1 Operativsystemer - oversigt

Hvorfor OS - virtualisering
Abstraktion (processer, virtual hukommelse, filer)
Kernel mode og user mode
Systemkald (slide 31)
Interrupts

At levere en **simplere model af computeren** til programmer, og **håndtere ekstern hardware**
Abstraktion - *Processer*, *virtuel hukommelse* og *filsystemer*
Virtualisering af resourcer (enhver process har egne resourcer)
Kernel mode og **user mode** (sikkerhed)

Systemkald - tilgang til hardware går gennem OS (sikkehed, abstraktion, simplicitet)
Process vil læse fil → OS gør dette, da OS styrer *filsystem*
Trap til kernen - Skift til kernelmode, OS arbjeder, returner til userspace (*slide 31*)
Fork() - opretter ny process og returnerer *pid*
Filsystem kald - hvis processer kan tilgå direkte, har OS ikke styr på filsystemet

Kernel mode - **complete access** to hardware and **any instruction** the computer can execute
User mode - can only access a **subset of machine instructions**, i.e. **I/O is forbidden**

Interrupts - **signal til CPU** om noget er sket (*fx keyboard input*)
Undgå busy wait
Kan afbryde nuværende beregning (*interrupts kan deaktiveres*)

2 Processer og tråde

Hvad er en process?
Hvad er tråde? (slide 34)
Process table
Schedulering - fairness balance

Proces - program under afvikling (*centralt begreb for virtualisering*)
- Har **eget adresserum** i memory
- Kan **kommunikere** via. semaforer, monitors og message passing (***mutual exclusion***)
- ***States*** - **running** (aktiv), **runnable** (klar), **blocked** (venter påydre enhed fx keyboard input)
- ***Multiprogrammering*** - afvikling af flere processer "samtidigt"

Thread - Har egen stak men fælles adresserum (*slide 34*)
- **Userspace threads -** hurtigere, men problem ved blocking syscalls
- Tillader **parallel udførsel**
- Processen kan arbejde videre selvom der ventes på input

Process table (*Process control blocks*) - Alt information om processerne er gemt her (*slide 14*)
- Skift af process → opdatering af process table
- Opdeles i generel information og trådspecifik information (*fx trådens stak*)

Scheduling - skift mellem threads eller processer
- **IO bound** vs **CPU bound** - blanding for maksimal CPU brug
- **Fairness** (alle får CPU-tid, undgå *starvation*), **balance** (flest muligt aktive enheder)

3 Proceskoordinering

Kritiske regioner
Race condition
Mutual exclusion - semaforer, monitors
Test-and-set-lock
Spin lock
Preemptive og nonpreemptive scheduling

Kritiske regioner - regioner der kan lede til race conditions (*spooler eksempel - overskrive værdier*)

Mutual exclusion - undgå race condition, tillad kun en process af gangen
Petersons algoritme (*slide 21*)
Starvation, deadlocks, livelocks
Semaforer - ikke negativt heltal. Kan implementeres ved Test-and-set-lock
Monitors - styrer tilgang til delt resource. Indbygget mutual exclusion (**nested monitors** → deadlocks)
Wait, **notify** kald (*ikke implementeret i C*)

Test-and-set-lock - atomar instruktion der tester og opdaterer en lagercelle
Spin locks - locks der benytter **busy waiting**. Brug kun når det vides at ventetiden er kort!

Nonpreemptive schedulering - der skal scheduleres ved *terminering*, *blokering* og ny *process*
Preemptive schedulering - *process start*, *process bliver runnable* og *efter tidskvantum*

4 Schedulering

Preemptive og nonpreemptive
Schedulering - prioriteter, fairness, balance
system typer - batch, interaktiv, realtid
first come first served
round robin
fair share
user threads og kernel thread (slide 17)

Preemptive *vs.* ***nonpreemptive*** - må CPU tages fra processen eller ej
Scheduling - skift mellem threads eller processer
Prioriteter - process med højest prioritet køres (*kan lede til starvation*)
IO bound vs **CPU bound** - blanding for maksimal CPU brug
Fairness (alle får CPU-tid, undgå *starvation*), **balance** (flest muligt aktive enheder)

Slide 8 - forskellige systemer
Batch systemer - *ingen bruger* venter på svar (*fx beregning af renter i banker*)
maximer udført arbejde, maximer CPU brug
Interaktive systemer - normalt system, preemption,
Hurtigt svar, opfyld brugerens forventninger
Realtidssystemer - afhængig af timing (multimedia, fylde flasker) (applikationer ofte kendt)
Overholde deadlines

Algoritmer
First come first served - *nonpreemptive* batchsystemer
Round-Robin - Vælg den næste i en normal kø (fairness problem ved flere brugere)
Fair share - giver lige meget CPU-tid til alle brugere

User threads vs **kernel threads** (*slide 17*)

5 Lageradministration - Basale elementer

Lager uden abstraktion (slide 6)
Swapping
statisk og dynamisk reallokering (slide 7)
håndtering af frie blokke

Register, cache, RAM, disk (*administreres af OS*) (*forskellig tid at tilgå*)

Lager uden abstraktion - absolutte adresser bruges (brugt i gamle dage (*slide 6*))
Multiprogrammering
- ***Swapping*** - Flyt alt ud på disk og hent ny process ind i ram (tager lang tid)
- ***Statisk reallokering*** - når et program loades adderes første memoryadresse til alle adressereferencer
- ***Dynamisk reallokering*** - *logiske* adresser konverteres til *fysiske* ved brug af **base-register**
  - ***Base*** - starten på processens adresserum (bruges ved logisk → fysisk konvertering)
  - ***Limit*** - længden af processens adresserum (*sørger for processer ikke tilgår andres data*)
  - **Swapping** bruges hvis der ikke er plads til alle processer (*memory compaction tager lang tid*)

Frie blokke kan håndteres af *liste* eller *bitmap* (*slide 14*)

Bloksammensmeltning - nabohuller skal sammensmeltes ved deallokering (*evt slide 17*)

6 Lageradministration - Virtuelt lager

Virtual memory
Overlays
Paging - page table og frame table
algoritmer (slide 33)
segmentation

Virtual memory - når processer skal bruge mere memory end der fysisk er

Overlays - opdel programmer i overlays og hav en overlay manager til at swappe disse ind og ud af memory
Paging - Opdel lageret i *page frames* af bestemt størrelse og load enkelte pages fra disk
MMU oversætter *virtuelle* adresser til *fysiske* adresser (*page, offset*) (*slide 21*)
Page faults - når en process skal bruge en side der ikke er i fysisk memory

TLB - hurtigere opslag end page table
Page table - indgang for hver virtuel adresse (*brug **TLB** for hurtigere opslag*)
Frame table - indgang for hver fysisk adresse, giver langsommere opslag (*omvendt page table*)
Indekseret efter haskoden på virtuel adresse

Page replacement algoritmer (*bruges ved **page faults***)
Aging - markerer pages der blev refereret i denne clockcycle (Software simul. af **LRU**) (slide 33)
Working Set Clock - sammensætning af **clock** og **workingset**

Segmentation - todimentionelt adresserum (*fx smart ved compilere*)
Kode og stak kan gro seperat af hinanden
Paging og segmentation bruges sammen (*slide 22*)

7 Filsystemer

Filbegrebet - filtyper, filstruktur, filattributter
Organisering af filer (slide 47)
Directories

Filbegrebet - abstraktion fra data på disken
- **Filtyper** - behandles forskelligt af programmer (*OS skal kun kende* ***eksekverbare***)
- **Filstruktur** - betragtes som sekvens af bytes. *OS kommer ikke i vejen* (benyttes af *UNIX* og *Windows*)
- **Filattributter** - attributter som *protection, password, creation time*

Organisering af filer på disk
- **Sammenhængende blokke** - *effektiv* og *simpel*, men giver *høj fragmentering*
- **Linked list** - **FAT** - *god udnyttelse af diskplads* - kan gemmes i ram eller pointer i hver blok (*slide 47*)
- **I-nodes** - indeholer data om filer, ligger et kendt sted på disken så de let kan slåes op (*slide 49*)

Directories - Samling af filer, hierarkisk system
- et **directory** er *en fil* der indeholder [i-number, filename] par
- **Hardlinks** - refererer til filens i-node
- **Symlinks** - indeholder stien til den linkede fil (ødelægges når filen slettes)

8 Input/Output - Basale elementer

Drivere
block og character devices
memory mapped IO
Polling, interrupts, DMA
Buffere

OS *kender ikke devices* så **drivere** leverer et *kendt interface* til enheden
Block devices - outputter og accepterer kun datablokke af bestemt størrelse (fx disk, usb)
Character devices - outputter og accepterer kun character streams (fx keyboard, højtaler)

Memory mapped IO - CPU kommunikerer med enheder *på samme måde* som lageret (*RAM*)
Fordele - samme interface som memory, nem tilgang fra højniveausprog, nem beskyttelse
Ulemper - undgå caching, devices skal lytte på memorybus, ligner brug af lager men er anderledes

Polling - busy wait, CPU-tid spildes på ingenting
Interrupts - mange kontekstskift og stor CPU overhead (interrupt ved hver byte)
DMA - polling, men DMA venter i stedet for CPU (interrupt ved *n* bytes), stjæler bussen, langsom CPU

Buffere - fx disk, hvor bytes kommer med konstant fart. Hvis bussen er optaget mistes de.

9 Input/Output - Devices

Diske - struktur, scheduling (slide 30 + 31)
Bad sectors
RAID
Stable storage - ECC

Diske - stort sekundært lager (*slide 7*)
opdelt i **sektorer** og **cylindere** - OS ser kun *virtuel opdeling*, disk controller tager sig af *fysisk placering*
Disk schedulering - *shortest seek first* og *elevator algorithm* (*slide 30 + 31*)

Bad sectors (*slide 32*)

RAID - flere diske der arbejder sammen (*højere hastighed* og/eller *højere stabilitet*)
RAID 0 - *Striping* - split filen op og distribuerer den over diske
RAID 1 - *Mirroring* - samme som RAID 0 men med backup diske
RAID 4 - Som RAID 0 med errorcorrection (*XOR alle strips*)
RAID 5 - Som RAID 4 men parity er spredt ud over diske (*undgå bottleneck*)

Stable storage - to identiske diske med **ECC** (*Error Correction Code*) til validering
Stable read (læs og valider med ECC), *stable write* (skriv til begge diske), *recovery*

10 Deadlocks

Deadlock - Coffman
Resursegrafer (slide 26)
Detection, avoidance, prevention (slide 38)
Starvation
Communication deadlock

Deadlock - processer er deadlocked hvis de venter på en event der kun kan forårsages af en anden process
Mutual exclusion, Incremental acquisition, No preemption, Wait-for-cycle

Resursegrafer - cykler er deadlocks (*slide 26*)

Detection - tillad deadlocks og fjern dem når de opstår
Opdag deadlocks (*matrix*) (*slide 38*)
Genopretning - preemption, rollback og stop processor
Aviodance - undgå deadlocks før de opstår
Skal vide hvilke resurser processerne vil benytte
Prevention - Opbyg systemet så deadlocks *ikke kan* opstå (fjern en af Coffmans betingelser)
Mutual exclusion - spool everything (*printer daemon*)
Incremental acquisition - tag resurser når processen startes
No preemption - Fjern resurser fra processer
Wait-for-cycle - giv alle resurser et nummer. Tag resurser i stigende orden

Starvation - en process bliver aldrig kørt da den blokeres af en anden
Communication deadlock - fx netværk, send pakke og vent på svar. pakken når aldrig frem