Introducción a los scripts de administración de Apple

O cómo perderles el miedo

Si es un administrador de Apple y tiene pánico a la creación de scripts, hoy es su día de suerte: esta guía es justo lo que necesita.

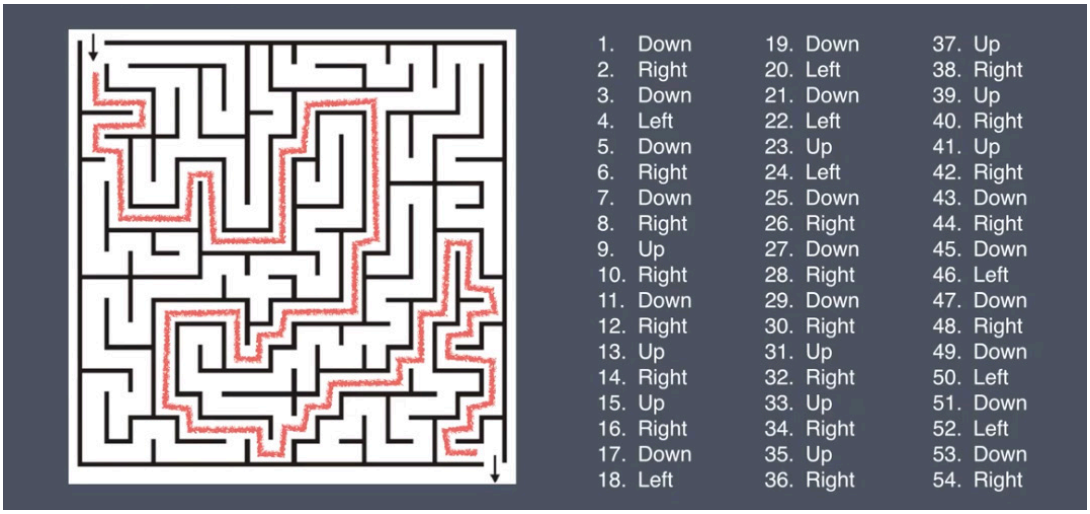
Con unos pocos scripts básicos conseguirá ganar velocidad y precisión y también mejorar la satisfacción de los usuarios finales.

> Si en algún momento ya había emprendido su viaje con los scripts y había terminado en la deriva, seguramente fue por algo: muchos tutoriales saltan rápidamente a una lista de scripts para memorizar, más o menos incomprensibles, sin explicar siquiera en qué consiste la creación de scripts y por qué merece la pena dominar este arte.
>
> **La buena noticia es que iniciarse en este camino puede ser fácil, siempre que entienda cómo funcionan los scripts y qué hacen antes de ponerse a memorizar cadenas de comandos.**

¿Para qué necesitamos scripts?

Los scripts nos ayudan a simplificar.

Algo tan complejo como moverse por un laberinto puede reducirse a un conjunto de instrucciones básicas. Solo hacen falta cuatro comandos: arriba, abajo, izquierda y derecha. Se trata de dividir el problema en porciones más pequeñas, como en este ejemplo:



Podemos resolver un desafío aparentemente complejo con solo cuatro comandos.

Los scripts nos permiten automatizar.

Para abrir un archivo en su equipo solo hay que hacer doble clic. Para abrir un archivo en su equipo y en otro hay que hacer dos dobles clics. ¿Y para abrir 24 archivos en 24 equipos a la vez?

Con una solución de MDM como Jamf Pro podemos automatizar esta orden en todos los equipos a través de un simple comando de una línea (open) combinado con la ubicación del archivo.

Los scripts minimizan los errores.

Cuando una tarea depende de una acción humana, también está expuesta al error humano. Y esto puede traducirse en más correcciones y en un aumento de las consultas al servicio de asistencia. Convirtiendo en scripts las tareas rutinarias nos aseguramos de que se ejecutan correctamente (y rápidamente) una y otra vez.

¿Cómo funciona Terminal?

Los intérpretes

¿No sabe qué son? En realidad, seguro que sí lo sabe:

Lleva años utilizándolos. Los intérpretes más habituales son el Explorador de Windows y el Finder del Mac. Su trabajo es interpretar sus clics en el ratón y los comandos del teclado y convertirlos en acciones en el equipo. Al hacer clic en un archivo, tanto el Explorador de Windows como el Finder del Mac interpretan este clic como una orden para seleccionar el archivo. Con un doble clic entienden que la orden es abrir el archivo.

Sin embargo, no todos los intérpretes interpretarán sus acciones de la misma forma. Por ejemplo, si selecciona un archivo en el Explorador y pulsa Intro, se abre el archivo. Si hace lo mismo en un Mac, puede modificar el nombre del archivo.

Terminal es donde los usuarios de Apple ejecutan sus scripts. Familiarizarse con el funcionamiento de Terminal es una buena forma de perder el miedo a los scripts.

Terminal no es exactamente un intérprete, sino una ventana que abre un intérprete llamado bash. En el Mac conviven varios intérpretes, pero este e-book se centra únicamente en bash.

Al abrir Terminal desde la carpeta Utilidades de Aplicaciones, conecta con el intérprete bash. Bash es la forma abreviada de Bourne-AgainSHell. Se trata de una versión revisada del shell intérprete creado por Steven Bourne 40 años atrás.

¿Cómo funcionan los comandos?

Resumido en pocas palabras, los comandos dicen al ordenador qué debe hacer. Permiten hacer todo lo que haría con el Finder. Por ejemplo, puede abrir un archivo haciendo doble clic encima. O puede abrir Terminal y usar el siguiente comando, sustituyendo el nombre del documento por cualquier cosa que tenga en su escritorio:

```
open /Users/jamfwebinar/Desktop/scripting101.docx
```

¿Y qué pasa si tenemos una página web en lugar de un archivo? Con el comando open también es posible abrirla. Solo hay que indicarle la URL y con qué navegador queremos que se abra:

```
open https://www.jamf.com –a Safari
```

Y ahora la parte interactiva: si quiere sacar el máximo provecho a este e-book, lo mejor es abrir Terminal e ir siguiendo la secuencia de pasos.

Aplicaciones > Utilidades > Terminal

Empecemos por el principio.

El primer programa que la mayoría de los usuarios aprenden en Terminal es **'Hello, World'**.

Parece un ejercicio sencillo, pero deberá utilizar el comando echo de muchas formas diferentes a medida que vaya familiarizándose con el mundo de los scripts, por lo que es importante empezar con buen pie.

En Terminal, escriba:

```
echo 'Hello World'
```

Y pulse Intro.

Como puede ver, el comando `echo` significa: **repetir lo que digo**.

Para dar un poco de sustancia a un comando, hay que añadir un argumento, es decir, el texto entrecomillado. El argumento es lo que dice al comando qué debe hacer.

Para aprender otros scripts sencillos y más aplicables a su día a día, puede empezar por las cosas más básicas que normalmente haría en su equipo. Un buen entrenamiento es hacerlo durante la jornada igual que realiza tareas rutinarias con el Finder, para así ir ganando confianza en el proceso de creación de scripts.

Por ejemplo, ya sabe perfectamente cómo realizar los siguientes pasos desde el Finder del Mac:

- Crear una carpeta llamada «MisCosas».
- Crear un archivo que tenga algún contenido.
- Colocar el archivo en la carpeta.

Ahora puede probar a hacerlos desde Terminal, con los siguientes comandos:

`mkdir` = crear directorio (es un comando nuevo, que crea una carpeta)
`echo` = repetir lo que diga
`mv` = mover (o cambiar nombre, aunque en este ejercicio lo usaremos como «mover»)

Y los comandos pueden usarse de diferentes formas.

Para crear una carpeta en su escritorio llamada «MisCosas», en la línea de comandos de Terminal escriba:

```
mkdir Desktop/MisCosas
```

Y pulse Intro.

Para crear un archivo con contenido, empiece primero por el contenido y después dirija (>) el contenido hacia dónde quiera guardarlo y en el formato que prefiera, utilizando el formato habitual para una dirección de ubicación de archivo:

```
echo “The quick brown fox jumped over the lazy dogs.” > Desktop/Quick-Brown-Fox.txt
```

Y pulse Intro.

Coloque el archivo en la carpeta. En la línea de comandos, escriba:

```
mv Desktop/Quick-Brown-Fox.txt Desktop/MisCosas
```

Y pulse Intro.

Una de las cosas interesantes de la línea de comandos de Terminal es que si le pedimos que haga algo lo hace. Sin embargo, no muestra ningún mensaje de confirmación (excepto si se lo pedimos).

Por tanto, si quiere asegurarse, compruebe en el Finder si tiene una nueva carpeta llamada «MisCosas» y si contiene el documento Quick-Brown-Fox.txt.

También puede abrir el documento a través de Terminal.

Escriba:

```
open Desktop/MisCosas/Quick-Brown-Fox.txt
```

Y pulse Intro.

Puede eliminar documentos y carpetas a través de Terminal, con el comando `rm`.

`rm` = eliminar
`rm -r` = eliminar de forma recursiva

Para eliminar un documento, solo necesita `rm`. En la línea de comandos, pruebe lo siguiente:

```
rm Desktop/MisCosas/Quick-Brown-Fox.txt
```

A través del Finder, compruebe en la carpeta si el documento Quick-Brown-Fox.txt ha desaparecido.

Puede probar lo mismo para eliminar la carpeta:

```
rm Desktop/MisCosas
```

Si Terminal no puede o no quiere hacer lo que le pide, aparecerá un mensaje de error:

```
rm: Desktop/MisCosas: is a directory
```

Para indicar a Terminal que de verdad quiere eliminar la carpeta/directorio y todo lo que hay dentro, deberá incluir el comando de eliminación recursiva o `rm -r`.

Pruébelo en Terminal y compruebe después si la carpeta MisCosas ha desaparecido.

```
rm -r Desktop/MisCosas
```

Nota: si quiere que el nombre de una carpeta o documento tenga espacios, solo tiene que encerrar el nombre entre comillas. Por ejemplo, “Mis Cosas” o “Quick Brown Fox.txt”.

Creación de scripts a partir de comandos de Terminal

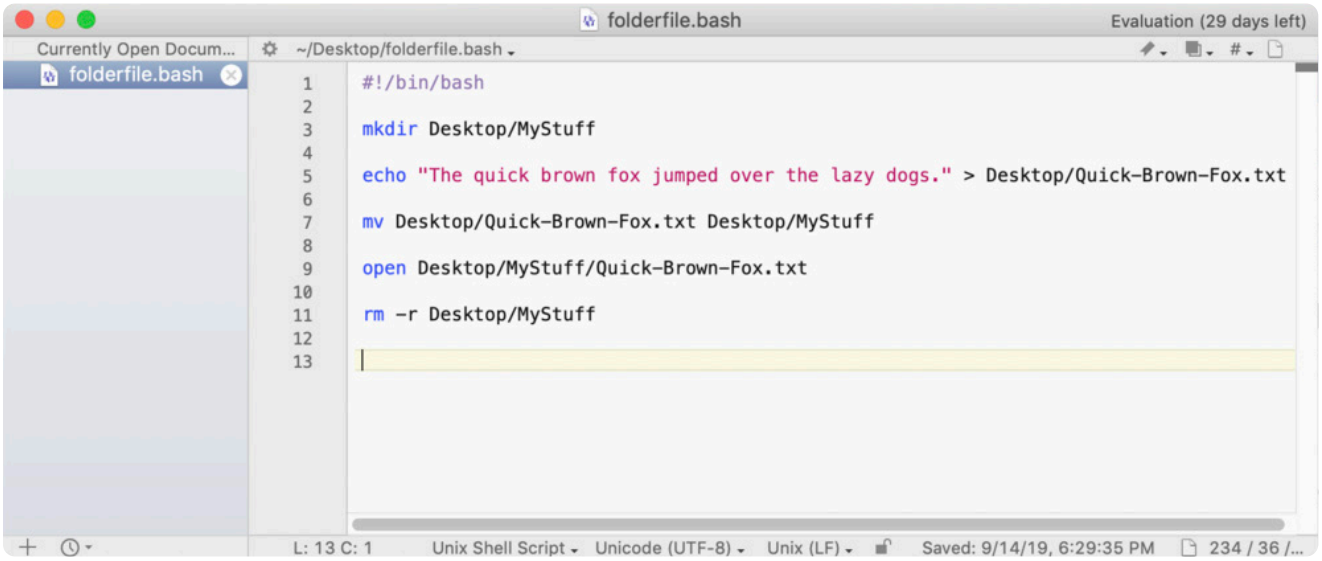
Ahora vamos a crear un script a partir de lo que acabamos de hacer en Terminal.

Para empezar, descargue y abra un editor de scripts. A nosotros nos gusta BBEdit, pero al final de este e-book encontrará una lista con otros recursos. Puede usar simplemente un editor de texto, pero un editor de scripts gratuito suele integrar resaltado de sintaxis, es decir, cambios en los colores de determinadas partes del script para poder ver rápidamente qué es cada cosa y también localizar y corregir errores. Esta función resulta especialmente útil para ayudar a los legos en la materia a entender en qué consiste la sintaxis de los scripts y también para resolver problemas. En este caso, los comandos son azules y los argumentos son de color rosa.

Antes de cada script debe escribir siempre algo llamado shebang, que es la combinación de hash + bang (#!). Y a continuación /bin/bash, para que al utilizar sus scripts Terminal sepa que debe usar bash como intérprete.

```
#!/bin/bash
```

Copie y pegue todas las líneas de comandos que ha utilizado en el editor de scripts. Si añade líneas en blanco, la lectura será más fácil.



Guarde el script en el escritorio como «**folderfile.bash**».

A continuación, arrastre este archivo a Terminal y pulse Intro.

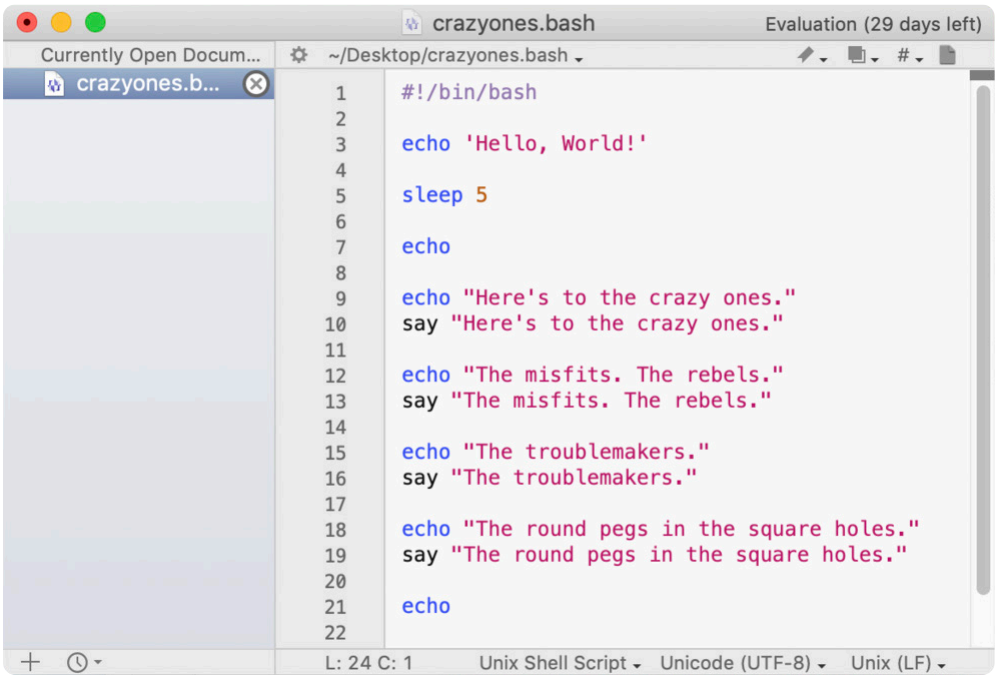
No pestañee si no quiere perdérselo: otra de las ventajas de los scripts es que son rapidísimos.

Y ahora otro ejercicio también interesante, pero en este caso empezaremos desde un editor de scripts.

Abra un editor de scripts y escriba:

```
#!/bin/bash
echo 'Hello, World!'
sleep 5
echo
echo “Here’s to the crazy ones.”
say “Here’s to the crazy ones.”
echo “The misfits. The rebels.”
say “The misfits. The rebels.”
echo “The troublemakers.”
say “The troublemakers.”
echo “The round pegs in the square holes.”
say “The round pegs in the square holes.”
echo
exit
```

Tendrá este aspecto:



Como puede ver, la tercera línea es un comando echo con «Hello, World!». echo es azul y las variables son rosa.

La quinta línea es `sleep 5`.

`Sleep` indica una pausa y 5 significa que su duración es de 5 segundos.

`Say` es un comando diferente y enseguida veremos qué significa.

Un comando `echo` en blanco es simplemente una forma de introducir un espacio: «echo [espacio en blanco]».

Todos estos comandos en una misma fila constituyen un script.

1. Guarde este script igual que el anterior, con la extensión `.bash`.
2. Seleccione el archivo y arrástrelo a Terminal.
3. Asegúrese de tener el sonido activado.
4. Pulse Intro y se ejecutará el script.

¿Ha visto lo que ha ocurrido? Ahora ya sabe qué significa `say`.

Comandos que ha aprendido hasta ahora:

`echo` = repetir lo dicho
`sleep` = pausa (utilizado junto con un argumento para indicar el número de segundos)
`say` = leer en voz alta
`open` = abrir cualquier archivo
`mkdir` = crear directorio/carpeta
`mv` = mover un archivo o cambiarle el nombre
`rm` = eliminar un archivo
`rm -r` = eliminar de forma recursiva (utilizado para eliminar una carpeta/directorio)

El funcionamiento de los scripts y el papel de Jamf

Variables: haciendo sitio para lo que vendrá

Las variables son espacios reservados para cosas que todavía no han ocurrido. Podemos asignarles valores como números, texto, nombres de archivo, dispositivos o cualquier otro tipo de datos. Una variable no es nada concreto, sino un indicador de unos datos que introduciremos más adelante.

Las variables son importantes porque, si las creamos correctamente, podemos ejecutar el mismo script muchas veces sin tener que volver a escribirlo, ya que simplemente se irán modificando los datos.

Veamos cómo funcionan a partir de la variable más simple: un nombre. Y lo haremos empezando desde atrás, como hicimos al crear un archivo con contenido.

En primer lugar decimos al equipo de qué datos se tratará. En este caso, escriba en Terminal:

```
myName=Martin
```

A continuación, pulse Intro. Utilice siempre un nombre de variable claro, transparente y fácil de recordar.

A continuación diremos al equipo qué debe hacer con los datos. Siempre que utilicemos una variable, indicaremos que se trata de una variable con el símbolo $. Por tanto, escribiremos en Terminal:

```
echo Hello, my name is $myName.
```

A continuación, pulse Intro.

En Terminal, veremos:

```
Hello, my name is Martin.
```

Definir una variable y utilizarla en un script es un recurso que puede tener muchas aplicaciones prácticas en su flujo de trabajo.

En una sola línea

Ahora veremos otro comando útil, modificado por una variable. Imaginemos que quiere saber si un Mac necesita actualizar su sistema operativo.

Una vez más, empezaremos desde atrás.

Primero decidiremos qué variable vamos a usar y usaremos el resultado de un comando para localizarla.

Para conseguirlo, vamos a necesitar dos comandos nuevos (en realidad uno, pero complementado):

`sw_vers` = versión de software
`sw_vers -productVersion` = solo la versión del producto del software

El comando `sw_vers` nos indica información sobre macOS: el nombre del sistema operativo, la versión que estamos utilizando y el número de compilación. Si solo quiere saber la versión, puede escribir el script con -productVersion después del comando y solo verá lo que necesita.

Pruébelo en Terminal como si fuera un comando.

Los resultados de la primera versión deberían ser más o menos estos:

```
ProductName:  Mac OS X
ProductVersion: 10.14.4
BuildVersion: 18E226
```

En la segunda versión, el resultado sería simplemente:

```
10.14.4
```

Al trabajar a partir de comandos existentes no hace falta definir la variable con antelación, pero sí es necesario indicar que estamos usando una variable. Para indicar que se trata de una variable, podríamos usar:

```
$( sw_vers -productVersion )
```

Seguimos usando el símbolo $, pero en este caso utilizamos como variable el comando de versión del software y lo indicamos entre paréntesis. Como si de álgebra se tratara, lo que está entre paréntesis se calcula primero y después viene todo lo que está fuera.

Pruebe a escribir lo siguiente en su línea de comando:

```
echo My operating system is $( sw_vers -productVersion ).
```

Para que nos resulte todavía más útil, tendremos que echar mano de un recurso conocido como condicionales.

Es una cuestión de lógica: los condicionales mejoran los scripts mediante comandos «if»/«then»

Los condicionales son formulaciones tipo «si»/«entonces». Este tipo de formulaciones permite tomar decisiones: les ponemos las condiciones que deben cumplirse y después les decimos que evalúen si se cumplen.

Por ejemplo:

SI tengo limones, ENTONCES haré limonada.

Para obtener resultados realmente útiles, puede combinar variables con condicionales.

Por ejemplo, puede escribir lo siguiente en un editor de scripts:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = “10.14.5” ]; then
   echo No upgrade needed.
fi
```

Empezando con una variable, un administrador puede consultar qué versión del SO tiene el Mac.

Añadiendo un condicional, empezando por `if`, y después un `echo`, Terminal comprobará qué versión del SO está ejecutando el Mac y, si es la versión que quiere el administrador, Terminal indicará que no hace falta ninguna actualización.

Los condicionales terminan en `fi`, que es «if» al revés. Curioso, ¿verdad?

Pruébelo en Terminal guardando este script y arrastrándolo a Terminal.

¿Qué ocurre si la condición `if` es falsa? Solo tiene que añadir un enunciado `else` y especificar qué quiere que ocurra. «Else» vendría a significar en este caso «de lo contrario».

Añádalo a su script, que quedaría así:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = “10.14.5” ]; then
   echo No upgrade needed.
else
   echo Update required.
fi
```

A continuación, ejecútelo en Terminal.

¿Puede ejecutar un script que no solo detecte qué versión del SO tiene un Mac sino que además instale la última versión si hace falta?

Naturalmente, con la ayuda de las políticas de Jamf.

Definición de políticas

En este ejemplo, utilizaremos el binario del asistente de Jamf para pedir al Mac que ejecute un evento: una actualización del SO.

En cada Mac inscrito, Jamf Pro instala su propia línea de comando jamf o herramienta binaria. El comando `jamf` ejecuta acciones específicas de la gestión de Jamf, como la aplicación de políticas. Podemos asignar a cada política un nombre único, conocido como activador personalizado, y después usar la herramienta de línea de comando `jamf` para recuperar esta política a partir del nombre de su activador personalizado y ejecutarla.
En este caso, el script que usaremos recuperará la política runUpgrade, que básicamente ejecuta una actualización.

Para hacerlo en varios equipos a la vez, necesitará un servicio de MDM como Jamf Pro.

Para añadir este script a Jamf Pro, vaya a **Ajustes** > **Gestión de ordenadores** > **Scripts** y seleccione el botón Nuevo.

Aquí puede asignarle un nombre diferente, como CheckOSVersionandRunUpgrade, añadir notas, guardar el script en la pestaña Script, indicar opciones y limitaciones, y guardarlo.

Indique lo siguiente en la pestaña Script:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = “10.14.5” ]; then
   echo No upgrade needed.
else
   jamf policy -event runUpgrade
fi
```

Jamf Pro proporciona una vista de scripts que indica los tipos de script, más o menos como un editor de scripts. Seleccione shell una vez en el menú desplegable y, a partir de entonces, se abrirá de forma predeterminada este editor simplificado.

Puede añadir comandos echo que después se reflejarán en sus registros de políticas. Esto le facilitará las cosas al resolver problemas y localizar dispositivos que no se han actualizado.

Para crear la política, cree una nueva política y sitúela dentro de un grupo inteligente. Conviértalo en un evento recurrente y asócielo al código que ha creado y al que ha asignado un nombre anteriormente. (Para obtener más información sobre la creación de políticas, consulte https://docs.jamf.com/10.1.0/jamf-pro/quickstart-computers/Create_a_Policy_to_Run_Software_Update.html).

While: el condicional que espera su momento

Otro condicional es el enunciado «while». En lugar de evaluar si algo es cierto o no, espera hasta que este algo sea cierto.

```
#!/bin/bash
while [ $( pgrep TextEdit )]
do
   echo Waiting for TextEdit to quit.
   sleep 2
done
echo TextEdit has quit.
```

El comando `pgrep` determina si un proceso está o no en ejecución. En este caso, la aplicación TextEdit. Los dos enunciados siguientes nos dicen qué debería ocurrir mientras TextEdit está en ejecución:

1. repetición de espera de cierre del proceso.
2. dos segundos de espera con comando de suspensión.
3. nueva evaluación de la condición «while».
4. cuando TextEdit se haya cerrado y la condición «while» sea falsa, el resto del script puede continuar.

Si abre TextEdit, ejecuta este script en Terminal y cierra TextEdit, verá cómo funciona este script.

Bucles: acciones en archivos o procesos

Vamos a ejecutar un bucle `for` para pasar una lista de archivos a través de un bucle `for` y cambiar el nombre de cada archivo de la lista hasta que terminemos.

En este ejemplo usaremos varios comandos:

`cd` = cambiar directorio
~ es un atajo para la carpeta de inicio.
`ls` = lista (crea una lista de ítems en el directorio que se seleccione)
`aFile` = variable que representa cualquier archivo y puede ser la frase que elija.
'`for`' en este caso significa tanto definir una variable como leer una variable. Lo que dice es: para cada archivo de la lista de archivos, hacer lo siguiente. Repetimos la orden de cambio de nombre de archivo y usamos el comando move para hacerlo.

En este caso vamos a modificar el nombre de cada archivo de la carpeta MisCosas con la palabra webinar.

Así es cómo lo haremos.

```
#!/bin/bash

cd ~/Desktop/MisCosas

filelist=$( ls )

for aFile in $filelist
do
      echo Renaming file $aFile
      mv $aFile webinar-$aFile
done
```

Ahora todos los archivos de su carpeta MisCosas deberían empezar con la palabra webinar-.

Una mirada al futuro

No se preocupe. ¡No esperamos que se sepa de memoria todos los comandos que aparecen a continuación! Sin embargo, seguramente se sorprenderá de lo bien que ha asimilado los conceptos básicos.

Ejemplo: Conseguir que los usuarios se actualicen a Mojave

Imaginemos que un administrador quiere animar a los usuarios a actualizarse a Mojave y después actualizar automáticamente los Mac que no se han actualizado cuando haya transcurrido un tiempo determinado.

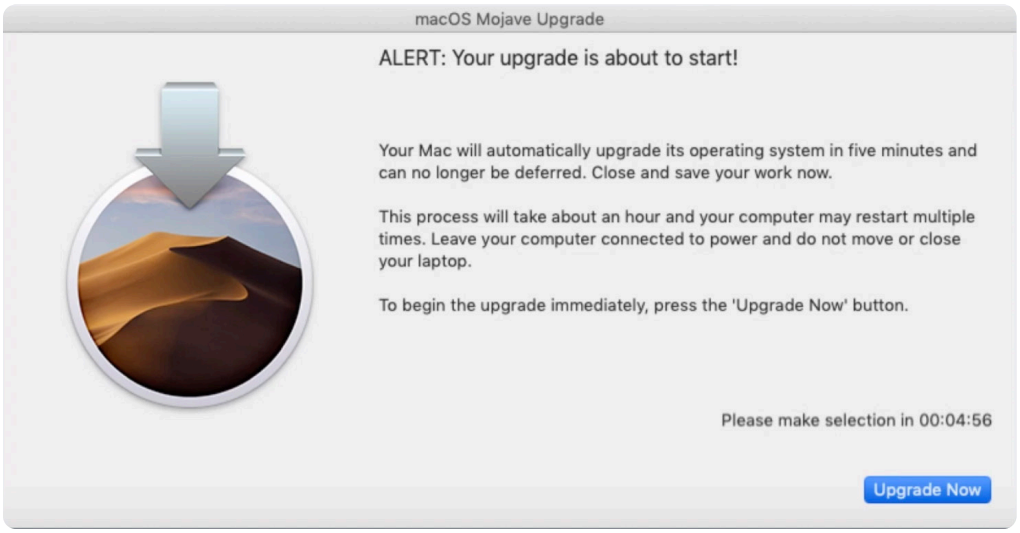
Así es cómo quedaría el script una vez finalizado el período de aviso.

```
#!/bin/bash

"/Library/Application  Support/JAMF/bin/jamfHelper.app/Contents/MacOS/jamfHelper" \
-windowType utility \
-lockHUD \
-title "macOS Mojave Upgrade" \
-heading "ALERT: Your OS upgrade is about to start!" \
-description "Your Mac will automatically upgrade its operating system in five
minutes and can no longer be deferred. Close and save your work now. The process
will take about an hour and your computer may restart multiple times. Leave your
computer connected to power and do not move or close your laptop. To begin the
upgrade immediately, select the 'Upgrade Now' button." \
-icon "/Applications/Install macOS Mojave.app/Contents/Resources/InstallAssistant.
icns" \
-iconSize 256
-button1 "Upgrade Now" \
-defaultButton 1 \
-countdown \
-timeout 300 \
-alignCountdown right

exit 0
```

Y así es cómo queda al ejecutarse:



Como puede ver, no cubre toda la pantalla (-iconSize 256). De este modo el usuario puede hacer clic y guardar su trabajo. La cuenta atrás aparece en la parte derecha y el botón por omisión es «Upgrade Now».

El botón, o el final de la cuenta atrás, debería desencadenar este proceso:

```
“/Applications/Install macOS Mojave.app/Contents/Resources/startosinstall”
—-agreetolicense &
```

Este script está integrado en la mayoría de los instaladores. Esta línea parte de la premisa de que Mojave ya está en la carpeta y el ampersand reinicia el ordenador.

Con este ejemplo de programación más avanzada no pretendemos asustarle. Nuestra intención es demostrarle que con unos conocimientos básicos puede hacer bastantes cosas con los scripts. De entrada, automatizar y mejorar sus flujos de trabajo y también la experiencia de los usuarios.

Y también le enseña otra cosa: aunque no haya escrito un código desde cero, también puede utilizarlo. ¿Quiere introducir algún cambio en un script? Búsquelo en Google, porque seguramente alguien ya lo ha hecho antes.

Y esto nos lleva a hablar de los recursos.

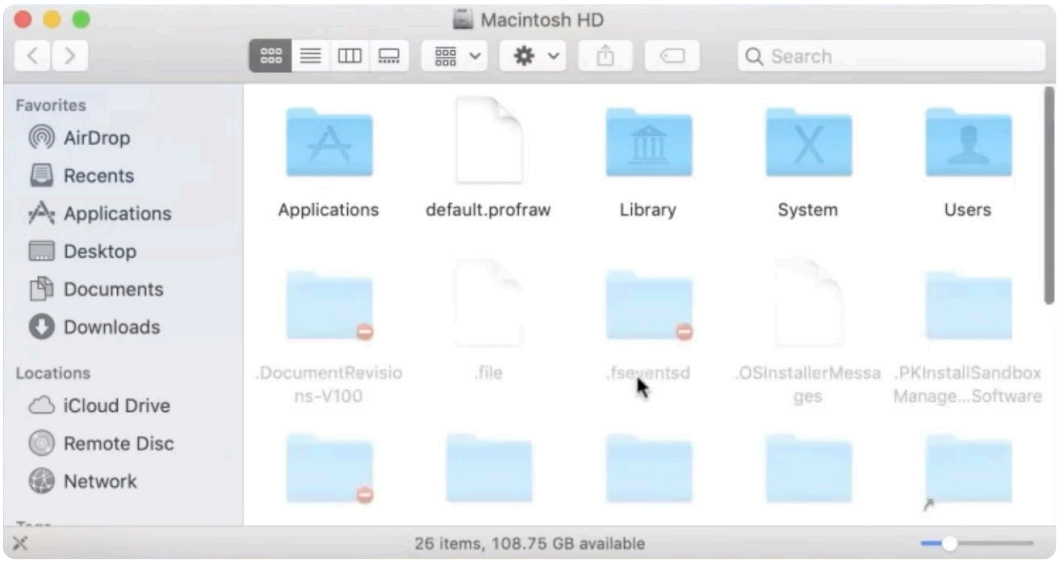
Próximo paso: recursos

Su Mac

Si quiere descubrir nuevos comandos, el recurso más valioso es el que tiene más a mano: su ordenador es por sí solo un recurso espectacular.

Pruebe lo siguiente:

Desde una ventana del Finder, seleccione Mayúsculas ⇧, comando ⌘, punto . para ver archivos y carpetas ocultos. En última instancia, los comandos de Terminal se apoyan en algo. Y su ordenador guarda todas las instrucciones asociadas a cada comando en un archivo separado.



La primera carpeta en la que debería mirar es bin. Bin es la forma abreviada de binario. Abra la carpeta para ver qué hay. En esta carpeta hay aproximadamente 35 ítems, como echo, bash, mkdir y otros. Seguramente algunos le suenan y otros no tanto.

No hace falta que sepa qué hace cada comando. Al escribir sus scripts, piense en lo que haría en el Finder y busque después comandos que hagan lo que necesita. ¡Confíe en Google! Si algo puede hacerse en el Finder, entonces existe una línea de comando para esa función.

También puede investigar otras carpetas como sbin y uxr (que contiene recursos del sistema de Unix). ¡Eche un vistazo!

En conjunto, tiene a tiro de piedra más de 1000 comandos.

Vídeos didácticos

Esta guía es una versión simplificada del webinar «Scripting for Beginners». Si le van los cursos de tipo magistral, seguramente le resultará bastante útil, aunque también encontrará más ejemplos en este otro curso: **Scripting 101 for Apple Admins**

Después de ver el vídeo, no se pierda la entrada del blog asociada, con ideas sobre los próximos pasos y cómo continuar aprendiendo, y con una lista más exhaustiva de recursos que los de esta guía: **Just 10 Commands, Then Keep Learning**

Trainingcatalog.jamf.com

Tiene a su disposición 15 horas de vídeos didácticos breves sobre scripts, con cursos básicos y avanzados. No se pierda nuestra serie dedicada a los scripts, disponible para clientes con suscripción.

La comunidad de código abierto

La comunidad de código abierto de **github.com/jamf** dispone de muchos scripts de código abierto gratuitos que puede utilizar. ¡Si algo ya existe no hace falta volver a inventarlo! Busque para ver si alguien ya ha escrito algo que pueda servirle. En Jamf Nation, un foro de mensajes independiente gestionado por Jamf y pensado para administradores de Apple, su gran familia de administradores de Mac está a su disposición para lo que necesite: **https://www.jamf.com/jamf-nation/**

Un buen editor de scripts

Esperamos haberle convencido de la importancia de usar un editor de scripts para poder ver claramente la sintaxis de los scripts. Aquí le proponemos algunos. La mayoría son gratis.

bbedit: **barebones.com** (versión básica gratis)

Atom: **atom.io** (versión básica gratis)

Coderunner: **coderunnerapp.com** 14,99 USD en el momento de la publicación de este documento

Esperamos que esta guía le haya servido para despejar sus dudas en relación con los scripts y animarle a lanzarse a la piscina, experimentar, aprender y mejorar sus procesos.

Y si los scripts por sí solos ya pueden aportarle mucho, combinados con una buena solución de MDM pueden convertir horas y horas de trabajo en algo tan fácil como hacer clic en un botón. Nosotros nos permitimos recomendarle Jamf Pro.

jamf | PRO

Solicitar prueba

jamf

www.jamf.com

O póngase en contacto con su distribuidor autorizado de Apple para realizar una prueba gratuita de Jamf Pro.