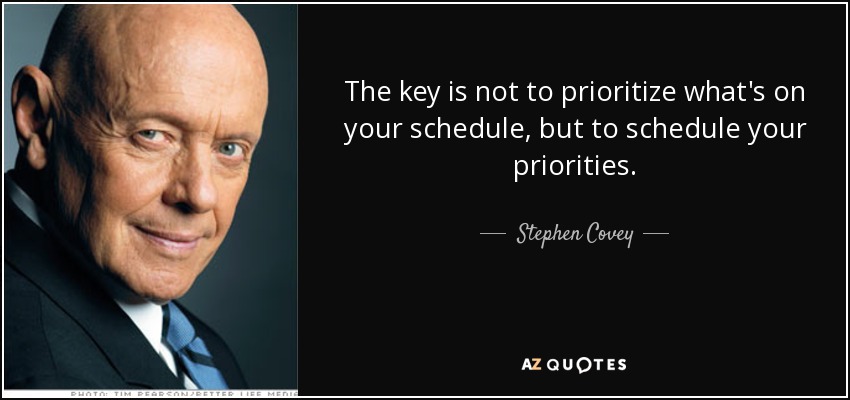
# Process 02

|  |  |
| --- | --- |
| **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:   * [Process Scheduling](https://www.youtube.com/watch?v=2h3eWaPx8SA&list=PLBlnK6fEyqRiVhbXDGLXDk_OQAeuVcp2O) (video: 11:18) * [Process Scheduling](https://www.youtube.com/watch?v=THqcAa1bbFU) (video: 5:26, det samme, men sagt med lidt andre ord) * [Queue explained](https://www.linkedin.com/learning/learning-c-sharp-algorithms/queue-explained?u=57075649) (video: 1:29) * [OS]1 (uddrag vedlagt i Planer):   + Afsnit 8–8.4, Process Management: Concepts, Threads and Scheduling  (s. 151 - 164)     - **Skimlæs afsnit 8.4.8**   + Kapitel 10, Basic Memory Management (side 209 – 223) * Du skal have løst øvelse 5.1 fra Ex51, da du ellers ikke kan løse dagens opgave – hvis du ikke nåede i mål sidste gang   1 [OS]: “Operating Systems – A Spiral Approach”, Elmasri, Carrick & Levine |

Dagens opgave dykker yderligere ned i procesbegrebet og ser bl.a. på, hvad ’process scheduling’ er, og ’hands on’ hvordan en prioritetskø fungerer.

Dagens ord:



Øvelse 1: Terminologi

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

* *“Process scheduling” (for eksempel:* ”*FIFO/FCFS*”,”Priority”)
* ”*Scheduling queues (Job queue, Ready queue)*”
* ”*Enqueue*”, ”*Dequeue*”
* ”*Time sharing*”,
* *“Context switch*”

Sørg for, at alle får mulighed for at tale.

*Tidsramme: ca. 20 minutter*

**Benyt parprogrammering i det følgende.**

# Øvelse 2: Mere om procesmodellen

Beskriv følgende sammen med din makker:

* Indholdet i en Process Control Block (PCB)
  + Forsøg at forklare så mange punkter, som du kan se i listen på side 153 i [OS], men brug ikke mere 5 minutter i alt
* *FCFS*, herunder ”*cooperative multitasking*”
* *“Priority scheduling algorithm*”, herunder ”*starvation*”
* “*preemption*”
* Forskellen på “*CPU bound*” og “*I/O bound*”
* “*multilevel queuing*”
* “*feedback*” ifm. multilevel queues

# Øvelse 3: Prioritetskø

Inden du tager hul på denne øvelse, kan du eventuelt hurtigt genopfriske, hvad en kø er her: [Queue explained](https://www.linkedin.com/learning/learning-c-sharp-algorithms/queue-explained?u=57075649) (video: 1:29).

Køen af processer i tilstanden ”*Ready*” implementeres ofte som en *Prioritetskø*. En prioritetskø adskiller sig fra en almindelig kø ved, at elementerne indsættes med en given prioritet. Hvis elementet har højere prioritet end det sidste element i køen, rykker det fremad i køen, indtil det støder på et element med samme prioritet. Her indsættes det nye element foran alle elementer med lavere prioritet. Hvis alle elementerne i køen har lavere prioritet end det nye element, bliver det således indsat forrest i køen.

Elementerne tages stadig ud af køen fra den forreste ende, idet elementerne er sorteret efter prioritet.

Du skal nu implementere og teste din egen prioritetskø i C#. Hertil skal du bruge den testkode, som er angivet i filen ’Ex52-Process02-UnitTest.docx’ udleveret i materiale-folderen på GitLab.

* Åben projektet ”Scheduling”, som du udarbejde i sidste opgave (Ex51)
* Kopier testkoden fra ”Ex52-Process02-UnitTest.docx” ind i projektet
* Opret C#-klassen ”PriorityQueue” i dit ”Scheduling” projekt, og implementér følgende metoder:
  + public void Enqueue(PCB pcb)
  + public void Dequeue()
  + public void Reprioritize(string processName, int newPriority)
  + public override string ToString()

**Bemærk**: Det er helt op til dig at vælge en passende intern datastruktur til at repræsentere køen i PriorityQueue-klassen. Det anbefales desuden, at du inspicerer unittestene, for at få et bedre indblik i, hvordan du skal løse opgaven.