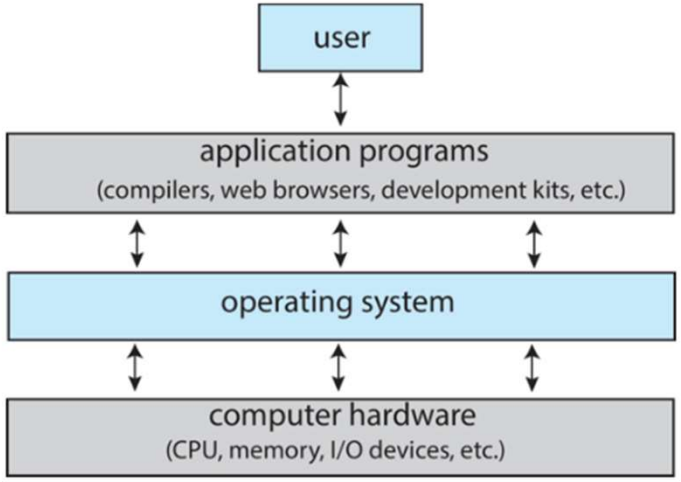
Operativ Systemer 2

Lavet af: Vivek Misra

Repetition af OS

- Som vi kendte fra den sidste slide, så handlede Operativ Systemer om følgende:
  - At kunne danne en kommunikation eller forbindelse mellem Brugeren og Hardware.
  - Forbindelse er forbundet igennem CPU'en som er Operativ Systemets Hjerne.
- MÅL:
  - Vi kan se, at målet med en Operativ System er, at kunne eksekvere brugerens programmer og derved gør computeren overskueligt at bruge.
  - Det er også målet, at bruge computeren på en effektiv måde for brugeren.

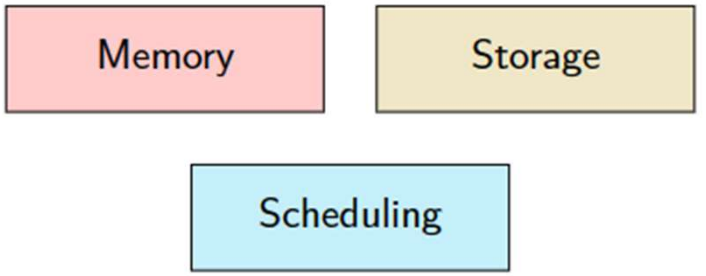


Repetition af OS

- Bruger Perspektiv:
  - Computer OS:
    - Nemt at bruge
    - Hurtigt respons fra computeren.
    - God udførelse.
  - Mobile OS:
    - Brugernemhed
    - Energi-Effektivitet ift. Lille enhed.
- Superuser Perspektiv:
  - Interprocess Communication: Det er der, hvor Klienten og Modtageren kommunikerer med hinanden indenfor et fælles maskine.
- System Perspektiv:
  - Allokerer Ressourcer på den rigtige måde ved at udnytte plads rigtigt i memory.
  - God Interface til Hardware.
  - Drivers og Firmware skal virke fint og effektivitet ift. Ressource Allokering.

Repetition af OS

- Vi kan se, at Operativ Systemet sørger for, at den uddeler tid til en opgave.
- OS sørger for, at benytte Memory Ressourcer samt at man også har adgang til Storage som eksempelvis læsning/skrivning.
- Så har man selve strukturen af computersystemet. Dette vil sige såsom CPU'en, Control Unit, Memory Unit og Hardware osv.

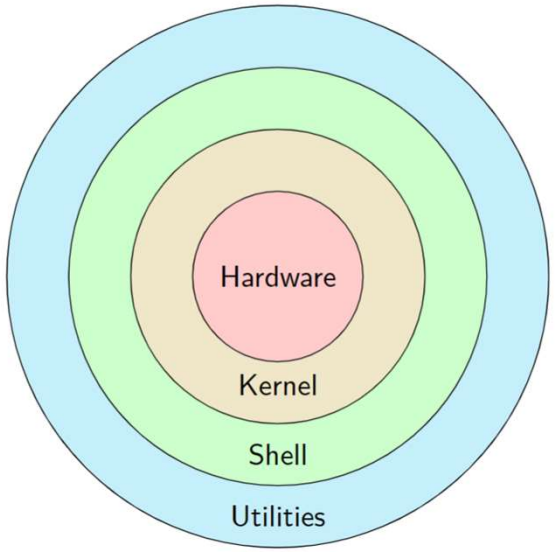


Unix Arkitektur og Filsystemer

Nu gennemgår vi Unix Arkitektur og hvordan filsystemer er generelt.

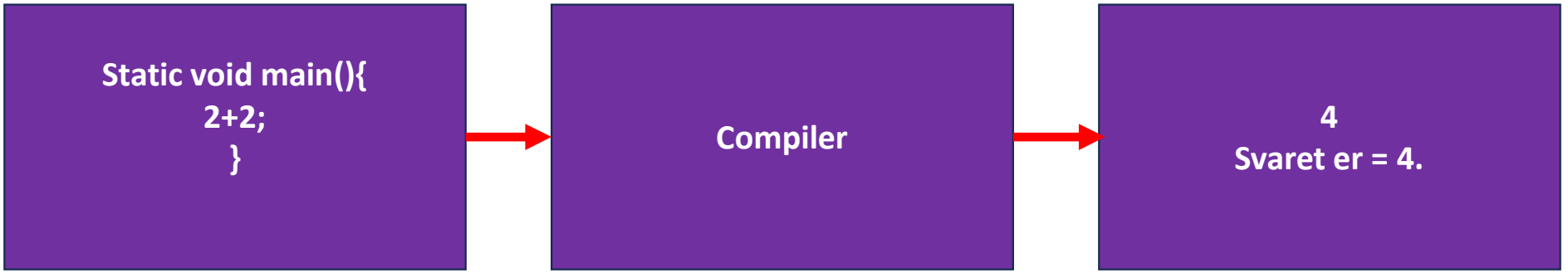
Unix Arkitektur

- Hensigten med Unix Arkitektur er, at kunne forklare hvordan Linux er opbygget.
- Unix Arkitektur består af følgende som vist på billedet til højre:
  - Hardware: Hardware er bare Computeren selv som en Fysisk Elektronisk Genstand.
  - Kernel: Den skaber kommunikation mellem Hardwaren og Software. Den sørger for, at allokere ressourcer, når der er nødvendighed for det. Eksempler kan være CPU, Memory, Input/Output Devices.
  - Shell: Det er kommandoprompten, som brugeren benytter sig af for at kunne interagerer med operativ systemet. Den skaber teknisk set kommunikation mellem brugeren og operativ systemet.
  - Utilities: Det er små software programmer, der udfører specifikt arbejder. Det kan eksempelvis være kommandoer i kommandoprompten såsom cp, mv mkdir, cat, more, less osv.



Compiler

- Du har måske hørt om Compilers.
  - Compilers er source-kode, der er skrevet af programmøren, der er oversat om til maskine kode eller eksekverbarkode som computeren kan forstå og eksekvere direkte.
  - Her har vi følgende Eksempel.



Virtual Memory

- Virtual Memory er der, hvor der bliver dannet ekstra plads i Hardwaren.
  - Dette kan ske, når RAM eller Random Access Memory er fyldt ud.

Physical Memory

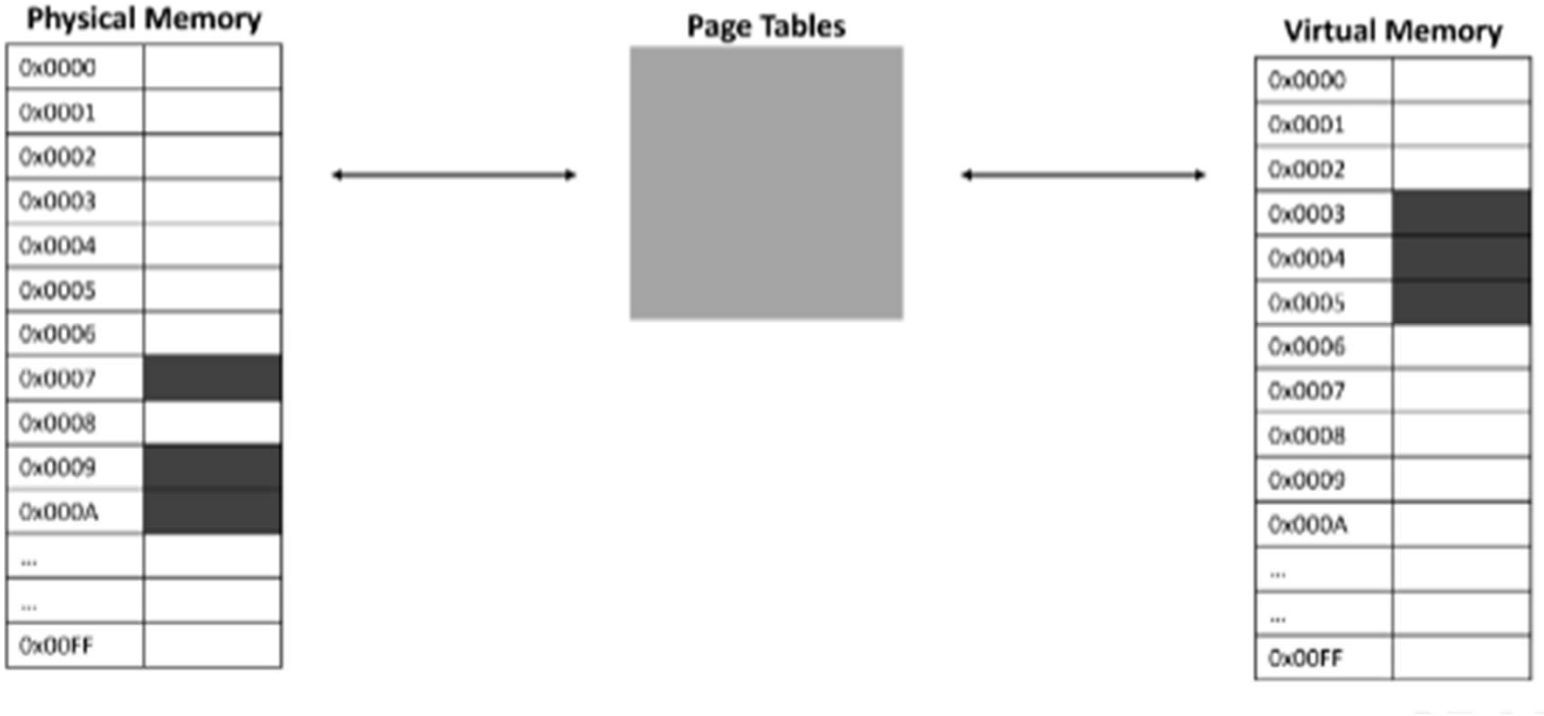
- Fysisk Memory er vores RAM som også er kendt som Random Access Memory.
  - Hensigten med RAM er, at gemme data som bliver brugt på nuværende tidspunkt.
  - RAM har begrænset plads, og er meget hurtig.

Page Table

- I sidste slide gennemgik, hvad forskellen mellem Virtual Memory og en Fysisk Memory var.
  - Vi fandt ud af, at vi havde Virtual Memory som dannede ekstra plads i Hardware, når den Fysiske Memory (RAM) var fyldt.
- For at kunne forbinde de to Memory sammen, anvender man Page Tables.
  - Forestil dig, at du befinder dig i et bibliotek men har udfordring med at forstå, hvordan man skal lede efter et specifik bog ved reolerne. Det er her, hvor du skal anvende Page Tables.
  - Den primære opgave med Page Table er at kunne ”mappe” addresser mellem virtual- og fysisk memory, således at Page table kan finde lokationen af den aktuelle data i Fysisk Memory.
  - Hvorimod hvis der mangles data i Fysisk Memory, og at dataet er midlertidig gemt i Virtual Memory, så vil Page Table hjælpe Operativ Systemet med at overføre data over i det fysiske memory.
  - På den måde kan der let veksles data mellem fysisk og virtuel memory vha. Page Tables.

Page Table

- For at kunne give et overblik over Page Table mellem Virtual Memory og Fysisk Memory. Så har vi tilføjet et billede fra Undervisningen her.

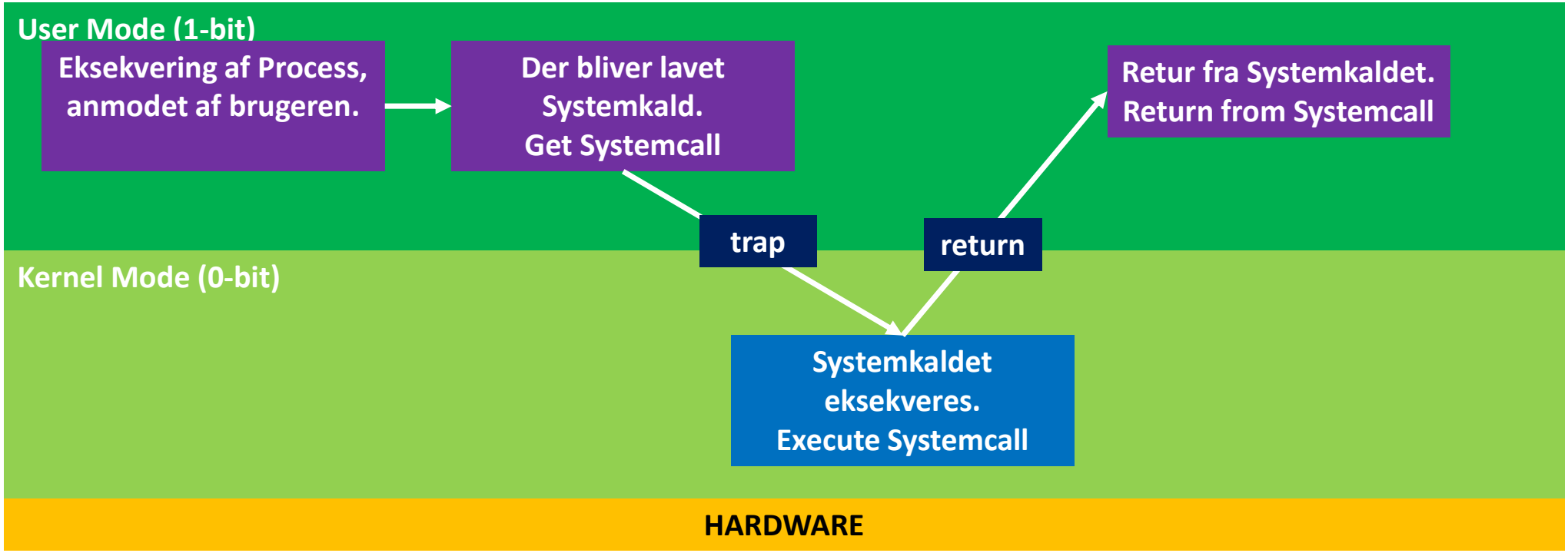


Dual Mode Operation

- Den primære hensigt med Dual Mode Operation er at kunne operere CPU'en i to tilstande. Den ene er User-Mode tilstanden og den anden er Kernel Mode tilstanden.
  - Det kan tydeligt ses, at når en bruger befinder sig i systemet, så ønsker brugerne at få adgang til Hardwarens Data.
  - For, at kunne få adgang til Hardwarens Data skal brugeren igennem Kernel.
  - Når man eksekvere et program, laver User Mode et signal kald til Kernel.
  - Dette resulterer, at der sker en "trap", som skifter fra User Mode til Kernel Mode.
  - Når man kommer til Kernel Mode, så bliver processen eksekveret.
  - Efter eksekvering af processen, bruges return til at kunne komme tilbage til User Mode.
- NOTE: Når man befinder sig i User-Mode er man i 1 bit. I Kernel Mode bliver man til 0 bit.

Dual Mode Operation

- Nu viser vi et billede, som skal illustrere Dual Mode Operation.



Filesystemers Opbygning

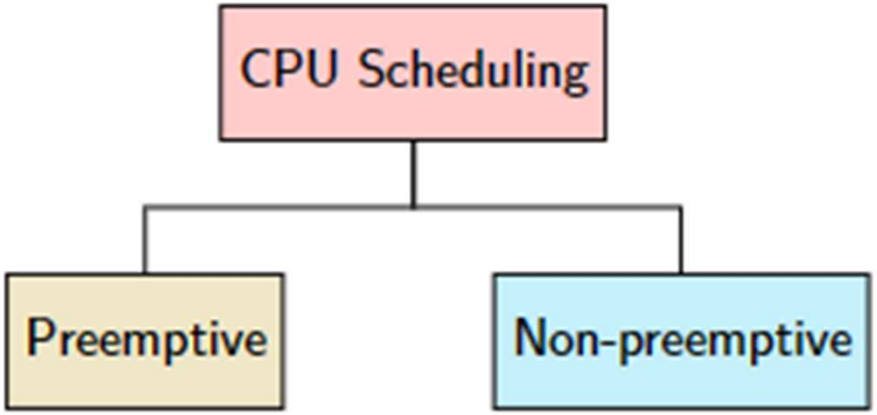
- Nu introducerer vi lidt til Filsystemer, således at det giver bedre mening.
- Når vi snakker om Filsystemer opbygning, så kan det opdeles i en logisk, fysisk og virtuel del.
- Logisk Filsystemer: Det repræsenterer filer og mapper sådan, at de er organiseret og tilgået gennem operativ systemet. Det definerer strukturering og navngivning således at man kan holde styr på filerne.
- Virtuel Filsystemer: Det er en abstrakter lager, som tillader operativ systemer i at interagerer med forskellige typer af storage devices samt i at få adgang til filsystemer på en nemmere måde.
- Fysisk Filsystemer: Dette referer til den aktuelle format og struktur brugt på storage device til at kunne lage og organisere filer. Det repræsenterer data som er fysisk gemt i storage devices.

Scheduleringsalgoritmer

Nu introduceres der til Scheduleringsalgoritmer såsom Round Robin osv.

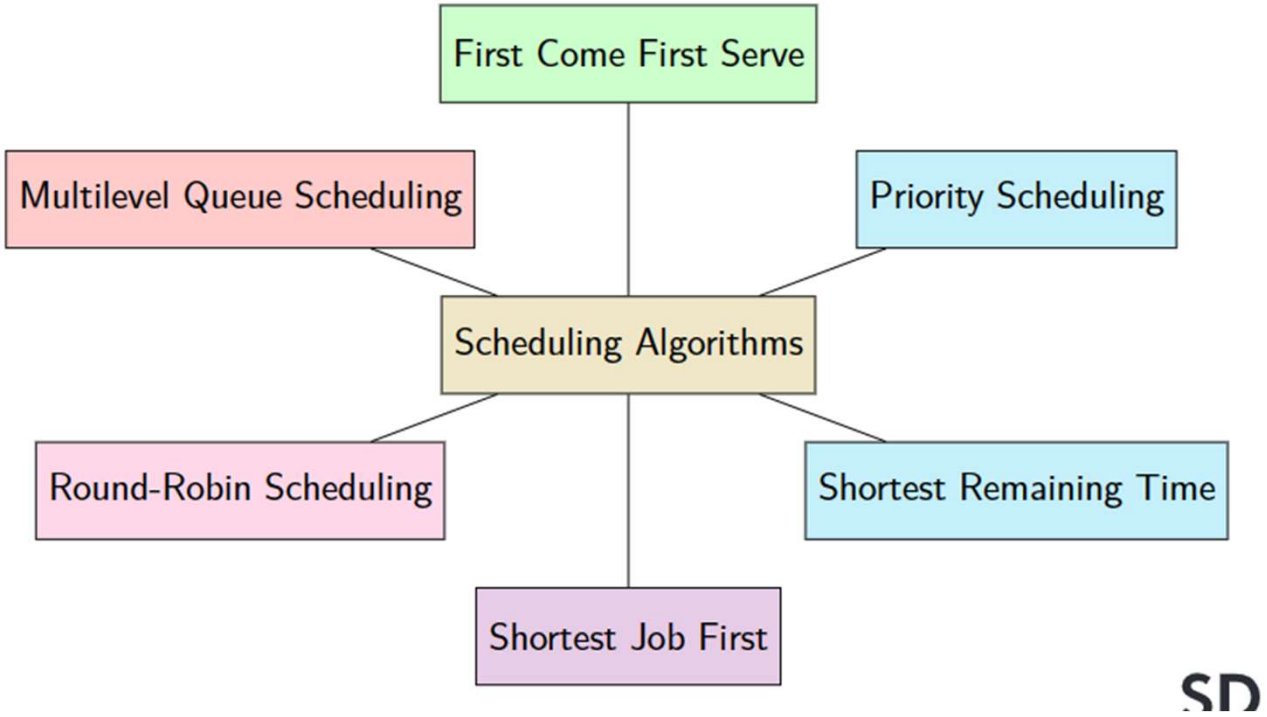
Scheduleringstyper

- Inden vi snakker om Scheduleringsalgoritmer, er det nødvendigt at snakke om Typerne.
- Vi har i tilfældet to Scheduleringsalgoritmer:
  - Non-Preemptive: Det er der, hvor CPU'en eksekvere processen fuldt ud.
  - Preemptive: Det er der, hvor CPU'en eksekvere kun delvis af processen.



Scheduleringsalgoritmer

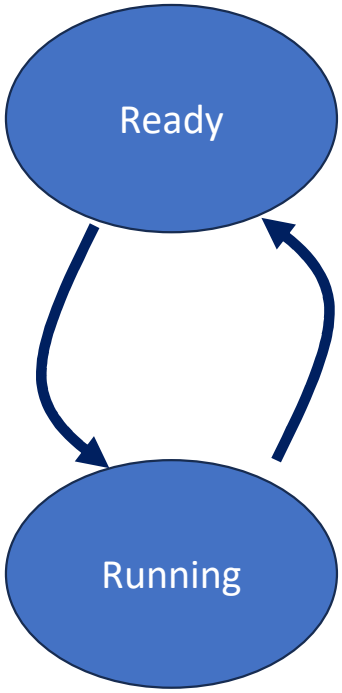
- Hvis vi går videre og snakker om Scheduleringsalgoritmer. Så findes der mange forskellige. Her har vi kun inkluderet et billede, for at give overblik.
- I næste slide vil vi introducere til Round-Robin Princippet, da det er den som er vigtigst.



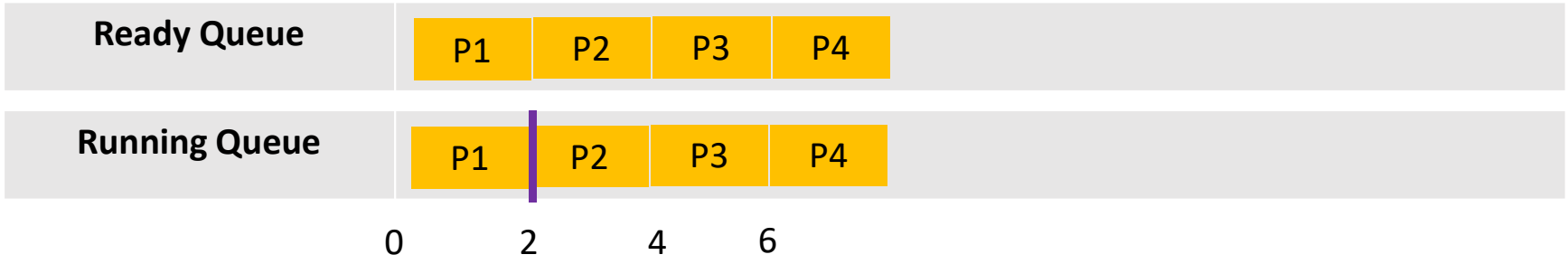
Round Robin Princippet

- Hensigten med Round-Robin princippet er at kunne eksekvere processer gentagne gange, på baggrund af deres Kvantum.
  - Vi kan se, at vi henter Ready-Queue, som befinder sig inden i RAM. Derfra vælger vi en Proces, og tager den i Running-Queue i CPU'en. Det betyder bare, at vi udvælger en process og tager den med i CPU'en og derved eksekvere det.
  - Men når vi snakker om Round-Robin princippet, så bruger CPU'en noget som hedder Kvantum. Kvantum betyder, at hvis en proces kommer i Running-State og CPU'ens Kvantum er 2. Så bliver Processen udført 2 gange og derved sat tilbage til Ready-State.
  - Vi kan se, at vi bruger Sekvenser til at udføre Processer.
  - CPU'en er IKKE Non-Preemptive og det betyder at efter at have eksekveret 2 gange ved CPU'en, bliver CPU'en klar til en nyt proces.
  - Context Switching betyder, at man gemmer den kørende proces og bruger den ind. Når den gamle kommer igen, så starter den fra Kvantumsafslutning.

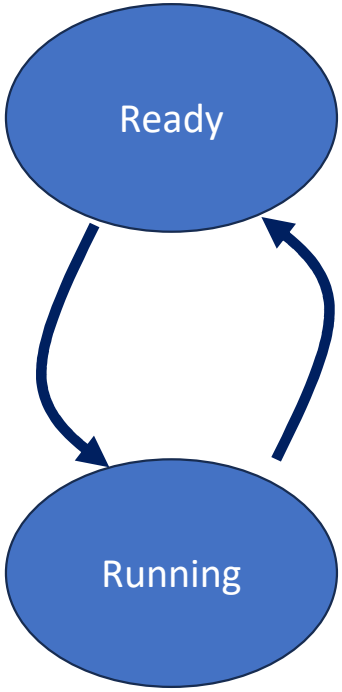
Round Robin Princippet



Process No:	Arrival Time	Bust Time	Completion Time	TAT	WT	RT
P1	0	5				
P2	1	4				
P3	2	2				
P4	4	1				

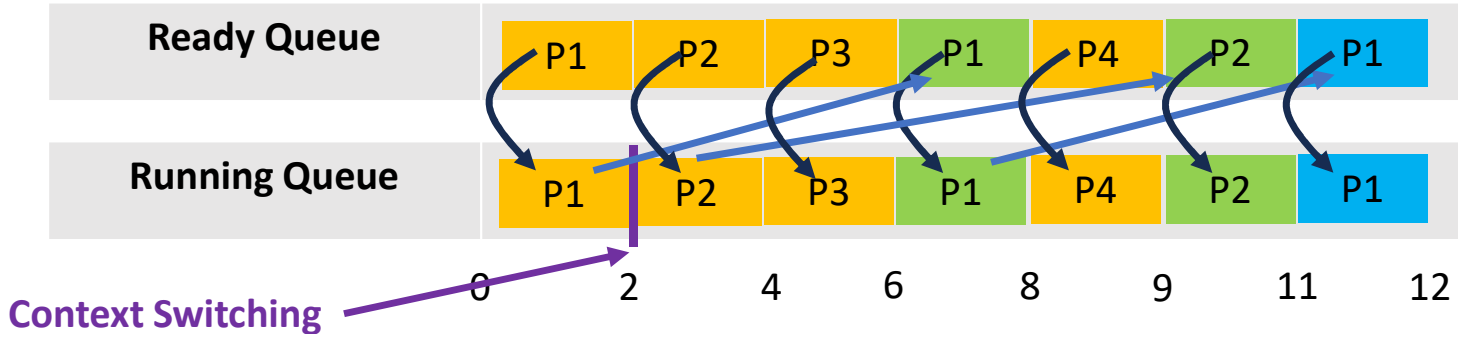


Round Robin Princippet



Det er vigtigt, at sige at processer der når 0, bliver ikke kørt videre. Hvorimod de andre som ikke har nået 0 ved Bust time, bliver kørt videre hvor de starter i Ready Queue og så over i Running Queue.

Process No:	Arrival Time	Bust Time	Completion Time	TAT	WT	RT
P1	0	5=>(5-2)=3 => (3-2)=1				
P2	1	4=>(4-2)=2 => (2-2)=0				
P3	2	2=> (2-2)=0				
P4	4	1=kørt en gang				



SLUT 2

Lavet af: Vivek Misra