Engineer

Bash

>_ оболонка GNU, для якої пишуть більшість скриптів

Що це таке

Bash (Bourne-Again SHell), що поширюється як GNU Bash і запускається командою `bash`, — це програма, яка читає команди, набрані в рядку запрошення або збережені у файлі, і виконує їх: запускає інші програми, з'єднує їхні вхід і вихід та додає змінні, цикли, перевірки й функції, щоб послідовність команд могла ухвалювати рішення. Назва обіграє прізвище Стівена Борна, чия оболонка `sh` постачалася із сьомою редакцією Research Unix у Bell Labs. Посібник описує Bash як здебільшого сумісну з тією оболонкою, з можливостями, запозиченими з Korn shell і C shell, і як таку, що має відповідати частині стандарту POSIX про оболонку. Поточний випуск — 5.3, а посібник до нього датовано травнем 2025 року.

Оболонку легко сплутати з вікном, у якому вона працює: термінал малює символи й передає натискання клавіш, а оболонка — це програма по інший бік, яка їх тлумачить. Bash працює інтерактивно, з рядком запрошення, історією та автодоповненням за клавішею Tab, або неінтерактивно, читаючи скрипт від початку до кінця без нагляду людини, і саме там ховається більшість її пасток.

Bash — одна з кількох програм, що можуть виконувати цю роль. POSIX визначає переносну мову оболонки, `sh`, а те, яка програма встановлена як `/bin/sh`, залежить від системи. У Debian та Ubuntu це dash — невелика оболонка POSIX, що походить від Almquist shell, а Debian Policy називає bash, dash і ksh найпоширенішими оболонками, які користувачі можуть обрати як `/bin/sh`. Zsh є типовою оболонкою для нових облікових записів у macOS від версії 10.15, хоча Mac досі містить `/bin/bash`. Скрипт обирає свій інтерпретатор першим рядком, так званим shebang: `#!/bin/sh` просить ту оболонку POSIX, яка є в системі, а `#!/bin/bash` або `#!/usr/bin/env bash` просить саме Bash.

Застосуйте поняття до рішення

Застосуйте «Що таке Bash?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Щоденний догляд за серверами, поштою, обліковими записами й оновленнями, щоб те, на що всі покладаються, працювало тихо й без сюрпризів.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Підтримувала 10 серверів PostgreSQL і MS SQL — 10 серверів баз даних розгорнуто й підтримувано в справному стані

Підтримувала 10 серверів PostgreSQL і MS SQL Зв'язатися з нами →

Чому це важливо

Bash — мова, що зшиває між собою все на серверах Unix. Вона є майже на кожній машині з Linux, а команди, які інженер набирає в рядку запрошення, уже написані на Bash, тож кілька таких команд, збережених у файлі, стають скриптом, який використовує вже встановлені в системі інструменти. Щоб з'єднати наявні програми в ланцюжок, наприклад зняти дамп бази даних, стиснути його й скопіювати деінде, мало яка мова буде такою ж стислою, бо канали, перенаправлення та коди завершення — її рідна лексика.

Ті самі риси перетворюються на витрати, коли скрипт росте. Кожне значення — це рядок, а арифметика працює лише з цілими числами, тож усе, що має структуру, як-от JSON, дати чи дробові суми, доводиться передавати іншому інструменту. Розгортання без лапок розбиваються на слова, а потім зіставляються з іменами файлів, тож значення з пробілом чи зірочкою змінює зміст команди, в яку потрапляє. Обробку помилок треба вмикати самому: типово скрипт працює далі після того, як команда завершилася з помилкою, а параметри, що це змінюють, `set -e` і `pipefail`, мають винятки, на перелік яких посібнику потрібно кілька абзаців. Робоче правило таке: щойно скрипту знадобилися структури даних, парсер чи набір тестів, він став програмою, і мову на кшталт Python дешевше підтримувати.

Переносність — теж рішення. Debian Policy вимагає, щоб скрипти для `/bin/sh` обмежувалися можливостями POSIX і коротким списком розширень, а скрипт, якому потрібно більше, називав свою оболонку в першому рядку, наприклад `#!/bin/bash`. Для пошуку порушень вона радить `checkbashisms` або запуск у суворішій оболонці, як-от posh, а в разі сумніву — `/bin/bash`. Стандарт рухається повільно: у Bash `pipefail` є з версії 3.0, а POSIX додав його лише у виданні 2024 року.

Оболонка перетворює текст на команди, а отже, стає межею безпеки всюди, куди потрапляє текст ззовні. Вразливість Shellshock, зареєстрована як CVE-2014-6271 у вересні 2014 року, показала, наскільки широкою може бути ця межа. Bash до версії 4.3 включно виконувала команди, що стояли після визначення функції у змінній середовища, тож усе, що копіювало текст зловмисника в середовище перед запуском Bash, ставало способом виконати код на машині: CGI-скрипти під Apache, механізм ForceCommand в OpenSSH і деякі клієнти DHCP. Перше виправлення виявилося неповним, і те, що лишилося, отримало власний ідентифікатор, CVE-2014-7169. Повсякденний урок скромніший: текст ззовні має потрапляти до команди як аргумент, а не як частина рядка, переданого `eval` чи `sh -c` .

Як це виглядає на практиці

У роботі компанії Bash має те місце, яке їй відводить сторінка технологій: Python відповідає за роботу з даними, автоматизацію та сервіси, а Bash з’єднує все це між собою і пишеться так, щоб його було легко читати.

У шести пайплайнах ETL і ELT, які наповнювали Google BigQuery даними зі сховища за IPsec VPN, завантажені архіви містили файли Parquet, CSV і .bak. Скрипти на Bash і Python діставали з них файли й розподіляли кожен за типом і схемою, після чого bcp, psql і SSIS завантажували дані в проміжні схеми, а gzip, Google Cloud Storage та утиліта bq доставляли результат у BigQuery. Автоматизація стиснула щонайменше три години ручної роботи до менше ніж 20 хвилин на весь цикл і перевела синхронізацію даних із щотижневої на щоденну.

Сто автоматизованих критичних резервних копій даних показують такий самий поділ праці. Barman відповідав за PostgreSQL, Google Cloud зберігав копії за межами основного майданчика, а Bash і Python керували всім процесом і перевіряли його: політики зберігання тримали копії актуальними, а перевірки цілісності підтверджували, що кожна з них справді придатна, а не просто існує. Саме таку роботу Bash виконує найкраще: послідовність наявних інструментів, де цінність скрипту полягає в перевірці того, що повернув кожен крок.

Де виникають проблеми

Ім'я файлу з пробілом перетворюється на два аргументи. `$file` без лапок розбивається за пробільними символами, а потім розгортається як шаблон glob, тож `report 2024.csv` доходить до `rm` як два імені, а випадкова `*` захоплює цілий каталог. Беріть у лапки кожне розгортання, як-от `"$file"`, передавайте списки аргументів далі як `"$@"`, зберігайте списки імен у масивах і доручіть решту ShellCheck.

Конвеєр резервного копіювання звітує про успіх, хоча дамп зірвався. Код завершення конвеєра — це код його останньої команди, тож у `pg_dump app | gzip > app.sql.gz` невдалий дамп однаково лишає gzip успішним на тих крихтах, які він отримав. Увімкніть `set -o pipefail`, щоб конвеєр повертав останній ненульовий статус, і перевіряйте результат командою `gzip -t` або, ще краще, пробним відновленням.

Скрипт із `set -e` іде далі після команди, що завершилася з помилкою. Параметр ігнорується в умові `if` чи `while`, у будь-якій частині списку `&&` або `||`, крім останньої, і в будь-якій функції, викликаній із цих місць, тож дамп нижче може зірватися, а скрипт однаково виведе повідомлення про успіх:

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

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

if backup; then # тут set -e всередині backup ігнорується
  echo "резервне копіювання завершено"
fi
```

Споріднена пастка — `local out=$(cmd)`: такий рядок повертає статус `local` і приховує помилку в `cmd`. Усередині функцій додавайте `|| return 1` після кожної важливої команди й оголошуйте змінну окремим рядком перед тим, як присвоїти їй значення.

Скрипт, що працює на одній машині, ламається на іншій. Файл, який починається з `#!/bin/sh`, але використовує перевірки `[[` або масиви, в одній системі запускається під Bash, а в наступній — під dash. Там `[[` — невідома команда, тож `if`, побудований на ній, виводить помилку й переходить до гілки else, а присвоєння масиву є синтаксичною помилкою, яка завершує скрипт. На Mac `/bin/bash` повідомляє версію 3.2.57, яка вийшла раніше за асоціативні масиви й `mapfile`, що з'явилися у 4.0, тож `declare -A` там зупиняється з помилкою про неприпустимий параметр. Указуйте в shebang ту оболонку, яка справді потрібна скрипту, і тестуйте на найстарішій версії, на якій він має працювати.

Рядок очищення видаляє набагато більше, ніж мав би. Якщо `build_dir` не задано, `rm -rf "$build_dir"/*` розгортається в `rm -rf /*`. `set -u` робить незадану змінну помилкою, а запис `"${build_dir:?}"` зупиняє неінтерактивний скрипт щоразу, коли змінна не задана або містить порожній рядок.

Що почитати далі

- GNU Bash Reference Manual — посібник до версії 5.3: розгортання, лапки, вбудована команда `set` і точний перелік випадків, коли `set -e` ігнорується.
- Shell Command Language — переносна мова `sh` у POSIX.1-2024, межа для будь-якого скрипту з `#!/bin/sh`.
- Debian Policy: Scripts — що має підтримувати `/bin/sh` у Debian, коли скрипт мусить просити Bash на ім'я, і поради щодо `set -e`.
- Bash NEWS — перелік нових можливостей від супровідника, випуск за випуском, щоб перевірити, якої версії потребує конструкція.
- CVE-2014-6271 — запис NVD про Shellshock із переліченими шляхами атаки та ідентифікатором подальшої вразливості.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Bash?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/bash/