TDIU11 – Föreläsning 1

Introduktion till operativsystem

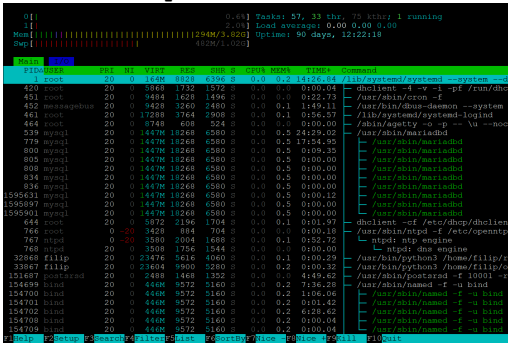
Filip Strömbäck

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

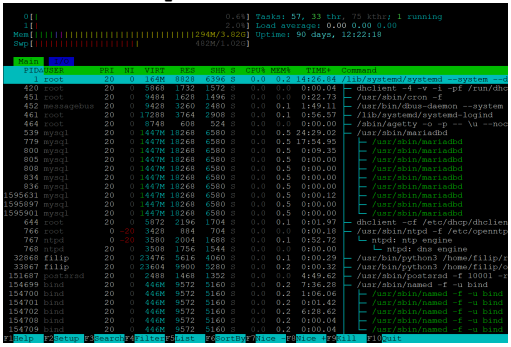
Överblick

Bild från `htop`:



Överblick

Bild från `htop`:

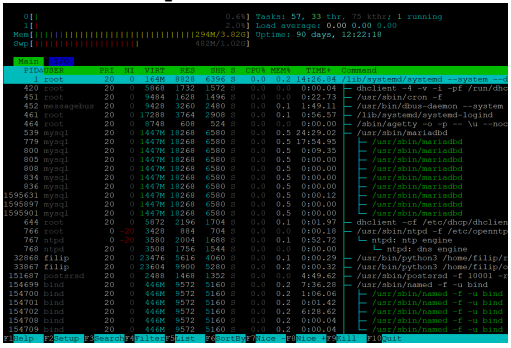


Intressanta frågor:

- Vi har 2 CPU-kärnor, men 57 "tasks". Hur fungerar det?
- Hur ser vi till att olika program kan samarbeta på ett säkert sätt?

Överblick

Bild från `htop`:

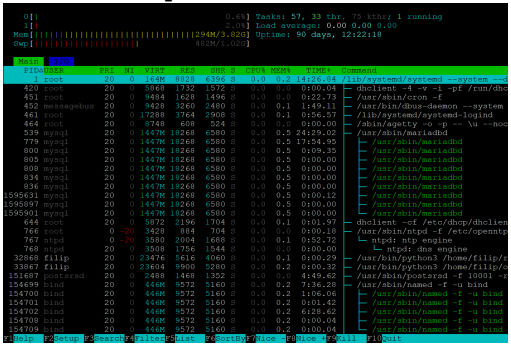


Intressanta frågor:

- Varför ser ”mem” full ut, trots att bara 294 MiB av 3,82 GiB används?
- Varför använder `systemd` 164 MiB ”virt”, men bara 9 MiB ”res” och 6 MiB ”shr”?
- Vad innebär ”swp”? När används den?

Överblick

Bild från `htop`:



Intressanta frågor:

- Hur lagrar vi olika filer på samma disk?
- Hur ser vi till att bara rätt program kan komma åt rätt information?

Mål med kursen

Få insyn i...

- ...*vad* ett operativsystem är
- ...*vad* det gör
- ...*varför* det behövs

Mål med kursen

Få insyn i...

- ...*vad* ett operativsystem är
- ...*vad* det gör
- ...*varför* det behövs

Så att vi kan...

- ...*förstå* hur våra program körs
- ...*förstå* de abstraktioner som OS introducerar
- ...*utnyttja* abstraktionerna effektivt
- ...*undvika* begränsningar som finns

LINKÖPING UNIVERSITY

Resurser

- Kurshemsida: `https://www.ida.liu.se/~TDIU11/`
- Litteratur: Operating System Concepts
- Kod för Pintos: `https://gitlab.liu.se/tdiu16-material/pintos`

Kursledare	Filip Strömbäck
	(Klas Arvidsson)
Kursassistent	Dag Jönsson
Assistenter	Christoffer Holm
	Malte Nilsson
	Mattias Karlsson
Administratör	Anna Grabska Eklund

Examination

UPG1 Problemlösning, 3 hp (U, G)
Förbereda och diskutera inlämningsuppgifter (*challenges*) till de fem tillfällena som finns under kursens gång.

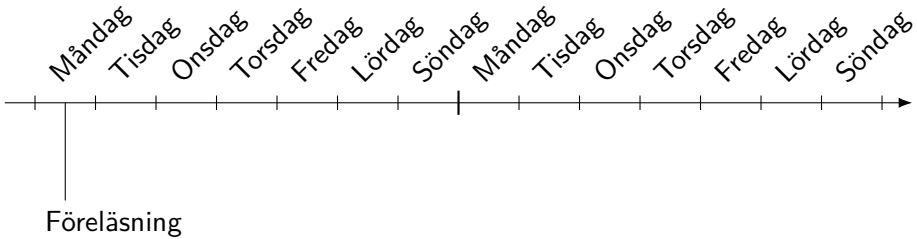
UPG2 Inlämningsuppgift, 1 hp (U, G)
Sammanfattning och diskussion om artiklar.

TEN1 Skriftlig tentamen, 2 hp (U, 3, 4, 5)
Skriftlig papperstentamen.

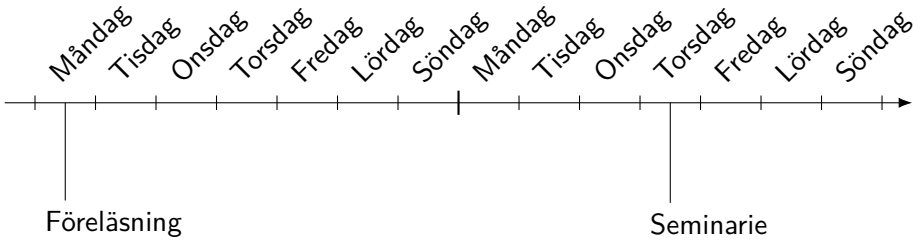
Ändringar från tidigare år

- Ny kursansvarig
- Uppdaterade föreläsningar
- Tydliggöra ambitionsnivån på uppgifter för UPG1

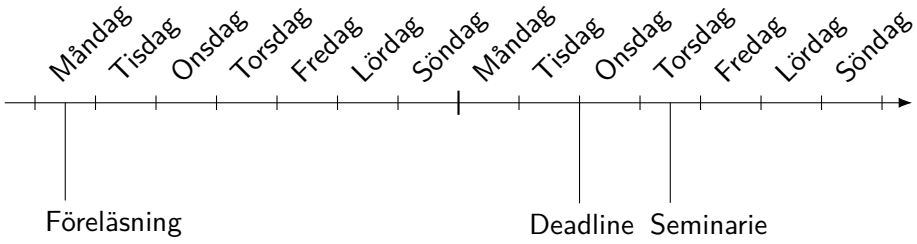
Struktur


Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Föreläsning

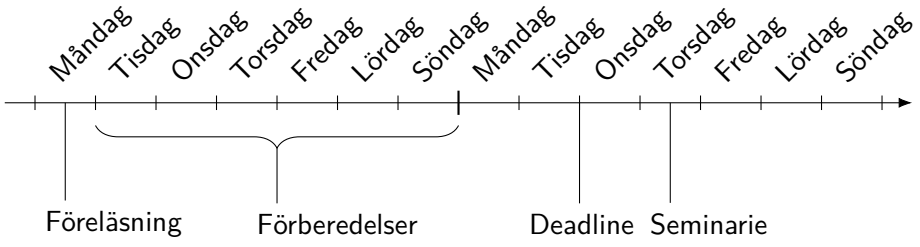
Struktur


Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Föreläsning
Seminarie

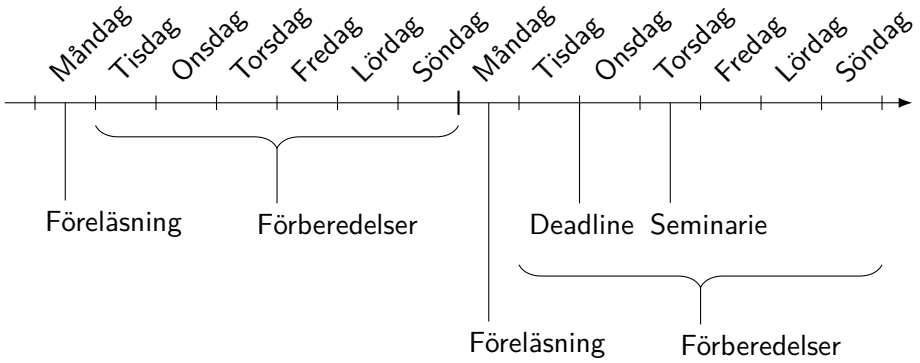
Struktur


Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Föreläsning
Deadline
Seminarie

Struktur


Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Föreläsning
Förberedelser
Deadline
Seminarie

Struktur


Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
Föreläsning
Förberedelser
Deadline
Seminarie
Föreläsning
Förberedelser

Föreläsningar

- Oftast måndagar
- Introducerar teori till kommande seminarier
- Läshänvisningar till kursboken finns på kurshemsidan
- Skumma gärna igenom boken innan föreläsningen – ställ frågor på sådant som är oklart
- Slides finns på kurshemsidan – mest som anteckningsstöd, inte tänkta att gå att läsa ur kontext

Seminarier: Challenges (UPG1)

- 5 tillfällen
- 8 problem/tillfälle
- 40 problem totalt
- 16 lösta problem ger G
- Var 4:e löst problem utöver detta ger 1 p på mot högre betyg på tentan

Förberedelse:

- Lös en delmängd av problemen
- Skicka lösningsanteckningar till din assistent innan deadline
  Ämne: `[TDIU11] Challenge N`
  Skriv också: *Jag har löst uppgift: 1, 3, 5, 7*
- Samarbete är OK och uppmuntrat. Skicka in dina *egna* lösningsanteckningar, var beredd att presentera.

Seminarier: Challenges (UPG1)

- 5 tillfällen
- 8 problem/tillfälle
- 40 problem totalt
- 16 lösta problem ger G
- Var 4:e löst problem utöver detta ger 1 p på mot högre betyg på tentan

Under seminariet:

- Assistenten väljer studenter slumpvis
- Presentation utifrån medtagna anteckningar
- Diskussion vid behov
- Kom gärna även om du inte löst något problem!

Notera: Lösningen behöver *inte* vara helt korrekt. Visar du att du har förstått det som krävs för att lösa problemet är det OK. Kan du inte det förlorar du *alla* poäng från seminariet.

Seminarier: Artiklar (UPG2)

- 3 tillfällen
- 1 artikel/tillfälle
- 2 för G

Inför seminariet:

- Läs och sammanfatta en OS-relaterad artikel
- Skicka in till assistent och urkund, samt skriv ut och ta med till seminariet
  Ämne: `[TDIU11] Summary N`

Under seminariet:

- Läs och ge feedback till en annan student
- Diskussion i större grupp

Tips: sista tillfället är reserv och ligger i tenta-p.
Undvik det för mer tid att förbereda inför tentan!

Planering

Vecka	Föreläsning	Seminarie
3	Processer och trådar	—
4	Minneshantering	Challenge 1: Schemaläggning
5	Virtuellt minne	Artikel 1: Schemaläggning
6	Filsystem och lagring	Challenge 2: Virtuellt minne
7	Säkerhet	Challenge 3: Filsystem
8	Repetition/utblickar	Artikel 2: Filsystem
9	—	Challenge 4: Säkerhet
10	Tentaförberedelse	Challenge 5: Repetition
11	(omtenta-p)	Artikel 3: Säkerhet (reserv)

Planering

- Finns material att arbeta med varje vecka
- Finns däremot flexibilitet vilka seminarier du fokuserar på
- Planera i förväg för sammanfattningarna – bra idé att läsa igenom artikeln en första gång i god tid!

Webreg

Anmäl er till en grupp i Webreg

- Finns två grupper, en för UPG1 och en för UPG2.
- Anmäl er i båda

Notera: Två seminarier i Dags grupp ligger utanför blocket (10–12 i stället för 8–10). Dubbelkolla krockar innan du väljer den gruppen.

LINKÖPING UNIVERSITY

Varför vill vi ha ett operativsystem?

Varför vill vi ha ett operativsystem?

Användbarhet:

- Användare förväntar sig viss funktionalitet
- Programmerare vill inte behöva implementera filsystem/IO/etc.

Portabilitet:

- Vi vill att program ska fungera på olika datorer med olika hårdvara
- Vill inte behöva implementera stöd för all möjlig hårdvara i varje program

Varför vill vi ha ett operativsystem?

Effektivitet:

- Vi vill nyttja hårdvaran effektivt:

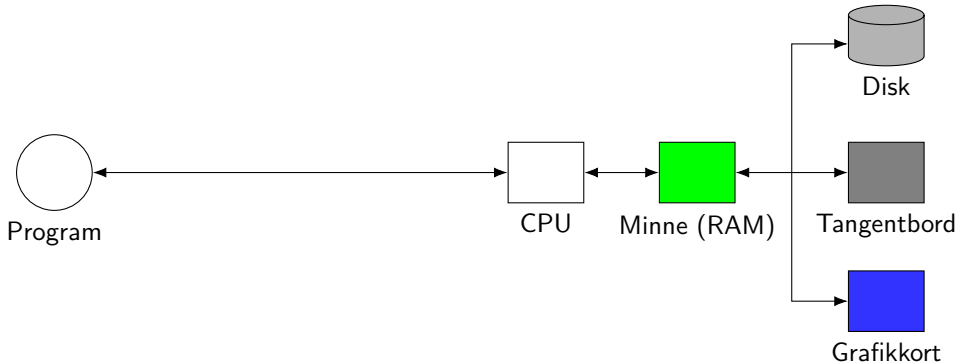
$\Rightarrow$ Vi vill kunna köra flera program samtidigt

$\Rightarrow$ Vi vill kunna låta flera användare använda systemet samtidigt

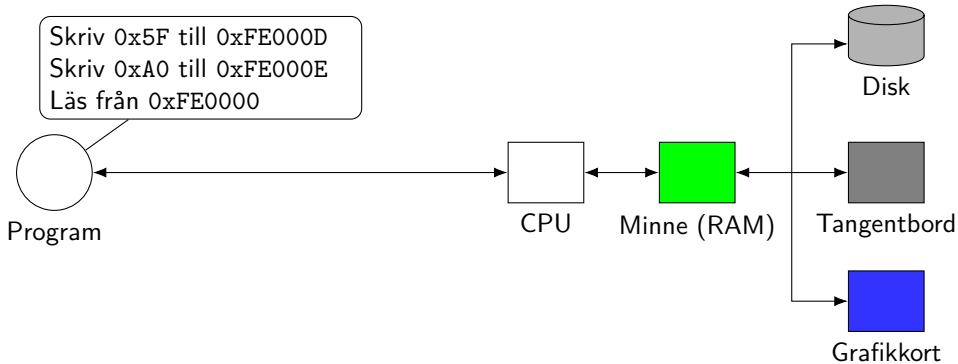
Säkerhet:

- Program vi kör samtidigt ska inte kunna förstöra för varandra
- Program ska inte kunna förstöra för OS
- Respektera användares begränsningar i åtkomst av data

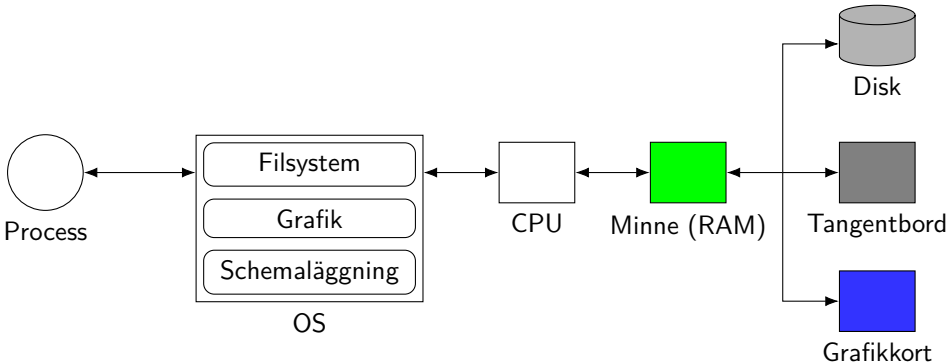
Översikt



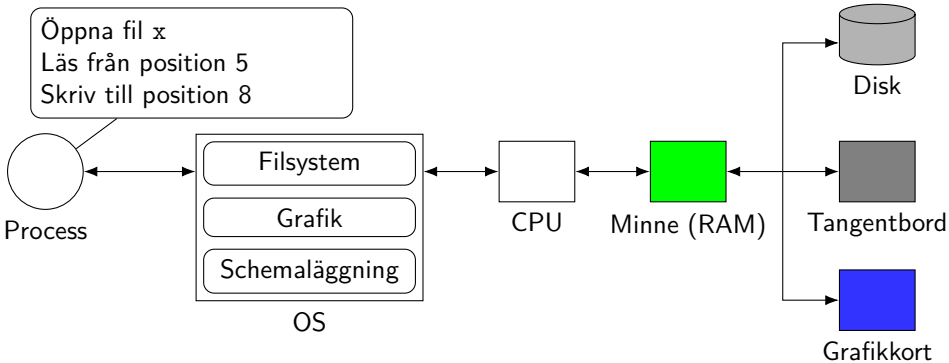
Översikt


Skriv 0x5F till 0xFE000D
Skriv 0xA0 till 0xFE000E
Läs från 0xFE0000
Program
CPU
Minne (RAM)
Disk
Tangentbord
Grafikkort

Översikt


Disk
Process
Filsystem
Grafik
Schemaläggning
OS
CPU
Minne (RAM)
Tangentbord
Grafikkort

Översikt


Öppna fil x
Läs från position 5
Skriv till position 8
Process
Filsystem
Grafik
Schemaläggning
OS
CPU
Minne (RAM)
Disk
Tangentbord
Grafikkort

Vad består ett operativsystem av?

- **Kernel/kärna**
  Delen av OS som *alltid* körs, koordinerar processer som körs
- **Användargränssnitt** (skal, shell)
  Program för att interagera med användaren
  Ex. vis: bash, terminal, fönsterhanterare, skrivbordsmiljö
- **Verktyg**
  Program som hjälper användaren att göra saker
  Ex.vis: `cp`, `g++`, `emacs`, `libreoffice`, `firefox`

Var går gränsen för vad som är ett operativsystem?

Fokus i kursen: kernel

Olika typer av operativsystem

- Single-user
  - Äldre system, tillåter oftast bara att ett program körs samtidigt
  - OS kan helt enkelt vara en samling funktioner som kan anropas
  - Exempel: DOS
- Batchsystem
- Time-sharing
- Real-time

Olika typer av operativsystem

- Single-user
- Batchsystem
  - Tidiga operativsystem – stordator i källaren
  - Byggda för att hantera *job* effektivt
  - Kör flera job samtidigt för att nyttja systemet effektivt
  - Inte byggt för interaktiv användning, svarstider kan vara långa
- Time-sharing
- Real-time

Olika typer av operativsystem

- Single-user
- Batchsystem
- Time-sharing
  - Moderna operativsystem
  - Kör flera processer "samtidigt", ibland åt olika användare
  - Har som mål att användas interaktivt, och strävar efter att låta alla processer köras relativt ofta
  - Exempel: Linux, Windows, MacOS, ...
- Real-time

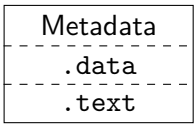
Olika typer av operativsystem

- Single-user
- Batchsystem
- Time-sharing
- Real-time
  - Ofta i inbyggda system
  - Har likt time-sharing stöd för flera processer
  - Ger *garantier* på när systemet svarar på externa händelser
  - Finns *soft* real-time och *hard* real-time
  - Exempel: FreeRTOS, (real-time Linux)

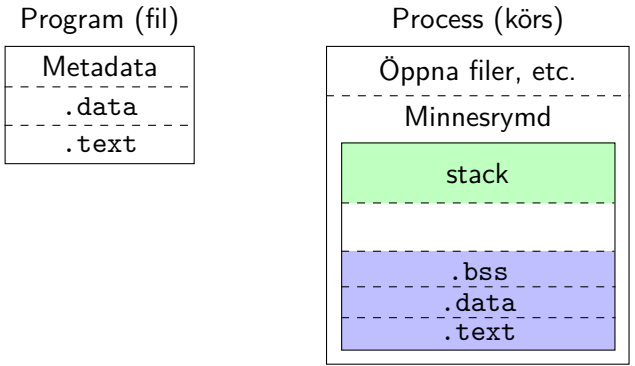
LINKÖPING UNIVERSITY

Processer och trådar

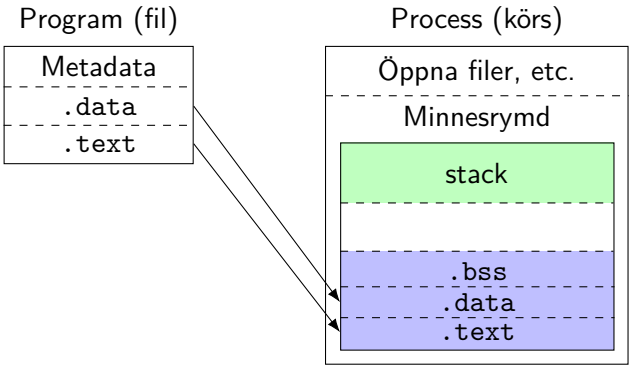
Program (fil)



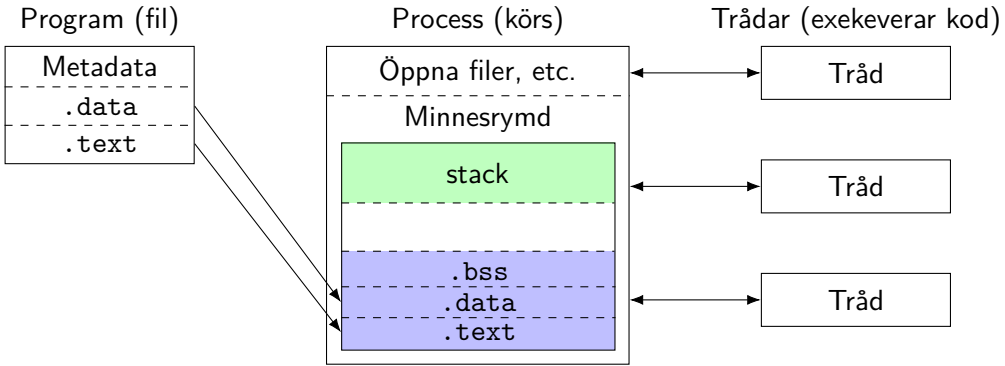
Processer och trådar



Processer och trådar



Processer och trådar



Hur hanterar vi flera processer på ett säkert sätt?

Hårdvaran ger oss:

- Dual-mode exekevering
- Virtuellt minne
- Avbrott (Interrupt)
- Timers

Dual-mode

Mål: Vill kunna ”låsa ner” CPU när vi kör kod utanför kernel

Lösning: vi har en ”mode-bit” i CPU (eller motsvarande)

Exempel: x86 har olika ”ringar” som får göra olika mycket med systemet:

0. Kernel mode
1. (används ej)
2. (används ej)
3. User mode

Dual-mode

x86 har olika ”ringar” som får göra olika mycket med systemet:

0. Kernel mode
1. (används ej)
2. (används ej)
3. User mode

Endast ring 0 får köra följande instruktioner:

`hlt`	Halt – avbryter exekevering tills interrupt kommer
`lgdt`	Load global descriptor table – ändrar hur virtuellt minne ser ut
`in`, `out`	I/O till hårdvara

Dual-mode

x86 har olika "ringar" som får göra olika mycket med systemet:

-2. System management
-1. Hypervisor
0. Kernel mode
1. (används ej)
2. (används ej)
3. User mode

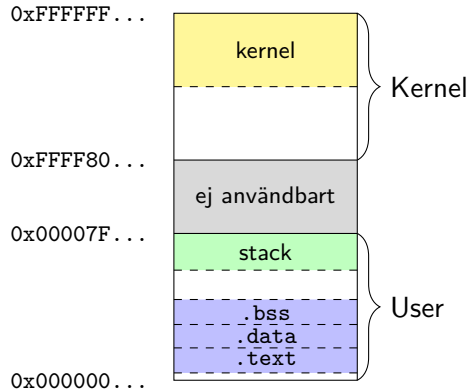
Endast ring 0 eller lägre får köra följande instruktioner:

`hlt`	Halt – avbryter exekevering tills interrupt kommer
`lgdt`	Load global descriptor table – ändrar hur virtuellt minne ser ut
`in`, `out`	I/O till hårdvara

Virtuellt minne

Vi vill också förhindra att processer kommer åt varandras och kernels minne

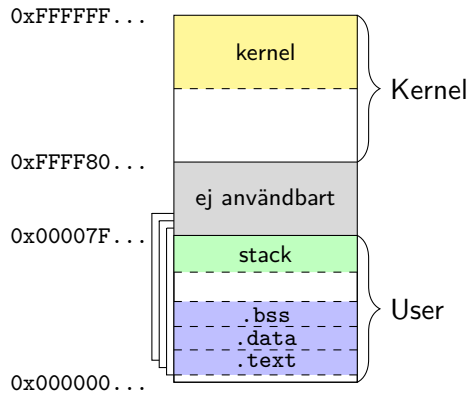
- CPU tillhandahåller *paging*
- Vi kan implementera *virtuellt minne*
- Varje process får egen *vy* av RAM
- Kan bara komma åt minne de "borde" kunna komma åt



Virtuellt minne

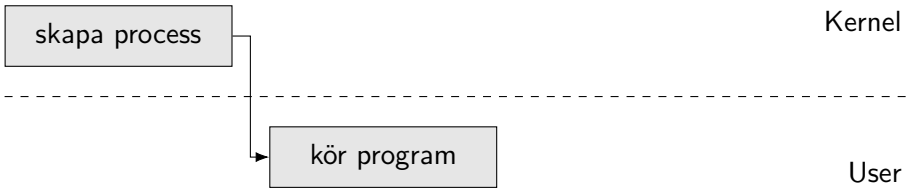
Vi vill också förhindra att processer kommer åt varandras och kernels minne

- CPU tillhandahåller *paging*
- Vi kan implementera *virtuellt minne*
- Varje process får egen *vy* av RAM
- Kan bara komma åt minne de "borde" kunna komma åt



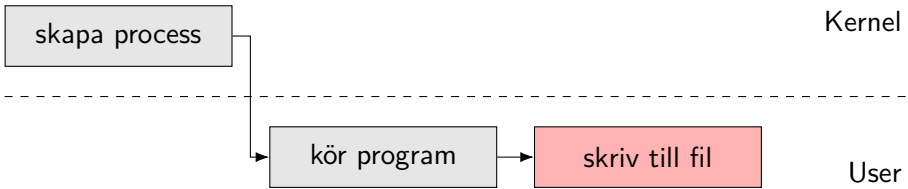
Problem så här långt

Vad vi kan göra nu:



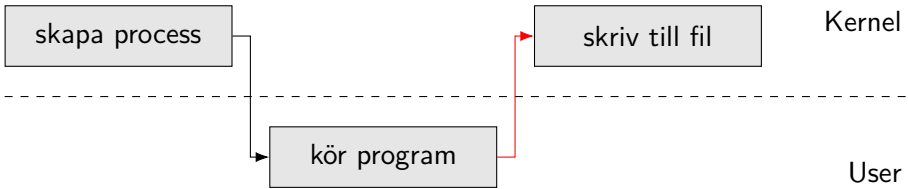
Problem så här långt

Vad vi kan göra nu:



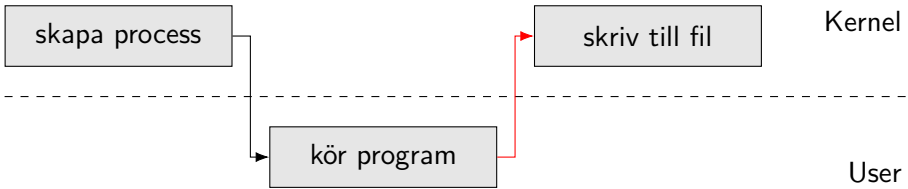
Problem så här långt

Vad vi kan göra nu:



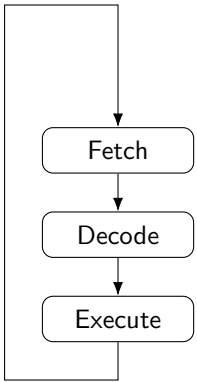
Problem så här långt

Vad vi kan göra nu:

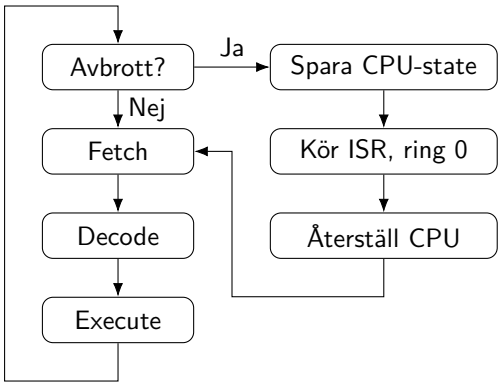


Hur löser vi detta?

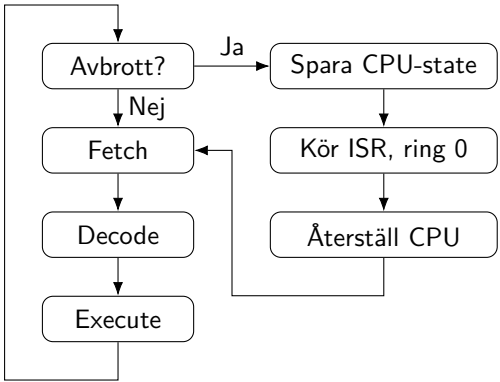
Avbrott (interrupt)



Avbrott (interrupt)



Avbrott (interrupt)



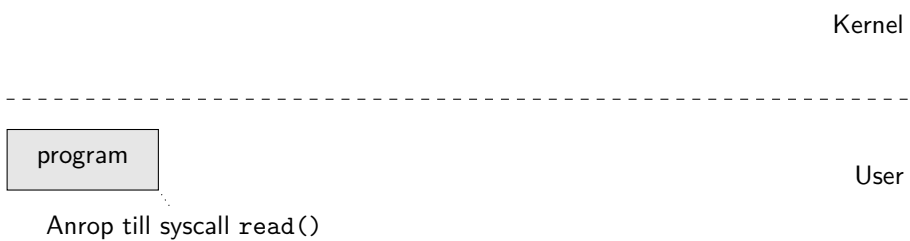
- ISR specifieras av *kernel*
- ISR körs som *kernel*
- Genereras av *hårdvara* eller *mjukvara*
- Kan användas för att *säkert* återgå till högre privillegier

Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:

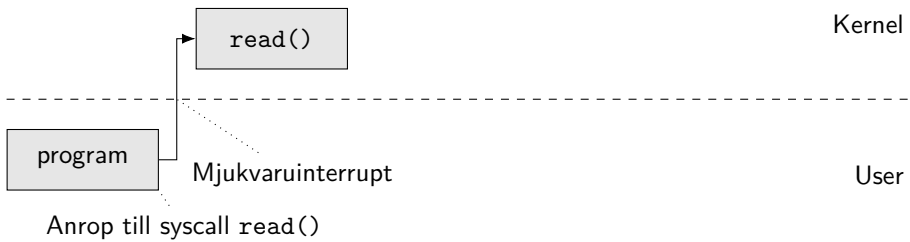
Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:



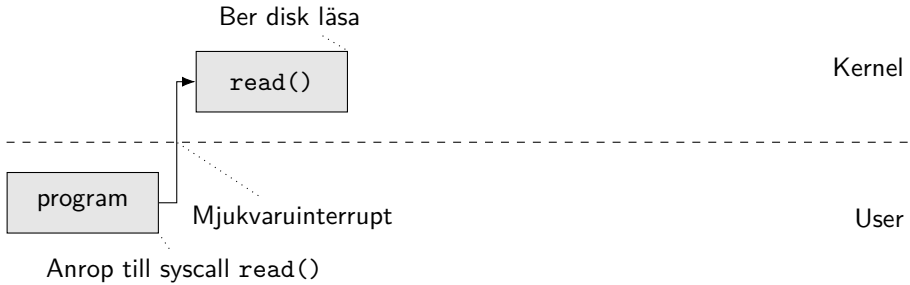
Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:



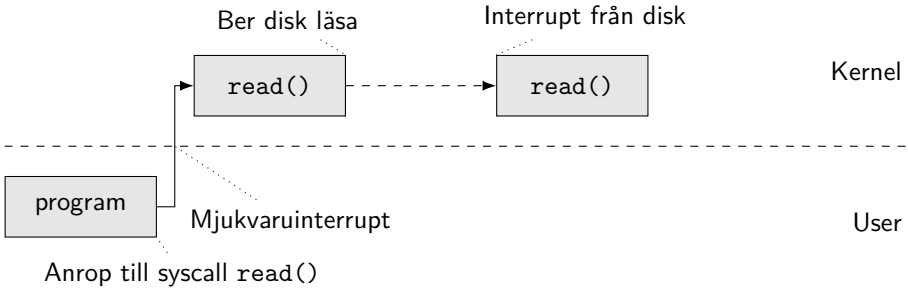
Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:



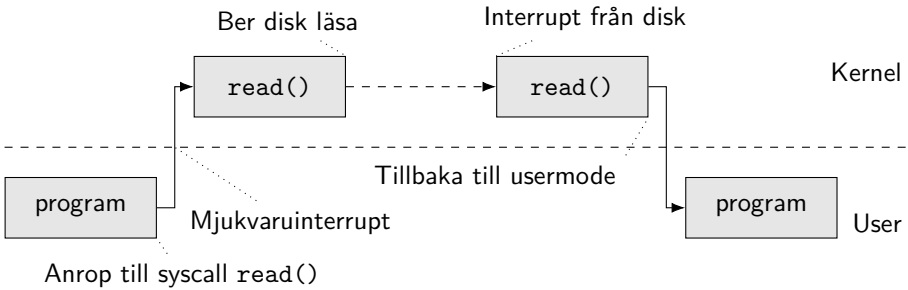
Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:



Systemanrop

Vi kan använda interrupt för att implementera *systemanrop*:



Filip Strömbäck

www.liu.se

LINKÖPING UNIVERSITY