# ISU: 1 Programs in relation to the OS and Kernel

**User space kontra kernels pace**

* User space: restricted instructions
* Kernel space: priviliged instructions

## Processes and threads

**Proces:**

* OS styrer processer. Lave, slette, allokere ressourcer, ind og ud af memory
* En del af et program der bliver afviklet –
* En entitet hvori programmet er – program-counter, processor registre, variabel værdier, process stack, variable 🡪 context
* Stack, heap, registers, descriptors mm. ????????????
* Processor har deres eget virtuelle memory space - to processor kan ikke tilgå hinandens
* Processor kan ”spawne” andre processor – evt. sig selv
* Processer kan også ”spawne” tråde i dens eget memory space
  + Kan indeholde flere tråde – mindst en!

**Tråde:**

* Kalde også tasks eller jobs – da det er hvad det er – en opgave der skal udføres
* Tråde deler proces memory space – der kan være flere tråde i en proces
  + Derfor skal man passe på IKKE at tråde ikke overskriver hinandens data
* Oftest vil man arbejde med flere tråde

## Threading Model

User level threading – Kernel level thread – Hybrid level threading

**User level:**

* Ingen kernel support
* Hurtige context skift, da kernen ikke skal involveres
* Ikke muligt med flere kerne, da kernen ”kigger på” mens alt foregår i user space

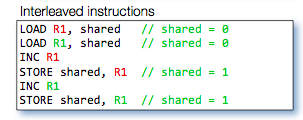
**Kernel level:**

* Kernen skal kende til threading
* Mapper direkte til en tråd, som scheduleren kan kontrollere
* Effektivt ved flere kerner
* Dyrere context switches -

**Hybrid level:**

* Komplekst
* Kræver god koordination mellem user og kernel space scheduler
  + Ellers er det ikke optimalt, da de skal vente på hinanden

### Multithreading – fordele og ulemper

* **Fordele**
  + Prioriteter
  + Udnyttelse af ressourcer – så der ikke skal ventes på I/O
* **Ulemper**
  + Shared data problem
  + 
  + Starvation – LP tråde kommer aldrig til – svært at gennemskue

## Process anatomy

Context:

* ”Miljøet” for den kørende proces – værdier i CPU registre, adress space mm.
* Switch foretages af OS’et der suspenderer den kørende for at genoptage en anden proces
* Steps: Interrupt igangværende, save context, restore context, resume execution

OS’et schedulerer proceserne for afvikling:

* Pre-emptiv: Tasks kan blive interruptet af scheduleren
* Non pre-emptiv: Tasks kan ikke blive interruptet, og skal selv ”slippe” CPU’en

## Virtual Memory

* Memory management unit (MMU)
  + Mapper virtuelt hukommelse i den fysiske
* Processer skal ikke selv håndtere eller bekymre sig om den reelle fysiske hukommelse
  + Den virtuelle fremstår for processen som uendelig stor
* Dette gør også at processer ikke deler memory space 🡪 øget sikkerhed

Stack:

* ”Gror” nedad
* Lokale variable, returværdier
* LIFO

Heap:

* ”Gror” opad
* ”Free-store” ?
* Dynamisk allokeret hukommelse

## Threads being executed on CPU, the as- sociated scheduler & Cache