Engineer

macOS

skrivbordsoperativsystemet från Apple

Vad det är

macOS är operativsystemet som Apple levererar med varje Mac. Det användaren ser är skrivbordet, Finder och de medföljande apparna; under dem ligger ett Unix-system med terminal, skal och de kommandoradsverktyg som en utvecklare känner igen från andra system. Apple bygger systemet för sin egen hårdvara och säljer de två tillsammans, så den som väljer macOS väljer i praktiken också en Mac.

Grunden heter Darwin, och Apple publicerar dess källkod. Dess systemkärna, XNU, är en hybrid: Mach-kärnan från Carnegie Mellon University kombinerad med komponenter från FreeBSD och IOKit, ett ramverk för drivrutiner skrivet i C++. Namnet utläses ”X is Not Unix”. Stora delar av användarlandet kommer också från FreeBSD, och manualsidorna för `sed` och `date` på en Mac beskriver FreeBSD-versionerna av de verktygen. Sedan macOS 10.15 Catalina är zsh standardskalet.

Under systemet har hårdvaran bytts ut på senare år. I slutet av 2020 började Apple flytta Mac från processorer från Intel till egna chip, Apple silicon, och i dag har varje Mac-serie, från MacBook Air till Mac Pro, modeller med Apple silicon. De chippen är arm64, och det får betydelse i samma stund som programvara som byggts på en Mac ska köras på en x86-64-server.

På bärbara och stationära datorer ställs macOS mot Windows och Linux.

Använd begreppet i ett beslut

Använd ”Vad är macOS?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Laptopen, skrivaren, inloggningen och e-posten, fixade snabbt och på ett begripligt språk, utan att någon behöver känna sig dum.

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. Etablerade en avdelning för teknisk support — 10,000 kunder som fick hjälp av en supportavdelning byggd från grunden

Etablerade en avdelning för teknisk support Hör av er →

Varför det spelar roll

För ett företag som skriver programvara har macOS två roller som lätt flyter ihop. Det är maskinen som många utvecklare arbetar på, och det är en plattform som kunder kör programvara på.

Som arbetsstation skiljer sig en Mac från Linux-servrarna den driftsätter till på tre punkter. Kommandoradsverktygen kommer från FreeBSD och inte från GNU, så samma flagga kan betyda olika saker i ett skript. Processorn är arm64, medan servern kan vara x86-64. Och det finns ingen Linux-kärna, medan Linux-containrar delar systemkärna med sin värddator. Verktyg som kör dem på en Mac startar därför först en virtuell Linux-maskin. Apple har ett eget verktyg, `container`, som kräver Apple silicon och macOS 26 och ger varje container en egen lätt virtuell maskin. Detta utesluter ändå inte en Mac. Det betyder att man testar på systemet man driftsätter till, i stället för att lita på en grön körning på en bärbar dator.

Som plattform väcker macOS frågan om hur man bygger för den. Ett plattformsoberoende skal, alltså en kodbas med en inbäddad webbläsarmotor som levereras till Windows, Linux och Mac, håller flera system i takt till låg kostnad. På en Mac bär det dock med sig en körmiljö för en webbläsare som en native applikation inte har någon användning för, och det ritar egna kontroller i stället för systemets. En native applikation i Swift och SwiftUI använder systemets kontroller och material och följer användarens inställningar för utseende, accentfärg, kontrast, textstorlek och minskad rörelse utan extra kod. Priset är en separat klient och personer som kan plattformen. Native lönar sig när kunderna sitter vid en Mac.

Att leverera till Mac innebär också att passera en grind. Programvara som distribueras utanför Mac App Store signeras med ett Developer ID-certifikat och skickas till notarietjänsten hos Apple, en automatisk granskning som letar efter skadligt innehåll och problem med kodsigneringen och som är skild från App Review. Sedan macOS 10.15 måste all programvara som byggts efter den 1 juni 2019 och distribueras med Developer ID vara notariserad. Apple skickar tillbaka en biljett (”ticket”) som häftas vid programvaran och publicerar samma biljett på nätet, där Gatekeeper letar efter den när användaren öppnar appen första

gången. Certifikatet ingår i Apple Developer Program, som kostar 99 dollar om året. Räkna in medlemskapet och signeringssteget redan inför den första releasen.

Så ser det ut i praktiken

En meddelandeprodukt hade på macOS en skrivbordsupplevelse som byggde på ett plattformsoberoende skal och som varken såg ut eller betedde sig som en Mac-app. Produktens vältestade kärna i Rust exponerar ett C-gränssnitt, och Swift kan anropa C, så en native klient i Swift och SwiftUI byggdes direkt på det gränssnittet. Rust-kärnan länkas in som ett statiskt arkiv från en låst uppströmsrevision, vilket håller bygget reproducerbart, och en tunn wrapper äger varje pekare som korsar gränsen mellan språken. Klienten startar som en Mac-app, använder plattformens egna material och kontroller och bär inte på någon inbäddad webbläsare. Sidan säger rakt ut vad som återstår: det finns ingen notariserad distribution och ingen uppdateringskanal, så det är en fungerande app snarare än en produkt som andra kör. Notariseringen ovan är steget mellan de två.

Plattformens utseende kan upprätthållas i stället för att imiteras. Ett designsystem med 48 namngivna tokens i sex grupper bygger färg och material på plattformens semantiska färger och glasmaterial i stället för fasta värden. Därför följer gränssnittet systemets utseende, accentfärgen och kontrastinställningarna utan någon kod som bevakar dem, och textstilarna skalas med användarens storleksinställning. En linter avvisar ett rått tal där en token hör hemma, en hårdkodad teckenstorlek och en animation som struntar i inställningen för minskad rörelse. Sidan anger också gränsen: tokens gör ett gränssnitt konsekvent, och det är inte detsamma som att göra det bra.

Komprimerat minne och swap låter en macOS-app växa långt förbi maskinens fysiska minne. En incidentbeskrivning dokumenterar diagnostikrapporter med 111 GB komprimerade sidor på en maskin med 36 GB den 2026-09-01 och ett separat stickprov av minnesförbrukningen som visade en ökning med ungefär 41 MB per sekund under en återförsöksslinga för anslutningen. Händelsevägen behöll core-logghändelser i en obegränsad huvudström och kopierade dem till prenumeranter som inte använde de händelserna. Buffertarna begränsades, prenumerationerna filtrerades efter händelsetyp och loggningen fick en taktbudget. En jämförelse av fullständiga testkörningar registrerade att 610 996 core-loggrader blev 1 411; det är mätningar från testkörningar, inte en mätning av minnesförbrukningen efter korrigeringen eller ett prestandatest i produktion.

Avvägningen är uttrycklig: upprepade diagnostiska händelser kan kastas bort och berört tillstånd kan läsas in på nytt i stället för att behålla ett obegränsat antal händelser. Separata tidsgränser vid avstängning behövde en separat korrigering.

Var det går fel

Aktivitetskontroll visar nästan inget ledigt minne på en Mac som går alldeles utmärkt. Programmet heter Activity Monitor på engelska. macOS cachar nyligen använda filer i minne som annars skulle stå oanvänt, och när minnet fylls komprimeras inaktiva appar för att ge plats åt de aktiva. Att använt minne ligger nära det fysiska och att det komprimerade minnet är stort är alltså normalt, och enligt Apple gör ledigt minne inte nödvändigtvis en Mac snabbare. Läs i stället diagrammet Minnestryck: grönt betyder att RAM-minnet används effektivt, gult att datorn så småningom kan behöva mer och rött att den behöver mer. Swap är den tydligaste signalen:

```
sysctl vm.swapusage
vm_stat | grep -E '^Swap(ins|outs)'
```

Räknarna i `vm_stat` räknar från uppstart, så jämför två avläsningar med en minuts mellanrum. Swapouts som fortsätter att stiga betyder att arbetet har vuxit ur RAM-minnet.

Ett skript som fungerar på en Mac stannar med ett märkligt fel på Linux-servern, eller tvärtom. Den FreeBSD-härledda `sed` på en Mac tolkar ordet efter `-i` som ändelsen för en reservkopia, så GNU-vanan `sed -i 's/a/b/' file` tar uttrycket som ändelse och filnamnet som skript. Mac-vanan `sed -i '' …` får i sin tur GNU-versionen av `sed` att läsa uttrycket som ett filnamn. En ändelse som sitter ihop med flaggan fungerar på båda:

```
sed -i.bak 's/^port=80$/port=8080/' app.conf && rm app.conf.bak
```

Den varaktiga lösningen är att köra skripten i CI på samma system som de möter i produktion.

Gatekeeper stoppar ett bygge som delats utanför teamet när mottagaren öppnar det. Appen signerades för utveckling, eller signerades men notariserades aldrig. Signera den med ett Developer ID-certifikat, med härdad körmiljö (”hardened runtime”) påslagen och en säker tidsstämpel, skicka den till notarietjänsten och häfta biljetten vid appen så att Gatekeeper hittar den även utan nätverk. Ett ZIP-arkiv kan notariseras men kan inte bära en biljett, så det är appen inuti som får den:

```
ditto -c -k --keepParent App.app App.zip
xcrun notarytool submit App.zip --keychain-profile notary --wait
xcrun stapler staple App.app
```

En container-image går på utvecklarens Mac och dör på servern med `exec format error`. Docker bygger som standard för plattformen där BuildKit-daemonen kör, och på en Mac med Apple silicon kör den daemonen i en virtuell arm64-maskin. En container delar värdens systemkärna, så utan emulering kör en x86-64-värd bara x86-64-kod. Kontrollera servern med `uname -m` och bygg sedan för den med `docker build --platform linux/amd64`, eller bygg i CI på samma processortyp som servern.

Läs mer

- Mac User Guide — handboken för macOS från Apple själva, uppdaterad för varje aktuell version.
- Check if your Mac needs more RAM — diagrammet Minnestryck och vad dess tre färger betyder.
- Notarizing macOS software before distribution — Developer ID, den härdade körmiljön och hur Gatekeeper använder biljetten.
- Multi-platform builds — varför värdar med arm64 och amd64 behöver olika images, och hur ett enda bygge tar fram båda.
- container — verktyget från Apple som kör Linux-containrar på en Mac, var och en i en egen lätt virtuell maskin.
- XNU — källkoden till systemkärnan i macOS, med en kort redogörelse för hur Mach, FreeBSD och IOKit hänger ihop.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv enheten eller tjänsten, observerade symtom och uppgifterna som måste bevaras. Vi kan fastställa det säkraste användbara undersökningssteget.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är macOS?” 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/macos/