

Comandos de shell y programación en csh

Diego Chaparro (dchaparro@acm.org)
22 de Mayo de 2006

Preámbulo

Este documento no pretende ser una guía completa de programación de shell, sino que solamente trata de hacer una introducción y una visión global de ésta, centrada en la shell csh.

Este documento no está libre de errores y está en continuo desarrollo.

Licencia

Este documento se distribuye bajo la licencia Creative Commons Attribution-ShareAlike. Para obtener la licencia completa véase:

<http://creativecommons.org/licenses/by-sa/2.1/es/>

Índice de contenidos

| | |
|---|----|
| 1. Unix Shells..... | 3 |
| 2. Comandos Básicos..... | 3 |
| 3. Editor vi..... | 11 |
| 4. Introducción a la programación de shell..... | 13 |
| 5. Variables..... | 14 |
| 6. Condicionales..... | 15 |
| 7. Bucles..... | 17 |
| 8. Entrada/salida..... | 18 |
| 9. Aliases y argumentos..... | 19 |
| 10. Expresiones regulares y grep..... | 20 |
| 11. Más comandos..... | 21 |
| 12. Procesos..... | 29 |
| 13. Historia..... | 31 |
| 14. Completado de sintaxis..... | 31 |
| 15. Sed y awk..... | 32 |
| 16. Entorno de usuario..... | 33 |
| Referencias..... | 34 |

1. Unix Shells

Una shell es un programa que lee comandos y los ejecuta: un intérprete de comandos.

Hay muchas shells, pero las más conocidas son: sh, bash, ksh, csh y tcsh.

Cada una de estas shells tienen algunas diferencias y similitudes con las otras con respecto a aspectos como:

- Historial de comandos
- Edición de la línea de comando
- Completado de sintaxis
- Entrada/salida
- ...

2. Comandos Básicos

2.1 Gestión de ficheros y directorios

ls

Descripción

Muestra el contenido del directorio que le especifiquemos como argumento. Si no especificamos el nombre del directorio, nos muestra el contenido del directorio actual.

Sintaxis

ls [argumentos] [directorio]

Argumentos

- a Muestra ficheros/directorios ocultos
- l Listado detallado
- C En columnas, ordenado alfabética y verticalmente
- R Listado recursivo

cd

Descripción

Cambia el directorio actual al que le especifiquemos como argumento. El directorio especificado puede ser un PATH relativo o absoluto.

Sintaxis

cd [directorio]

pwd

Descripción

Muestra el nombre del directorio actual. Esta información está también almacenada en la variable de entorno \$PWD

Sintaxis

pwd

mkdir

Descripción

Crea el/los directorio/s que le especifiquemos

Sintaxis

mkdir [argumentos] directorio...

Argumentos

-p Crea los directorios padre que falten para cada directorio a crear

rm

Descripción

Borra el/los ficheros o directorios especificados. Por lo normal, no borra directorios, para eso se debe utilizar el argumento -r o -R.

Sintaxis

rm [argumentos] fichero...

Argumentos

| | |
|---------|--|
| -i | Pide confirmación para borrar cada fichero o directorio |
| -f | Fuerza el borrado y no pregunta si el fichero o directorio no es modificable |
| -r o -R | Borra recursivamente un directorio |

cp

Descripción

Copia ficheros y opcionalmente directorios. Se puede copiar de varias formas: 1) copiar un fichero a un fichero de destino o 2) copiar uno o varios ficheros a un directorio de destino. En el primer caso el destino debe ser un fichero, y en el segundo debe ser un directorio.

Sintaxis

cp [argumentos] fichero fichero
cp [argumentos] fichero... directorio

Argumentos

| | |
|---------|--|
| -i | Pregunta si sobrescribir ficheros destino existentes |
| -r o -R | Copia directorios recursivamente |

mv

Descripción

Mueve o renombra ficheros o directorios. Si el último argumento es un directorio existente, mueve cada uno de los otros ficheros/directorios a ese directorio con el mismo nombre. Si no existe el destino, renombra el primero al segundo.

Sintaxis

```
mv [argumentos] origen destino  
mv [argumentos] origen... destino
```

Argumentos

-f No pide confirmación si existe el destino a sobrescribir
-i Pide confirmación cuando el destino existe y se sobrescribiría

which

Descripción

Devuelve la localización de los comandos que pueden ser ejecutados por el usuario, buscando éstos en los directorios definidos por la variable de entorno PATH

Sintaxis

```
which [argumentos] comando...
```

Argumentos

-a Muestra todas las coincidencias en la búsqueda

touch

Descripción

Cambia la fecha de acceso y/o modificación del archivo especificado. Las fechas a cambiar son modificadas a la fecha actual. Si el fichero no existe lo crea, que es otro uso que se le suele dar a este comando, crear ficheros vacíos.

Sintaxis

```
touch [argumentos] archivo...
```

Argumentos

-a Cambia la fecha de acceso del archivo

file

Descripción

Comprueba cada argumento e intenta averiguar el tipo de fichero de cada uno de ellos

Sintaxis

file [argumentos] fichero...

Argumentos

-z Intenta mirar dentro de ficheros comprimidos

2.2 Tratamiento de ficheros de texto

Ahora veremos algunos comandos que nos permiten realizar operaciones sobre ficheros de texto.

cat

Descripción

Concatena ficheros y escribe el resultado en la salida estándar. Un uso habitual suele ser especificar un solo fichero y se usa para ver su contenido (no concatenamos nada).

Sintaxis

cat [argumentos] [fichero...]

Argumentos

-n Numera todas las líneas de salida

more

Descripción

Muestra un fichero de texto de forma paginada, mostrando una pantalla cada vez. Solo permite ver el fichero hacia delante, no permite volver hacia atrás.

Sintaxis

more [argumentos] [fichero...]

Teclas utilizadas

| | |
|---------|------------------|
| ESPACIO | Página siguiente |
| ENTER | Línea siguiente |
| q | Salir |

less

Descripción

Muestra un fichero de texto de forma paginada, mostrando una pantalla cada vez. Pero proporciona mayor funcionalidad que el comando more, porque permite movimiento hacia delante y hacia atrás en el fichero, y permite otra serie de movimientos muy útiles

Sintaxis

less [argumentos] [fichero]

Teclas utilizadas

| | |
|---------------|----------------------------------|
| ESPACIO | Avanza hacia la página siguiente |
| ENTER | Avanza hacia la línea siguiente |
| CURSOR ARRIBA | Retrocede una línea hacia atrás |
| CURSOR ABAJO | Avanza una línea hacia delante |
| g | Va a la línea 1 |
| G | Va a la última línea |

WC

Descripción

Filtra un fichero de texto y cuenta los caracteres, palabras y líneas que contiene el mismo. El resultado lo escribe en la pantalla.

Sintaxis

wc [argumentos] fichero

Argumentos

| | |
|----|--------------------------------|
| -c | Cuenta el número de bytes |
| -m | Cuenta el número de caracteres |
| -l | Cuenta el número de líneas |
| -w | Cuenta el número de palabras |

head

Descripción

Muestra las n primeras líneas de un fichero de texto. El valor de n por defecto es 10

Sintaxis

head [argumentos] [fichero...]

Argumentos

-n Muestra las n primeras líneas.

tail

Descripción

Muestra las n últimas líneas de un fichero de texto. El valor de n por defecto es 10

Sintaxis

tail [argumentos] fichero...

Argumentos

-n Muestra las n últimas líneas.

2.3 Otros comandos útiles

echo

Descripción

Muestra la línea de texto especificada como argumento. También se le puede pasar una variable de entorno como argumento y mostraría su valor.

Sintaxis

echo [argumentos] [texto...]
echo [argumentos] [variable...]

Argumentos

-n No escribe un final de línea al final

3. Editor vi

3.1 Modos

- Modo comando:
 - Permite ejecutar comandos como borrar, moverse, salir, ...
 - Por defecto se entra en modo comando. Para volver a él se pulsa la tecla ESC
- Modo edición:
 - Permite insertar contenido en un documento
 - Para entrar en modo inserción: A, a, I, i, o, O, s, r, R

3.2 Comandos de movimiento

| | |
|---------------|--|
| h | Moverse a la izquierda |
| l | Moverse a la derecha |
| j | Moverse hacia abajo |
| k | Moverse hacia arriba |
| w | Mueve el cursor una palabra hacia delante |
| CTRL+F | Mueve el cursor una pantalla hacia delante |
| CTRL+B | Mueve el cursor una pantalla hacia atrás |
| nG | Mueve el cursor a la línea <i>n</i> |
| G | Mueve el cursor al final de fichero |

3.3 Comandos para pasar a modo inserción

| | |
|----------|---|
| I | Insertar caracteres al comienzo de línea |
| i | Insertar caracteres inmediatamente antes del cursor |
| a | Añade caracteres inmediatamente después del cursor |
| A | Añade caracteres al final de la línea |
| O | Abre una línea antes de la que está el cursor y se queda en modo inserción |
| o | Abre una línea debajo de la que está el cursor y se queda en modo inserción |

3.4 Comandos de borrado

| | |
|-----------|---|
| x | Borra el carácter sobre el que está el cursor |
| X | Borra el carácter situado antes del cursor |
| dd | Borra la línea sobre la que está el cursor |
| dw | Borra la palabra situada pbajo el cursor |
| J | Junta las líneas actual y siguiente |

3.5 Comandos para deshacer cambios

| | |
|------------|---|
| u | Deshace la última modificación |
| uu | Deshace lo que hizo la primera <i>u</i> . Es equivalente a no hacer nada |
| uuu | Es igual a una sola <i>u</i> Algunos clónicos han variado el comportameiento del <i>u</i> para deshacer los últimos <i>n</i> cambios |

3.6 Comandos de búsqueda

| | |
|----------------|--|
| /cadena | Busca la cadena |
| n | Vuelve a buscar la cadena |
| N | Vuelve a buscar la cadena en sentido inverso de búsqueda |

3.7 Comandos para salir

| | |
|-------------------|---|
| :q | Salte sin guardar el resultado (si no hemos modificado) |
| :wq | Salte y guarda el fichero |
| ZZ | Salte y guarda el fichero |
| :q! | Salte y no guarda aunque hayamos modificado |
| :n fichero | Abre ese fichero |

4. Introducción a la programación de shell

- Un shell script es un conjunto de comandos de shell que se ejecutan de forma secuencial, uno detrás de otro, y que se almacenan en un fichero.
- Todos los shell scripts empiezan con una línea que tiene los siguientes caracteres:
`#!`
- Y después se especifica el PATH del intérprete de comandos que se desea usar en el script. Ej:
`#!/bin/bash`

4.1 Comentarios

Los comentarios se especifican poniendo un carácter almoadilla delante del comentario. Ej:

```
# Esto es un comentario
```

4.2 Permiso de ejecución

Un shell script se guarda en un fichero y para poder ejecutarlo hay que darle permiso de ejecución:

```
chmod +x      # Permiso de ejecución para todos
chmod u+x    # Permiso de ejecución para el dueño
chmod g+x    # Permiso de ejecución para el grupo
chmod o+x    # Permiso de ejecución para el resto
```

4.3 Ejecución

Un shell script se ejecuta especificando su PATH absoluto o relativo:

```
$ /home/prueba/script.sh
$ ./script.sh
```

4.4 Primer script

El ejemplo más sencillo de shell script:

```
#!/bin/csh
echo "Hola mundo"
```

5. Variables

Hay dos tipos de variables:

5.1 Variables de entorno

Son variables inicializadas cuando se inicia el intérprete de comandos y son heredadas por los programas que se abran desde el mismo.

Para mostrar las variables de entorno existentes:

```
$ setenv
```

Para modificar el valor de una variable o añadir una nueva:

```
$ setenv VARIABLE VALOR
```

Variables de entorno importantes:

PATH Directorios donde se buscaran los binarios

HOME Directorio que pertenece al usuario

SHELL Intérprete de comandos en uso

USER Nombre del usuario

PWD Directorio actual

5.2 Variables locales

Variables que son usadas solamente por el intérprete de comandos y no se pasan a ningún otro programa.

Para mostrar las variables que hay definidas:

```
$ set
```

Para definir una variable o cambiarle el valor a una existente:

```
$ set VAR = VALOR
```

6. Condicionales

6.1 if

```

if ( EXPRESION ) ...
ó
if ( EXPRESION ) then
  ...
else
  ...
endif

```

6.2 switch

```

switch (VARIABLE)
  case VALOR:
    ...
    breaksw
  case VALOR:
    ...
    breaksw
  default:
    ...
    breaksw
endsw

```

6.3 Expresiones

Las expresiones están formadas por operadores:

- Numéricos:

+ - * / %

- Comparacion:

== != =~ !~ <= >= < >

- Condicionales:

// && !

- Para ficheros:

r Permiso de lectura
w Permiso de escritura
x Permiso de ejecución
e Existencia del fichero
o Propietario del fichero
z Fichero vacío
f Fichero ordinario
d Directorio

- Para asignar valores resultantes de una operación numérica a una variable se utiliza el operador:

@

Ejemplos:

\$a > \$b

5 + 3

-e /tmp/prueba.txt

@ a = 2 + 5

7. Bucles

7.1 foreach

```
foreach VAR ( VALORES )  
end ...
```

VALORES: cadenas separadas por un espacio en blanco
VAR: cada iteración del bucle se asigna un valor de la lista de *VALORES*

7.2 while

```
while ( EXPRESION )  
end ...
```

7.3 repeat

```
repeat NUMERO COMANDO
```

Ejecuta *NUMERO* veces el comando especificado

8. Entrada/salida

| | |
|------------------------------|---|
| <i>< fichero</i> | Utiliza el <i>fichero</i> como entrada estándar |
| <i>> fichero</i> | Utiliza el <i>fichero</i> como salida estándar. Sobreescribe el fichero |
| <i>>& fichero</i> | Utiliza el fichero como salida estándar y como salida de errores. Sobreescribe el fichero |
| <i>>> fichero</i> | Utiliza el <i>fichero</i> como salida estándar. Añade al final del fichero |
| <i>>>& fichero</i> | Utiliza el fichero como salida estándar y como salida de errores. Sobreescribe el fichero |
| <i>com1 com2</i> | Envía la salida estándar del comando <i>com1</i> a la entrada estándar del comando <i>com2</i> |
| <i> &</i> | Envía la salida estándar y la salida de errores del comando <i>com1</i> a la entrada estándar del comando <i>com2</i> |

9. Aliases y argumentos

9.1 Aliases

Un alias es un comando de shell definido por el usuario, en csh es la forma de definir funciones. Para mostrar el listado de alias definidos:

```
$ alias
```

Para definir un nuevo alias:

```
$ alias ll "ls -l"
```

Para borrar un alias existente:

```
$ unalias ll
```

9.2 Argumentos

Los argumentos de un alias o una shell se identifican mediante las siguientes variables:

| | |
|------|--------------------------------|
| argv | Lista con todos los parámetros |
| \$0 | Comando ejecutados |
| \$1 | Primer parámetro |
| \$2 | Segundo parámetro |

10. Expresiones regulares y grep

Una expresión regular es una notación que permite la búsqueda de texto que coincide con un determinado patrón. Por ejemplo: el texto que empieza por "a".

Las expresiones regulares están formadas por:

- Caracteres ordinarios
- Caracteres especiales. Algunos de ellos son:

| | |
|----------|--|
| * | Cualquier carácter: 0 o más |
| . | Cualquier carácter excepto el nulo |
| [ab] | Carácter <i>a</i> o <i>b</i> |
| [a-f] | Cualquier carácter entre la <i>a</i> y la <i>f</i> |
| [^ab] | Cualquier carácter excepto el especificado |
| ^patron | Línea que comienza con el <i>patron</i> |
| patron\$ | Reconoce el <i>patron</i> al final de la línea |
| + | Reconoce una o más apariciones del patrón anterior |
| ? | Reconoce cero o una aparición del patrón anterior |

grep es un programa que utiliza expresiones regulares para seleccionar las líneas de texto que encajan en el patrón definido.

Sintaxis:

```
grep [argumentos] patron [ficheros]
```

Argumentos:

| | |
|----|--|
| -i | Ignora diferencias entre mayúsculas y minúsculas |
| -w | Busca coincidencias de la palabra entera |

Ejemplos:

```
grep hola fichero.txt
```

```
grep '^Hola' fichero.txt
```

```
grep '[Hh]ola' fichero.txt
```

```
grep 'adios$' fichero.txt
```

11. Más comandos

split

Descripción

Parte un fichero en trozos de 1000 líneas por defecto. Cada trozo lo numero como [prefijo].aa, [prefijo].ab, ...

Sintaxis

split [opciones] [fichero [prefijo]]

Argumentos

- n Lo separa en trozos de tamaño *n*
- Especifica que la entrada sea la entrada estándar

Ejemplos:

```
split fichero
split -200 fichero
split -200 fichero trozo.
```

cut

Descripción

Seleccionar columnas de las líneas de un fichero

Sintaxis

cut [opcion] fichero

Argumentos

- c Seleccionamos las columnas
- f Seleccionamos los campos separados por el delimitador TAB por defecto
- d Especificamos el delimitador
- s No aparece la línea en la que no hay delimitador, por defecto si saldría

Ejemplos

```
cut -c1-3,7
cut -d: -f1,2
```

paste

Descripción

Concatena las líneas de varios ficheros de entrada

Sintaxis

paste [opcion] [ficheros]

Argumentos

-d Especificamos el delimitador que queremos que nos ponga entre las líneas de cada fichero. Por defecto es un tabulador

Ejemplos

```
paste f1 f2 f3
paste -d: f1 f2 f3
```

join

Descripción

Enlaza líneas de dos ficheros a partir de un campo clave. Join supone que los ficheros están ordenados por el campo clave

Sintaxis

join [opcion] fichero1 fichero2

Argumentos

-j1 num El campo *num* es el campo clave del primer fichero
 -o num1.num2 La salida sea el campo *num2* del fichero *num1*
 -t delim Delimitador entre los campos

Ejemplo

Tenemos dos ficheros:

CURSOS

```
1,shell,22/05
2,c++,29/05
3,solaris,19/05
4,linux,12/05
```

ALUMNOS

```
antonio,1
juan,1
pepe,1
maria,1
maria,2
```

```
$join -j1 1 -j2 2 -t, cursos alumnos
$join -j1 1 -j2 2 -t, -o 2.1,1.2,1.3 cursos alumnos
```

tr

Descripción

Cambia caracteres, uno por otro de la entrada estándar, no se le puede especificar un fichero

Sintaxis

tr [opcion] conjunto1 [conjunto2]

Especificamos en el conjunto1 los caracteres que queremos cambiar.

Y en el conjunto 2 los caracteres por los que los queremos cambiar.

Se pueden especificar rangos: a-z

Argumentos

-c Cambia lo que no esté en conjunto1

-s Elimina las repeticiones

-d Borra los caracteres especificados, no los cambia

tee

Descripción

Lee de la entrada estándar y lo escribe en un fichero y en la salida estándar. Nos vale para crear dos copias, normalmente una para guardarla y otra para procesarla en ese momento

Sintaxis

tee [opcion] fichero

Argumentos

-a Añade al fichero, en lugar de sobrescribirlo

sort

Descripción

Ordena las líneas tomando como base la ordenación de los caracteres ASCII

Sintaxis

sort [option] [fichero]

Argumentos

-k num Ordena a partir del campo número *num*
-t delim Especifica el delimitador de campos, por defecto es tabulador o espacios
-r Ordenación inversa
-n Ordenación numérica

Ejemplos

sort -k 1 fichero
sort -t: -k 2 fichero

uniq

Descripción

Elimina las líneas repetidas consecutivas

Sintaxis

uniq [opcion] [ficheroentrada] [ficherosalida]

Argumentos

-c Muestra el número de veces que se repiten las líneas
-d Muestra solo las líneas que se repiten

comm

Descripción

Compara dos conjuntos. Deben ser dos ficheros ordenados y sin duplicados

Sintaxis

comm [opcion] fichero1 fichero2

Cada línea es un elemento del conjunto

Muestra el resultado en tres columnas:

Los elementos que están en el conjunto 1 y no en el 2

Los elementos que están en el conjunto 2 y no en el 1

Los elementos que están en los dos

Argumentos

Podemos especificar las columnas que NO queremos ver con la opción -col1col2

Ejemplo

comm -12 fichero1 fichero2

cmp

Descripción

Compara cualquier tipo de ficheros.

Sintaxis

cmp [opcion] fichero1 [fichero2 [skip1 [skip2]]]

Argumentos

Hace la comparación byte a byte. Su principal objetivo es saber si son iguales o no.

Si son iguales no saca ningún resultado. Si no lo son señala la posición del primer byte en el que difieren

diff

Descripción

Compara ficheros cuyo contenido son textos
Solo es de utilidad cuando los ficheros a comparar tienen una mayoría de líneas idénticas. Muestra las diferencias entre esos dos ficheros especificando las líneas de cada uno de los ficheros el tipo de diferencia (c: cambio, a: solo presente en el segundo fichero, d: solo presente en el primero). El contenido de cada fichero (< son los datos del primer fichero y > del segundo)
También tenemos **diff3**, para comparar tres ficheros

Sintaxis

diff [opcion] ficheros

mail

Descripción

Programa para enviar correo. También sirve para leer el correo.

Sintaxis

mail [argumentos] DIRECCION

Argumentos

-s SUBJECT Especifica el título del mensaje
-c DIRECCION Especifica dirección a la que mandar copia
-b DIRECCION Especifica dirección a la que mandar copia oculta

Ejemplo

```
$ mail -s "Esto es un correo de prueba" pepe@pepe.com  
$ mail -s "prueba2" -b jefe@pepe.com pepe@pepe.com
```

Cuando acabamos de escribir el cuerpo del mensaje para finalizar se escribe:

CTRL+D
ó un punto en una línea que solo contenga ese carácter.

find

Descripción

Busca ficheros en una jerarquía de directorios

Sintaxis

find [path] [expresion]

Expresion

-name *X* Busca ficheros que tengan el nombre *X*
-type *X* Busca ficheros del tipo *X* (b, c, d, f o l)

exit

Descripción

Termina la ejecución de la shell activa

Sintaxis

exit [argumentos]

Argumentos

(*X*) Devuelve el código de estado *X*

Ejemplos

exit (1)

tar

Descripción

Se usa para archivar varios ficheros en uno solo, manteniendo la estructura de directorios. Se deben usar PATH relativos, nunca absolutos porque habría problemas al desempaquetar

Sintaxis

tar [opcion] [fichero_tar] [ficheros]

Argumentos

- c Crea un nuevo archivo
- f Especifica el nombre del archivo tar
- t Muestra los archivos contenidos en un tar
- x Extrae el contenido de un tar
- v (Verbose) Muestra lo que va haciendo
- w Modo interactivo. Pregunta cada archivo a extraer
- r Añade un nuevo archivo al un tar existente
- u Actualiza o añade un fichero a un tar existente

gzip

Descripción

Comprime un fichero
Se borra el archivo original

Sintaxis

gzip [opcion] [fichero]

Argumentos

- v Muestra información sobre el fichero comprimido, ratio de compresión, etc...

Ejemplo

```
$ gzip fichero
```

Nos devolvería un fichero de nombre fichero.gz y borraría el original
Podemos empaquetar y comprimir a la vez:
tar -cvzf archivo.tar.gz ó
tar -cf archivo.tar . | gzip archivo.tar

12. Procesos

Un proceso es un programa en ejecución y la shell nos permite ejecutar procesos.

Cada proceso tiene un identificador único llamado identificador de proceso (PID). Los PID comienzan en el número 1 y se van incrementando

top

Muestra un listado de los procesos en ejecución, además de información sobre uso de memoria y swap.

Lo muestra en forma interactiva, permitiendo ordenar por uso de recursos. Comandos:

| | |
|---|-------------------------------------|
| M | Ordenar procesos por uso de memoria |
| P | Ordenar procesos por uso de CPU |
| h | Ayuda |

ps

Muestra un listado de los procesos en ejecución

Argumentos:

| | |
|------|---|
| -aux | Muestra todos los procesos incluyendo el nombre de usuario y la fecha de inicio |
|------|---|

&

Ejecuta un proceso en segundo plano.

jobs

La shell asocia los procesos abiertos desde ella, manteniendo un listado de ellos y asociándoles un número de job. Se puede mostrar esta información con el comando:

\$ jobs

CTRL+z

Suspende el proceso que se está ejecutando en primer plano.

CTRL+c

Termina la ejecución de un proceso que se ejecuta en primer plano.

fg

Pone un job en primer plano. Hay que especificar el número de job.

Ejemplo:

```
$ fg %2
```

bg

Pone un job en segundo plano. Hay que especificar el número de job.

Ejemplo:

```
$ bg %2
```

13. Historia

La shell guarda una historia de todos los comandos que se han introducido, y tenemos varias formas de acceder a esa información.

Lo primero de todo, para activar la funcionalidad de historial en la shell hay que definir una variable llamada *history* y asignarle el número de comandos que queremos que nos guarde. Ej:

```
$ set history = 100
```

Para ver la historia de comandos que hemos introducido:

```
$ history
```

Para recuperar los últimos comandos introducidos se puede hacer de las siguientes formas:

| | |
|---------------|-------------------------------|
| <i>\$!!</i> | Recupera el último comando |
| <i>\$!77</i> | Recupera el comando 77 |
| <i>\$!-2</i> | Recupera el penúltimo comando |

14. Completado de sintaxis

El completado de sintaxis en csh ayuda a completar el nombre de ficheros o directorios usando la tecla Escape.

Para activar el completado de sintaxis hay que definir la variable local *filec*:

```
$ set filec = 1
```

Para usarlo, cuando se está escribiendo el nombre de un fichero o directorio, se pulsa la tecla escape y la shell completa el nombre del fichero si es la única posibilidad desde los caracteres que hayamos escrito.

15. Sed y awk

15.1 sed

sed es un editor de texto no interactivo. Se utiliza para realizar modificaciones repetitivas sobre uno o varios ficheros.

Primero hay que definir qué operaciones queremos realizar sobre el fichero, y sed va procesando el fichero de entrada línea a línea.

sed no cambia el fichero original, lo copia, lo procesa y el resultado lo muestra en la salida estándar.

Sintaxis:

```
sed [-e commands] [-f scriptfile] fichero [>salida]
```

Argumentos:

- e Indica que los comandos están a continuación, en la línea de comandos
- f Indica que los comandos están en el fichero indicado

Sintaxis de los comandos:

```
[ direccion [, direccion ] [!]] funcion [argumentos]
```

Si no ponemos dirección, la función se aplica a todas las líneas

La dirección puede ser un número de línea o una expresión

El carácter ! indica todo menos lo seleccionado con las direcciones

Funciones:

| | |
|---|------------|
| p | imprimir |
| d | borrar |
| s | substituir |
| r | leer |

Ejemplos

```
sed -e '3,5p'
```

```
sed -e '4,6d' -e '10,$d'
```

```
sed -e 's/lunes/dia/'
```

Muestra las líneas de la 3 a la 5

Borra las líneas 4-6 y 10 en adelante

15.2 awk

awk es un lenguaje de programación diseñado para procesar ficheros de texto y seleccionar información de ellos.

Es útil, por ejemplo para seleccionar campos de un fichero que use un delimitador:

```
awk '{ print ($5)}'  
awk -F : '{ print ($5)}'
```

16. Entorno de usuario

Hay una serie de ficheros en cada directorio de usuario que se utilizan para definir variables de entorno, alias, ...

Estos ficheros son:

| | |
|-----------|---|
| ~/.cshrc | Ejecutado cuando se inicia la shell |
| ~/.login | Después de .cshrc si la shell utiliza login |
| ~/.logout | Ejecutado al hacer logout |

Referencias

[1] UNIX shell differences and how to change your shell:

<http://www.faqs.org/faqs/unix-faq/shell/shell-differences/>

[2] Csh Programming Considered Harmful

<http://www.faqs.org/faqs/unix-faq/shell/csh-whynot/>

[3] Using csh & tcsh

<http://www.kitebird.com/csh-tcsh-book/>

[4] Unix shells

<http://cbbrowne.com/info/unixshells.html>