erasmus

HOGESCHOOL BRUSSEL

IT Essentials

Deel III: Operating Systems

1: Introductie

INTRODUCTIE

Wat is een OS?

Geschiedenis van OS'en

Computer Hardware herhaling en relevantie ervan t.o.v. OS'en

Verschillende soorten OS'en

OS concepten (processen, Address Spaces, bestanden, I/O,…)

System Calls

Verschillende OS structuren

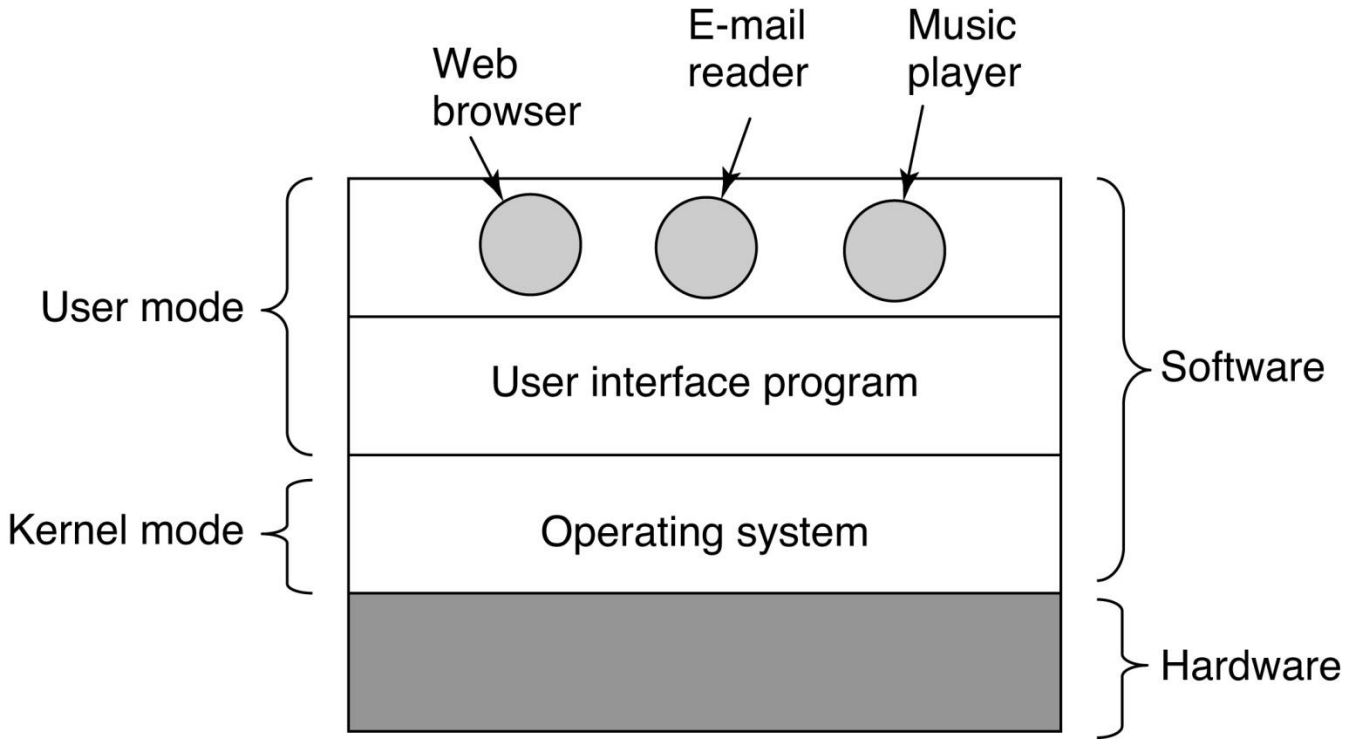
erasmus
HOGESCHOOL BRUSSEL

WAT IS EEN OS?

- Vaak een foutief gebruikte benaming:
  - Windows 11?
  - Debian Linux?
- Veel te ruime benadering
- OS: in kernel mode
- Applicaties in user mode

erasmus
HOGESCHOOL BRUSSEL

WAT IS EEN OS?


Web browser
E-mail reader
Music player
User mode
Kernel mode
User interface program
Operating system
Software
Hardware

WAT IS EEN OS?

Het OS als extended machine

- Van boven naar beneden perspectief
- Werken met hardware is een vuile opdracht
- Vaak zeer gespecialiseerd, ingewikkeld, inconsistent
- Het OS biedt abstracties aan van de onderliggende hardware

erasmus
HOGESCHOOL BRUSSEL

WAT IS EEN OS?

Het OS als resource manager

- Van beneden naar boven perspectief
- Veel beschikbare resources: geheugen, processor(tijd), HDD's, NIC's,…
- "conflictbemiddeling"
- Zorgen voor een stabiele en gecontroleerde manier in het beheren van al de beschikbare resources en deze aldus aanbieden aan de bovenliggende applicaties

erasmus
HOGESCHOOL BRUSSEL

WAT IS EEN OS?

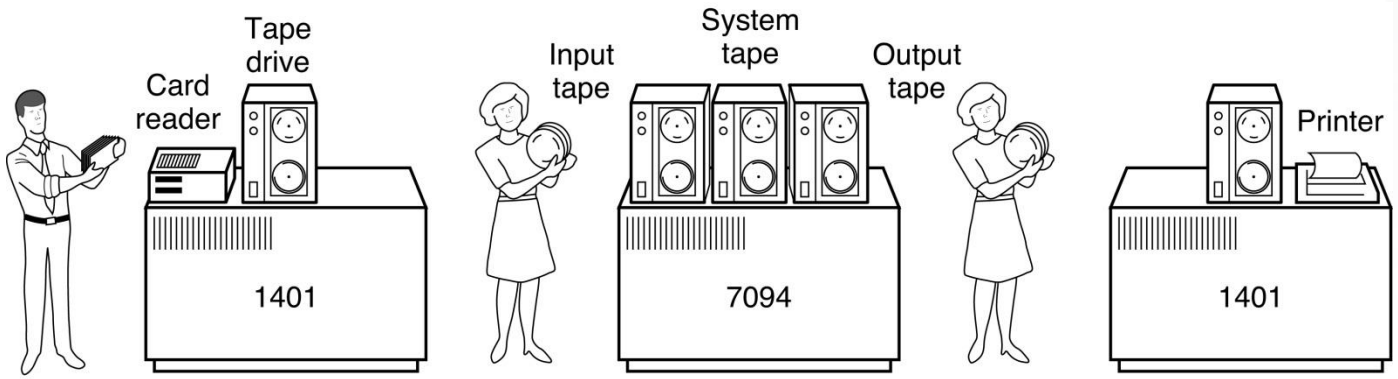
Het OS als resource manager

- **Multiplexing**: delen van resources
  - In tijd: CPU, printer, muis,…
  - In ruimte: HDD, geheugen,…

erasmus
HOGESCHOOL BRUSSEL

GESCHIEDENIS VAN OS'EN

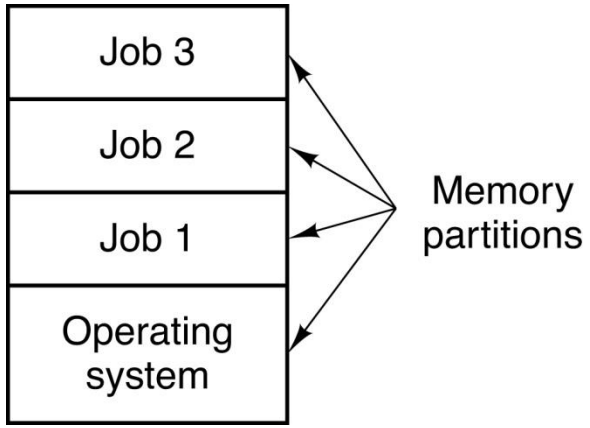
- De **1ste generatie** (1945–55): vacuum tubes
  - Geen programmeertalen, geen OS
- De **2de generatie** (1955–65): transistors en batch systemen
  - Mainframes en ponskaarten



erasmus HOGESCHOOL BRUSSEL

GESCHIEDENIS VAN OS'EN

- De **3de generatie** (1965–1980): ICs en multiprogramming
  - Introductie van ICs leidt tot OS/360 van IBM: 1 OS dat werkt op alle modellen
  - **multiprogramming** om CPU-tijd te optimaliseren wordt een andere job uitgevoerd terwijl een eerste nog wacht op I/O



GESCHIEDENIS VAN OS'EN

- **spooling**:
  meerdere input of output wordt klaargezet en automatisch verwerkt door de CPU wanneer deze klaar is voor de volgende taak
- **timesharing**:
  meerdere gebruikers maken gebruik van 1 computer die zijn resources verdeelt
- Multics wordt ontwikkeld, zeer invloedrijk OS (introductie van cloud computing?)
- Ontwikkeling van BSD, Posix, Minix en Linux

erasmus
HOGESCHOOL BRUSSEL

GESCHIEDENIS VAN OS'EN

- De **4de generatie** (1980–heden): personal computers
  - Large scale integration (LSI) circuits maken microprocessors mogelijk met betaalbare PC's als gevolg
  - CP/M ontwikkeld voor de 8080
  - MS DOS (+ BASIC): gigantisch succes door koppeling aan PC-verkoop (OEM)
  - OSX met focus op GUI (WYSIWYG)
  - Windows 3.1
  - Windows 95-ME
  - Windows NT
    - Client
    - Server

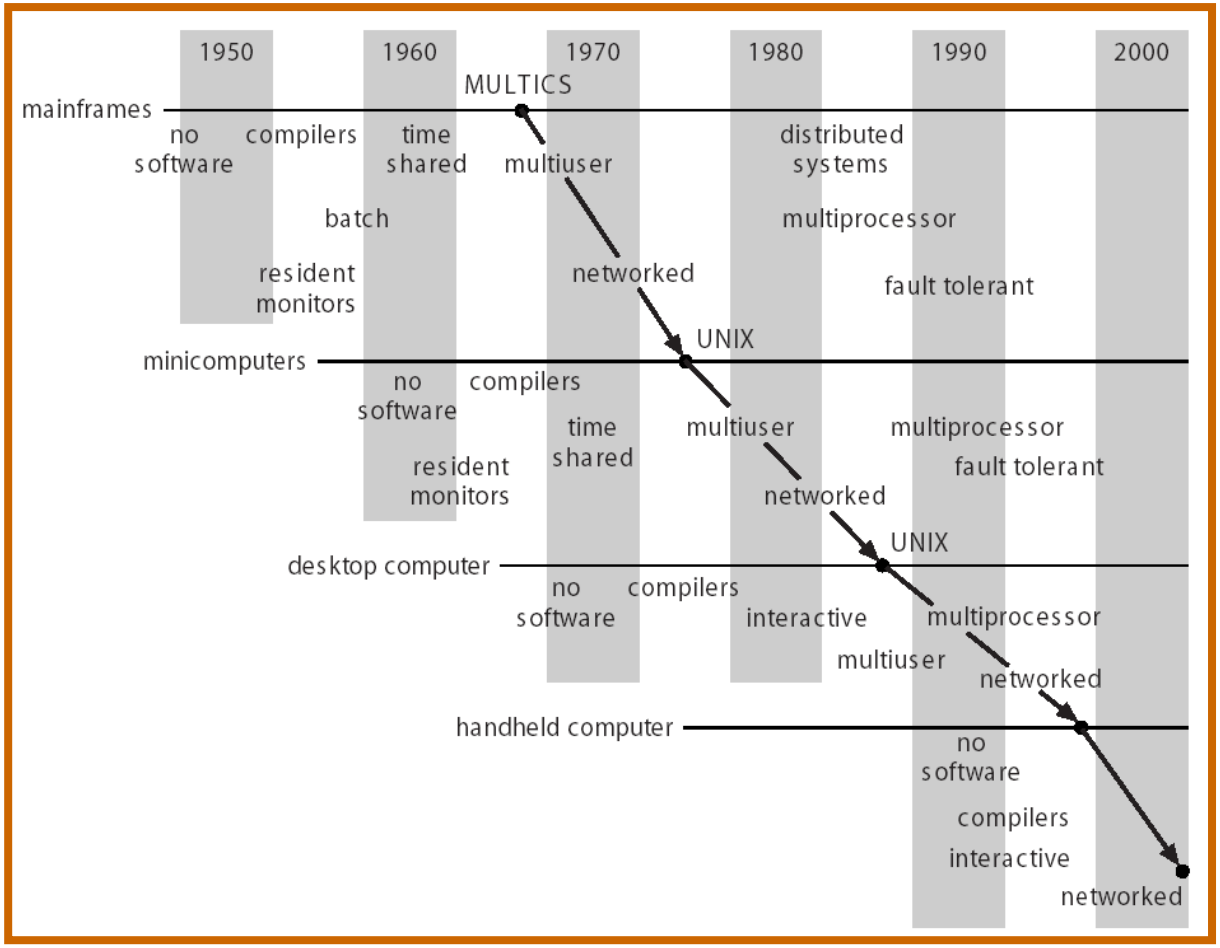
erasmus HOGESCHOOL BRUSSEL

GESCHIEDENIS VAN OS'EN

- De **5de generatie** (1990–heden):Mobile Computers
  - Samensmelting van PDA met GSM
  - Symbian OS
  - Blackberry (RIM)
  - IOS (Apple)
  - Android

erasmus
HOGESCHOOL BRUSSEL

GESCHIEDENIS VAN OS'EN



COMPUTER HARDWARE

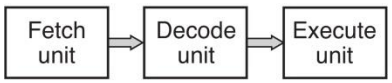
Processors

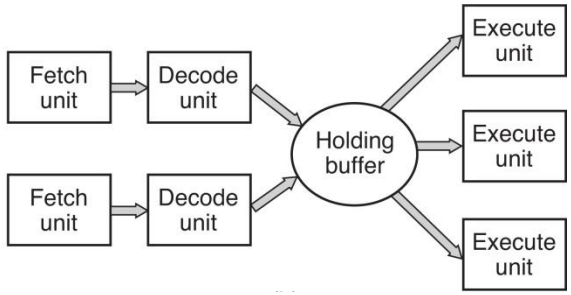
- Gewone registers:
  tijdelijke opslag rechtstreeks op CPU van variabelen en tijdelijke resultaten
- Speciale registers:
  - **Program counter**: adres van volgende instructie die moet opgehaald worden
  - **Stack pointer**: wijst naar de top van de stack
  - **PSW** (program status word): control bits, waaronder de user/kernel mode bit

COMPUTER HARDWARE

Processors

- **pipelining**:
  - Meerdere instructies worden ingeladen in een "lopende band" waardoor de gemiddelde output verhoogt
- **superscalar**:
  - meerdere uitvoeringseenheden zijn aanwezig (floating point, boolean, integer berekeningen)





erasmus HOGESCHOOL BRUSSEL

COMPUTER HARDWARE

Processors

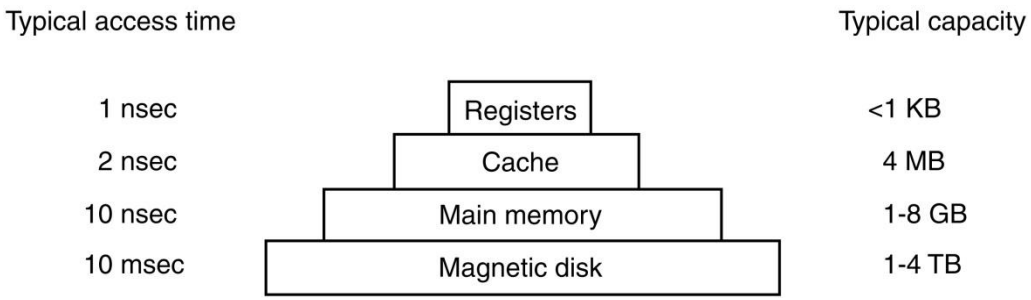
- **system call**
  - Applicatie in user mode heeft een een service nodig die enkel de kernel kan leveren (kernel mode is vereist)
  - Een system call wordt gegenereerd die de kernel trapt
  - “Software interrupt”
- **multi(-of hyper)threading**
  - CPU feature waarbij op 1 CPU in een nanoseconde kan geswitcht worden naar een andere thread
  - Geen echt parallelisme maar mogelijks tijdswinst
  - Mogelijks ook tijdsverlies: 2 threads op 1 core in multicore multithreading CPU?

erasmus
HOGESCHOOL BRUSSEL

COMPUTER HARDWARE

Memory

- (veel) trager dan CPU: nood aan caching
- L1, L2, L3



Disks

- Te weinig RAM wordt (vaak) opgelost via virtual memory

erasmus
HOGESCHOOL BRUSSEL

COMPUTER HARDWARE

I/O devices

- Bestaat uit een controller en het apparaat zelf
- Drivers
  - speciale software die de controller beheert
  - Aangepast voor elke controller en voor elk OS
  - Geleverd door fabrikant van de controller
  - Draaien meestal in kernel mode

erasmus
HOGESCHOOL BRUSSEL

COMPUTER HARDWARE

I/O devices

- 3 methodes voor I/O (zie hoofdstuk I/O)
  1. **busy waiting**
  2. **interrupt**
  3. **DMA**

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Mainframe OS'en

- Grootste OS'en
- 1000 disken en miljoenen aan gigabyte
- Tegenwoordig in de vorm van high-end webservers, commerce sites (Amazon, Ebay,…)
- Veel jobs tegelijkertijd
- 3 diensten
  - Batch (vaste, grote berekeningen op grote hoeveelheden data)
  - transaction processing (massa's kleine bewerkingen)
  - timesharing
- Bijvoorbeeld OS/390 en Linux

erasmus HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Server OS'en

- Diensten aanbieden over het netwerk aan vele gebruikers en computers
- Diensten: file, print, authenticatie, web, mail,…
- Bijvoorbeeld Windows Server, Linux, BSD,…

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Multiprocessor OS'en

- Meerdere CPU's op 1 moederbord
- Vaak ook meerdere cores op 1 CPU
- Vereist speciale aanpassing aan het OS (meestal al ingebouwd in de meest recente OS'en)
- High-End (server) OS'en bieden meer mogelijkheden volgens editie

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

PC OS'en

- Bedoeld voor 1 gebruiker (c.f.r. "personal")
- Ook wel client of desktop OS'en genoemd als tegenhanger van de server OS'en
- Bijvoorbeeld Windows 11, Linux, MacOS,…

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Handheld Computer OS'en

- PDA's, smartphones, tablets
- Steeds toenemende hardwarecapaciteiten (multicore CPU's, grote hoeveelheid RAM,…)
- Veel sensoren en locatiegevoelige toepassingen
- Laag stroomverbruik vereisten

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Embedded OS'en

- Microgolf ovens, TV's, dvd-spelers, HDD-recorders, klassieke telefoons…
- Geen mogelijkheid tot het toevoegen van (onvertrouwde) software = security is minder een issue
- Software is ingebakken in ROM
- Niet altijd een strikte scheiding tussen user mode en kernel mode

erasmus
HOGESCHOOL BRUSSEL

VERSCHILLENDE SOORTEN OS'EN

Real-time OS'en

- Tijd is essentiële parameter: instructies moeten uitgevoerd zijn binnen een strikt tijdsschema
- Bijvoorbeeld industrieprocessen zoals autobouw aan een band: robot moet op het exacte moment lassen
- Vaak bij zeer kritieke/gevoelige processen zoals luchtvaart, militaire toepassingen, nucleaire installaties…
- Soft-Realtime system: tijd blijft de belangrijkste factor, maar er mag al eens gemist worden
  - Digitale audio of video: een gemist frame kan niet zoveel kwaad

erasmus HOGESCHOOL BRUSSEL

OS CONCEPTEN (INTRODUCTIE)

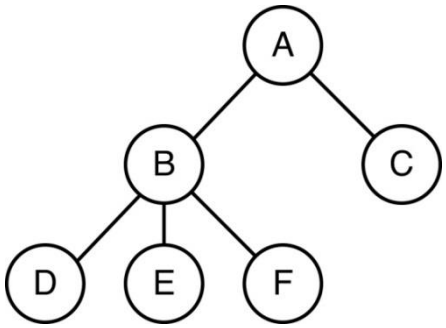
Processen

- Programma in uitvoering
- ‘container’ waarin info wordt bewaard:
  - Programma zelf, data, en stack wordt opgeslagen in de **address space** (van 0 tot bepaalde max grootte)
  - In gebruik zijnde registers en files (en de locatie in die files), openstaande alarmen, gerelateerde processen worden opgeslagen in de **process table**

OS CONCEPTEN (INTRODUCTIE)

Processen

- Processen kunnen nieuwe processen aanmaken via system calls: **child processen** vormen **process tree**
- Communicatie tussen deze processen: **interprocess communication**
- **Process alarm**: signaal van het OS aan het proces om de huidige activiteit (tijdelijk) op te schorten en eerst een andere taak af te handelen. Vergelijkbaar met hardware interrupt.
- Process wordt geassocieerd met **Uid**, die lid kan zijn van **Gid**



erasmus HOGESCHOOL BRUSSEL

OS CONCEPTEN (INTRODUCTIE)

Address space

- Multiprogramming vereist multiplexing vereist meerdere programma's in werkgeheugen vereist **beschermingsmechanisme** om interferentie te vermijden
- Vereiste geheugenruimte soms groter dan beschikbare ruimte: **virtual memory**

erasmus
HOGESCHOOL BRUSSEL

OS CONCEPTEN (INTRODUCTIE)

Files

- **Abstractie** van onderliggende hardwareconcepten
- Groeperen van files wordt ook geabstraheerd: **directories**
- Beheren van files en directories gebeurt via system calls
- **Boomstructuur**, met bovenin de root
  - Absoluut pad
  - Relatief pad (working directory)

erasmus HOGESCHOOL BRUSSEL

OS CONCEPTEN (INTRODUCTIE)

Files

- Windows: **drive letters**
- Unix:
  - **mounting**
  - **Special files**: block (storage) en character (bv. printers)
- **Piping**: pseudofile die output doorstuurt als input (zie ook interprocess communication)

erasmus HOGESCHOOL BRUSSEL

SYSTEM CALLS

Procedure

- 1 van de taken van het OS is het abstraheren van de onderliggende (hardware)functionaliteiten
- User programs doen beroep op het OS om de vereiste taken m.b.t. deze abstracties uit te voeren
- Dit gebeurt via **SYSTEM CALLS**

erasmus HOGESCHOOL BRUSSEL

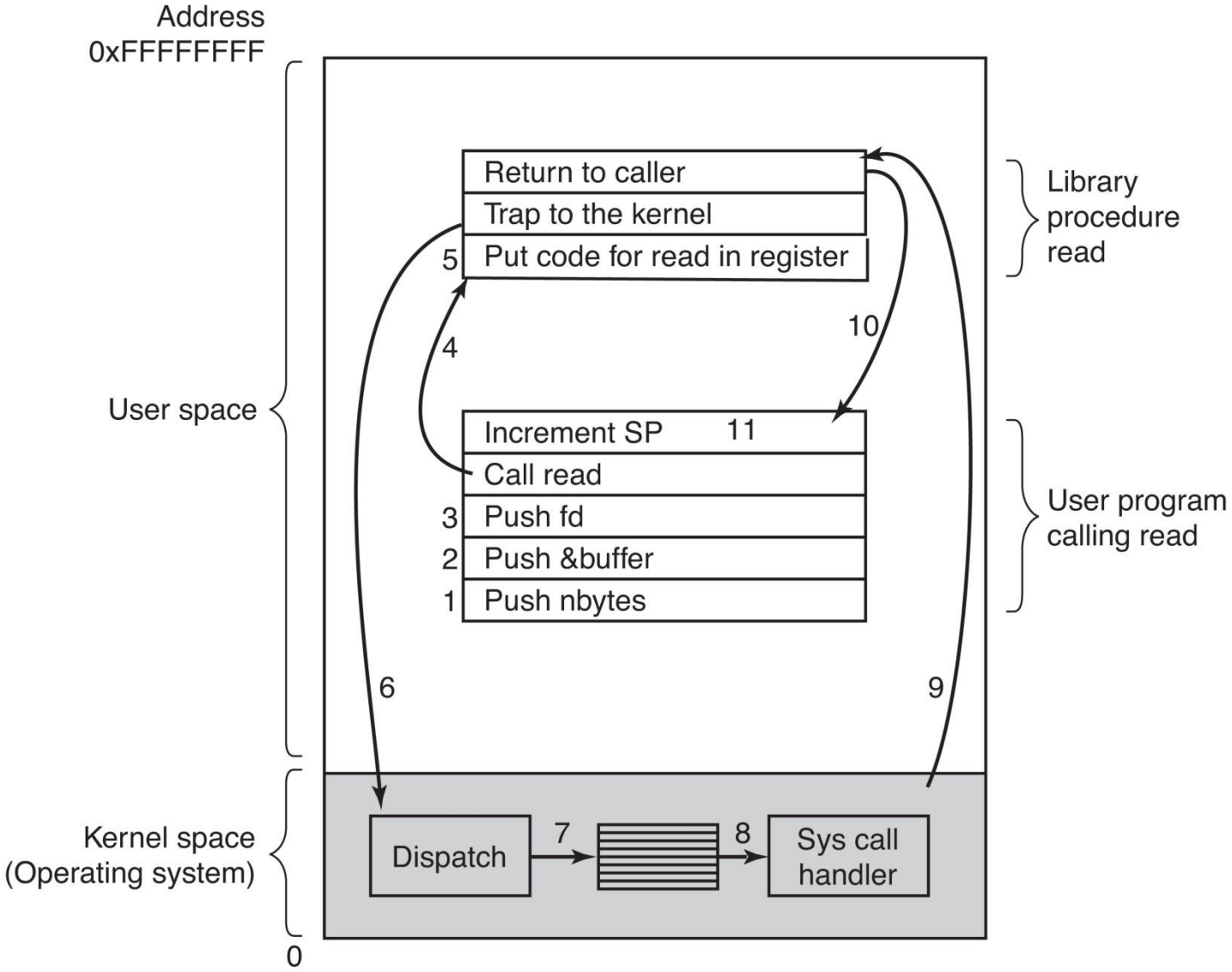
SYSTEM CALLS

Procedure

- Voorbeeld aan de hand van de system call *read* (in POSIX) die een bestand zal gaan lezen
  - Bestaat uit 3 parameters
  - fd: welke file?
  - Buffer: waar moet de data geschreven worden
  - Nbytes: hoeveel bytes moeten er gelezen worden
  - Maakt gebruikt van library met dezelfde naam read in een vorm als:

    Count=read(fd,buffer,nbytes);
    met count de hoeveelheid bytes die gelezen wordt

erasmus HOGESCHOOL BRUSSEL


Address
0xFFFFFFFF
User space
Kernel space
(Operating system)
0
Return to caller
Trap to the kernel
5
Put code for read in register
4
10
Increment SP
11
Call read
3
Push fd
2
Push &buffer
1
Push nbytes
6
9
Dispatch
7
8
Sys call
handler
Library
procedure
read
User program
calling read

SYSTEM CALLS

Voorbeelden van System Calls in Posix

- System calls worden uitgevoerd voornamelijk voor onderstaande taken:
  - Process management
  - File management
  - Directory- en filesystem management
  - Allerlei

erasmus HOGESCHOOL BRUSSEL

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory- and file-system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

SYSTEM CALLS

Process management

- **Fork**:
  - aanmaken van nieuwe processen
  - Het child process is een exacte kopie van de parent
- **Waitpid**:
  - Het parent process wacht tot het child klaar is
- **Exec(ve)**:
  - System call die door het child process gebruikt wordt om de taak uit te voeren
- **Exit**:
  - Om aan te duiden dat een process klaar is en mag afgesloten worden

erasmus
HOGESCHOOL BRUSSEL

SYSTEM CALLS

Process management

Voorbeeld: "cp file1 file2" commando in een Posix shell

- De shell maakt een kopie van zichzelf via *fork*
- De parent krijgt een *waitpid* tot de child klaar is
- Het child process start een exec om het cp commando uit te voeren
  - Name: naam van het uit te voeren programma, in dit geval *cp*
  - Argv: argumenten, in totaal 3, met 0=cp, 1=file1 en 2=file2
  - Environp: variabelen die mee worden gegeven door de omgeving. 0 indien geen
- Eens *cp* klaar is, krijgen we een *exit* system call, die de exit status doorgeeft aan de eerdere *waitpid* in de *statloc* parameter, met als gevolg dat het child process wordt afgesloten, en het parent process terug verder mag gaan, bijvoorbeeld met het volgende commando

erasmus HOGESCHOOL BRUSSEL

SYSTEM CALLS

File management

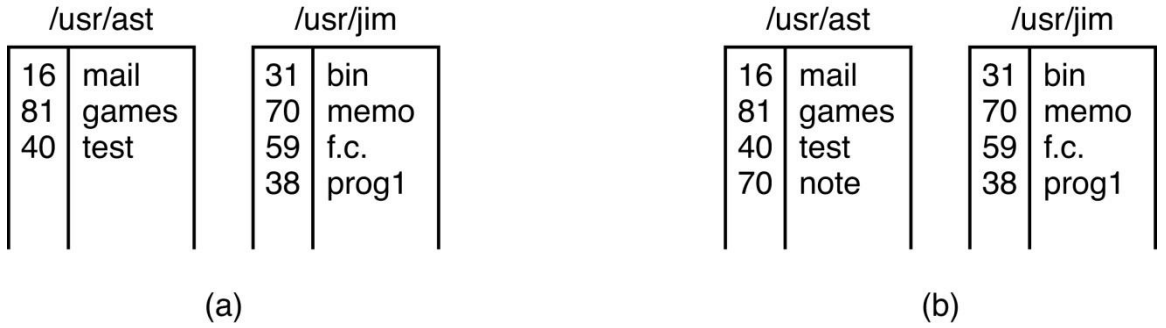
- Alvorens er operaties kunnen uitgevoerd worden op een file dient dit eerst geopend te worden: ***open***
  - Naam: afhankelijk van absoluut of relatief pad
  - O_RDONLY, O_WRONLY, O_RDWR, O_CREAT
- Eens open kan er ***read*** of ***write*** gebruikt worden
- ***close*** sluit de file weer
- ***lseek*** laat toe te werken met pointers naar specifieke locaties in de file, zodat er niet uitsluitend sequentieel dient gelezen of geschreven te worden
- ***stat*** geeft de mogelijkheid informatie over een file op te vragen

erasmus
HOGESCHOOL BRUSSEL

SYSTEM CALLS

Directory management

- ***mkdir*** en ***rmdir*** voor het maken en verwijderen van directories
- ***Link*** voor het maken van links
  - is niet gelijk aan een kopie
  - Legt een extra link naar de i-number van de file in een andere directory



- ***mount*** en ***umount*** voor het mounten van storage devices in een reeds bestaand bestandssysteem

SYSTEM CALLS

Allerlei

- ***chdir*** voor het wijzigen van de working directory
- ***chmod*** voor het aanpassen van de modus (geheel van rechten) op een file
  - *chmod("file","0644")*
- ***kill*** voor het sturen van signalen naar processen, meestal om dit geforceerd af te sluiten
- ***time*** geeft de tijd weer

erasmus HOGESCHOOL BRUSSEL

SYSTEM CALLS

Posix versus Windows

- **Posix**: ongeveer 100 system calls, bijna 1 op 1 gekoppeld met library
- **Windows**: duizenden system calls, niet gekoppeld met bijhorende libraries
- In Windows: **Win32 API**: lijst van te gebruiken procedures door programmeurs
- Door de loskoppeling kan MS de system calls wijzigen met behoud van backward compatibility
- Win32 API wordt uitgebreid per nieuwe versie OS, basis blijft.
- **API calls**: soms system calls, maar vaak ook gewoon in de user space

erasmus
HOGESCHOOL BRUSSEL

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount, so no umount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time