**Engineer** Software

# Har indført Swift 6 strict concurrency over C

Har indført Swift 6's complete strict concurrency uden actors og bygget bro fra en blokerende C-hændelsesløkke til main actor gennem én producent, én forbruger og én rækkefølge — efter at have fastslået, at en opgave pr. hændelse mister den rækkefølge, grænsefladen afhænger af.

september 2026

## strict concurrency med én rækkefølge og ingen actors

**Situation.** Swift 6's fuldstændige strenge samtidighedstjek gør datakapløb til kompileringsfejl i stedet for lejlighedsvise nedbrud. Det er, når det skal indføres op mod et C-bibliotek, at det bliver svært: kernens hændelsesløkke er et blokerende kald, der skal køre uden for hovedtråden for evigt, og de værdier, den leverer tilbage, er pointere uden nogen som helst samtidighedsgarantier. Oversætteren kan ikke ræsonnere om noget af det og vil afvise alt, indtil grænsen er beskrevet udtrykkeligt.

**Opgave.** Fuldstændigt tjek skulle slås til uden nogen nødudgange, hvilket betød at udforme overgangen fra en blokerende C-løkke til hovedaktøren i stedet for at annotere sig uden om den.

**Handling.** Den nærliggende fremgangsmåde er en aktør pr. delsystem, og den blev forkastet på grundlag af måling og ikke smag. At starte en opgave pr. indkommende hændelse lader runtime-miljøet planlægge dem i vilkårlig rækkefølge, og kernens hændelsesstrøm er ordnet — en besked-ændret-hændelse, der overhaler den besked-oprettet-hændelse, den henviser til, frembringer en grænseflade, der viser en redigering af noget, der endnu ikke findes. At aktører er reentrante, gør dette værre, ikke bedre, for en aktør kan suspendere midt i en metode og behandle et andet kald. Det, der erstattede den, er bevidst enkelt: én producenttråd, der ejer den blokerende løkke, én forbruger, én kø imellem dem og et enkelt hop over på hovedaktøren til sidst. Rækkefølgen bevares, fordi der er præcis én vej, og intet overhaler noget. De usikre typer, der krydser den grænse, er pakket ind i typer, hvis trådsikkerhed

hævdes ved indpakningen i stedet for at blive taget for givet, og påstanden er dokumenteret med, hvorfor den holder — pointeren ejes af én tråd og kopieres, før den overdrages.

**Resultat.** Programmet kompilerer under fuldstændigt strengt samtidighedstjek uden undertrykkelser, og hændelsesrækkefølgen er en strukturel egenskab og ikke et håb. Prisen er, at designet er mindre parallelt, end det kunne være: alt ledes gennem én forbruger, og hvis den forbruger nogensinde bliver en flaskehals, vil rettelsen kræve, at man på ny udleder, hvilke hændelser der trygt kan omordnes, hvilket er præcis den analyse, dette undgik.

### Brug erfaringen i jeres projekt

Brug »Har indført Swift 6 strict concurrency over C« som udgangspunkt for en lignende udfordring i jeres organisation. Ydelserne bag arbejdet kan også bruges i jeres projekt. Systemets form lagt fast på papir først, så delene passer sammen, og prisen er kendt, før nogen skriver kode.

**Se vores ydelser** Kontakt os →

### Se bag resultatet

Læs om situationen, beslutningerne og resultatet i eksemplet. Vi kan drøfte, hvad der passer til jeres vilkår, og hvad der kræver en anden tilgang. Har indført Swift 6 strict concurrency over C — strict concurrency med én rækkefølge og ingen actors

**Se vores ydelser** Kontakt os →

## Vil du have samtidighed, compileren kan bevise er sikker?

Tal om jeres projekt med den ingeniør, der har ansvaret for leverancen. Vi aftaler omfang, prisgrundlag og overdragelse skriftligt, før arbejdet begynder.

Rolle i opgaven: macOS-applikationsudvikler. Det trækker på Løsningsarkitektur, Backend-udvikling, Frontend-udvikling, Driftssikkerhed & backup, Optimering af ydeevne og Test & QA. Og når I alligevel er her, er vi ret dygtige til Platform- & løsningsarkitektur, Backend- & API-udvikling og Teknisk ledelse & rådgivning.

**Kontakt os** Se vores ydelser

## Ét stykke af **en længere fortegnelse** — og det hele ligger her på sitet.

Hver post er skrevet på samme måde: situationen, opgaven, hvad vi gjorde, og hvad der ændrede sig.

**Se hele fortegnelsen**

### Definér den ønskede forbedring

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

**Se, hvad det koster** Kontakt os →

# Vil du have samtidighed, compileren kan bevise er sikker?

Spørg til beslutningerne bag »Har indført Swift 6 strict concurrency over C«. I taler med mennesker, der kan forklare arbejdet og vurdere jeres eget projekt.

**Se vores ydelser**

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

Tag denne side med dig

Visitkort

**Lavet med kærlighed af D.**

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

**Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.**

**Udskrevet fra https://software.engineer.company/da/portfolio/adopted-swift-6-strict-concurrency-over-c/**