

# IN1805 I - Operating System Concepten

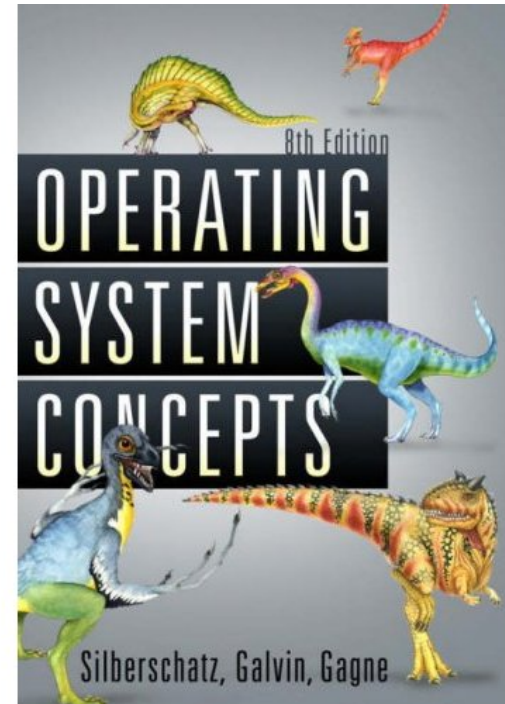
**Koen Langendoen**

email:

[k.g.langendoen@tudelft.nl](mailto:k.g.langendoen@tudelft.nl)

# Praktische Zaken

- Voorkennis: C + computer systemen
- College:  $7 \times 3 = 5 \times 4 = 5 \times (\text{wo} + \text{do})$
- Tentamen: 2 april + 1 juli
- Boek



# Doel van een Operating System

## Wat is een Operating System?

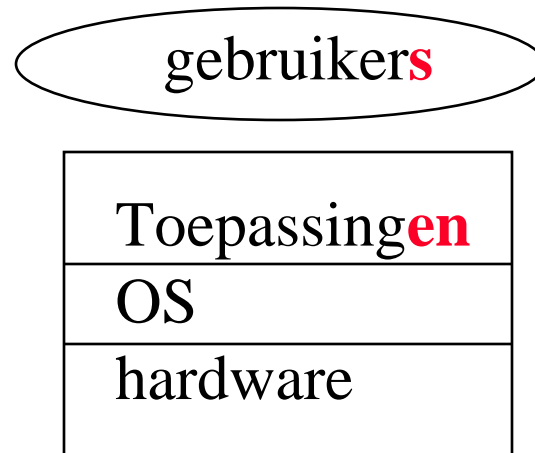
Een Operating System is die software die het mogelijk maakt op een eenvoudige en efficiënte manier programma's uit te voeren op een computer

## Twee belangrijkste Doelstellingen:

- **Makkelijkere** (mooiere) machine bieden
- **efficiënt** gebruik van de machine

Vaak strijdig

## 3-Lagen model



opm:

1. Grens tussen toepassingen en OS niet scherp, b.v. plaats van systeemprogramma's.
2. Ontwikkelingen op gebied van hardware en OS beïnvloeden elkaar
3. De *kernel* van een OS is dat deel dat permanent in het (beschermde) geheugen aanwezig is

# Interrupts

- Interrupts
  - Onderbreken processing
  - Gegeneerd door de hardware
  - Is interface van hardware naar kernel
- Traps (Soft Interrupts, exceptions)
  - Onderbreken processing
  - Veroorzaakt door de software (b.v delen door 0, of System Call)

Voor iedere interrupt of trap bestaat een afhandelingsroutine,  
Wordt vaak gevonden d.m.v. interrupt vector

- Traps: synchron, Interrupts: asynchroon

# Enkele termen: Batch verwerking

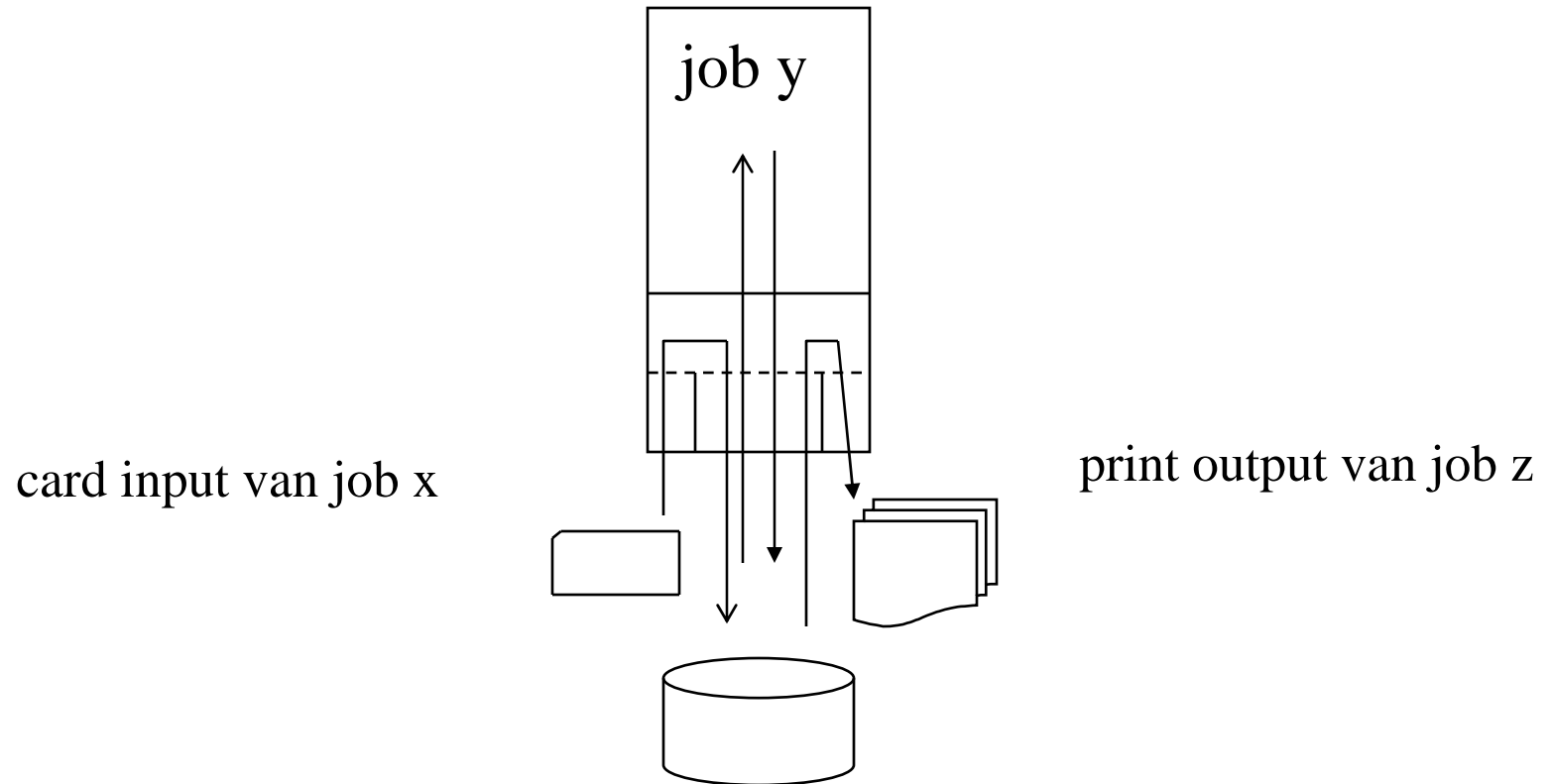
- Batch processing

letterlijk: “partij-gewijs” of “groepsgewijs”:

- Gebruikers leveren programma’s aan in de vorm van *jobs*
- Opdrachten, b.v. “compileren” en “uitvoeren” worden gegeven door middel van een Job-Control taal (JCL)
- een job wordt in zijn geheel verwerkt, geen interactie met de gebruikers.
- Er wordt gebruik gemaakt van *spooling*
- Jobs wachten aanvankelijk in de *job pool*
- *Job scheduling* bepaalt welke job in executie te nemen.

# spooling

(Simultaneous Peripheral Operation On Line)



job y leest zijn card input van disk en schrijft print output naar disk

# Enkele termen: Multiprogrammering en Timesharing

- Multiprogrammering: Het simultaan uitvoeren van verschillende programma's.
  - Geeft betere benutting van CPU, want als een programma moet wachten op b.v I/O, kan een ander de CPU gebruiken.
  - *CPU-scheduling* bepaalt welk programma de CPU krijgt
- Time Sharing: vorm van multiprogrammering, met speciale kenmerken:
  - Het is interactief: gebruikers hebben tijdens executie contact met hun programma's
  - CPU gaat beurtelings naar verschillende programma's (ook als er geen geen sprake is van wachten op b.v I/O)
- Timer interrupt nodig om programma te onderbreken

# Dual mode operation (1)

- Doel: bescherming
- processor kent 2 soorten instructies:
  - privileged
  - non-privileged
- processor kan in 2 verschillende modes werken
  - user mode
  - monitor mode
- user programma's draaien in user mode
- *kernel* draait in monitor mode

## Dual mode operation (2)

- In monitor mode privileged en non-privileged instructies uit te voeren
- In user mode alleen non-privileged instructies uit te voeren
- Bij interrupt of trap (soft interrupt, b.v. System Call):
  - processor switcht naar monitor mode
  - code van de afhandelingsroutine wordt uitgevoerd
  - bij terugkeer naar geïnterrupteerd proces, gaat processor terug naar user mode.

# Componenten (1)

## 1. Proces Management

*Definitie: Een proces is de uitvoering van een programma*

- Zorgt voor creëren, verwijderen, onderbreken, hervatten van processen
- Problemen: synchronisatie en deadlock

## 2. Main memory management

- Zorgt voor beheer van main memory (primary storage)

## Componenten (2)

### 3. File-SystemManagement

*definitie: een file is een verzameling bij elkaar horende informatie*

- Zorgt voor creëren en verwijderen van files
- Verantwoordelijk voor de directories

### 4. Secondary storage management (mass storage management)

- Zorgt voor beheer van diskruimte
- snelheid bepalend voor performance van het hele systeem

## Componenten (3)

### 5. I/O management

- Maakt gevarieerde hardware eenvoudig bruikbaar  
(verbergt details van specifieke apparatuur)
- Bevat o.a. *caching* en *device drivers*

# Caching

- Geheugen kent hiërarchie, b.v.
  1. Registers
  2. Main memory
  3. Disk
  4. Tape
- Hoger in de hiërarchie betekent: sneller, kleiner, duurder
- *Caching* houdt in: Als data normaal op niveau N1, dan een deel tijdelijk op niveau N2 te plaatsen (N2 hoger dan N1). Dit verhoogt de snelheid.
- Cache management bepaalt grootte en inhoud van de cache

# Device drivers

- Device drivers zijn onderdeel van de kernel
- Device driver per device controller
- Kent details voor specifieke devices,
  - B.v. printer van bepaald merk en type
- Stuurt controller aan, en handelt interrupt af
- Biedt een uniform interface naar de rest van het OS

# Protectie en Security

## 6. Protectie systeem

- Zorgt voor bescherming.  
o.a. memory, CPU, data
- Authenticatie  
o.a. user/group ID

# Bijzondere systemen (1)

## 1. Parallele systemen (multiprocessor systemen)

- tightly coupled, delen (sharen) bus, klok en vaak geheugen

voordelen:

- verhogen throughput ( niet lineair)
- delen van resources (b.v. behuizing, power supply)
- verhogen van de beschikbaarheid (gracefull degradation, fault tolerant)

onderscheid:

- symmetrische multiprocessing
- asymmetrische multiprocessing

# Bijzondere systemen (2)

## 2. Gedistribueerde systemen

- fysiek gescheiden (delen geen klok of geheugen)
- communicatie door middel van netwerken
- distributie transparant voor de gebruiker

voordelen: in principe dezelfde als bij parallelle systemen,

- nu ook communicatie voor gebruikers, b.v. E-mail

Complex vanwege:

- heterogene systemen
- synchronisatieproblemen

# Bijzondere systemen (3)

## 3. Real time systemen

- strenge eisen m.b.t. de responstijd
- te gebruiken voor besturing, b.v. van industriële processen

onderscheid:

- hard real time systemen
  - niet te realiseren in general purpose systeem
  - maakt geen gebruik van de meest geavanceerde onderdelen van een OS
- soft real time systemen
  - te realiseren in general purpose systeem, door real time taak prioriteit te geven.

# Bijzondere systemen (4)

## 4. Embedded Systemen

- Meestal real-time systemen met strikte eisen aan responstijd
- Ingebouwd in allerlei apparatuur, b.v. auto's, mobieltjes, etc.
- Komen in zeer grote aantallen voor
- Nauwelijks of geen user interface

## 5. Multimedia Systemen

- Multimedia datastromen (audio en video) moeten volgens het juiste tijdschema worden weergegeven

# Trend?

- Traditioneel: gecentraliseerd; Mainframes, eventueel te gebruiken m.b.v. telefoonlijnen en terminals
- Nu: overvloed aan PC's en kleinere machines, vaak gekoppeld aan snelle netwerken.

Maken gebruik van services op basis van

- Client/server model, of
- Peer-to-peer model
- Veel principes op gebied van OS, zijn geformuleerd in de mainframe periode, maar zijn nog steeds geldig.

PAUZE

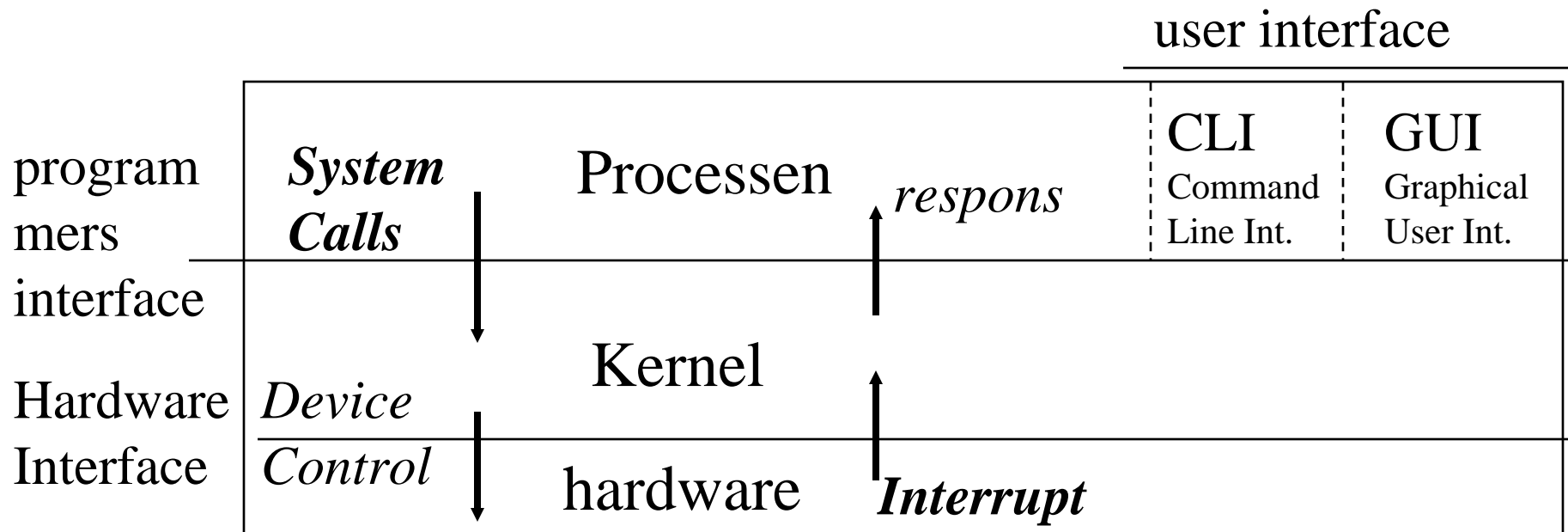
# OS Services (1)

- Direct voor de gebruiker
  - User interface
    - Command Line Interface (CLI)
    - Graphical User Interface (GUI), b.v. Windows
  - Programma uitvoering
    - Laden, uitvoeren, beëindigen
  - I/O operaties
  - File systeem manipulatie
  - Communicatie
    - Tussen processen op dezelfde machine of op verschillende machines
  - Foutdetectie

## OS Services (2)

- Ten behoeve van het systeem
  - Resource toekenning (allocation)
    - Verschillende processen kunnen om dezelfde resources vragen, toekenning moet zodanig gebeuren dat processen niet nodeloos moeten wachten , of dat resources onbenut blijven
  - Accounting,
    - doorberekening van de kosten aan individuele gebruikers
  - Protectie
    - Beschermt resources tegen ongeauthoriseerde toegang

# Interfaces (1)



De Command Line Interpreter is in veel systemen een proces,  
b.v. in UNIX: shell

## Interfaces (2)

- Command Line Interpreter (CLI)
  - soms onderdeel van de kernel, soms (systeem) proces
    - voorbeeld: unix shell, MSDOS shell
  - functie:
    - leest commando regel, b.v. `cd ../programs1`
    - herkent commandonaam
      - voert commando zelf uit, of
      - zoekt programma (met die naam) en start proces, geeft parameters door aan het proces
  - voordeel: eenvoudig nieuwe commando's toe te voegen

# Interfaces (3)

- Graphical User Interface (GUI)
  - Gebruikt monitor, toetsenbord, muis
  - Maakt gebruik van windows, menus, iconen
  - Voorbeelden:
    - MSWindows
    - UNIX met X11
    - Linux met KDE

# Interfaces (4)

## System Call

- Is interface tussen proces en kernel
- vaak assembly language instructie (trap)
- onzichtbaar voor meeste programmeurs (wordt vaak aangeroepen d.m.v een libraryfunctie)
- verschillende mogelijkheden voor doorgeven van parameters aan OS:
  - via registers
  - via data block en een register dat wijst naar het data block
  - via stack, door middel van **push** en **pop**

## Interfaces(5)

- Een Application Programmers Interface (API) is de verzameling functies die een programmeur kan gebruiken om de diensten van een OS aan te roepen.
- Functies hebben vaak één op één relatie met System Calls, maar zijn eenvoudiger in het gebruik
- Portability

# Interfaces (6)

System calls in 5 groepen

1. Proces beheer
2. File manipulatie
3. Device manipulatie
4. Systeem-informatie uitwisseling
5. Communicatie

2 en 3 vaak gecombineerd

(uniforme benadering van files en devices)

# Interfaces (7)

## Systeemprogramma's (en system utilities)

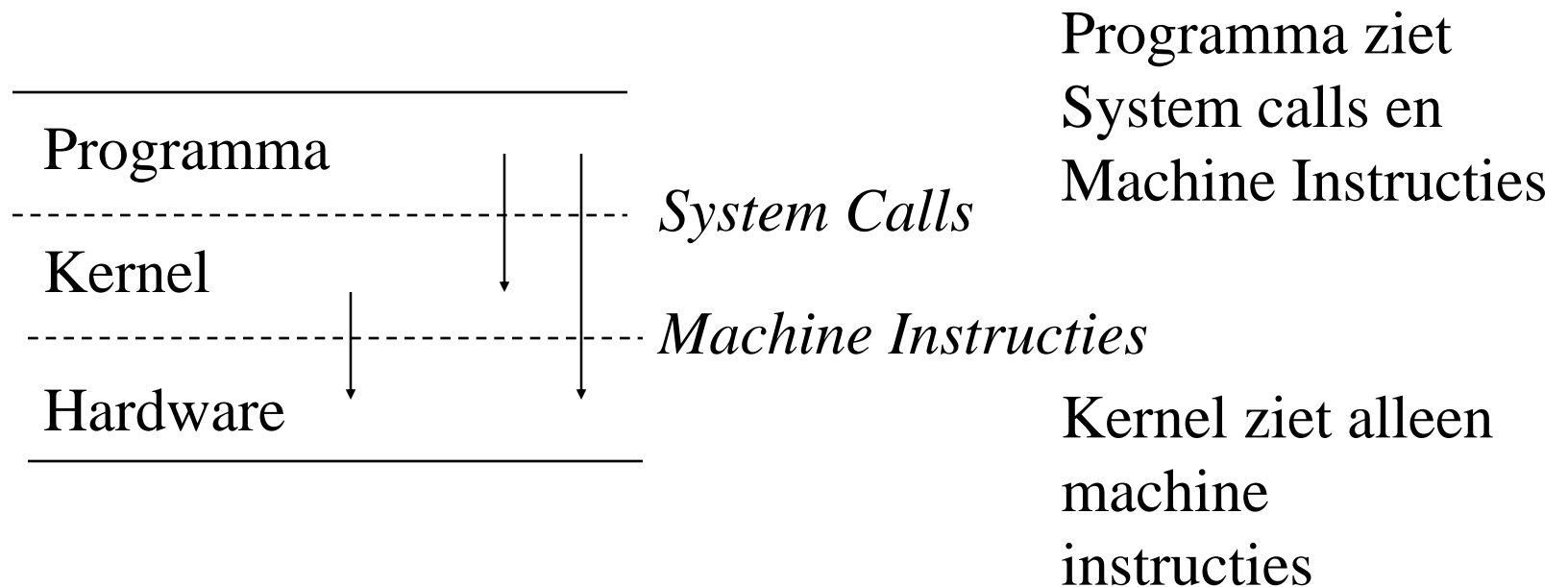
- Leveren de omgeving waarin makkelijk programma's te ontwikkelen en uit te voeren.
- voorbeelden:
  - file management programma's
  - assemblers, compilers
  - Command Line Interpreter
- Beeld wat een gebruiker heeft van een systeem wordt voor een belangrijk deel bepaald door het User Interface en de systeem programma's.

# Gelaagde Opbouw

- OS en kernel worden vaak opgebouwd uit lagen
- Iedere laag gebruikt alleen functies uit de direct onderliggende laag
- voordelen:
  - eenvoudiger debugging
  - verificatie
- problemen:
  - keuze en volgorde van de lagen
  - overhead
- Veel systemen echter historisch bepaald

# Virtuele Machines(1)

- Lagenstructuur maakt Virtuele Machines mogelijk
- Virtuele Machine = machine gerealiseerd door software
- Verschillende niveaus:



## Virtuele Machines (2)

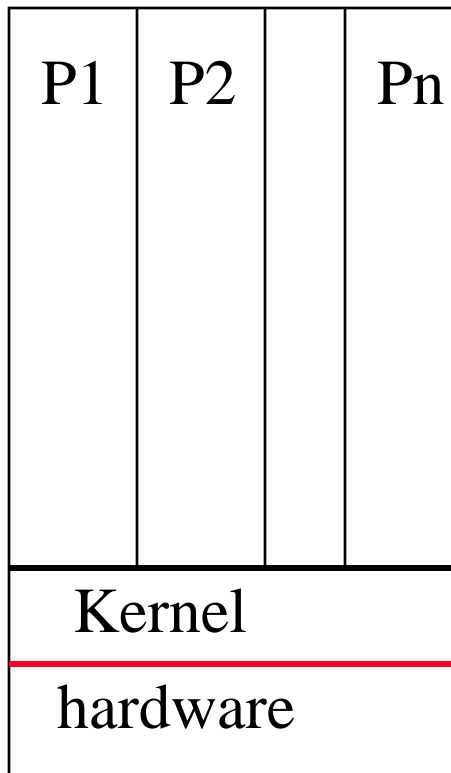
‘Klassieke’ Virtuele Machine (IBM: VM, VM/370) gebruikt virtualiserende Kernel.

Virtualiserende kernel (VK) biedt aan ieder “proces” het interface van een kale machine (de kale hardware).

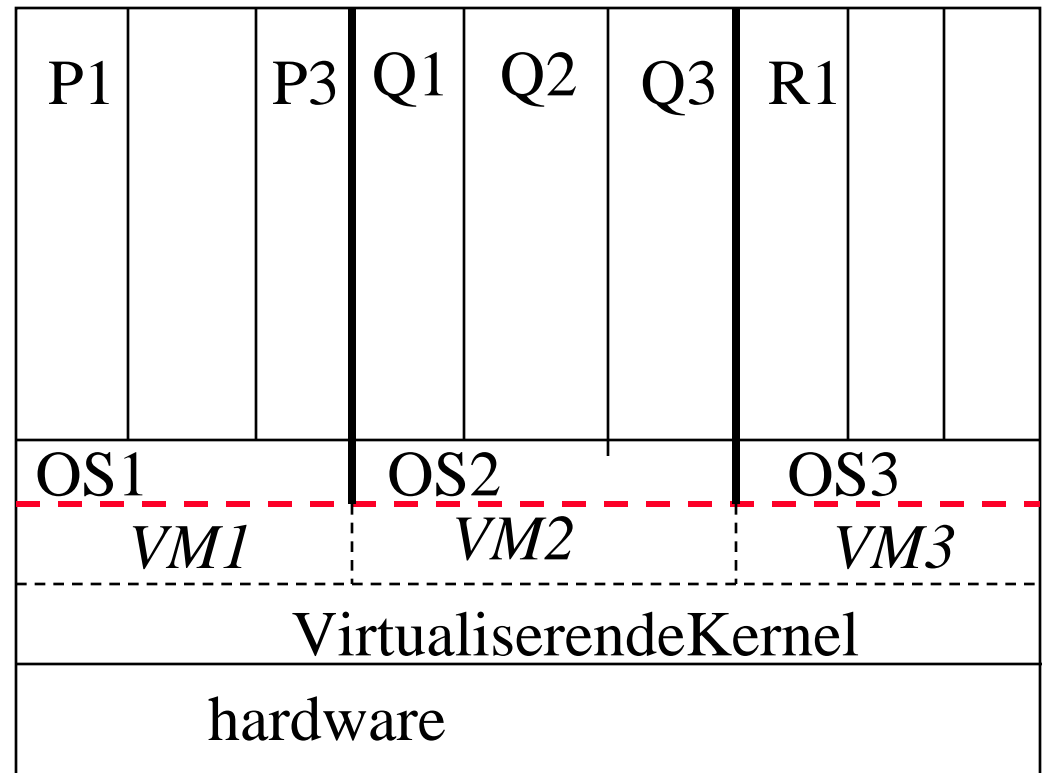
- processor en geheugen worden gedeeld
- kaartlezers en printers via spooling
- minidisks i.p.v. echte disks
- Ieder “proces” op de VK een OS
- Binnen ieder OS weer processen te starten
- Deze Virtuele Machine nuttig voor:
  - testen van OS'en
  - draaien van verschillende OS'en binnen 1 machine

# Virtuele Machines (2A)

Klassieke Virtuele machine biedt naar boven een interface dat eruit ziet als de oorspronkelijke hardware

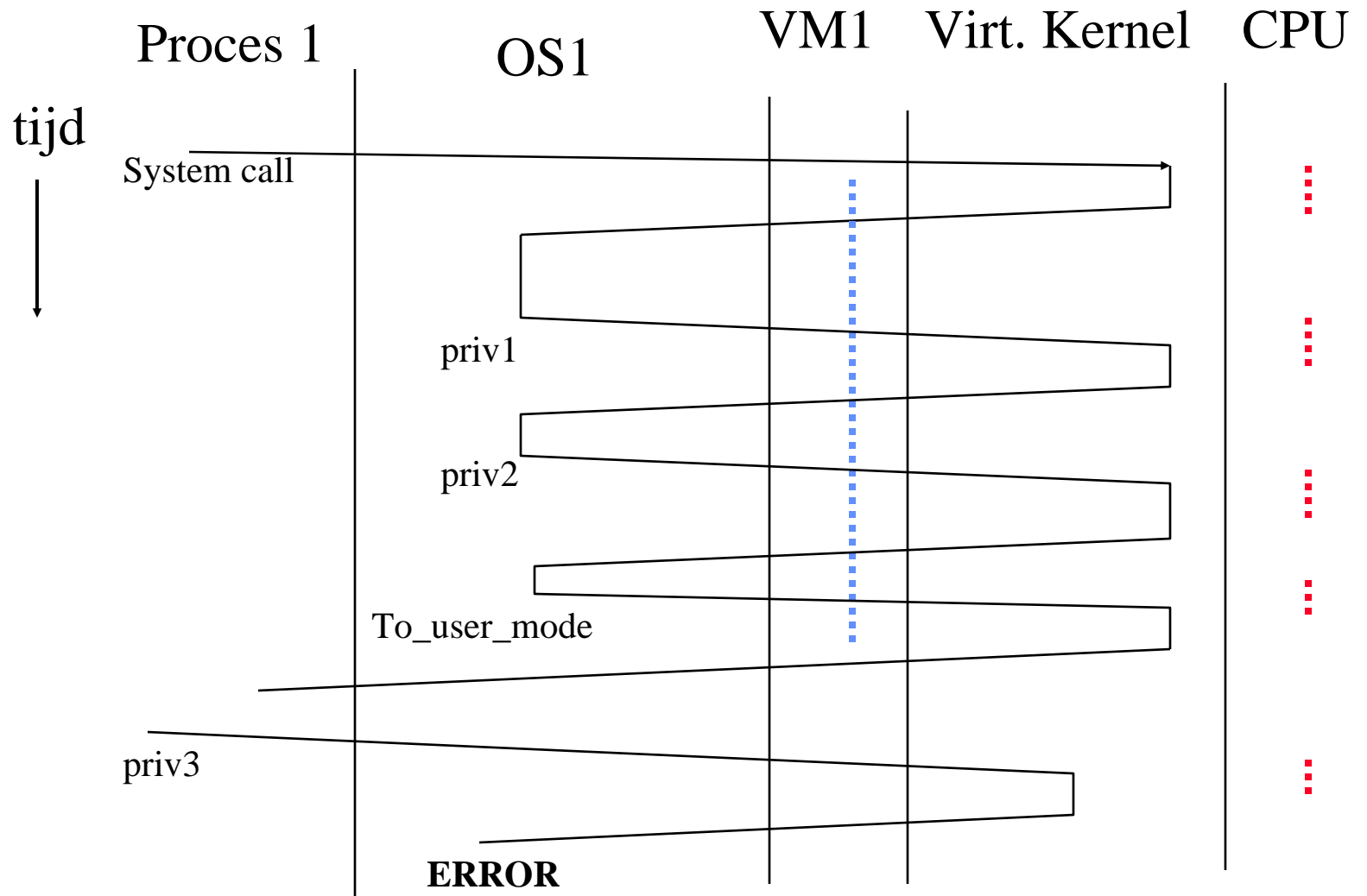


normale kernel



Virtualiserende kernel

# Virtual User en Monitor mode in een VM



(virtual) monitor mode aangegeven door .....

VM1 is normaal in virtual usermode. VM1 komt in virtual monitor mode als een System Call wordt gegeven; echte CPU blijft in user mode. 1-36

## Virtuele Machines (3)

Andere vormen van virtuele machines op een hoger niveau

- Java Virtuele Machine (JVM) biedt machine die Java bytecode verwerkt.
  - JVM vaak op het niveau van een toepassingsprogramma, b.v. Internet browser)
- Software die OS en machine-instructies van andere processorotypen ondersteunen.
  - Voorbeeld: MS-DOS programma's op Sun/Solaris (wabi)

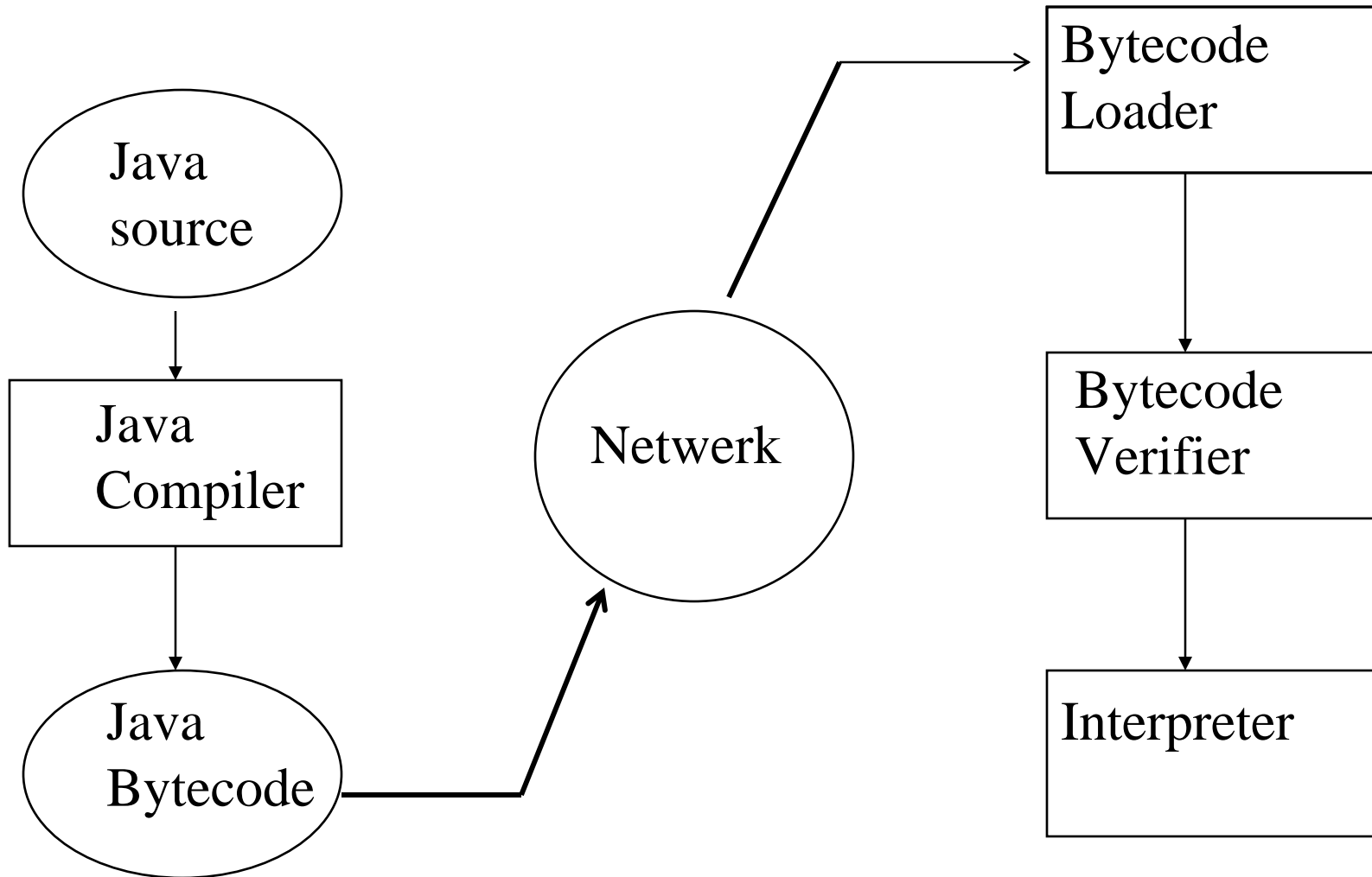
# Virtuele Machines (3A)

## Java Virtual Machine

- *Java* is een object-oriented programmeertaal
- wordt gecompileerd naar bytecode
- bytecode wordt geïnterpreteerd door *Java Virtual Machine* (JVM)
- JVM vaak geïmplementeerd als programma
  - zelfstandig,, of
  - onderdeel van browser, b.v. Netscape of Microsoft Explorer
- Java op ieder platform te draaien waarvoor een JVM bestaat.

# Virtuele Machines (3B)

## Java Virtual Machine



# Ontwerp en Implementatie (1)

- Scheiding van Policy en Mechanisme
  - Policy heeft betrekking op **Wie** of **Wat**
  - Mechanisme zegt **Hoe**voorbeeld: *microkernel systeem*
- Programmeertaal
  - oude systemen vaak in assembler
  - tegenwoordig: hogere talen, vaak: C
    - voordelen: sneller te schrijven, beter te begrijpen, beter te porten
    - nadeel: performance verlies ??

## Ontwerp en Implementatie (2)

- **Systeem generatie (Sysgen)**  
Het aanpassen van het OS aan een specifieke machine,  
Verschillende werkwijzen voor verschillende systemen:
  - volledige compilatie
  - selectie uit geprecompileerde libraries
  - alleen aanpassen van tabellen, dynamische configuratie
- **Bootstrap**  
OS wordt geladen door een bootstrap programma