Engineer

Systemkärna

kärnan i ett operativsystem

Vad det är

En systemkärna, oftast bara kallad kärnan, är det program i centrum av ett operativsystem som körs med processorns fulla privilegier och styr åtkomsten till hårdvaran. Den bestämmer vilket program som får köra härnäst, delar ut minne och håller varje programs minne åtskilt från de andras, läser och skriver diskar genom filsystem, flyttar paket genom nätverksstacken och upprätthåller reglerna för vem som får komma åt vilken fil, port eller enhet. Allt annat körs i användarrymden med färre privilegier och ber kärnan att agera åt det.

De förfrågningarna är systemanrop, som Linux-manualen beskriver som det grundläggande gränssnittet mellan en applikation och Linux-kärnan. Program gör dem sällan för hand: de anropar omslag i ett C-bibliotek som glibc, som skickar argumenten vidare och gör om kärnans svar till ett returvärde och en felkod. När ett program stannar med ”Permission denied” är den formuleringen oftast kärnans svar på ett sådant anrop, översatt av biblioteket.

Kärnor skiljer sig åt i hur mycket de håller inne. Linux håller drivrutiner, filsystem och nätverksstacken i själva kärnan och kan ladda in mer som moduler medan den kör, en design som brukar kallas monolitisk. Varje drivrutin körs då med kärnans fulla privilegier. En mikrokärna behåller bara det minsta möjliga. seL4 tillhandahåller mekanismer för att styra åtkomsten till det fysiska adressutrymmet, avbrott och processortid, och kör drivrutiner, filsystem och nätverksstackar som vanliga program i användarläge; den verifierade 64-bitarsversionen för RISC-V omfattar ungefär 10 000 rader kod, med ett formellt bevis för att implementationen i C gör det som specifikationen säger. Kärnan i macOS, XNU, är en hybrid som kombinerar mikrokärnan Mach med delar av FreeBSD.

En uppdatering av kärnan får normalt verkan först när maskinen startar med den, och kärnans versionsnummer, t.ex. 6.12, är skilt från versionsnumret på distributionen runt omkring den.

Använd begreppet i ett beslut

Använd ”Vad är en systemkärna?” 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

För ett företag avgör kärnan tre frågor: hur långt isär arbetslasterna egentligen ligger, vad en säkerhetsrättning kostar i driftstopp, och var sanningen om en maskin finns.

Först isoleringen. En container är en uppsättning vanliga processer som kärnan har placerat i namnrymder, som ger varje grupp en egen bild av process-id:n, nätverksgränssnitt, monteringar och användare, och i kontrollgrupper (cgroups), som fördelar processortid, minne och diskens in- och utdata. Varje container på en värd körs på värdens enda kärna, så en brist i den kärnan är en brist under dem alla på en gång, och en avbildning som tar med sina egna bibliotek förlitar sig ändå på de kärnfunktioner som värden har. En virtuell maskin startar en egen kärna. För ett teams egna tjänster är containrar en rimlig gräns; för kunder som inte litar på varandra bör varje kund få en egen kärna.

Sedan rättningarna. Ett patchat kärnpaket på disken ändrar ingenting förrän maskinen startar med det, och på en server är den omstarten kostnaden. Patchning i drift, som kärnan kallar livepatch, finns eftersom användare enligt kärnans dokumentation drar sig för att starta om: den leder om enskilda funktioner till rättade versioner medan systemet kör. Bara funktioner som kärnan kan spåra går att patcha på det sättet, så livepatch köper tid för en brådskande rättning snarare än att ersätta omstarter. En plan med regelbundna, inövade omstarter kostar mindre än en som utgår från att omstarten aldrig behöver ske.

Sist håller kärnan det aktuella tillståndet. En brandväggsfil beskriver regler; reglerna i kärnans paketfilter avgör ett pakets öde. En fil i `/etc/sysctl.d` beskriver en inställning; värdet under `/proc/sys` är det som gäller. En tjänst kan rapportera sig som frisk oavsett om dess åtgärd någonsin nådde kärnan. Kontroller som läser kärnans eget tillstånd fångar fel som en granskning av konfigurationsfiler skulle släppa igenom.

Så ser det ut i praktiken

En härdningsgranskning av en produktionsserver i augusti 2026 frågade om serverns skydd mot brute force-attacker via SSH fungerade. Alla tre var konfigurerade och syntes i varje rapport, så granskningen prövade dem mot vad kärnan faktiskt gör med ett paket i stället för mot filerna som beskriver vad som borde hända. Sett från kärnans håll hade alla tre varit verkningslösa ända sedan servern sattes upp. Varje spärr som jailen skrev gällde port 22, men SSH lyssnade sedan den första härdningen på port 1986. Brandväggens användarregler prövas uppifrån och ned och den första träffen vinner, så hastighetsbegränsningen, placerad under en bredare tillåtande regel, var död kod som ändå syntes i varje statuslistning. Spärråtgärden anropade ett program för paketfiltrering som paketsystemet bara rekommenderar, så jailen startade, räknade och fattade beslut och fallerade sedan i just det ögonblick den försökte spärra. Granskningen lämnade två regler efter sig: ett säkerhetsskydd bevisas med en kontroll i stället för att beskrivas i en kommentar, och en brandvägg bedöms efter var reglerna står i ordningen, inte efter att de finns.

Spärrflödet fick sedan ett test som körs varje gång konfigurationen tillämpas. Varje körning spärrar en adress ur ett block som är reserverat för dokumentation, läser tillbaka den resulterande regeln från kärnans paketfilter och häver sedan spärren i ett städsteg som körs oavsett utfall. Det är återläsningen som gör det till ett test: spärrtjänstens egen status skulle rapportera framgång för en spärr som inte gav upphov till någon regel. En reserverad adress tillhör inget verkligt nätverk, så en spärr som lämnas kvar efter en misslyckad körning stänger inte ute någon. Kostnaden är några sekunder per körning och en brandväggsregel som skrivs till produktion och tas bort varje gång.

En värd på 464 MB utan plats för en stack för systemmätvärden fick sju roller som bara läser och återger maskinens aktuella tillstånd som Markdown; ögonblicksbilden av värdens fakta beräknar processorbelastningen ur två avläsningar av kärnans egen räknare. Säkerhetsrapporten skriver ut brandväggsreglerna i matchningsordning, bredvid den aktiva regeluppsättningen

för spärrar, och alla sju rollerna lyder under samma krav: varje siffra de visar har uppmätts av just den körningen. Bland det rapporterna hittade fanns två döda säkerhetsåtgärder och en lyssnande port som saknade deklaration.

Var det går fel

En kärninställning som ändrats för hand återgår efter nästa omstart. `sysctl -w` skriver värdet bara i den körande kärnan, så det håller till nästa omstart. Det omvända misstaget: en fil under `/etc/sysctl.d` läses vid uppstart, så ändringen väntar, utan att säga något, på någon senare omstart. Skriv inställningen till en fil, ladda den nu och läs tillbaka det gällande värdet:

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

En graf över processorn ligger platt genom en lasttopp. Raden `cpu` i `/proc/stat` räknar tid sedan uppstart, i enheter av USER_HZ (hundradels sekunder på de flesta arkitekturer), så en enda avläsning ger genomsnittet sedan uppstart, som knappt rör sig på en maskin som har varit igång i månader. Läs två gånger och dividera skillnaderna:

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

Upptagen tid är förändringen i user, nice, system, irq och softirq, ledig tid är förändringen i idle och iowait, och belastningen är upptagen tid delat med summan av upptagen och ledig tid. Manualen kallar iowait opålitlig, så behandla den som en fingervisning snarare än ett larm.

En kraschrapport kommer tillbaka med en begäran om att återskapa felet på en kärna som inte är tainted. Kärnan markerar sig själv som tainted, alltså befläckad, när något händer som kan spela roll för en senare utredning, t.ex. att en proprietär, extern eller osignerad modul laddas, och dess dokumentation varnar för att utvecklare ofta bortser från felrapporter från befläckade kärnor. Markeringen ligger kvar till omstart, även sedan modulen har laddats ur. Läs den innan du skickar in rapporten:

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

Noll betyder obefläckad; varje annat tal är en summa av flaggor, bland dem `1` för en proprietär modul, `4096` för en modul utanför kärnans källträd och `8192` för en osignerad. Återskapa felet efter att ha startat utan den modul som satte flaggan.

Ett nytt nätverkskort syns i hårdvarulistan men systemet erbjuder inget gränssnitt för det. Drivrutiner är en del av kärnan, och en kärna som är äldre än hårdvaran kan sakna en. `lspci -k` visar varje enhets drivrutin i bruk och de moduler som kan hantera den; en enhet utan en rad för drivrutin i bruk har ingen bunden. Installera en nyare kärnserie från distributionen i stället för att bygga leverantörens modul för hand: en modul utanför källträdet måste byggas om för varje ny kärna, och att ladda den gör kärnan befläckad.

Läs mer

- The Linux kernel user's and administrator's guide — kärnprojektets egen handbok för att köra Linux.
- syscalls(2) — varje systemanrop i Linux, med den kärnversion som införde det.
- namespaces(7) — slagen av namnrymder och hur containrar byggs av dem.
- proc_stat(5) — varje fält i `/proc/stat`, varningen om iowait inräknad.
- Tainted kernels — varje taint-flagga och hur talet avkodas.
- seL4 FAQ — vad en mikrokärna lämnar utanför, och antagandena bakom beviset för seL4.

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 en systemkärna?” 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/kernel/