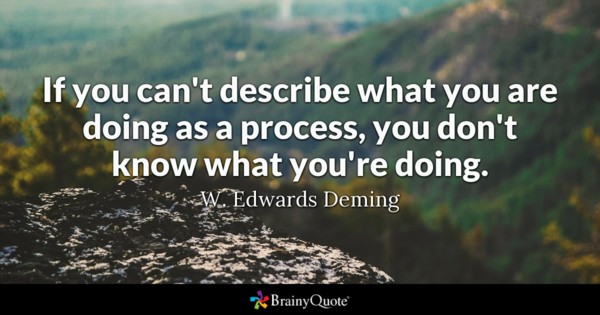
# Process 01

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf5:** anvende centrale metoder og teknikker til at designe og konstruere programmer som samarbejdende processer […] * **1Tv1:** udviklingsbaseret viden om tidssvarendeoperativsystemer […] herunder deres opbygning og faciliteter * **1Tf2:** anvende centrale faciliteter i […] operativsystemer på en hensigtsmæssig måde * **1Tk1:** i en struktureret sammenhæng, tilegne sig ny viden og færdigheder om nye operativsystemer […] |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan forstå, hvad en **proces** er, samt hvorledes den afvikles i en computer under administration af et **operativsystem**. |
| **Din forberedelse** | Programmeringssporet:   * [How does a CPU work?](https://www.youtube.com/watch?v=RIrQwd_srRw) (video: 7:50) * [Process Management (Processes and Threads)](https://www.youtube.com/watch?v=OrM7nZcxXZU) (video: 7:31) * [Process State](https://www.youtube.com/watch?v=jZ_6PXoaoxo&list=PLBlnK6fEyqRiVhbXDGLXDk_OQAeuVcp2O) (video: 8:52) * [Process Control Block](https://www.youtube.com/watch?v=4s2MKuVYKV8&list=PLBlnK6fEyqRiVhbXDGLXDk_OQAeuVcp2O) (video: 7:01) * [OS]1 (uddrag vedlagt i Planer):   + Afsnit 2.1 – 2.3, Operation System Concepts, Components and Architectures (side 19 – 29)   + Afsnit 4.4: Memory Management (side 75 – 80) * Genlæs: [A Basic Introduction To C# Unit Test For Beginners](https://www.c-sharpcorner.com/article/a-basic-introduction-of-unit-test-for-beginners/)   + Bemærk: Vi benytter os af .Net og ikke .Net Framework   1 [OS]: “Operating Systems – A Spiral Approach”, Elmasri, Carrick & Levine |

Denne opgave introducerer temaet for denne og næste uges opgaver, nemlig **processer og tråde**. Som du nok har opdaget, kan alle computere, mobiletelefoner, tablets, mm. i dag køre mange programmer/apps på en gang, eller mere korrekt, mange processer på en gang. Dette er en fantastisk og helt nødvendig egenskab, vi ikke kan undvære, men introducerer også en masse spændende problemstillinger, som du skal kunne arbejde med i programmering. Der er dog en del begreber knyttet til processer og tråde samt operativsystemer, der først skal på plads. Dette skal du arbejde på i denne opgave.

# Dagens ord:



Øvelse 1: Terminologi

Del teamet op i to mindre grupper, og brug Ordet rundt til at reflektere over følgende:

* Formålet med en ”*CPU*” og ”*RAM*”
* Forskellen mellem et “*Program*” og en “P*roces*"
* Formål og rolle for en ”*Process Control Block*” (”*PCB*”)
* Forskellen mellem “*User or application processes*”, “*System program processes*” og “*OS processes*”
* Forskellen mellem ”*privileged mode*” og ”*user mode*”

Sørg for, at alle får mulighed for at tale.

*Tidsramme: ca. 20 minutter*

**Benyt parprogrammering i det følgende.**

# Øvelse 2: Procesmodellen

Beskriv følgende sammen med din makker:

* *Process state*
* *State transition* ifm. proces-tilstande
* Figur 2.2, s. 26 i [OS], herunder de 8 tilstandsændringer
* Forskellen på ”*Wait-mode*” og ”*Ready-mode*” i figur 2.2
* Forskellen på tilstandsændringerne ”*5 – Interrupted*” og ”*3 – Needs something*”
* Hvad kræves af OS for, at tilstandsændringen ”*4 – Got what it needed*” kan opstå?
* Hvad kræves af et multi-process OS for, at tilstandsændringen ”2 *– Gets CPU time*” kan udføres, når der er flere processer i tilstand ”*Ready*”?

# Øvelse 3: Inspektion af Jobliste (Task Manager)

I ”gamle dage” betød tastekombinationen ”Ctrl+Alt+Delete”, at PC’en genstartede (*reboot’ede*), men i dag giver tastekombinationen (i Windows 10) bl.a. adgang til ”Jobliste”, hvilket du skal se nærmere på i denne øvelse.

|  |
| --- |
| **Advarsel:** Via ”Jobliste” kan du ændre betingelser for kørende processer, herunder standse dem. Hvis du manipulerer med operativsystemets processer, kan du ændre betingelser for andre processer, som afhænger af dem, hermed kan dit system blive ustabilt og uforudsigeligt.    Det er ekstremt sjældent, at sådanne konsekvenser ikke kan neutraliseres med en genstart af systemet, men vi giver ingen garantier.    Det anbefales derfor, at du **ikke** manipulerer med processer, medmindre du har et komplet overblik over konsekvenserne. |

## Øvelse 3.1: Udforsk og overvej

Gå sammen med din sidemakker, og udfør følgende:

* Åbn ”*Jobliste*” på din PC, og udforsk de enkelte faneblade overfladisk (det er helt ufarligt *så længe du ikke piller*… 😊).
* Udforsk fanebladet ”Processer” nøjere.
  + Bemærk, at flere processer har en foranstillet pil (”>”) som gør, at du kan få yderligere informationer om bagvedliggende processer. Bemærk, også at du får flere muligheder, hvis du højreklikker på et procesnavn.
* Prøv om du kan gætte, hvad forskellen mellem ”*Apps*”, ”*Baggrundsprocesser*” og ”*Windows-processer*” er?

# Øvelse 4: C# processer

Du skal gå ind i materialefolderen i GitLab og hente dagens materiale i bilaget:

* *Ex51-Process01-Bilag.docx*

**Benyt parprogrammering i de følgende øvelser.**

## Øvelse 4.1: Hvad kommer der til at ske?

Udfør følgende:

* Læs først koden igennem og se, om du kan forudsige, hvad den gør.
* Vend dine tanker med din sidemakker.

## Øvelse 4.2: Flere processer

Udfør følgende:

* Opret en konsolapplikation med projektnavnet “Process01” (VIGTIGT!)
* Erstat klassen ”*Program*” med koden i bilaget
* Kør koden, og lad programmet starte 3 processer
  + Var din forudsigelse fra tidligere korrekt?
* Åbn ”*Jobliste*”, og undersøg de kørende Apps
  + Bemærk, hvad der sker, når dine processer standser

## Øvelse 4.3: Virtuel hukommelse

Du har måske tænkt tanken: ’*Hvad så, hvis jeg løber tør for RAM? Fryser maskinen så?*’

I ovenstående tilfælde vil din maskine benytte sig af den virtuelle hukommelse, altså i det tilfælde, at du har benyttet dig af alt den RAM, du har i din maskine (for eksempelvis: 8GB eller 16GB RAM). Men ”Hvad er virtuel hukommelse?”. Virtuel hukommelse er i virkeligheden blot en sidefil, der oprettes på en af dine eksisterende partitioner (en opdeling af diskens lager) i maskinen, hvilket også betyder, at hastigheden af denne sidefil er begrænset af hastigheden på din disk. I gamle dage, hvor man havde harddiske (HDD), var denne sidefil meget langsom. Dette er dog ikke et problem, man som sådan bemærker i moderne maskiner, hvor man har Solid State Diske (SSD).

Formålet med denne øvelse er at fylde din RAM ud og observere, at denne sidefil tages i brug.

Udfør følgende:

* Åbn ”*Avancerede systemindstillinger*”, og klik på ”*Indstillinger*” ved ”*Ydeevne*”. Under indstillinger for ydeevne, skal du vælge fanen ”*Avanceret*” og se, hvor stor en sidefil, der er tildelt som virtuel hukommelse. Husk dette tal
* Åbn ”*Denne pc*” og start det ovenstående program igen, denne gang med markant flere processer (prøv dig frem, notér, hvornår at sidefilen tages i brug)
  + Bemærk, hvad der sker med den ledige plads, hvis du vælger at starte programmet med mange processer, og bemærk, hvad der sker med den ledige plads, når dine processer standser
  + **Note*:*** Hvis du ikke kan se nogen ændring under ”Denne pc”, så benyt dig af F5

# Øvelse 5: Process Control Block (PCB)

I denne opgave skal du oprette en C#-applikation, hvor du skal implementere en process control block – denne øvelse arbejder vi videre på i næste opgave (Ex52), hvor vi vil implementere en prioritetskø (priority queue).

## Øvelse 5.1: Oprettelse af PCB

Udfør følgende:

* Opret en ny konsolapplikation med navnet ”Scheduling”
* Opret en C#-klasse med navnet ’PCB’ til at repræsentere en ’process control block’, som du har hørt om i forberedelsen. PCB-klassen er meget forsimplet og har kun følgende properties og metode:
  + public string ProcessName { get; set; }  
    Angiver navnet på processen, som PCB’en repræsenterer.
  + public int ProcessPriority { get; set; }  
    Angiver processens prioritet, hvor 1 er højeste prioritet og 2, 3, osv. er tilsvarende lavere prioriteter. Prioriteten anvendes til at placere processen korrekt i en prioritetskø.
  + public ProcessStateType ProcessState { get; set; }  
    Angiver processens prioritet, hvor 1 er højeste prioritet og 2, 3, osv. er tilsvarende lavere prioriteter. Prioriteten anvendes til at placere processen korrekt i en prioritetskø.   
    Enumeration-typen ProcessStateType skal defineres som:
    - public enum ProcessStateType { New, Ready, Run, Exit, Wait };
  + public override string ToString()  
    Returnerer en tekstrepræsentation af PCB’en med formatet: <navn>(<prioritet>). F.eks. for en proces med navnet ’A’ og prioriteten 2, returnerer ToString() tekststrengen ”A(2)”.

## Øvelse 5.2: Oprettelse af unittest

Du og dit team skal udarbejde en unittest til opgaven.

**Hint:** Hvis I har glemt, hvordan I opretter unittest: [A Basic Introduction To C# Unit Test For Beginners](https://www.c-sharpcorner.com/article/a-basic-introduction-of-unit-test-for-beginners/)

## Øvelse 5.3: Upload til Google Drive

Du og dit team skal uploade jeres udarbejdede UnitTest-klasse. I skal benytte jer af følgende navnekonvention:

* Team[NR]-UnitTest.cs
  + For eksempel: Team12-Unittest.cs

Link til afleveringsmappe:

* <https://drive.google.com/drive/folders/1vXlAQnKQcCyW5pN_b8GEBepzUmTHacl1?usp=share_link>