bin/sh

#
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg
if [ -z $1 ]
then
  rental="*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
  rental=$1
fi

case $rental in
   "car") echo "For $rental Rs.20 per k/m";;
   "van") echo "For $rental Rs.10 per k/m";;
   "jeep") echo "For $rental Rs.5 per k/m";;
   "bicycle") echo "For $rental 20 paisa per k/m";;
   *) echo "Sorry, I can not gat a $rental for you";;
esac
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo IV

```
#!/bin/sh

function cal()
{
   n1=$1
   op=$2
   n2=$3
   ans=0
   if [ $# -eq 3 ]; then
       ans=$(( $n1 $op $n2 ))
       echo "$n1 $op $n2 = $ans"
       return $ans
   else
      echo "Function cal requires atleast three args"
   fi
 return
}

 cal 5 + 10
 cal 10 - 2
 cal 10 / 2
 echo $?
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo V

```
#!/bin/sh
# Script to create simple menus and take action according to that selected
# menu item
while :
 do
    clear
    echo "-------------------------------------"
    echo " Main Menu "
    echo "-------------------------------------"
    echo "[1] Show Todays date/time"
    echo "[2] Show files in current directory"
    echo "[3] Show calendar"
    echo "[4] Start editor to write letters"
    echo "[5] Exit/Stop"
    echo "======================"
    echo -n "Enter your menu choice [1-5]: "
    read yourch
    case $yourch in
      1) echo "Today is `date` , press a key. . ." ; read ;;
      2) echo "Files in `pwd`" ; ls -l ; echo "Press a key. . ." ; read ;;
      3) cal ; echo "Press a key. . ." ; read ;;
      4) vi ;;
      5) exit 0 ;;
      *) echo "Opps!!! Please select choice 1,2,3,4, or 5";
         echo "Press a key. . ." ; read ;;
 esac
done
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo VI - I

```
#!/bin/sh

# Declare the array of 5 subscripts to hold 5 numbers
declare nos[5]=(4 -1 2 66 10)

#
# Prints the number befor sorting
#
echo "Original Numbers in array:"
for (( i = 0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done

# Now do the Sorting of numbers
#
for (( i = 0; i <= 4 ; i++ ))
do
   for (( j = $i; j <= 4; j++ ))
   do
      if [ ${nos[$i]} -gt ${nos[$j]}  ]; then
           t=${nos[$i]}
           nos[$i]=${nos[$j]}
           nos[$j]=$t
      fi
   done
done
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo V - II

```
#
# Print the sorted number
#
echo -e "\nSorted Numbers in Ascending Order:"
for (( i=0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo VI

```
#!/bin/sh

echo
echo "Digital Clock for Linux"
echo "To stop this clock use command kill pid, see above for pid"
echo "Press a key to continue. . ."

while :
do
    ti=`date +"%r"`
    echo -e -n "\033[7s"    #save current screen postion & attributes
    #
    # Show the clock
    #

    tput cup 0 69           # row 0 and column 69 is used to show clock

    echo -n $ti             # put clock on screen

    echo -e -n "\033[8u"    #restore current screen postion & attributs
    #
    #Delay fro 1 second
    #
    sleep 1
done
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Ejemplo VII

```
#! /bin/sh

# aminusc - reemplaza archivos con su nombre en minusculas

for i in $*; do
   tmp=`echo $i | tr "[A-Z]" "[a-z]"`
   if [ -f $i -a "$i" != "$tmp" ]; then
          echo -n "¿Cambiar $i a ${tmp}? [s/n] "
          read resp
          if [ "$resp" != "s" -a "$resp" != "S" ]; then
                 echo no se cambiara $i
          else
                 if [ -f "$tmp" ]; then
                        echo -n "$tmp ya existe. ¿Reemplazar? [s/n] "
                        read resp
                        if [ "$resp" = "s" -o "$resp" = "S" ]; then
                               mv -f $i $tmp
                               if [$? -ne 0 ]; then
                                  echo error en mv, $i queda sin cambios
                               fi
                        fi
                 fi
          fi
   fi
done
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Utilidad dialog

- **Utilidad para la generación de cajas de información, menús, mensajes, de entrada de texto, etc...**

- **Sintaxis:**

```
dialog --title {title} --backtitle {backtitle} {Box options}
where Box options can be any one of following
--yesno {text} {height} {width}
--msgbox {text} {height} {width}
--infobox {text} {height} {width}
--inputbox {text} {height} {width} [{init}]
--textbox {file} {height} {width}
--menu {text} {height} {width} {menu} {height} {tag1 item1}...
```

- **Retorna: 0 – ok ; 1-no; 255 -esc**

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Utilidad dialog- Ejemplo I

```
#!/bin/sh
dialog --title "Ejemplo mensaje" \
--backtitle "Administracion SSOO" \
--msgbox "Este mensaje es un ejemplo de utilizacion. Pulse tecla" 9 50
```

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Utilidad dialog- Ejemplo II

```
#!/bin/sh
dialog --title "Ejemplo de caja de texto de entrada" \
--backtitle "Administracion SSOO" \
--inputbox "Introduzca login: " 9 50 2>/tmp/text.$$
```

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Utilidad dialog- Ejemplo III

```
#!/bin/sh
dialog --backtitle "Administracion SSOO "    \
--title "Ejemplo de menu" \
--menu "Seleccionar con [UP] [DOWN],[Enter] " 15 50 3 \
Fecha "Mostrar la fecha y hora" \
Calendario "Motrar calendario" \
Usuario "Mostrar info usuario" 2>/tmp/menuitem.$$

menuitem=`cat /tmp/menuitem.$$`

case $menuitem in
Fecha) date;;
Calendario) cal;;
Usuario) echo "Usuario: $USER";;
esac
```


Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)

Programación Bourne Shell

Utilidad dialog- Ejemplo III

Ramón Manjavacas Ortiz (Ramon.Manjavacas@uclm.es)