Engineer

Bash

>_ el shell de GNU para el que se escribe la mayoría de scripts

Qué es

Bash, Bourne-Again SHell, distribuido como GNU Bash e iniciado con el comando `bash`, lee comandos introducidos tras el indicador de comandos o guardados en un archivo y los ejecuta. Inicia otros programas, conecta sus entradas y salidas y añade variables, bucles, pruebas y funciones para que la secuencia tome decisiones. El nombre juega con Stephen Bourne, cuyo `sh` llegó con Seventh Edition de Research Unix en Bell Labs. El manual lo describe como compatible en gran medida con aquel intérprete, con características tomadas del Korn shell y del C shell, y diseñado para ajustarse a la parte de POSIX dedicada al shell. La versión actual es 5.3 y el manual 5.3 está fechado en mayo de 2025.

Se confunde fácilmente con la ventana: el terminal dibuja caracteres y transmite pulsaciones; el intérprete es el programa al otro lado que las interpreta. Bash funciona interactivamente, con indicador, historial y autocompletado por tabulación, o sin interacción, leyendo el script de principio a fin sin vigilancia. Ahí residen la mayoría de sus trampas.

Bash es uno de varios programas que pueden desempeñar el mismo papel. POSIX define un lenguaje de shell portátil, `sh`, y el programa instalado como `/bin/sh` varía según el sistema. En Debian y Ubuntu es dash, un pequeño intérprete POSIX descendiente del intérprete Almquist, y Debian Policy enumera bash, dash y ksh como los más habituales allí. Zsh es el intérprete de inicio de sesión predeterminado para las cuentas nuevas de macOS desde la versión 10.15, aunque el Mac sigue incluyendo `/bin/bash`. Un script elige su intérprete mediante la primera línea, el shebang: `#!/bin/sh` pide el intérprete POSIX que tenga el sistema, mientras que `#!/bin/bash` o `#!/usr/bin/env bash` pide Bash por su nombre.

Utilice el concepto en una decisión

Aplique «¿Qué es Bash?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. El cuidado diario de servidores, correo, cuentas y actualizaciones para que los servicios de los que depende todo el mundo sigan funcionando sin sorpresas.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 10 servidores PostgreSQL y MS SQL en Ubuntu — 10 servidores de bases de datos desplegados y mantenidos

10 servidores PostgreSQL y MS SQL en Ubuntu Póngase en contacto →

Por qué importa

Bash es el lenguaje que conecta servidores Unix. Está en casi toda máquina Linux y los comandos que un ingeniero escribe ya son Bash; guardarlos en un archivo crea un script que usa lo disponible. Para encadenar programas —volcar una base, comprimir el volcado y copiarlo— pocos lenguajes son tan breves, porque tuberías, redirección y estados de salida son vocabulario nativo.

Esas mismas características se convierten en costes a medida que crece un script. Todos los valores son cadenas y la aritmética solo trabaja con enteros, por lo que cualquier dato con estructura, como JSON, fechas o cantidades decimales, debe delegarse en otra herramienta. Las expansiones sin comillas se dividen en palabras y después se comparan con nombres de archivo, de modo que un valor con un espacio o un asterisco cambia el significado del comando al que llega. El manejo de errores debe activarse: de forma predeterminada, un script continúa después de que falle un comando, y las opciones que cambian eso, `set -e` y `pipefail`, tienen excepciones que el manual necesita varios párrafos para enumerar. Una regla práctica es que, cuando un script necesita estructuras de datos, un analizador o una batería de pruebas, se ha convertido en un programa y resulta más barato mantenerlo en un lenguaje como Python.

La portabilidad también es una decisión. Debian Policy exige que los scripts `/bin/sh` se limiten a las funciones POSIX más una lista breve de extensiones, y pide que cualquier script que necesite más nombre su intérprete en la primera línea, por ejemplo `#!/bin/bash`. Sugiere `checkbashisms` o una ejecución con un intérprete más estricto, como posh, para detectar incumplimientos, y aconseja `/bin/bash` en caso de duda. El estándar avanza despacio: Bash ofrece `pipefail` desde la versión 3.0 y POSIX no lo añadió hasta su edición de 2024.

Un intérprete de shell convierte texto en comandos, por lo que constituye una frontera de seguridad dondequiera que le llegue texto externo. Shellshock, registrado como CVE-2014-6271 en septiembre de 2014, mostró la amplitud que puede tener esa frontera. Bash hasta la versión 4.3 ejecutaba los comandos que seguían a una definición de función en una variable de entorno, de modo que cualquier componente que copiara texto controlado por un atacante al entorno antes de iniciar Bash, incluidos scripts CGI bajo Apache, la función ForceCommand de OpenSSH y algunos clientes DHCP, se convertía en una vía para ejecutar código en la máquina. La corrección inicial era incompleta, y lo que quedaba recibió su propio identificador, CVE-2014-7169. La lección cotidiana es más pequeña: el texto externo debe llegar a un comando como argumento, nunca como parte de una cadena entregada a `eval` o `sh -c` .

Cómo se aplica en la práctica

En el trabajo de la empresa, Bash mantiene el papel de la página de tecnologías: Python para datos, automatización y servicios; Bash como nexo, escrito para ser leído.

En los seis flujos ETL y ELT que alimentaban Google BigQuery desde un almacén de datos detrás de una VPN IPsec, los archivos comprimidos descargados contenían archivos Parquet, CSV y .bak. Los scripts Bash y Python los descomprimían y clasificaban cada archivo por tipo y disposición de columnas antes de que bcp, psql y SSIS los cargaran en esquemas staging y gzip, Google Cloud Storage y la herramienta bq llevaran los resultados a BigQuery. Con la extracción y clasificación automatizadas, una ejecución que había requerido al menos tres horas manuales terminaba en menos de 20 minutos, y los datos podían sincronizarse cada día en vez de una vez por semana.

Las cien copias automatizadas de datos críticos muestran la misma división del trabajo. Barman se ocupaba de PostgreSQL, Google Cloud conservaba las copias fuera de las instalaciones y Bash y Python orquestaban y validaban todo el sistema, con políticas de retención que mantenían las copias al día y comprobaciones de integridad que confirmaban que cada una era válida y no simplemente que existía. Ese es el trabajo adecuado para Bash: una secuencia de herramientas existentes en la que el valor del script reside en comprobar qué devolvió cada paso.

Dónde surgen los problemas

Un nombre de archivo con un espacio se convierte en dos argumentos. Un `$file` sin comillas se divide por espacios en blanco y después se expande como un patrón de nombres de archivo, por lo que `report 2024.csv` llega a `rm` como dos nombres y un `*` accidental coincide con todo un directorio. Ponga entre comillas cada expansión, como en `"$file"`, transmita las listas de argumentos como `"$@"`, conserve las listas de nombres en arrays y deje que ShellCheck encuentre el resto.

La tubería de copia informa de éxito tras fallar el volcado. Su estado es el del último comando: en `pg_dump app | gzip > app.sql.gz`, gzip termina correctamente con lo poco recibido aunque falle el volcado. Active `set -o pipefail` para devolver el último estado no cero y compruebe con `gzip -t` o, mejor, restaurando.

`set -e` **deja continuar tras un fallo.** Se ignora en la condición de `if` o `while`, en todos los elementos de `&&` o `||` salvo el último y dentro de funciones llamadas desde esos lugares. El siguiente volcado puede fallar y aun así imprimir éxito:

```
#!/usr/bin/env bash
set -euo pipefail

backup() {
  pg_dump app > /var/backups/app.sql
  gzip -f /var/backups/app.sql
}

if backup; then # set -e is ignored inside backup here
  echo "backup done"
fi
```

Otra trampa es `local out=$(cmd)`: devuelve el estado de `local` y oculta el fallo de `cmd`. En funciones, añada `|| return 1` tras cada comando importante y declare la variable y asígnele un valor en una línea aparte.

Un script que funciona en una máquina falla en otra. Un archivo que empieza por `#!/bin/sh` pero usa pruebas `[[` o arrays se ejecuta bajo Bash en un sistema y bajo dash en el siguiente. Allí `[[` es un comando desconocido, por lo que un `if` construido con él imprime un error y toma su rama else, y una asignación de

array es un error sintáctico que termina el script. En un Mac, `/bin/bash` informa de la versión 3.2.57, anterior a los arrays asociativos y a `mapfile`, que llegaron en 4.0, por lo que `declare -A` se detiene allí con un error de opción inválida. Nombre en el shebang el intérprete que realmente necesita el script y pruebe contra la versión más antigua en la que deba ejecutarse.

La limpieza borra mucho más de lo previsto. Sin `build_dir`, `rm -rf "$build_dir"/*` se expande a `rm -rf /*`. `set -u` convierte una variable no definida en error y `"${build_dir:?}"` detiene un script no interactivo cuando falta o contiene una cadena vacía.

Lecturas adicionales

- Manual de referencia GNU Bash — el manual 5.3: expansiones, comillas, el comando interno `set` y los casos exactos en los que se ignora `set -e`.
- Shell Command Language — el lenguaje `sh` portátil de POSIX.1-2024, el límite para cualquier script `#!/bin/sh`.
- Debian Policy: Scripts — qué debe admitir `/bin/sh` en Debian, cuándo un script debe pedir Bash por su nombre y los consejos sobre `set -e`.
- Bash NEWS — la lista de nuevas funciones del mantenedor, versión por versión, para comprobar qué versión requiere una construcción.
- CVE-2014-6271 — el registro NVD de Shellshock, con las vías de ataque que enumera y el identificador posterior.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Bash?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/bash/