Engineer

Bash

>_ det GNU-skal som de flesta skript skrivs för

Vad det är

Bash, Bourne-Again SHell, som distribueras som GNU Bash och startas med kommandot `bash`, är ett program som läser kommandon, inskrivna vid en prompt eller sparade i en fil, och utför dem: det startar andra program, kopplar ihop deras in- och utdata och lägger till variabler, loopar, tester och funktioner så att en följd av kommandon kan fatta beslut. Namnet är en ordlek med Stephen Bourne, vars `sh` följde med Seventh Edition av Research Unix vid Bell Labs. Manualen beskriver Bash som i stort sett kompatibelt med det skalet, med funktioner hämtade från Korn-skalet och C-skalet, och som avsett att följa skaldelen av POSIX-standarden. Den aktuella versionen är 5.3, och manualen för 5.3 är daterad maj 2025.

Ett skal förväxlas lätt med fönstret det körs i: terminalen ritar tecken och skickar vidare tangenttryckningar, och skalet är programmet på andra sidan som tolkar dem. Bash körs interaktivt, med en prompt, historik och tabbkomplettering, eller icke-interaktivt, när det läser ett skript uppifrån och ned utan tillsyn, och det är där de flesta av dess fällor finns.

Bash är ett av flera program som kan fylla samma roll. POSIX definierar ett portabelt skalspråk, `sh`, och vilket program som är installerat som `/bin/sh` varierar mellan system. På Debian och Ubuntu är det dash, ett litet POSIX-skal som härstammar från Almquist-skalet, och Debian Policy nämner bash, dash och ksh som de vanligaste skalen att använda där. Zsh har varit standardskal vid inloggning för nya konton i macOS sedan version 10.15, även om en Mac fortfarande har en `/bin/bash`. Ett skript väljer sin tolk med sin första rad, shebang-raden: `#!/bin/sh` ber om det POSIX-skal som systemet har, medan `#!/bin/bash` eller `#!/usr/bin/env bash` ber om Bash vid namn.

Använd begreppet i ett beslut

Använd ”Vad är Bash?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Daglig skötsel av servrar, e-post, konton och uppdateringar, så att det som alla förlitar sig på förblir odramatiskt.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu — 10 databasservrar som driftsattes och hölls i gott skick

Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu Hör av er →

Varför det spelar roll

Bash är det sammanbindande språket på Unix-servrar. Det finns på nästan varje Linux-maskin, och kommandona som en ingenjör skriver vid en prompt är redan Bash, så en handfull av dem som sparas i en fil blir ett skript som körs på det som maskinen redan har. För att kedja ihop befintliga program, t.ex. dumpa en databas, komprimera dumpen och kopiera den någon annanstans, är få språk lika kortfattade, eftersom rör, omdirigering och slutstatus är dess modersmål.

Samma egenskaper blir kostnader när ett skript växer. Varje värde är en sträng, och aritmetik fungerar bara på heltal, så allt med struktur, som JSON, datum eller decimalbelopp, måste lämnas över till ett annat verktyg. Expansioner utan citattecken delas upp i ord och matchas sedan mot filnamn, så ett värde som innehåller ett mellanslag eller en asterisk ändrar betydelsen av kommandot det hamnar i. Felhanteringen är något man måste slå på: som standard fortsätter ett skript efter att ett kommando har misslyckats, och de alternativ som ändrar det, `set -e` och `pipefail`, kommer med undantag som manualen behöver flera stycken för att räkna upp. En användbar regel är att när ett skript behöver datastrukturer, en parser eller en testsvit har det blivit ett program, och ett språk som Python är billigare att underhålla.

Portabilitet är också ett beslut. Debian Policy kräver att `/bin/sh`-skript håller sig till funktioner i POSIX plus en kort lista med tillägg, och ber varje skript som behöver mer att namnge sitt skal på första raden, t.ex. `#!/bin/bash`. Den föreslår `checkbashisms` eller en körning under ett striktare skal som posh för att hitta överträdelser, och råder till `/bin/bash` vid tvekan. Standarden rör sig långsamt: Bash har erbjudit `pipefail` sedan version 3.0, och POSIX lade till det först i sin utgåva från 2024.

Ett skal gör text till kommandon, vilket gör det till en säkerhetsgräns överallt där text utifrån når det. Shellshock, registrerad som CVE-2014-6271 i september 2014, visade hur bred den gränsen kan vara. Bash till och med version 4.3 körde kommandon som följde efter en funktionsdefinition i en miljövariabel, så allt som kopierade text som en angripare styrde in i miljön innan Bash startades, bl.a. CGI-skript under Apache, funktionen ForceCommand i OpenSSH och vissa DHCP-klienter, blev ett sätt att köra kod på maskinen. Den första rättningen var ofullständig, och det som återstod fick en egen identifierare, CVE-2014-7169. Den vardagliga lärdomen är mindre: text utifrån når ett kommando som ett argument, aldrig som en del av en sträng som lämnas till `eval` eller `sh -c`.

Så ser det ut i praktiken

I företagets arbete har Bash den uppgift som teknikstacken ger det: Python för dataarbete, automatisering och tjänster, och Bash för limmet, skrivet för att läsas.

I de sex datapipelines för ETL och ELT som förde data till Google BigQuery från ett datalager bakom ett IPsec-VPN i ett uppdrag som dataingenjör mellan 2022 och 2024, innehöll de nedladdade arkiven Parquet-, CSV- och .bak-filer. Skript i Bash och Python packade upp dem och klassificerade varje fil efter typ och schema innan bcp, psql och SSIS läste in dem i staging-scheman, och gzip, Google Cloud Storage och verktyget bq förde resultaten vidare till BigQuery. Med uppackningen och sorteringen skriptade gick en körning som hade tagit minst tre timmar för hand på under 20 minuter, och data kunde synkas varje dag i stället för en gång i veckan.

De hundra automatiserade kritiska säkerhetskopiorna av data visar samma arbetsfördelning. Barman tog hand om PostgreSQL och Google Cloud förvarade kopiorna på annat håll, medan Bash och Python styrde körningarna och kontrollerade dem. Regler för lagringstid höll kopiorna aktuella, och integritetskontroller visade att varje kopia var i gott skick, inte enbart att den existerade. Det är den uppgift som Bash passar för: en följd av befintliga verktyg, där skriptets värde ligger i att kontrollera vad varje steg returnerade.

Var det går fel

Ett filnamn med ett mellanslag blir två argument. En `$file` utan citattecken delas vid blanktecken och expanderas sedan som ett glob-mönster, så `report 2024.csv` når `rm` som två namn och en vilsekommen `*` matchar en hel katalog.

Citera varje expansion, som i `"$file"`, skicka argumentlistor vidare som `"$@"`, håll listor med namn i arrayer och låt ShellCheck hitta resten.

En rörledning för säkerhetskopiering rapporterar framgång efter att dumpen har misslyckats. En rörlednings (pipeline) slutstatus är det sista kommandots, så i `pg_dump app | gzip > app.sql.gz` lyckas gzip ändå med det lilla den fick, även när dumpen misslyckades. Slå på `set -o pipefail`, som får rörledningen att returnera den sista statusen som inte är noll, och kontrollera resultatet med `gzip -t` eller, hellre, en provåterställning.

Ett skript med `set -e` fortsätter förbi ett kommando som misslyckas. Alternativet ignoreras i testet för en `if` eller `while`, i varje del av en `&&` - eller `||` -lista utom den sista och inuti varje funktion som anropas från de ställena, så dumpen nedan kan misslyckas medan skriptet ändå skriver ut sitt meddelande om framgång:

```
#!/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
```

En närliggande fälla är `local out=$(cmd)`, som returnerar statusen för `local` och döljer ett fel i `cmd`. Inuti funktioner, följ varje kommando som spelar roll med `|| return 1`, och deklarera en variabel på en egen rad innan den tilldelas ett värde.

Ett skript som fungerar på en maskin går sönder på en annan. En fil som börjar med `#!/bin/sh` men använder `[[` -tester eller arrayer körs under Bash på ett system och under dash på nästa. Där är `[[` ett okänt kommando, så en `if` som bygger på det skriver ut ett fel och tar sin else-gren, och en tilldelning till en array är ett syntaxfel som avslutar skriptet. På en Mac rapporterar `/bin/bash` version 3.2.57, äldre än de associativa arrayer och det `mapfile` som kom i 4.0, så

`declare -A` stannar där med ett fel om ogiltigt alternativ. Namnge det skal som skriptet verkligen behöver i shebang-raden, och testa mot den äldsta version det måste köras på.

En städrad raderar mycket mer än den borde. När `build_dir` inte är satt expanderas `rm -rf "$build_dir"/*` till `rm -rf /*`. `set -u` gör en variabel som inte är satt till ett fel, och att skriva `"${build_dir:?}"` stoppar ett icke-interaktivt skript så fort variabeln saknas eller är satt till en tom sträng.

Läs mer

- GNU Bash Reference Manual — manualen för 5.3: expansioner, citering, det inbyggda kommandot `set` och exakt i vilka fall `set -e` ignoreras.
- Shell Command Language — det portabla språket `sh` i POSIX.1-2024, gränsen för varje `#!/bin/sh` -skript.
- Debian Policy: Scripts — vad `/bin/sh` måste stödja i Debian, när ett skript måste be om Bash vid namn, och rådet om `set -e`.
- Bash NEWS — förvaltarens lista över nya funktioner, version för version, för att kontrollera vilken version en konstruktion kräver.
- CVE-2014-6271 — NVD:s post för Shellshock, med de angreppsvägar den listar och den uppföljande identifieraren.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är Bash?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/bash/