Hur delas CPU?

Del 1 av de tre viktigaste resursfördelningsdelarna.

Vinster med att göra saker samtidigt

- Multiprogramming
  - Nyttja resurser effektivt genom att exekvera tillräckligt med processer för att hålla varje resurs upptagen hela tiden. Behöver en process vänta på något växlas till en ny som är redo att jobba.
- Time share
  - Genom att ofta växla mellan flera processer kan man få det att se ut som att de exekverar samtidigt.

Vårt produktionsbolag igen

Resurser:

- En operatör som kan utföra manuell bearbetning av produktämnet, eller flytta ämnet mellan olika maskiner
- Automatisk bearbetning av produktämnet med hjälp av 3 maskiner, A, B och C

Problem: Begäran om tillverkning om olika produkter kommer in på löpande band. Alla produkter är olika, men kan tillverkas med de tre maskinerna, och en del manuell bearbetning. Utför tillverkningen så snabbt och effektivt som möjligt.

Produkt 1

1. Bearbetas i maskin A i 3 minuter
2. Bearbetas manuellt i 2 minuter
3. Bearbetas i maskin C i 5 minuter
4. Bearbetas manuellt i 1 minut

Produkt 2

1. Bearbetas manuellt i 10 minuter
2. Bearbetas i maskin B i 5 minuter
3. Bearbetas i maskin A i 4 minuter
4. Bearbetas manuellt i 4 minuter

Produkt 3

1. Bearbetas i maskin A i 2 minuter
2. Bearbetas i maskin B i 2 minuter
3. Bearbetas i maskin C i 2 minuter
4. Bearbetas i maskin D i 2 minuter

Produkt 4

1. Bearbetas manuellt i 15 minuter

Vad måste operatören hålla reda på?

…

Vad måste operatören hålla reda på?

- Vilken produkt håller han på med just nu?
- Vilka produkter har han påbörjat tillverkning av?
- Hur långt har varje påbörjad produkt hunnit?
- Vilka produkter är redo att bearbetas vidare?

- Har han fler tillverkningar av samma produkt igång samtidigt?

Operatörens arbetsordning

- När tillverkning av ett produktexemplar startas tas en kopia på listan över steg som behöver utföras.
- Vartefter steg blir klara stryks den bearbetning som just avslutats från listan.
- Nästa steg påbörjas. Listan placeras vid den maskin som används. Är maskinen upptagen läggs listan på hög vid den maskinen.
- När en maskin signalerar att den är klar stryks det steget från listan, och nästa bearbetning i den maskinen fortsätter genom att ta nästa lista från kön.
- Operatören har liksom varje maskin en egen kö med produktlistor som väntar på manuell bearbetning.
- När sista steget stryks flyttas listan till en pärm med ”avslutat jobb”, och produkten är klar.

I operativsystemet är CPU operatör

Program

- En samling instruktioner lagrade på disk som tillsammans utför en bestämd uppgift.
- Består av:
  - exekverbar binärkod
  - en datamängd med initialvärden
  - en datamängd utan initialvärden

Jämför: Produkt, Bok

Process

- En instans av ett program som exekverar. Aktivitet pågår.
- Består av:
  - Allt som är ett program
  - Resurser som skapats eller allokerats under körning, t.ex. filer, minne, prioritet, och minst en tråd som håller reda på exekveringsstack och processorregister när processen tar paus
- Alla resurser tillhör endast processen, de är inte tillgängliga för andra processer.
- Operativsystemet kontrollerar när processen får exekvera, och när och hur den får tillgång till begärda resurser.
- Processen kontrollerar hur den styr sin exekveringstid internt.

Jämför: Produkt under tillverkning, Läsning av bok

Tråd

- Minsta ”exekveringsenheten”. Håller reda på var i en sekvens instruktioner vi jobbar just nu.
- Består av:
  - en samling processorregister
  - en exekveringsstack
- Exekverar inom en process (användartråd) eller inom operativsystemet (kerneltråd). Processen eller OS styr över när tråden får exekvera.
- Tråden har tillgång till alla resurser inom sin domän (OS eller process).

Jämför: Kryss på arbetslista, Bokmärke

TCB och PCB

OS måste definiera en ”behållare” för tråddata och processdata, samt hålla reda på en lista av trådar för varje resurs.

OS exekverar och schemalägger endast kerneltrådar.

Processer körs genom att kerneltråden återvänder från ett interrupt (och växlar därmed från kernel till user mode).

User-trådar körs genom att processen internt schemalägger egna trådar och växlar mellan dem.

Moderna OS kan koppla user-trådar till kerneltrådar för att öka prestanda på multiprocessorer.

Exempel: Visa Pintos.

Skydd av resurser

- Vad händer om en tråd eller process:
  - Aldrig blir klar med CPU?
  - Använder hårdvaruresurser på fel sätt?

- Andra trådar eller processer blir lidande.
- Hårdvara kan ta skada.

Timeravbrott

(timer interrupt)

- Bredvid CPU finns en liten enhet som varje klocktick räknar upp en intern räknare.
- Räknaren har ett begränsat antal bitar. När den ”slår över” skickas en signal till CPU, och räknaren startar om på 0. Signalen genererar ett timeravbrott (asynkront).
- Operativsystemet kan ställa in hur ofta räknaren skall generera avbrott.
- Operativsystemet kan ställa in vilken kod som skall hantera avbrottet.

Exempel: Visa Pintos.

Trådbyte

- När OS-koden som reagera på ett timeravbrott exekverar så har OS möjligheten att undersöka hur länge nuvarnde tråd exekverat och eventuellt byta till en annan tråd.
- När OS på detta sätt tvingar en process att släppa en resurs (i detta fall CPU) kallas det preemption.
- Detta är funktionaliteten som gör att OS kan garantera alla trådar rättvis exekveringstid.
- Det finns enkla ”OS” där varje tråd själv måste se till att byta till en annan tråd. Dessa är inte lika säkra men duger bra där programmeraren har total kontroll på hela systemet och de trådar som kör (t.ex. ett inbyggt system).
- Eftersom avbrottet är asynkront kan det uppstå precis var som helst i koden, och överraskande fel kan uppstå om en tråd arbetar på samma data som en annan. Mer om detta i andra halvan av kursen.

Begrepp: cooperative OS, preemptive OS

Dual-mode exekvering

- CPU ges två exekveringslägen:
  - kernel mode
    - full access till alla instruktioner och resurser
  - user mode
    - endast access till resurser explicit tilldelade
- Operativsystemet exekverar kerneltrådar och har alltid tillgång till kernel mode
- Processer exekverar i user mode och måste be operativsystemet utföra resursåtkomst (inte be om lov, utan be att få begäran utförd av OS och endast få tillbaka resultatet)

Systemanrop

- När OS utför en tjänst på uppdrag av en process och ger resultatet till processen
- Processen kör hela tiden i user mode och har inte möjlighet utföra tjänsten själv då tjänsten i fråga kräver tillgång till kernel mode
- Operativsystemet kör hela tiden i kernel mode och kontrollerar noga att tjänsten går att utföra och att eventuella indata från processen är korrekta, samt utför tjänsten

Gränssnitt till OS

- Systemanrop utgör gränssnittet till OS
- Abstraherar skillnader i hårdvarukonfiguration, operativsystemet tar hand om eventuella skillnader och presenterar alltid ett enhetligt gränssnitt till applikationsprogrammeraren
- Program kan enklare göras portabla
- Operativsystemet tillhandahåller ett bibliotek som presenterar systemanropen som vanliga funktionsanrop, själva systemanropet döljs inuti dessa funktioner.

Exempel: Visa Pintos

Vanliga systemanrop

- Filhantering:
  - skapa filer
  - öppna filer
  - läsa filer
  - skriva filer
  - stänga filer
  - ta bort filer
- Minneshantering:
  - allokera minne
  - avallokera minne
  - dela minne med andra processer
- Processhantering:
  - starta processer
  - invänta processers avslut
  - stoppa processer
  - visa alla exekverande processer
- Trådhantering:
  - skapa trådar
  - starta trådra
  - pausa tråd som kör
  - stoppa trådar
  - synkronisera trådar

Systemanrop steg 1 (user mode)

- Processen anropar önskad funktion i biblioteket över systemanrop och anger rätt argument
- Biblioteksfunktionen placerar argumenten på processens (nu exekverande tråds) exekveringsstack
- Biblioteksfunktionen placerar en kod för ”sitt” systemanrop på samma stack
- Biblioteksfunktionen exekverar assemblerinstruktionen som skapar ett mjukvaruinterrupt (synkront).

Systemanrop steg 2 (avbrott)

- CPU reagerar med att i denna instruktion:
  - Växla till kernel mode
  - Spara SP temporärt
  - Återställa SP till den exekveringsstack OS använder i kernel mode (en för varje kernel-tråd)
  - Spara alla register på exekveringsstacken så som de såg ut just innan mjukvaruinteruptet exekverades (inklusive temporärts sparade SP)
  - Ersätta IP och PC med adress och instruktion på den adress operativsytemed specificerat för just detta interrupt

Systemanrop steg 3 (kernel mode)

- Nu exekverar OS-kod i kernel mode.
- OS kontrollerar de indata och ev. plats för utdata som placerades på processens stack.
- OS utför begäran på ett kontrollerat och effektivt sätt.
- OS placerar resultatet på förutbestämd plats (i register eller minne angivet av processen)
- OS exekverar instruktionen för att återvända från interrupt.

Systemanrop steg 4 (återhopp)

- CPU reagrear med att i en instruktion:
  - Återställa alla sparade register, inklusive SP, IR och PC (några kan vara manipulerade av OS)
  - Byta till user mode
- Nu är vi återigen i user mode
- Biblioteksfunktionen fortsätter och returnerar resultatet till anropande process.

Processens kod har aldrig haft direkt åtkomst till av OS kontrollerade resurser, endast OS kod har haft det.