erasmus

HOGESCHOOL BRUSSEL

IT Essentials

Deel III: Operating Systems

5: I-O

INHOUD

- I/O hardware
- I/O software
- I/O software lagen
- Clocks

erasmus
HOGESCHOOL BRUSSEL

INTRODUCTIE

- OS
  - Abstractie naar boven toe
  - Beheren/toegankelijk maken van I/O devices
    - Met een makkelijke interface
    - Voor zover mogelijk een interface die gelijk is voor alle devices
- I/O
  - Bekeken vanuit zowel hardware als software perspectief

I/O HARDWARE

- 2 soorten devices

Block devices	Character devices
Data opgeslagen in fixed size blocks, elk met een eigen adres	Data als stroom van karakters (geen blocks)
Elke block kan apart gelezen/geschreven worden	Niet adresseerbaar
Seek mogelijk	Geen seek mogelijk
Vb: HDD, USB sticks,...	Vb: printers, NIC's, muizen,...

- Niet perfect als concept: touchscreen, clocks passen niet goed

erasmus HOGESCHOOL BRUSSEL

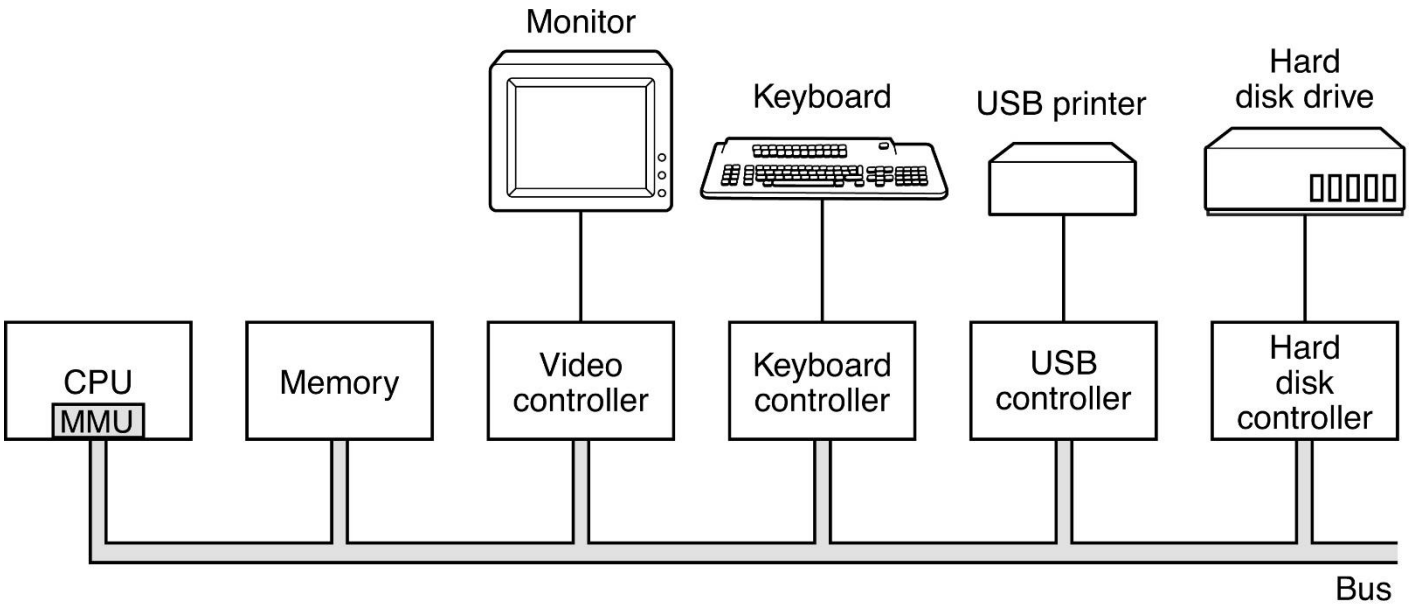
I/O HARDWARE

Device controllers

- Een device bestaat vaak uit een mechanisch en een electronisch component
- electronische component = device controller
- Chip op het MOBO of controller card
- Verbinding gebeurt (meestal) met kabel
- Vaak via standaarden: SATA, SCSI, USB,...

erasmus HOGESCHOOL BRUSSEL

I/O HARDWARE


Monitor
Keyboard
USB printer
Hard disk drive
CPU
MMU
Memory
Video controller
Keyboard controller
USB controller
Hard disk controller
Bus

I/O HARDWARE

Device controllers

- communicatie tussen controller en device meestal zeer low-level
  - Vb block device: preamble-data(blockcontent)- checksum
  - Seriële stream
- 3 taken:
  - Zet de seriële stream om naar block (tijdelijk opgeslagen in buffer)
  - Voert de checksum uit
  - Kopieert data naar RAM

erasmus HOGESCHOOL BRUSSEL

I/O HARDWARE

Memory-mapped I/O

- Elke controller heeft een aantal **registers**, gebruikt voor *communicatie met het OS*
  - OS geeft commando aan device door dit in de registers te schrijven
  - OS leest status van device uit de registers
- Elke controller heeft een aantal **buffers**
  - Vaak rechtstreeks lees/schrijfbaar door programma's en OS
  - Vb: video ram

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

Doel van I/O software

Taken van OS m.b.t. I/O

- Device independance

  Toegang tot I/O onafhankelijk van device

- Uniform naming
  - Benaming van devices onafhankelijk van soort device (windows versus linux?)
- Error handling
  - Zo veel mogelijk door de hardware, zo laag mogelijk in de lagen
    - » Controller->driver->…

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

Doel van I/O software

- Synchronous (blocking) versus asynchronous (interrupt driven)
  - Meestal asynchronous, maar sommige applicaties vragen synchronous: OS toont interrupt-driven als blocking
- Buffering
  - Inkomende data van I/O moet vaak eerst opgeslagen worden voor het kan doorgestuurd worden
    - » NIC: pakket moet eerst onderzocht worden
    - » Audio: niet te veel / te weinig data doorsturen binnen bepaalde tijd

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

Doel van I/O software

- Sharable versus dedicated devices
  - sommige devices moeten gebruikt kunnen worden door meerdere gebruikers tegelijkertijd: vb HDD
  - Sommige devices mogen niet gebruikt worden door meerdere gebruikers tegelijkertijd: vb printer

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

3 I/O methodes

- Programmed I/O (busy waiting)
- Interrupt driven I/O
- DMA I/O

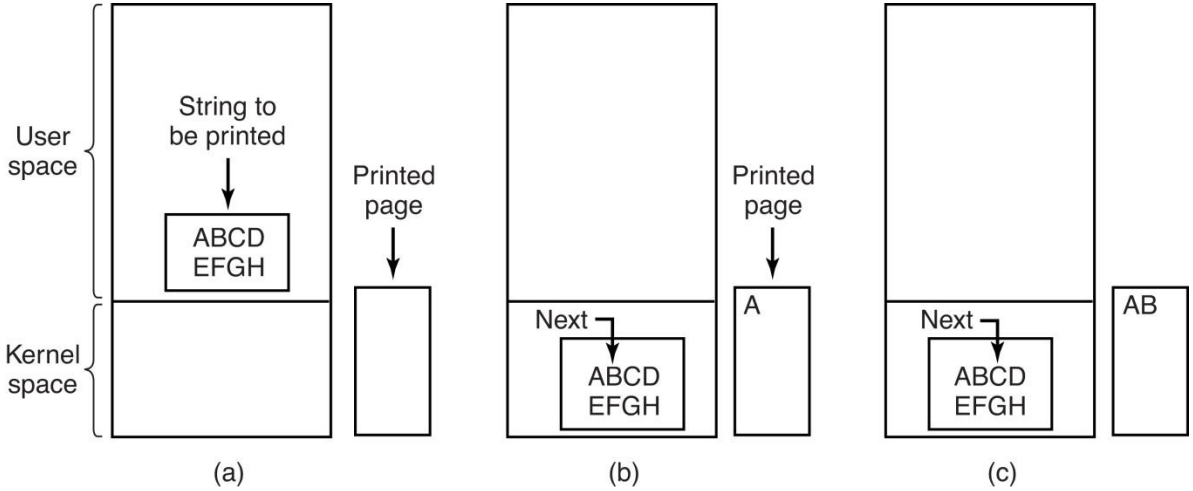
Aan de hand van voorbeeld printen van "ABCDEFGH" via seriële interface

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

Programmed I/O

- OS (in kernel space) kopieert “A” naar data register op device controller => status register op device controler wordt “unavailable”
- OS checkt deze status voortdurend tot deze terug op “available” komt te staan: “A” is geprint en B mag doorgestuurd worden
- CPU is “**busy waiting**”.



I/O SOFTWARE

Programmed I/O

- OK als CPU geen andere taken heeft of als wachttijd zeer kort is
- Anders te CPU intensief en te veel tijdsverlies

erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE

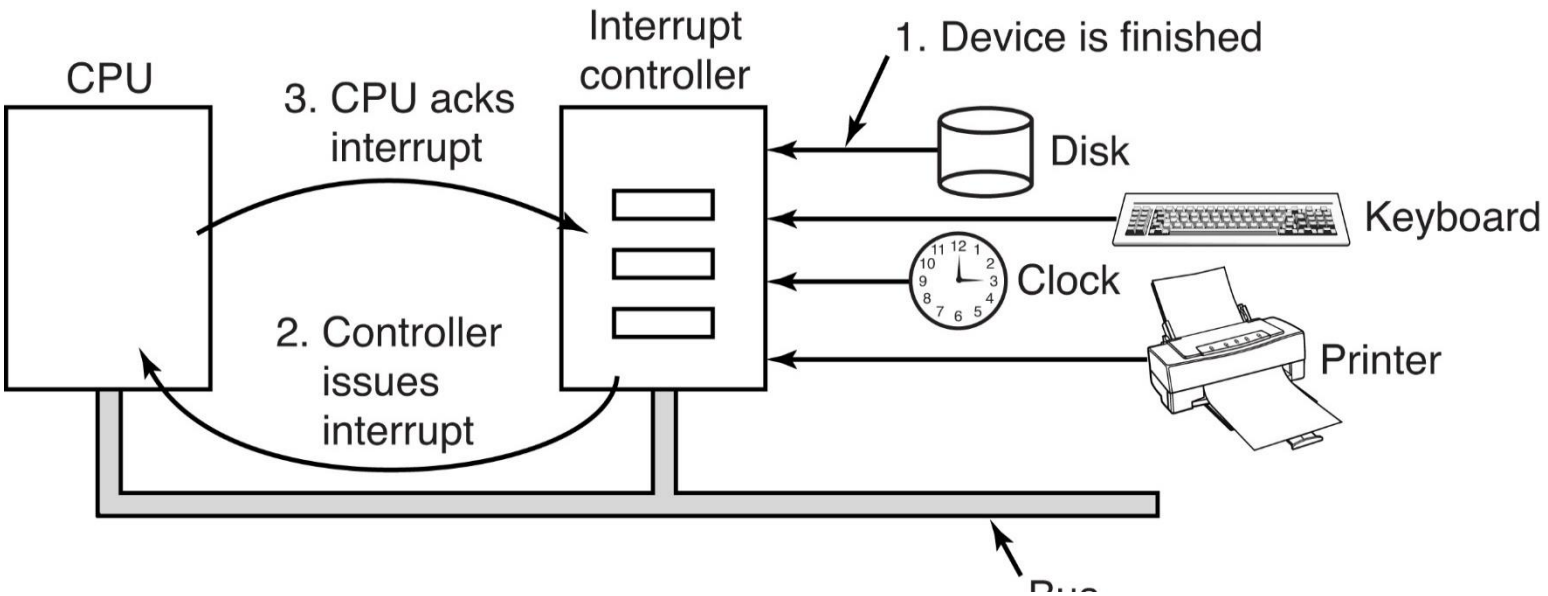
Interrupt-driven I/O

- CPU start ander process (scheduler) terwijl er gewacht wordt op interrupt van device controller via de interrupt controller
- CPU moet nog steeds elke letter 1 per 1 doorgeven
- De caller (user program) wordt in de tussentijd geblokkeerd
- Niet alle interrupts worden steeds dadelijk aanvaard

erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE

Interrupt-driven I/O



I/O SOFTWARE

DMA I/O

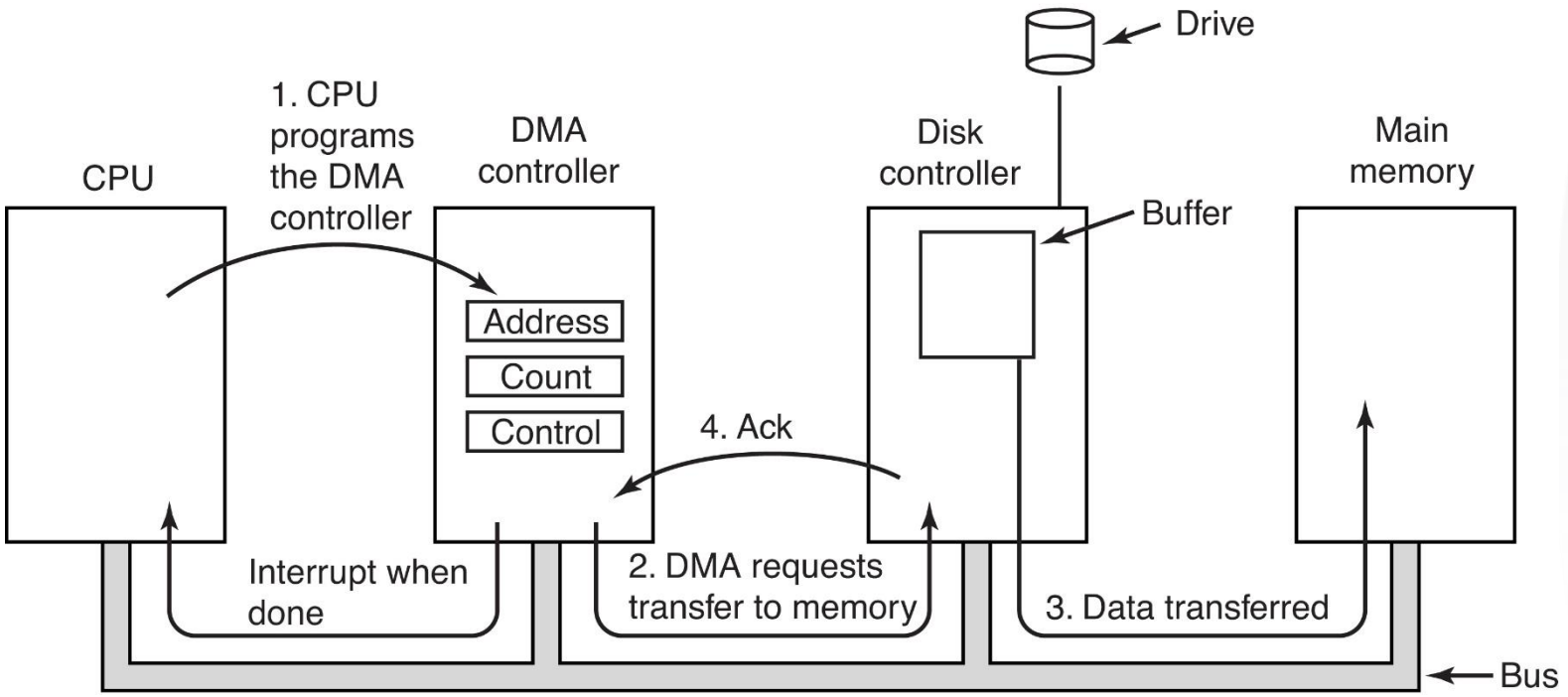
- CPU schrijft “ABCDEFGH” naar de DMA controller
- DMA controller geeft de letters door aan device controller
- CPU intussen vrij voor andere processen
- DMA zelf werkt via programmed I/O
- Vereist aparte hardware (DMA controller)
- DMA controller veel trager dan CPU: soms verlies aan snelheid

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE

DMA

- Werking



I/O SOFTWARE

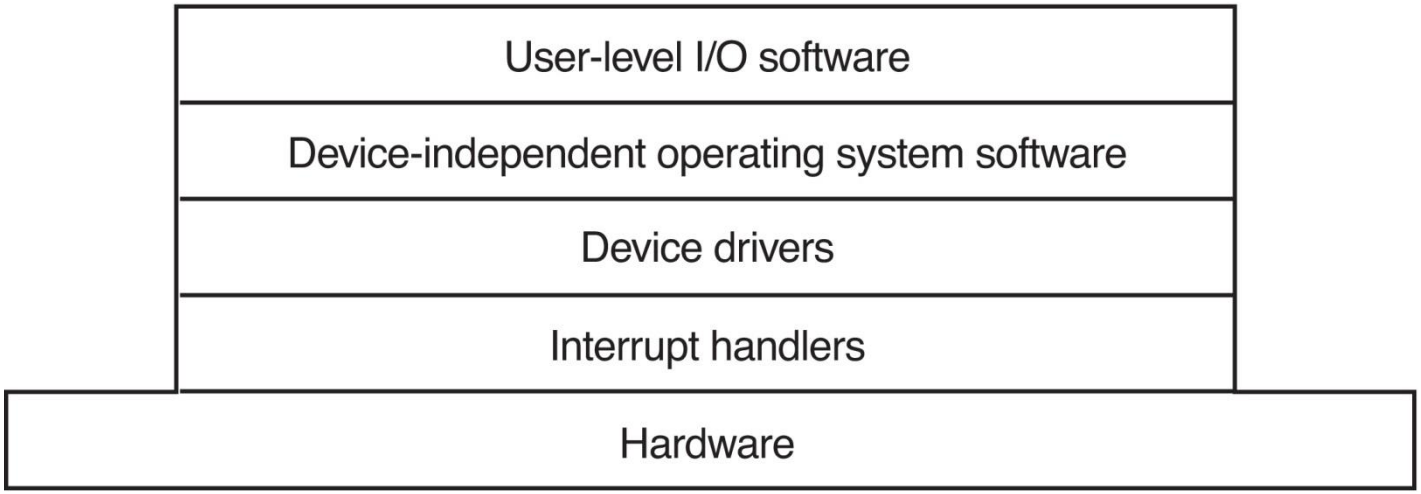
DMA

- Stappen 2-4 in loop tot hele proces is afgewerkt
- Eens alles afgewerkt: interrupt naar CPU
- Na interrupt moet de CPU de data niet meer naar RAM copiëren: reeds gedaan door DMA
- Meerdere devices tegelijkertijd mogelijk
  - Meerdere devices pending op ACK

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN

- Vergelijkbaar met networking
- Elke laag:
  - Duidelijk afgebakende taak
  - Specifieke interface met boven- en onderliggende laag



erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN

Interrupt handlers

- Meestal blockt de driver zichzelf
- Opvallend complex systeem met veel stappen en CPU-instructies waaronder:
  - Saven, kopiëren en laden van registers (context-switch)
  - Ack van interrupt controllers
  - Selecteren van welk process te laten lopen (scheduler)
  - …

erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN

Device drivers

- Device controllers gebruiken eigen registers voor data en status
- Registers zijn sterk afhankelijk van device (vb. muis versus HDD)
- Code nodig voor het controleren van device: device driver
- Aparte driver voor elk device (device dependant) en (type) OS
- Meestal aangeleverd door fabrikant device

erasmus
HOGESCHOOL BRUSSEL

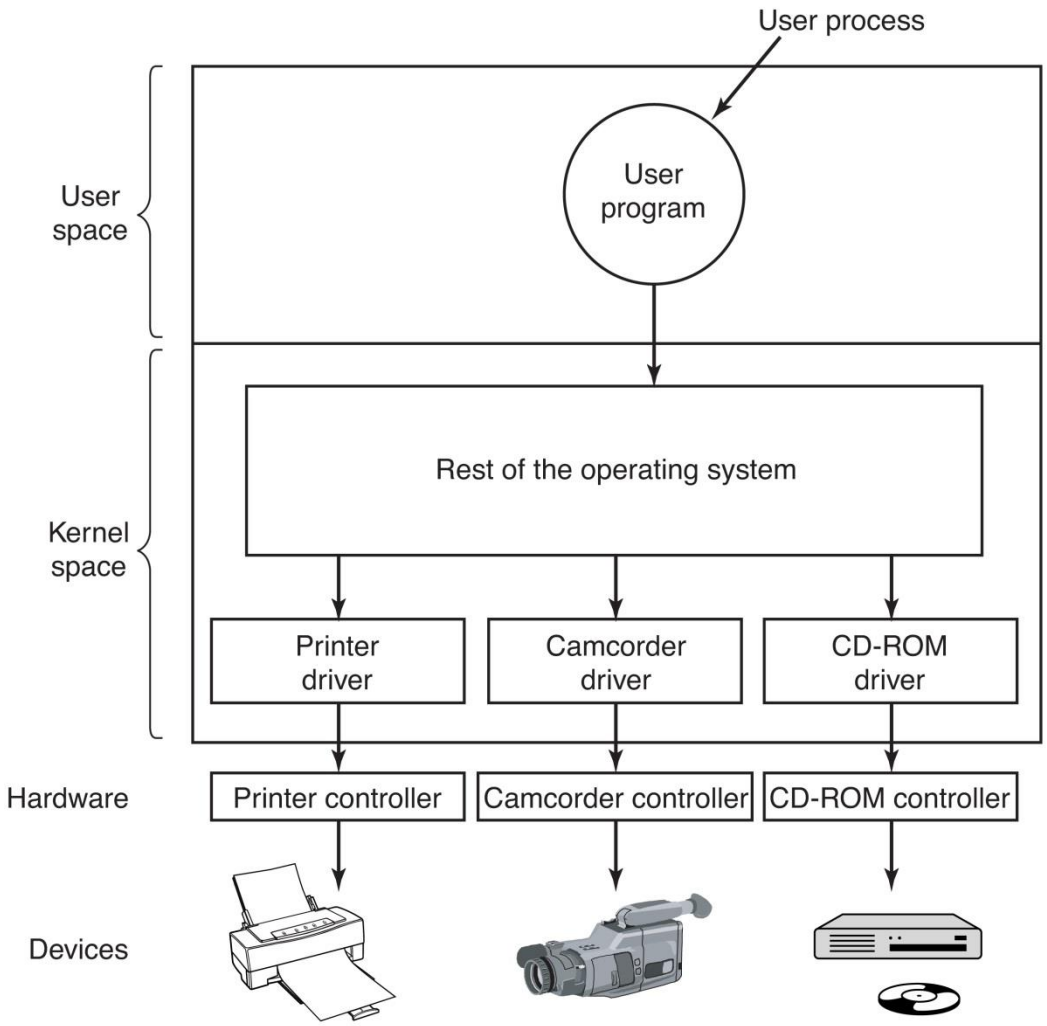
I/O SOFTWARE LAGEN

Device drivers

- Soms gelaagde structuur, met de onderste lagen identiek voor erg uiteenlopende devices
  - Vb. USB: enkel bovenste laag specifiek voor device, al de andere decoderen van seriële stream en algemene usb-functionaliteit
- Meestal onderdeel van OS-kernel
  - soms in user-mode voor stabiliteit: Minix
  - Onderaan de rest van het OS geplaatst

erasmus HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN



I/O SOFTWARE LAGEN

Device drivers

- 2 soorten: block en character
  - Uniforme interface voor elk die door elke driver moet gebruikt worden
- Hot-pluggable devices
  - OS moet lopende opdrachten en opdrachten in queue netjes wegwerken
  - Onbeschikbaarheid van apparaat moet gemeld worden
  - Eventueel aanpassing van IRQ-lijnen
- Mogen geen system calls uitvoeren
  - Toegang tot bepaalde onderdelen van de kernel via specifieke calls

I/O SOFTWARE LAGEN

Device independent OS software

- Taken

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN

Device independent OS software

- Uniforme interface voor drivers
  - nieuw device zelfde interface i.p.v. aanpassing aan de kernel
  - Per klasse van device vaste set van functies vastgelegd die een driver moet volgen
  - Vastgelegd in tabel met pointers naar de functies in driver
  - Adres van tabel opgenomen in OS die deze zal gebruiken voor aanspreken van functies

erasmus
HOGESCHOOL BRUSSEL

I/O SOFTWARE LAGEN

Device independent OS software

- Uniforme interface voor drivers
  - Interface is ook verantwoordelijk uniforme benaming
  - Symbolische device name mapped naar
    - » Major device number: welke driver
    - » Minor device number: welk device voor deze driver