Toegepaste Informatica

MBI07a

2022-2023

# Computer Systems

# Processen

Jeroen Jean, Rudi Swennen, Tiebe Van Nieuwenhove, Frédéric Vogels

# Deel 1: Process Scheduling

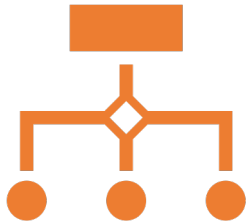
# Programma’s en processen

Wat is een programma?

# Programma’s

**Computers hebben een heleboel programma’s**

Browsers, tekstverwerkers, rekenmachine, bestandverkenner, terminal, shell



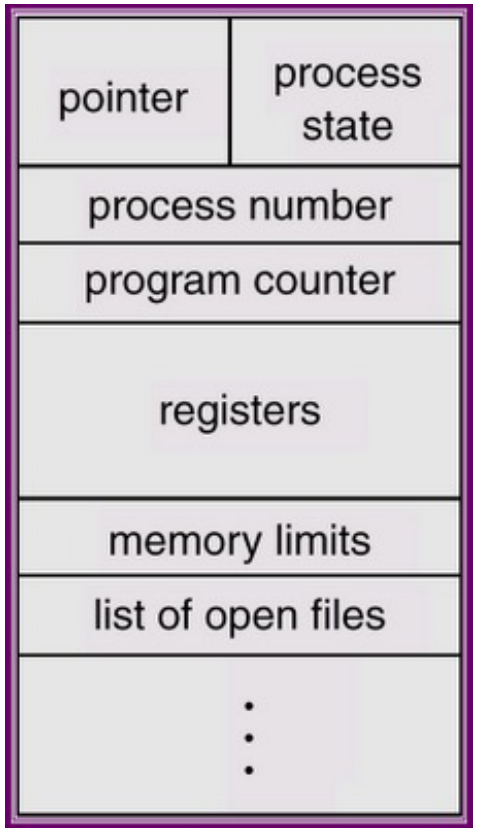
**Ieder programma heeft code, instructies die het uitvoert**

Instructies: een serie machineinstructies

Ieder programma heeft een ‘pointer’ om bij te houden welke instructie het moet uitvoeren

# Proces

- Een programma in uitvoering
- Houdt info bij:
  - Status programma
  - Gebruikte bestanden
  - Geheugengebruik
  - Procestoestand (hierover later meer)
  - …



# Processen uitvoeren

- De processor voert processen (een serie instructies uit)
- In de simpelste vorm neemt een processor een proces, voert dit uit, en gaat naar het volgende
  - Bijvoorbeeld, processen A, B en C

# Processen uitvoeren: een reality check

- In realiteit is een proces echter zelden rechtlijnig, 1 rekensom
  - Wat doet een tekstverwerker als er niks getypt wordt? Wachten
  - Wat doet een browser als je bezig bent met lezen? Wachten


- Processen zijn vaak onderbroken, met pauzes waarin de processor niks doet

  ➔ INEFFICIËNT!

- Processen moeten op elkaar wachten

# Multiprogrammatie

- Kunnen we processen door elkaar plannen?

- Wanneer het 1e proces wacht, laten we het volgende lopen
  - Optimaler gebruik van de processor

- Oplossing? We gaan processen ‘inplannen’ met een ***scheduler***

# Scheduling algoritmes

- Niet preëmptieve algoritmes
  - Kunnen lopende processen **NIET** onderbreken
- Preëmptieve algoritmes
  - Kunnen lopende processen onderbreken

# First come first served

- Wachtrij volgens aankomst (First In, First Out)


| Proces | Aankomst | Uitvoertijd | Start | Omlooptijd | Relatieve omlooptijd |
|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 1 |
| B | 1 | 100 | 1 | 100 | 1 |
| C | 2 | 1 | 101 | 100 | 100 |
| D | 3 | 100 | 102 | 199 | 1,99 |

# First come first served (niet preëmptief)

**Voordelen**

- Goed voor processen met lange CPU bursts

**Nadelen**

- Korte processen die later binnen komen zullen relatief lang moeten wachten
- Processen die vaak op I/O wachten moeten telkens achteraan aanschuiven

# Shortest Job First

- Het kortste proces wordt altijd als eerste uitgevoerd


| Proces | Aankomst | Uitvoertijd | Start | Omlooptijd | Relatieve omlooptijd |
|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 100 | 2 | 102 | 1,02 |
| C | 1 | 1 | 1 | 1 | 1 |
| D | 1 | 100 | 102 | 201 | 2,01 |

# Shortest job first (niet preëmptief)

**Voordelen**

- Korte processen zullen snel afgehandeld worden

**Nadelen**

- Processen met langere bursts worden uitgesteld ➔ **Starvation**

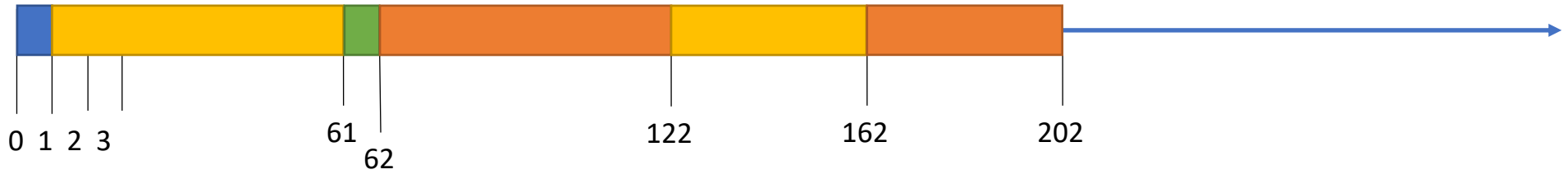
# Starving?

- Doordat er altijd andere processen voor een lang process kunnen springen, kan het zijn dat een proces eeuwig in de queue blijft staan

- Oplossing? **Aging**
  - Nadat een proces een bepaalde tijd in de queue staat het toch op de processor zetten, ookal zijn er kortere processen

# Round Robin

- Wachtrij volgens aankomst (First In, First Out)
- Processen hebben een tijdsquantum (bijv. 60)
  - Een proces kan maximaal 60 tijdseenheden per keer worden uitgevoerd




| Proces | Aankomst | Uitvoertijd | Start | Omlooptijd | Relatieve omlooptijd |
|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 1 |
| B | 1 | 100 | 1 | 161 | 1,61 |
| C | 2 | 1 | 61 | 60 | 60 |
| D | 3 | 100 | 62 | 199 | 1,99 |

# Round Robin(preëmptief)

**Voordelen**

- Minder lange wachttijden dan FCFS
- Geen starvation

**Nadelen**

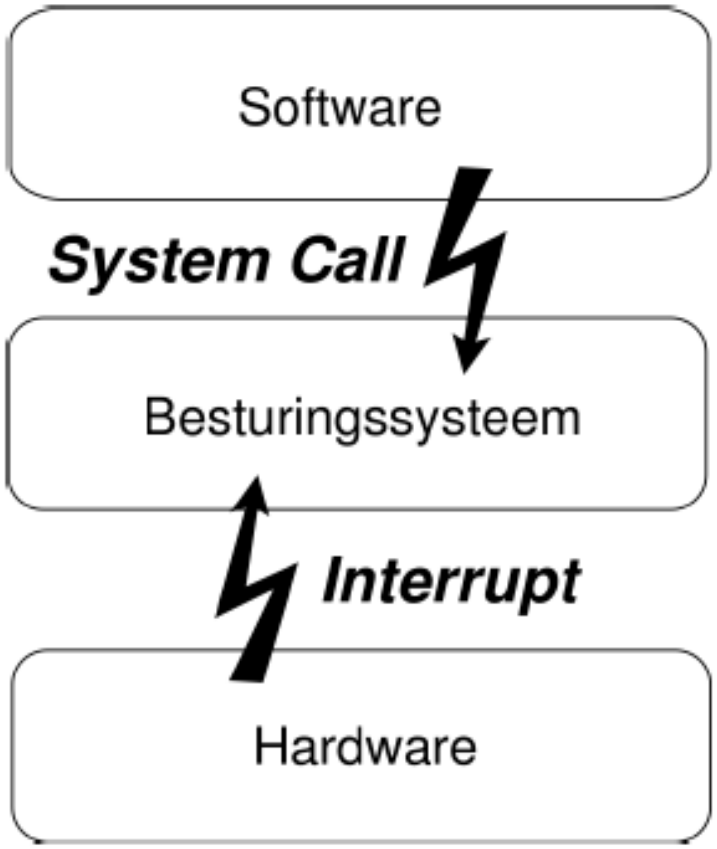
- Vaak wisselen tussen processen is trager
- Processen die vaak op I/O wachten moeten telkens achteraan aanschuiven

# Deel 2: System calls & Interrupts

# System Calls & Interrupts

- Processor voert één programma tegelijk uit
  - Besturingssysteem is zelf een programma

- Wanneer moet het besturingssysteem in actie komen?
  - Signaal vanuit de hardware
    - bijv. netwerk pakket komt aan
  - Vraag van hoger gelegen software
    - bijv. applicatie wil bestand wegschrijven



# Interrupt requests

- Stel: een netwerkpakket komt binnen op de netwerkkaart
  1. Netwerkkaart stuurt een interrupt request naar de processor
     - = berichtje dat het OS iets moet afhandelen
  2. Processor onderbreekt het actieve programma
     - Springt naar code van het OS om de request af te handelen

- Geen interrupt requests = geen onderbreking van het actieve programma (na iedere instructie controlleert PC op interrupts)

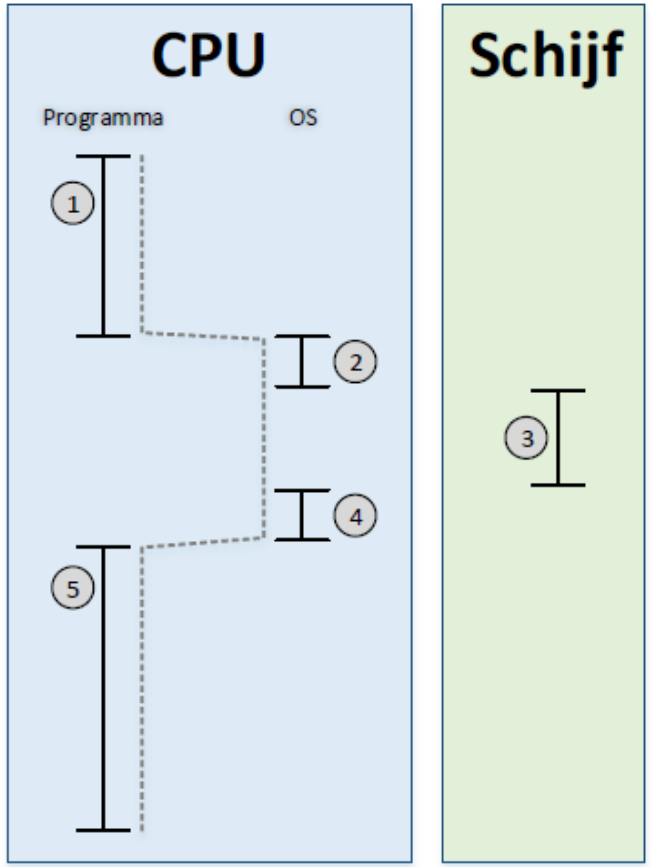
# Wat als er een interrupt binnenkomt?

1. De processor stopt het proces in uitvoering
2. De gegevens (instruction pointer, registers, …) over het proces worden in het geheugen opgeslagen
3. De processor laadt de nodige code in om de interrupt af te handelen
4. De processor handelt de interrupt af
5. De processor laadt de oorspronkelijke code terug in

Stap 2, 3 en 5 = ***Context Switch***

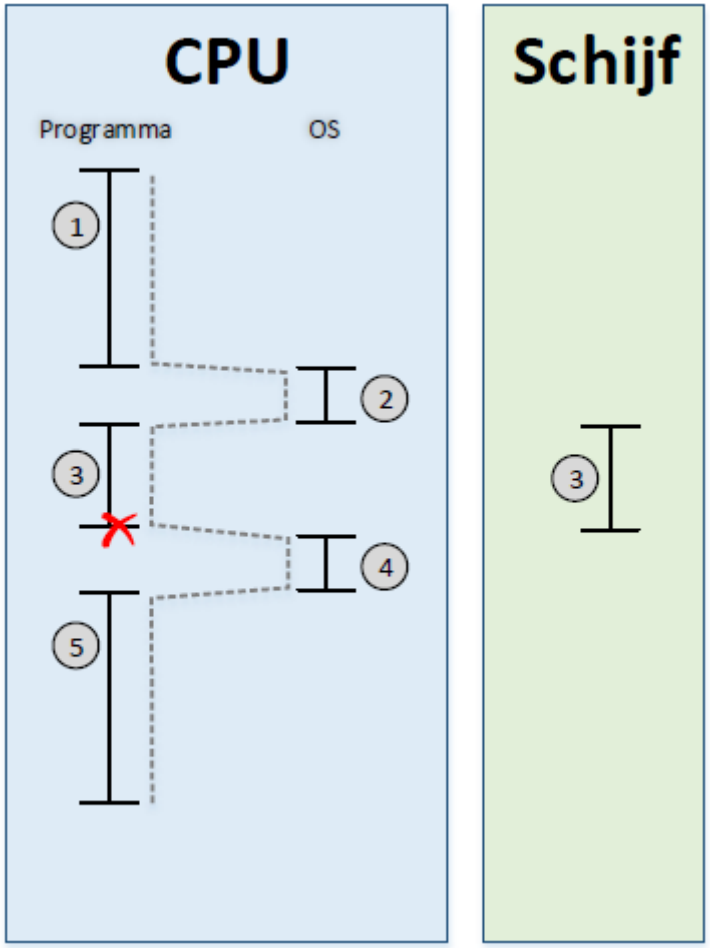
# Snelheidswinst interrupts: zonder interrupt

- Het programma (1) moet een bestand van de schijf inlezen
- Zolang het leeswerk (3) bezig is, moet het proces wachten op de leesoperatie
- Wanneer de leesoperatie gedaan is, kan het process hervatten (5)



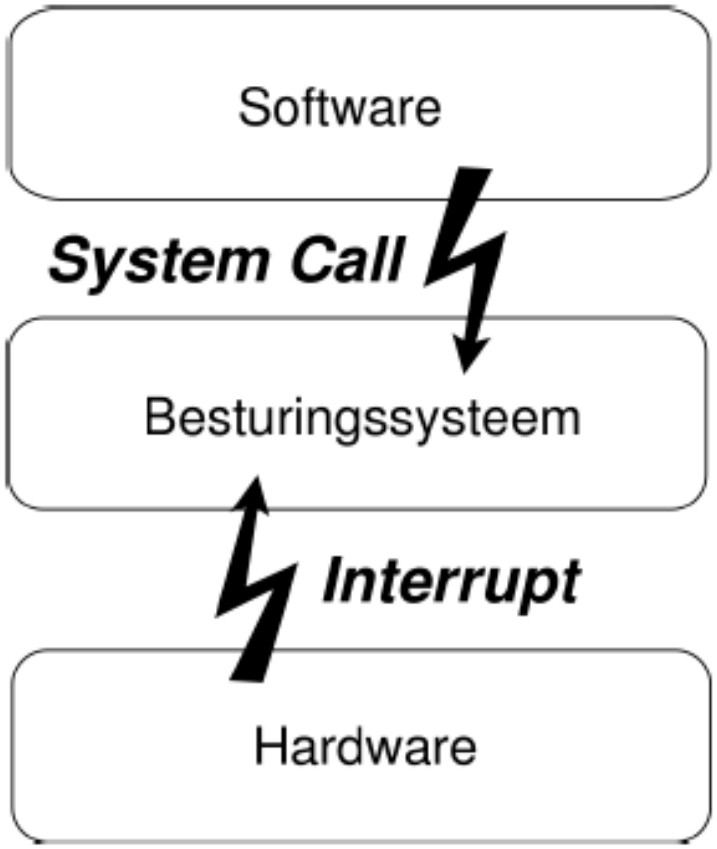
# Snelheidswinst interrupts: met interrupt

- Terwijl de leesoperatie bezig is, runt de CPU een ander proces (3)
- Wanneer de leesoperatie gedaan is, krijgt de CPU een interrupt (kruisje)
- Dit is het teken om het oorspronkelijke proces te hervatten



# System calls

- Zoals interrupts signalen van de hardware aan het OS zijn, zijn system calls signalen van de software aan de OS



# System calls: voorbeelden

- Opstarten en beëindigen van programma's
- Programma laten wachten op een gebeurtenis
- Toekennen en vrijgeven van geheugen
- Bestandsbeheer
  - Creëren en vernietigen van bestanden en directories
  - Openen en sluiten van bestanden
  - Lezen en schrijven in bestanden
  - Bepalen van toegang tot bestanden

# Voorbeelden

- read() leest bestanden in
- write() schrijft output
- exec() start nieuw programma op
- …

- Wil je hiermee eens op verkenning?
  - Voer in je terminal eens het commando strace [commandonaam] uit
  - Dit toont je de system calls die gebeuren voor dat proces