Engineer

Ядро операційної системи

основа операційної системи

Що це таке

Ядро операційної системи — це програма в самому її центрі, яка працює з повними привілеями процесора і контролює доступ до обладнання. Воно вирішує, яка програма отримає процесор наступною, розподіляє пам’ять і тримає пам’ять кожної програми окремо від решти, читає й записує диски через файлові системи, передає пакети через мережевий стек і забезпечує виконання правил, які визначають, хто має право доступу до якого файла, порту чи пристрою. Усе інше працює в просторі користувача з меншими привілеями й просить ядро діяти від свого імені.

Такі запити називаються системними викликами, і власний посібник Linux називає системний виклик основним інтерфейсом між застосунком і ядром. Програми рідко роблять їх напряму: вони викликають функції-обгортки з бібліотеки C, як-от glibc, які передають аргументи далі й перетворюють відповідь ядра на значення, що повертається, і код помилки. Коли програма зупиняється з повідомленням «Permission denied», це формулювання зазвичай є відповіддю ядра на один із таких викликів, перекладеною бібліотекою.

Ядра різняться тим, скільки тримають усередині. Linux тримає драйвери пристроїв, файлові системи й мережевий стек у самому ядрі та може довантажувати інші як модулі під час роботи; таку будову зазвичай називають монолітною. Кожен драйвер тоді працює з повними привілеями ядра. Мікроядро залишає собі лише необхідний мінімум. seL4 надає механізми керування доступом до фізичного адресного простору, перериваннями та процесорним часом, а драйвери, файлові системи й мережеві стеки запускає як звичайні програми в режимі користувача; його перевірена 64-бітна версія для RISC-V має близько 10 000 рядків коду, а

формальне доведення показує, що реалізація мовою C робить те, що каже її специфікація. Ядро macOS, XNU, — гібрид, що поєднує мікроядро Mach із частинами FreeBSD.

Оновлення ядра зазвичай набуває чинності лише тоді, коли машина завантажується з новим ядром, а номер версії ядра, наприклад 6.12, окремий від номера випуску дистрибутива навколо нього.

Застосуйте поняття до рішення

Застосуйте «Що таке ядро операційної системи?» 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

Для бізнесу ядро вирішує три питання: наскільки насправді розділені робочі навантаження, скільки простою коштує виправлення безпеки і де зберігається правда про машину.

Спершу ізоляція. Контейнер — це набір звичайних процесів, які ядро помістило в простори імен, що дають кожній групі власний погляд на ідентифікатори процесів, мережеві інтерфейси, точки монтування й користувачів, і в контрольні групи, що розподіляють процесорний час, пам’ять і дисковий ввід-вивід. Усі контейнери на хості працюють на одному ядрі хоста, тож вада в цьому ядрі є вадою під усіма ними водночас, а образ із власними бібліотеками однаково залежить від тих можливостей ядра, які є на хості. Віртуальна машина завантажує власне ядро. Для сервісів однієї команди контейнери — розумна межа; для клієнтів, які не довіряють одне одному, дайте кожному власне ядро.

Далі виправлення. Оновлений пакунок ядра на диску нічого не змінює, доки машина не завантажиться з ним, і на сервері ціною стає саме цей перезапуск. Живе латання існує тому, що користувачі, за словами документації ядра, неохоче перезавантажують систему: воно перенаправляє окремі функції на виправлені версії, поки система працює. Так можна залатати лише функції,

які ядро здатне трасувати, тож живе латання дає час для термінового виправлення, а не замінює перезавантаження. План із регулярним, відрепетируваним перезавантаженням коштує менше, ніж план, який припускає, що перезавантаження ніколи не знадобиться.

Нарешті, живий стан зберігає саме ядро. Файл брандмауера описує правила; що станеться з пакетом, вирішують правила у фільтрі пакетів ядра. Файл у `/etc/sysctl.d` описує параметр; чинним є значення в `/proc/sys` . Служба може звітувати, що вона справна, незалежно від того, чи її дія взагалі дійшла до ядра. Перевірки, що читають власний стан ядра, ловлять відмови, які огляд файлів налаштувань визнав би справними.

Як це виглядає на практиці

Огляд захищеності робочого хоста в серпні 2026 року з'ясовував, чи працює його захист SSH від перебору паролів. Кожен засіб був налаштований і з'являвся в кожному звіті, тож огляд перевірив їх проти того, що ядро насправді робить із пакетом, а не проти файлів, які описують, що мало б статися. Усі три були бездіяльними від дня створення хоста. Профіль блокування (jail) стежив за портом 22, коли SSH уже перенесли на порт 1986. Обмеження частоти на брандмауері стояло нижче ширшого правила, яке збігалося першим. Дія блокування викликала програму фільтра пакетів, яку система пакунків лише рекомендує, тож профіль блокування запускався, рахував й ухвалював рішення, а відмовляв саме тоді, коли намагалася заблокувати. Звідси з'явилися два правила: засіб безпеки отримує перевірку-твердження, а не коментар, а брандмауер перевіряють за позицією правил, а не за їхньою наявністю.

Шлях блокування потім отримав тест, який виконується щоразу, коли застосовують конфігурацію. Кожен прогін блокує адресу з блоку, зарезервованого для документації, зчитує з ядра отримане правило у фільтрі пакетів, а потім знімає блокування на кроці прибирання, що виконується за будь-якого результату. Саме зчитування робить це тестом: власний статус служби повідомив би про успіх для блокування, яке не створило правила. Зарезервована адреса не належить справжній мережі, тож блокування, яке лишив невдалий прогін, нікого не відріже. Ціна — кілька секунд на прогін і правило брандмауера, яке щоразу записують у робоче середовище й відкликають.

Хост на 464 МБ, де не було місця для стека метрик, отримав сім ролей лише для читання, що подають живу машину в Markdown; його знімок фактів обчислює завантаження процесора з двох зразків власного лічильника ядра. Звіт безпеки друкує правила брандмауера в порядку, в якому вони спрацьовують, поруч із живими правилами блокування. Для всіх семи ролей діє одне правило: друкується лише те число, яке прогін виміряв. Ці звіти виявили два мертві засоби безпеки й незадекларований порт, що слухає.

Де виникають проблеми

Параметр ядра, змінений вручну, повертається до старого значення після наступного перезавантаження. `sysctl -w` записує значення лише в ядро, що працює, тож воно живе до перезапуску. Зворотна помилка так само поширена: файл у `/etc/sysctl.d` читається під час завантаження, тож зміна непомітно чекає на якийсь пізніший перезапуск. Запишіть параметр у файл, завантажте його зараз і прочитайте живе значення:

```
echo 'vm.swappiness = 10' | sudo tee /etc/sysctl.d/90-local.conf
sudo sysctl --system
sysctl vm.swappiness
```

Графік процесора лишається пласким під час сплеску навантаження. Рядок `cpu` у `/proc/stat` рахує час від завантаження в одиницях USER_HZ (соті частки секунди на більшості архітектур), тож одне зчитування дає середнє від завантаження, а воно на машині, що працює місяцями, ледве рухається. Прочитайте рядок двічі й поділіть різниці:

```
head -1 /proc/stat; sleep 5; head -1 /proc/stat
```

Зайнятий час — це приріст user, nice, system, irq і softirq, час простою — приріст idle та iowait, а завантаження — це зайнятий час, поділений на суму зайнятого часу й часу простою. Посібник називає iowait ненадійним, тож вважайте його підказкою, а не підставою для тривоги.

Звіт про збій повертається з проханням відтворити його на незаплямованому ядрі. Ядро позначає себе заплямованим (tainted), коли стається щось, що може мати значення для подальшого розслідування, наприклад завантаження пропрієтарного, зібраного поза деревом ядра або

непідписаного модуля, а документація попереджає, що розробники часто ігнорують звіти про помилки із заплямованих ядер. Позначка лишається до перезавантаження, навіть після вивантаження модуля. Прочитайте її, перш ніж подавати звіт:

```
cat /proc/sys/kernel/tainted
```

Нуль означає, що ядро не заплямоване; будь-яке інше число — це сума прапорців, серед них `1` для пропрієтарного модуля, `4096` для модуля, зібраного поза деревом ядра, і `8192` для непідписаного. Відтворіть збій після завантаження без модуля, який установив прапорець.

Нова мережева карта з'являється в списку обладнання, а система не пропонує для неї інтерфейсу. Драйвери — частина ядра, і ядру, старішому за обладнання, може бракувати драйвера для нього. `lspci -k` показує для кожного пристрою драйвер ядра, що використовується, і модулі, здатні з ним працювати; пристрій без рядка «Kernel driver in use» не має прив'язаного драйвера. Установіть новішу серію ядра з дистрибутива, а не збирайте модуль постачальника вручну: модуль поза деревом ядра доводиться перезбирати для кожного нового ядра, а його завантаження заплямовує ядро.

Що почитати далі

- The Linux kernel user's and administrator's guide — власний посібник проєкту ядра з експлуатації Linux.
- syscalls(2) — усі системні виклики Linux із версією ядра, яка запровадила кожен із них.
- namespaces(7) — види просторів імен і те, як із них будують контейнери.
- proc_stat(5) — кожне поле `/proc/stat`, разом із застереженням щодо iowait.
- Tainted kernels — кожен прапорець і те, як розшифрувати число.
- seL4 FAQ — що мікроядро лишає поза собою і на яких припущеннях стоїть доведення seL4.

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 ядро операційної системи?» 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/kernel/