Les 1: Inleiding

"Adding manpower to a late software project makes it later."
Brooks' law – Fred Brooks – *The Mythical Man-Month*

best1-1

Overzicht

- Wat is een besturingssysteem
- Een korte geschiedenis
- Soorten besturingssystemen
- Architectuur van besturingssystemen
- Concepten in besturingssystemen

best1-2

In de allereerste plaats is een besturingssysteem een programma. Het is het programma dat als eerste programma opgestart wordt als een computersysteem aangezet wordt. De voornaamste opdracht voor het besturingssysteem is te bemiddelen tussen de hardware en de toepassingen die door de computer uitgevoerd worden. Hiervoor beschikt het over twee interfaces: een interface naar de hardware en een interface naar de toepassingen. Vroeger waren besturingssystemen vaak gekoppeld aan specifieke hardware. Tegenwoordig probeert men het besturingssysteem in de mate van het mogelijke onafhankelijk te maken van de onderliggende hardware.

Besturingssysteemdiensten

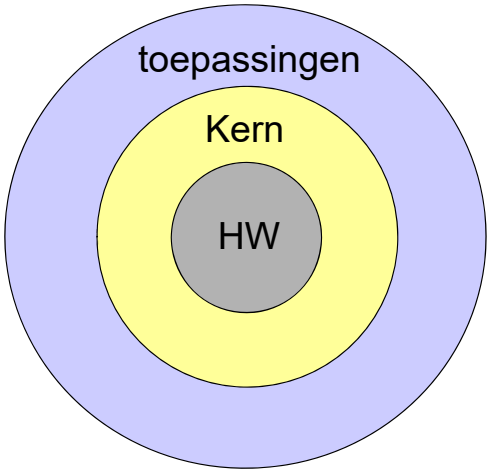
- User interface: GUI – CLI – Batch
- Uitvoering van programma's
- Input-output
- Bestandsmanipulatie
- Communicatie
- Foutdetectie
- Systeemmiddelenbeheer
- Boekhouding
- Protectie en beveiliging

best1-4

Een besturingssysteem hoort tal van diensten aan te bieden. Meest zichtbaar is de interactie met de gebruiker: grafisch (GUI), commandogebaseerd (CLI) of batch (niet-interactief). Daarnaast moet het uiteraard programma's kunnen uitvoeren, de input en output in goede banen leiden, bestanden kunnen manipuleren, kunnen communiceren met andere systemen, fouten detecteren, de systeemmiddelen adequaat beheren en het gebruik ervan bijhouden, en ten slotte zorgen voor de nodige afscherming en beveiliging.

Wat is een besturingssysteem? minimalistische visie

Enkel de software die in geprivilegieerde mode uitgevoerd wordt – de kern



best1-5

In zijn minimale definitie bevat een besturingssysteem enkel die code die dichtst bij de hardware staat. Alle bovenliggende lagen zoals een venstergebaseerde visualisatielaag kan men dan als een toepassing beschouwen. Dit minimale deel noemt men de kern van het besturingssysteem. Het is maar een heel klein deel van alle software die op een installatieCD van een besturingssysteem staat.

Wat is een besturingssysteem? maximalistische visie

Alles wat je meekrijgt als je een besturings-systeem koopt

Windows Mediaplayer?

Internet explorer?

Spelletjes?

best1-6

Doorgaans verstaan gebruikers onder een besturingssysteem veel meer dan enkel maar de kern. Het bevat alle software die op de installatieCD voorkomt. Dit is ook de visie van Microsoft. Door een sterke koppeling te maken tussen Internet Explorer en Windows hebben zij gepoogd om het marktaandeel van andere browsers in te pikken. Het gerecht is hen daarin echter niet gevolgd en heeft gesteld dat een browser geen onderdeel is van een besturingssysteem.

Gebruikersprogramma's vs. Systeemprogramma's

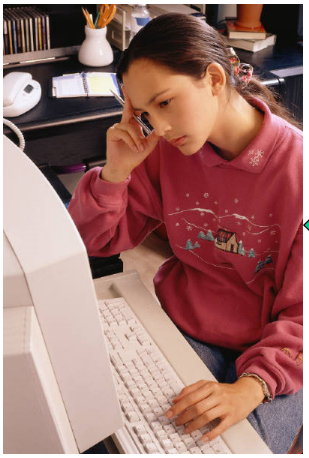
- Gebruikersprogramma's
  - Tekstverwerker
  - Databank
  - Mail-programma
  - …
- Systeemprogramma's
  - Configuratiebeheer
  - Backup/restore
  - Virusscanner
  - Commandoshell
  - …

Onderscheid niet steeds duidelijk of relevant

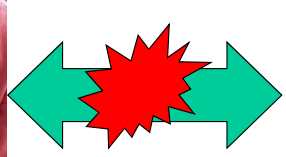
best1-7

Op het niveau van de toepassingen kan men een onderscheid maken tussen gebruikersprogramma's en systeemprogramma's. Systeemprogramma's zijn die programma's die nodig zijn om het besturingssysteem goed te kunnen gebruiken, te configureren, in stand te houden, enz. Zij zijn doorgaans standaard aanwezig (of zouden het moeten zijn). De gebruikersprogramma's kunnen sterk verschillen van systeem tot systeem, afhankelijk van de taken die de eigenaar van de computer wenst uit te voeren.

Ontwerpsdoelstellingen



BS gebruiker

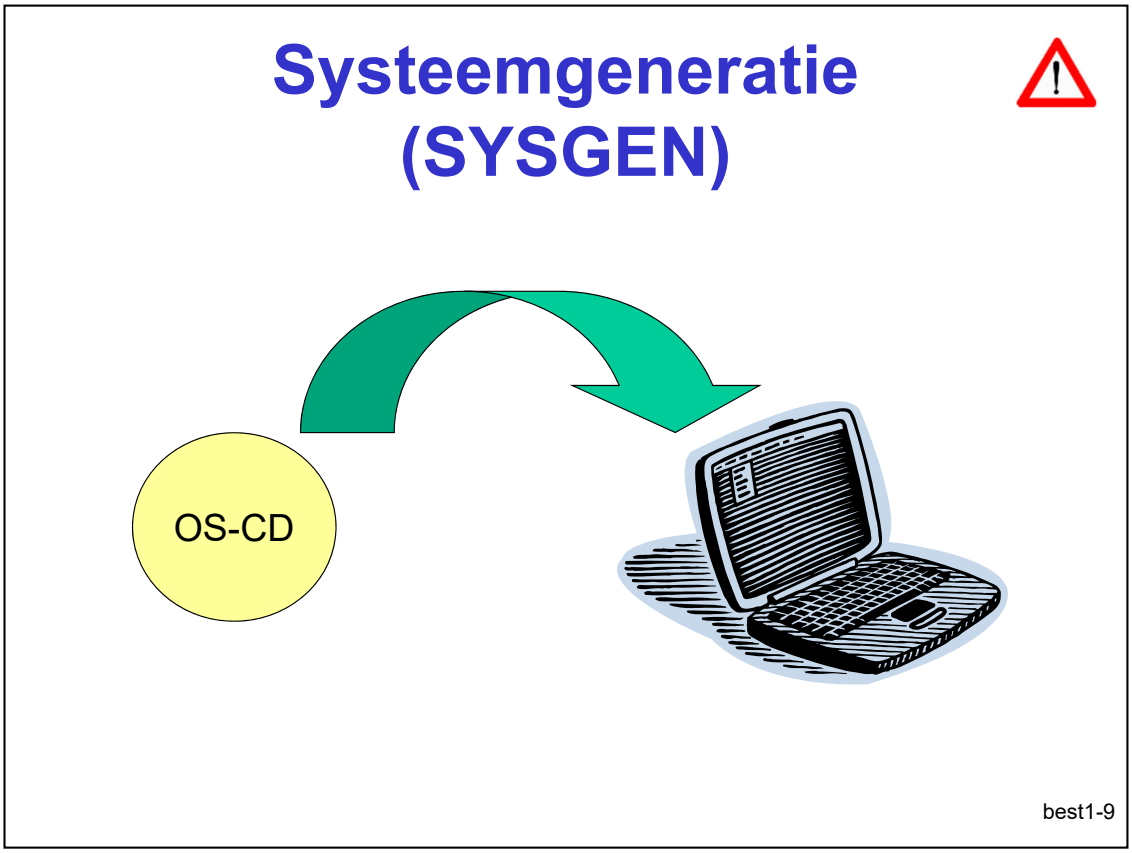


BS ontwerper

best1-8

De ontwerper van een besturingssysteem zal steeds moeten proberen om de wensen van de gebruikers en de programmeurs van besturingssystemen te verzoenen. De gebruiker wenst een besturingssysteem dat gemakkelijk in gebruik is, betrouwbaar, snel, veilig, … De ontwerper van het besturingssysteem wenst echter iets te ontwerpen dat gemakkelijk te ontwerpen en flexibel te onderhouden is.

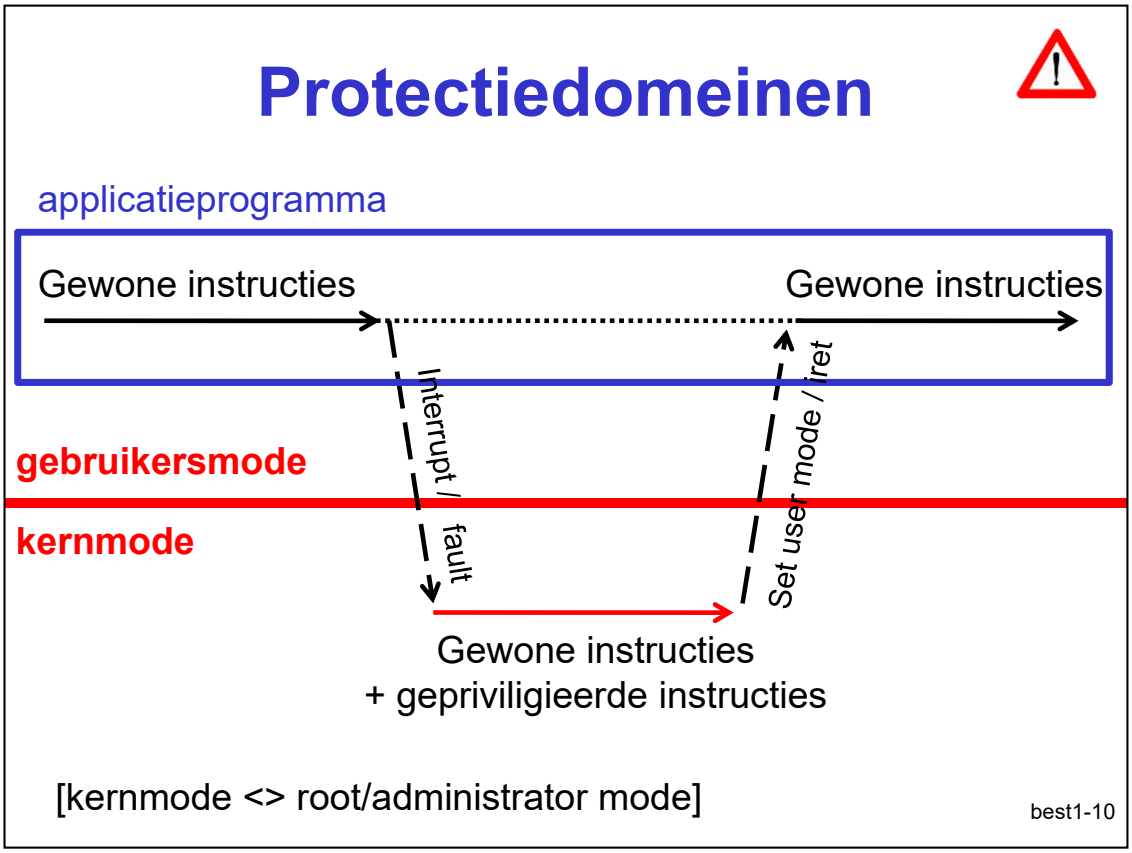
Deze tweestrijd is duidelijkst zichtbaar in de Windowswereld waar de gebruikers een systeem willen dat zowel 32-bit als 64-bit applicaties ondersteunen. Voor de ontwerpers is dit een nachtmerrie en één van de oorzaken van de instabiliteit van Windows. Al dit gebruiksgemak heeft onnoemelijk veel consequenties voor de interne architectuur en complexiteit van het besturingssysteem.



Een besturingssysteem wordt ontworpen voor een familie van hardware (b.v. van het PC-type). Het moet aangepast worden aan de concrete omstandigheden van een bepaalde PC (soort van grafische kaart, schijf, hoeveelheid geheugen, maar ook allerlei instellingen die landafhankelijk zijn, enz.).

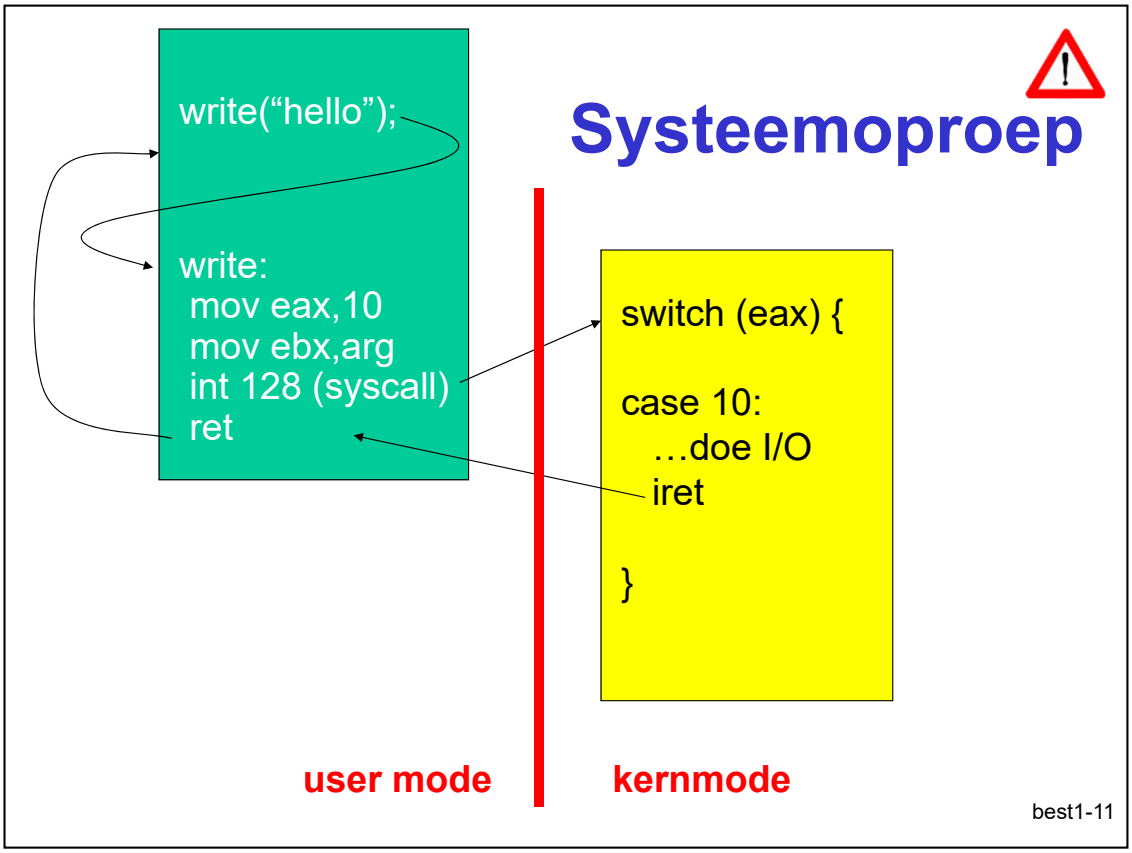
Bij de systeemgeneratie wordt de hardwareomgeving gedetecteerd en wordt een gespecialiseerde versie van het besturingssysteem geïnstalleerd. Dit gebeurt tijdens de installatie van het besturingssysteem. De systeemgeneratie kan geruime tijd duren (15 min tot meer dan 1 uur).

Indien het besturingssysteem geïnstalleerd dient te worden op een groot aantal identieke systemen, dan kan men het eenmaal installeren, en dan klonen. Naargelang de installatie kunnen er nog een paar kleine aanpassingen nodig zijn om de machines naderhand uniek te maken.



Protectie verwijst naar een mechanisme om toegang door programma's, processen, gebruikers tot de systeemmiddelen te controleren. Bij de uitvoering van processen zijn er 2 protectiedomeinen: kern/monitor/systeem mode en gebruikersmode. De kern van het besturingssysteem wordt doorgaans uitgevoerd in kernmode. De toepassingsprogramma's worden uitgevoerd in gebruikersmode.

Kernmode mag niet verward worden met de gebruiker 'administrator' of 'root'. Alles wat de systeembeheerder uitvoert wordt in principe in gebruikersmode uitgevoerd. Wel zal de systeembeheerder meer bevoegdheden hebben dan gewone gebruikers (zoals het creëren van nieuwe gebruikers, het omzeilen van paswoorden, enz.). Dit is van een totaal andere orde van b.v. het opvragen van de tijd aan de hardware. Noch de gewone gebruikers noch de systeembeheerder zullen dit rechtstreeks kunnen doen. Zij zullen dit beide aan de kern moeten vragen die dat in kernmode zal uitvoeren. Voor de kern van het besturingssysteem is er in principe geen verschil tussen een gewone gebruiker en een systeembeheerder: beide zijn gebruikers.



Systeemoproepen regelen de overgang tussen de gebruikersmode en de kernmode.

In de figuur wordt het voorbeeld gegeven van printf. Printf is een C-bibliotheekroutine die uiteindelijk zal uitmonden in een systeemoproep die de geformatteerde tekst zal wegschrijven. De kernmode wordt opgeroepen door middel van een speciale instructie ('int' in het geval van de IA32). Via de registers worden de argumenten van de systeemoproep meegedeeld aan de kern. Eénmaal binnen in de kern wordt in functie van de zgn. functiecode (hier in eax) de bijpassende code uitgevoerd.

Het adres van onderbrekingsroutine 128 haalt de processor uit de onderbrekingstabel. Het adres van de systeemoproeproutine zit met andere woorden niet hard in het toepassingsprogramma gecodeerd (enkel het nummer 128). Dit zorgt ervoor dat het besturingssysteem later kan vervangen worden door een nieuwe versie zonder het programma te moeten aanpassen.

Soorten systeemoproepen

- Procesbeheer
- Bestandsbeheer
- Apparaatbeheer
- Informatiebeheer
- Communicatiebeheer

best1-12

De systeemoproepen kunnen in een vijftal groepen ingedeeld worden.

. Procesbeheer staat in voor het creëren en manipuleren van processen en draden, voor de synchronisatie ertussen en voor het beheer van hun systeemmiddelen.

. Bestandsbeheer staat in voor het creëren en manipuleren van bestanden

. Apparaatbeheer staat in voor het manipuleren van randapparaten

. Informatiebeheer staat in voor het ter beschikking stellen van informatie zoals de tijd, en allerhande andere eigenschappen.

. Communicatiebeheer staat in voor de communicatie tussen processen en systemen

Overzicht

- Wat is een besturingssysteem
- Een korte geschiedenis
- Soorten besturingssystemen
- Architectuur van besturingssystemen
- Concepten in besturingssystemen

best1-13

Korte geschiedenis

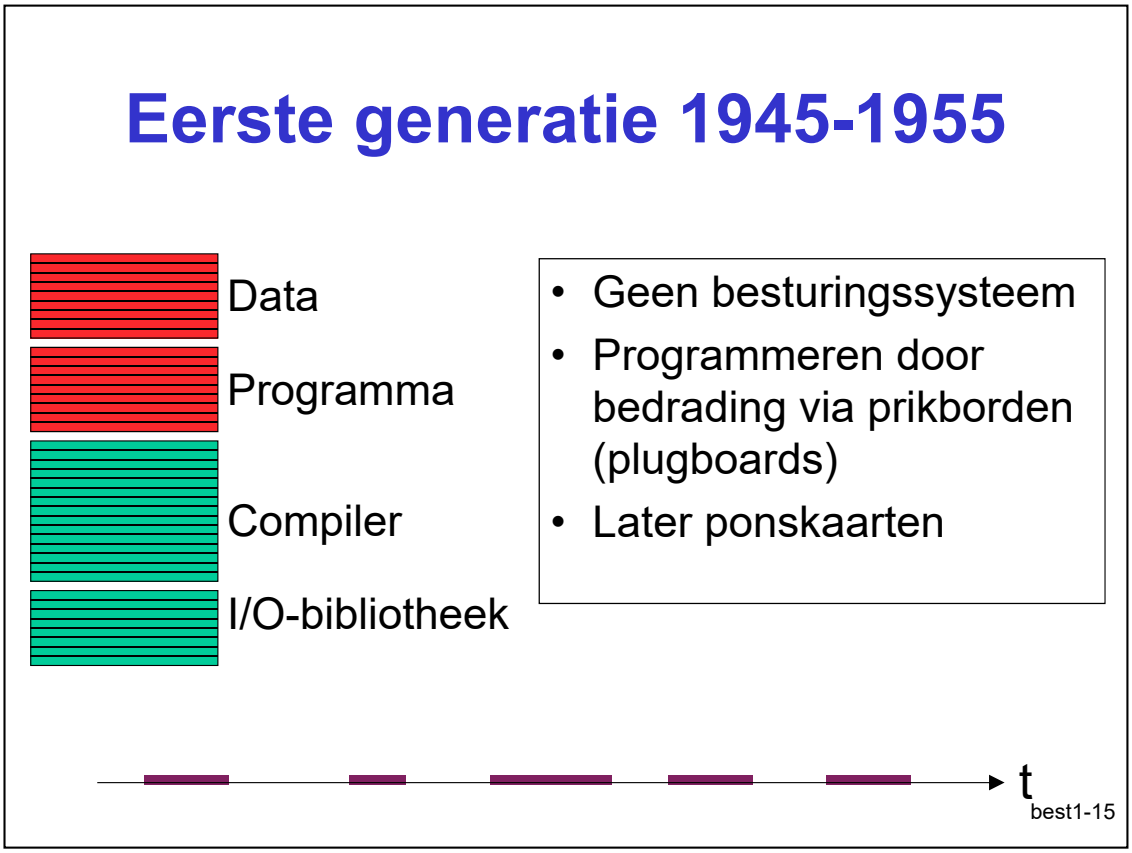
- Eerste generatie 1945 - 1955
  vacuümbuizen, prikborden
  paar computers ter **wereld**
- Tweede generatie 1955 - 1965
  transistors, batchsystemen, mainframes
  één computer per **bedrijf**
- Derde generatie 1965 – 1980
  IC's en multiprogrammering, minicomputers
  één computer per **afdeling**
- Vierde generatie 1980 – 1995
  microprocessors, pc's, laptops
  één computer per **persoon**
- Vijfde generatie 1995 – heden
  ingebedde systemen, smartphones, tablets
  verschillende computers per persoon

best1-14

Besturingssystemen ontstonden kort na het ontstaan van de eerste computers. In het begin bestonden ze uit een verzameling code die niets met de applicatie te maken heeft (zoals bibliotheken om gegevens in te lezen of te printen). In navolging van de programma's uit die tijd werden deze besturingssystemen in assembler geschreven.

Tegenwoordig worden de besturingssystemen in hoofdzakelijk in C of C++ geschreven waardoor ze gemakkelijker ontwikkelbaar en porteerbaar zijn. Een hedendaags besturingssysteem bestaat uit miljoenen regels code.

De ontwikkeling van besturingssystemen loopt grosso modo parallel met de ontwikkeling van de computersystemen. Men onderscheidt 5 generaties.

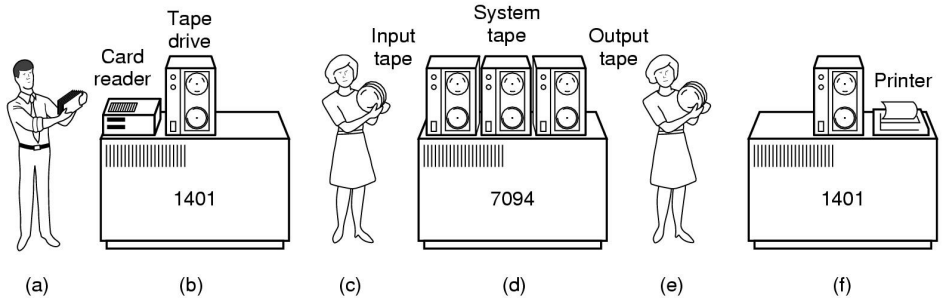


In de eerste generatie was er nog geen sprake van een besturingssysteem, maar veeleer van een IO bibliotheek. Programma's werden in het geheugen ingeladen van ponskaart. Men begon met een stapeltje ponskaarten met daarop de IO-bibliotheek. Daarop legde men dan de ponskaarten met daarop de compiler die men wilde gebruiken, gevolgd door het programma dat moest gecompileerd worden, en nadien de data waarmee het programma moest werken. Deze volledige stapel ponskaarten (dit werd een job genoemd) werd toen ingelezen en het geheel werd uitgevoerd. Bij ontstentenis van on-line opslag moest dit proces per programma-uitvoering herhaald worden. Gelukkig waren de meeste jobs toen nog vrij klein.

Als we naar het processorgebruik kijken, dan zien we dat er tussen de uitvoering van twee jobs een vrij groot ongebruikt tijdsinterval ligt. In de tweede generatie probeert men daar iets aan te verhelpen.

Tweede generatie 1955-1965

- Opkomst van de mainframes
- Off-line verwerking met tapes



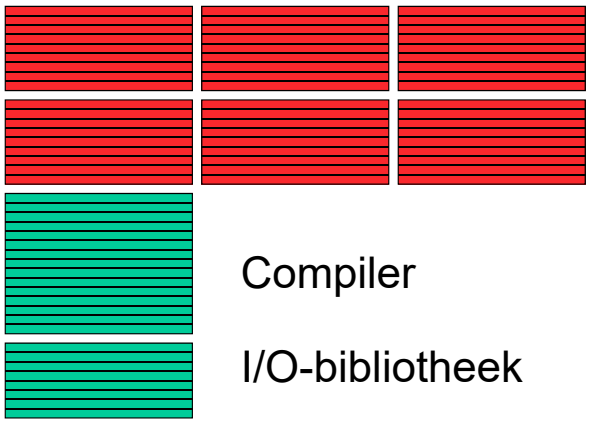
Nadeel: doorvoertijd wordt groter

best1-16

De tweede generatie wordt gekenmerkt doordat men het proces van het inlezen van een job, en het wegschrijven/afdrukken van het resultaat losgekoppeld wordt van de eigenlijke uitvoering. Op een afzonderlijke machine worden de verschillende jobs ingelezen, gesorteerd en op een tape weggeschreven. Die inleesmachines konden gemakkelijk ontdubbeld worden om meer jobs per tijdseenheid te kunnen inlezen. Eenmaal de tape klaar kon deze overgebracht worden naar de centrale computer (mainframe genoemd) die de verschillende jobs snel na elkaar kon inlezen van tape, uitvoeren en het resultaat teug naar tape kon schrijven. Op een derde computer konden de resultaten van de berekening dan verder verwerkt worden (afdrukken op ponskaart of afprinten).

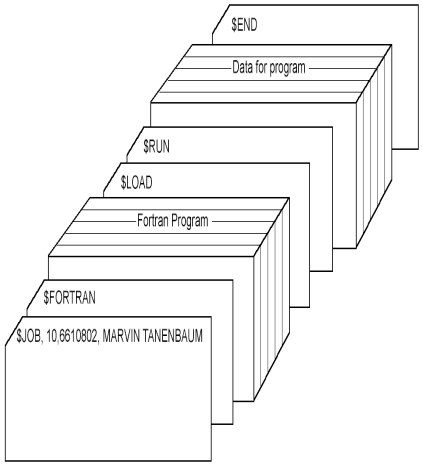
De rekencapaciteit van de mainframe werd hierdoor beter benut, maar de doorvoertijd (de tijd die verloopt tussen het aanbieden van de job en het terugkrijgen van het resultaat) werd hierdoor wel aanzienlijk langer.

Tweede generatie 1955-1965



Compiler

I/O-bibliotheek



Vb.: FMS: Fortran Monitor System: residente monitor

best1-17

Nu er meerdere jobs tegelijk aangeboden werden was het b.v. ook mogelijk om jobs samen te voegen. Het werd bijvoorbeeld mogelijk om alle jobs in dezelfde programmeertaal samen te voegen en de compiler slechts éénmaal in te laden. Op die manier ging er minder tijd verloren tussen het inladen van de verschillende jobs.

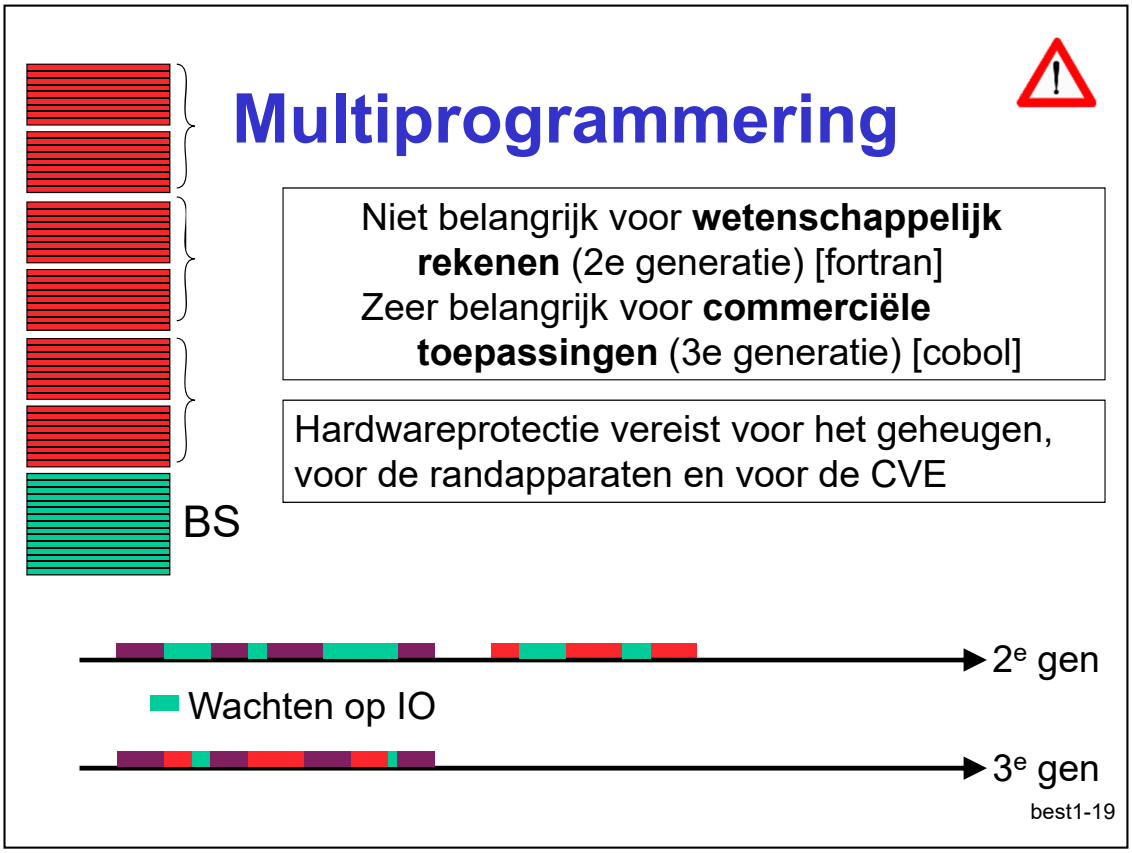
Uit die tijd stamt FMS of het Fortran Monitor System en JCL of Job Control Language. JCL is een eenvoudige scriptingtaal die toelaat het proces van inladen, compileren en uitvoeren van een taak wat te controleren.

Derde generatie 1965–1980

- Opkomst van de computerarchitectuur (IBM 360, 370, 4300, 3080, 3090)
- Opkomst van de **minicomputers** (PDP-1, 1961)
- Eerste echte BS: OS/360: verschrikkelijk complex, talloze bugs…
- Introductie van **multiprogrammering**
- Introductie van **spooling**
- Introductie van **time sharing**

best1-18

De derde generatie is eigenlijk de bloeitijd van de besturingssystemen. In deze periode werden nagenoeg alle belangrijke concepten bedacht en geïmplementeerd. Het begon allemaal met de opkomst van de IBM 360 die eigenlijk de eerste machine was met een echte computerarchitectuur. Vanaf nu kon men een besturingssysteem ontwikkelen dat ook geschikt zou zijn voor de volgende generaties van hardware. Het loonde dus de moeite om te investeren in systeemsoftware. Het resultaat was OS/360. Het resultaat was echter niet fraai: het had monsterachtige afmetingen, het was weinig stabiel, had meer gekost dan geraamd, enz. Het was meteen ook het begin van de software engineering als discipline. Als voornaamste nieuwigheden uit de derde generatie vermelden we: multiprogrammering, spooling en timescharing.



Multiprogrammering houdt in dat er in plaats van 1 job meerdere jobs in het geheugen gehouden worden zodat er snel tussen de verschillende jobs kan omgeschakeld worden. De voornaamste reden om dit te doen was het verhogen van de rendement van de mainframes. Men had immers vastgesteld dat zelfs indien de jobs elkaar zonder onderbreking opvolgden, de belasting van de CVE toch geen 100% was. De oorzaak was te zoeken in de wachttijden die ontstaan bij invoer/uitvoer. Bij sommige operaties (b.v. het terugspoelen van een tape) kan deze wachttijd aanzienlijk zijn. Als er zich op dat ogenblik een tweede job in het geheugen bevindt, is het interessanter om tijdens deze wachttijd aan een andere job te werken. Op die manier kon men het rendement van de CVE verder verhogen. De enige kostprijs was voldoende RAM-geheugen om de diverse jobs te kunnen inladen en de noodzaak voor een protectiemechanisme om de verschillende jobs van elkaar af te schermen in het geheugen.

Het probleem van de wachttijden hing voor een stuk samen met de opkomst van de administratieve toepassingen. Voor wetenschappelijke berekeningen waren er niet zo heel veel input/outputbewerkingen vereist – deze waren vooral rekenintensief. Administratieve toepassingen hadden op dit punt een totaal ander gedrag omdat zij veel vaker gegevens op schijf of op tape moesten raadplegen en de eigenlijke berekeningen relatief eenvoudig waren.

Spooling

Simultaneous Peripheral Operation On-Line

- disk i.p.v. tape
- niet langer sequentieel
- minder wachttijden (terugspoelen, nieuwe tapes opleggen,…)

best1-20

SPOOLing betekent dat één randapparaat door verschillende simultaan uitvoerende jobs kan gedeeld worden. Dit was noodzakelijk om tot multiprogrammering te kunnen overgaan. In essentie kwam het erop neer dat de interactie niet langer met de fysieke randapparaten gebeurde, maar wal via bestanden die dan op schijf werden opgeslagen. Op die manier konden verschillende simultaan uitvoerende jobs b.v. dezelfde printer gebruiken. Ze schreven allemaal naar een bestand, en van zodra de job beëindigd was, werd het bestand afgeprint. Doordat schijven sneller waren dan tapes, boekte men hier ook een prestatiewinst. Verder was met niet langer gebonden aan de sequentiële toegang van tapes, maar kon men gemakkelijk toegang krijgen tot verschillende delen van het bestand.

Time sharing

- **CTSS**: Compatible Time Sharing System, MIT 1962, eerste **interactieve** vorm van multiprogrammering
- Opvolger **MULTICS**: MULTiplexed Information and Computing Server (MIT, Bell Labs, General Electric), later gecommercialiseerd door Honeywell
- Ken Thompson (Bell Labs) porteerde Multics voor de PDP-7, hetgeen later resulteerde in **Unix** en zijn varianten

best1-21

De laatste vernieuwing werd gerealiseerd met het interactief gebruik van de computer. Om verschillende gebruikers toe te laten om interactief te werken, moest men de verschillende programma's uiteraard in het geheugen houden (multiprogrammering), en doordat het interactieve programma's waren, waren de wachttijden enorm groot (invoer van het toetsenbord gebeurt op een secondeschaal). Het interactief werken met een computer opende een hele resem nieuwe toepassingen en was enkel mogelijk door de invoering van een nieuw randapparaat: de terminal – eerst een toetsenbord met een printer, later een videoterminal.

Dit onderzoek is na verloop van tijd uitgemond in de ontwikkeling van Unix dat de voorvader geworden is van nagenoeg alle hedendaagse interactieve besturingssystemen.

Vierde generatie 1980–1995

- Microcomputers of PC's
- Gary Kildall, Digital Research: 1974 CP/M (Control Program for Microcomputers)
- IBM PC (1980): MS-DOS, Xenix
- Apple (1983): introductie van de GUI
- Vanaf 1995: Windows NT, 2000, XP, Vista, 7, 8; convergentie met Windows 95, 98, 10
- Daarnaast ook Unix/Linux
- Tegenwoordig allemaal **netwerkbesturingssystemen**

best1-22

Met de opkomst van de microcomputers en later de PC werd het aspect processorrendement minder belangrijk. Er was toch maar één gebruiker en één job. Het eerste besturingssysteem voor microcomputers was CP/M. Later zou een kloon van CP/M uitgroeien tot DOS en via Microsoft evolueren tot het dominante besturingssysteem voor de vroege PC's. Apple computer had meteen het nut van grafische user interfaces ingezien en deze heel vroeg op de markt gebracht (1983). Het heeft tot 1995 geduurd vooraleer Microsoft voor een passend antwoord kon zorgen. Tegenwoordig is Windows het dominante besturingssysteem voor de desktop. Sinds ongeveer 2000 wordt ook Linux gebruikt als desktopbesturingssysteem.

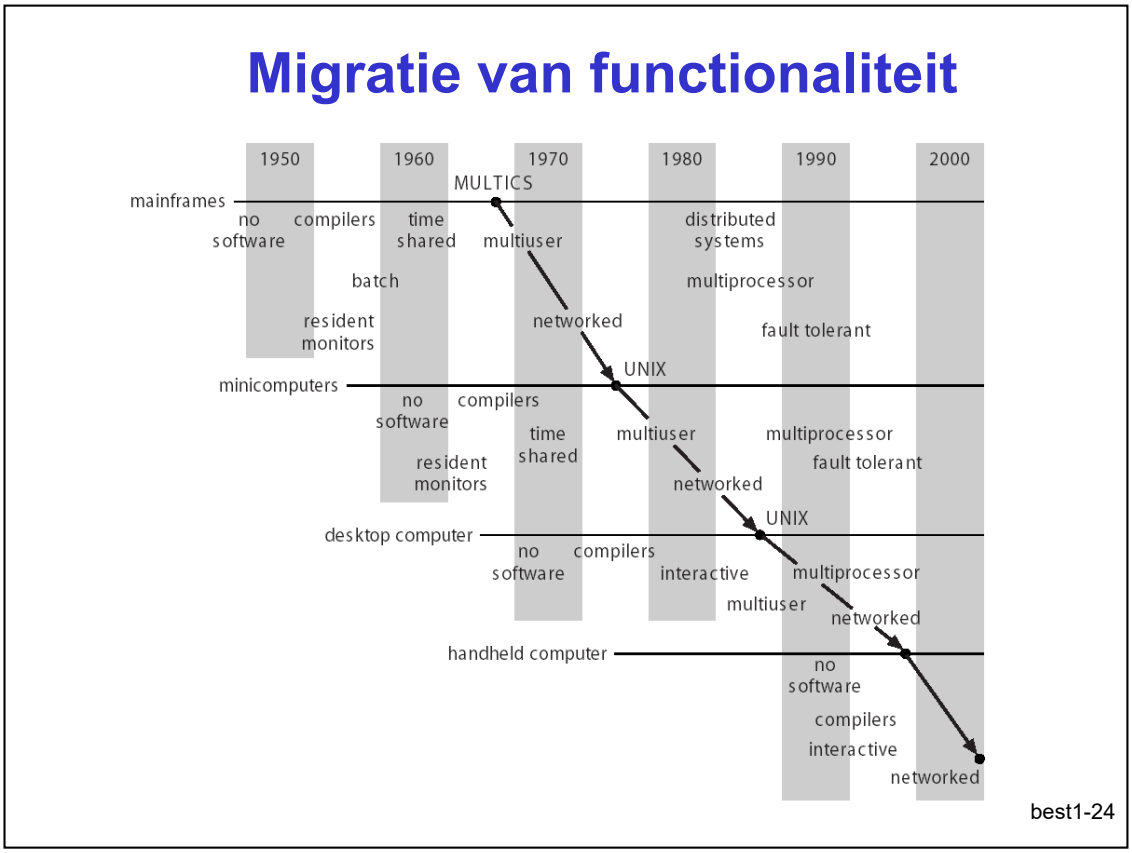
Vijfde generatie: 1995–heden

- Ingebedde processors en besturingssystemen in GSM's, PDA's, GPS'en, MP3-spelers, enz.
- Communicatie en verwerking van media belangrijk, quality of service (QoS), beveiliging, …
- Windows Mobile, Symbian, Android, Palm OS, iOS, …

best1-23

Sinds 1995 vond er nog een bijkomende evolutie plaats. De computers zijn zo klein geworden dat ze draagbaar geworden zijn. Voorbeelden zijn MP3-spelers, GSM's, PDA's. Ook deze systemen hebben een besturingssysteem: Pocket PC, Symbian, Palm OS, Windows Mobile,…

Belangrijk voor deze systemen is: goede communicatiemogelijkheden met de buitenwereld, een efficiënte verwerking van multimediale informatie zoals geluid en beeld, een goede beveiliging, …



Als we de geschiedenis van besturingssystemen even overlopen dan zien we dat nieuwe systemen steeds vertrekken van een beperkte functionaliteit, die – naarmate de systemen populairder worden – uitgebreid wordt. Zo was DOS bij zijn introductie vergelijkbaar met een besturingssysteem van begin de zestiger jaren. Geleidelijk aan werden diverse aspecten zoals ondersteuning van het netwerk, multiprogrammering, spooling toegevoegd. Ook de eerste PDA's en GSM's hadden een zeer beperkte functionaliteit die snel is opgeklommen tot die van een volwaardig computersysteem.

Overzicht

- Wat is een besturingssysteem
- Een korte geschiedenis
- Soorten besturingssystemen
- Architectuur van besturingssystemen
- Concepten in besturingssystemen

best1-25

Soorten besturingssystemen

- Mainframes
- Supercomputers
- Servers
- Desktop/Laptop
- Mobiele toestellen
- Ware-tijdsbesturingssystemen
- Chipkaartsystemen/Sensornetwerken

best1-26

Mainframes

- Sterk in IO (TB opslagcapaciteit)
- Prestatie gemeten in transacties/s
- Zeer betrouwbaar en veilig (redundantie)
- Achterwaarts compatibel: z/Architecture is nog steeds compatibel met IBM 360
- Vooral commerciële toepassingen
  - Batch
  - Transacties (databanken)
- Vb. z/OS (IBM)

best1-27

Supercomputers

- Sterk in rekenkracht
- Prestatie gemeten in floating point operations/s (Tflops, Pflops, Eflops, …)
- Vooral wetenschappelijke toepassingen
  - Oorspronkelijk vooral voor de berekening van complexe wiskundige modellen
  - Maar nu ook voor big data berekeningen
  - Recent ook voor deep learning
- Vooral open source systeemsoftware
- Vaak uitgerust met acceleratoren (bv GPUs)

best1-28

Servers

- Vooral terug te vinden in datacenters, maar ook alleenstaande file servers, mail servers, web servers, …
- Clouddiensten: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), …

best1-29

Desktop/Laptop

- Vooral Windows, macOS
- Vooral de gebruikerservaring primeert
- Business-2-Consumer (B2C) markt: marges zijn relatief klein
- x86 architectuur domineert
- Veel randapparaten beschikbaar

best1-30

Mobiele toestellen

- Smartphones, tablets, ...
- B2C markt, veel concurrentie
- Nieuwe features geïntroduceerd via high-end modellen
- ARM architectuur dominant
- OS: Android, iOS, Tizen, ..

best1-31

Ware-tijdsbesturingssystemen

- Ware tijd ≠ snel ! Ware tijd = op tijd.
- Harde ware tijd
  - Missen van een deadline = fout (bv noodstop)
- Zachte ware tijd
  - Missen van een deadline = kwaliteitsverlies (bv video)
- Interactie met de omgeving
- Gespecialiseerde besturingssystemen: VxWorks, QNX, eCos, RTLinux

best1-32

Chipkaart/sensor netwerken besturingssystemen

- Extreme randvoorwaarden qua verbruik, snelheid, enz.
- Minimale kern
- Soms een Java Virtuele Machine
- TinyOS, LiteOS, Contiki, …

best1-33

Overzicht

- Wat is een besturingssysteem
- Een korte geschiedenis
- Soorten besturingssystemen
- Architectuur van besturingssystemen
- Concepten in besturingssystemen

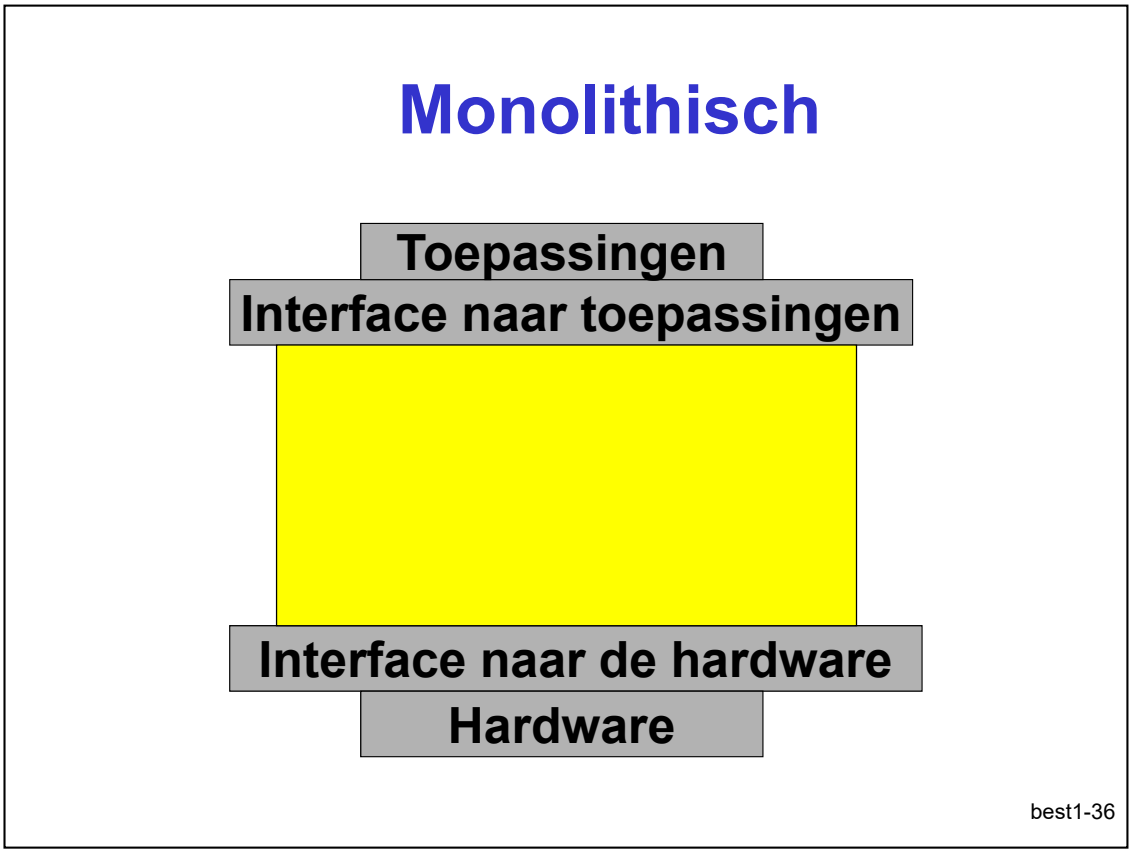
best1-34

Architectuur van besturingssystemen

- Monolitisch
- Gelaagd
- Kernmodules
- Microkern
- Virtuele machines

best1-35

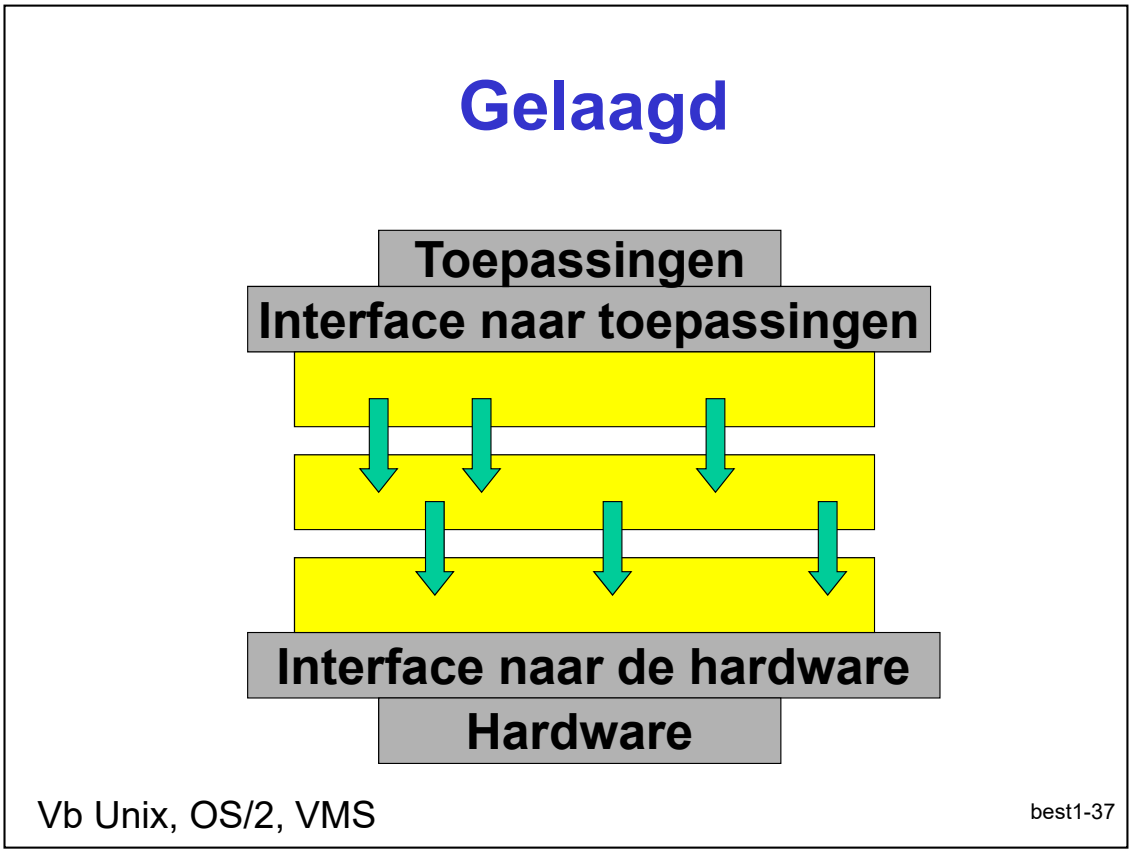
Besturingssytemen kunnen intern op verschillende manieren opgebouwd zijn. We bespreken hier een aantal populaire architecturen.



De vroegste besturingssystemen waren monolithisch wat neerkomt op een verzameling van bibliotheken die elkaar zonder bijkomende beperkingen kunnen oproepen. Een dergelijke architectuur is moeilijk onderhoudbaar. Ook de beveiliging is problematisch omdat een fout in een bibliotheek ongehinderd het gehele systeem onklaar kan maken.

DOS is een voorbeeld van een monolithisch besturingssysteem. Het toepassingsprogramma hoorde normaal gezien via de diverse lagen de BIOS aan te sturen, maar kon dit ook rechtstreeks – alle bescherming omzeilend.

Deze manier van werken die – in een tijdperk van minder krachtige processors – heel courant was om applicaties te versnellen heeft ertoe geleid dat bepaalde ontwerpsbeslissingen naderhand niet meer ongedaan konden gemaakt worden. Zo vonden heel wat ontwikkelaars het handiger om rechtstreeks in het videogeheugen te schrijven i.p.v. via het besturingssysteem rond te gaan. Het gevolg was dat het videogeheugen naderhand niet meer van plaats veranderd kon worden, en dat DOS altijd is blijven worstelen met de beperking van 640 KiB conventioneel geheugen.



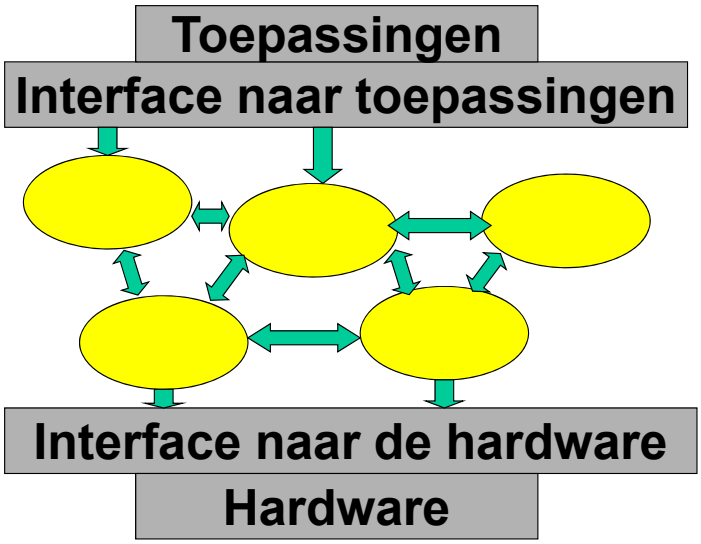
Een betere structuur is de gelaagde structuur waarbij diensten van een hoger niveau verplicht moeten steunen op diensten uit een lager niveau die dan op een welomschreven manier moeten opgeroepen worden. Op die manier kan men verhinderen dat een fout in de bovenste laag schade aanricht in de onderliggende lagen.

Heel wat besturingssystemen uit de jaren ’70 en ’80 hebben geprobeerd om deze architectuur te implementeren. Echt eenvoudig was dit echter niet. Probleem is dat de te implementeren functionaliteit niet steeds te stratifiëren (in lagen op te delen) is.

De basisroutines voor output moeten op de hoogste laag gedefinieerd worden omdat deze door de toepassingen moeten kunnen opgeroepen worden. Als er zich echter een fout voordoet in de onderste laag, dan zal deze ook van dezelfde routine gebruik willen maken om een foutboodschap te genereren, maar dit is in een gelaagde architectuur in principe niet mogelijk.

Unix kan men beschouwen als een systeem met 2 lagen: de gebruikerslaag en de kernlaag. De gebruikerslaag kan niet zomaar de kerngegevens veranderen, en ook de kern kan in principe de gebruikerscode niet uitvoeren (dat zou wel kunnen aan de hand van een callback, maar dat is inherent onveilig). De kern zal dan ook geen gebruik maken van de printf-functie om output te produceren, maar de speciale printk-functie gebruiken.

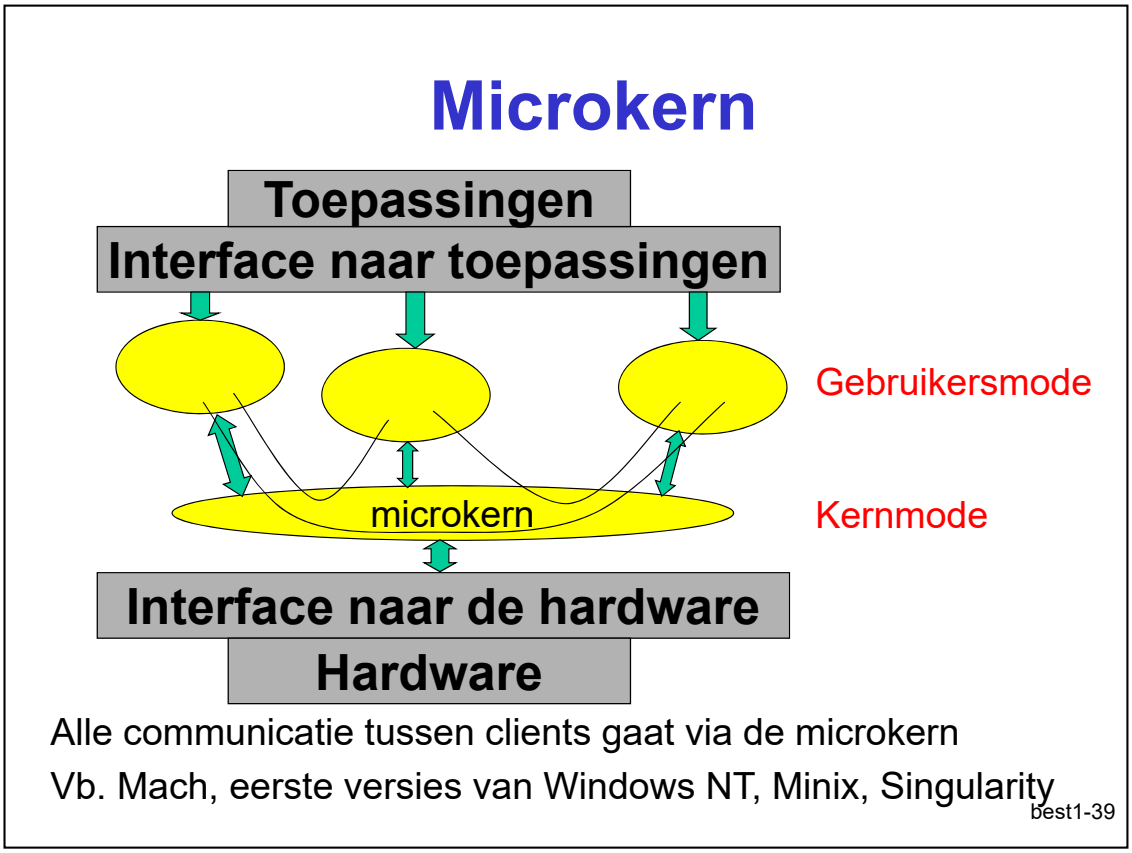
Kernmodules



[Modules kunnen soms dynamisch ingeladen worden]

best1-38

Een modernere architectuur is de client-server architectuur. Hierbij wordt de kern van het besturingssysteem gevormd door een aantal modules – elk met hun eigen goedgedefinieerde interface – die elkaars diensten kunnen oproepen. Verder kunnen modules naar eigen goeddunken vervangen worden door alternatieve implementaties – zolang de interface met de buitenwereld maar gerespecteerd blijft. Op deze manier is het b.v. mogelijk om verschillende bestandensystemen te gebruiken in een besturingssysteem (in de Windowswereld kan men b.v. kiezen tussen FAT en NTFS, ook in Linux heeft men de keuze uit een aantal systemen zoals ext2, ext3). Soms kunnen modules zelfs tijdens de uitvoering ingeladen worden. Solaris, de UNIX variant door SUN, maakt gebruik van diverse kernmodules. Op deze manier kan men het besturingssysteem configureren naar zijn eigen smaak.

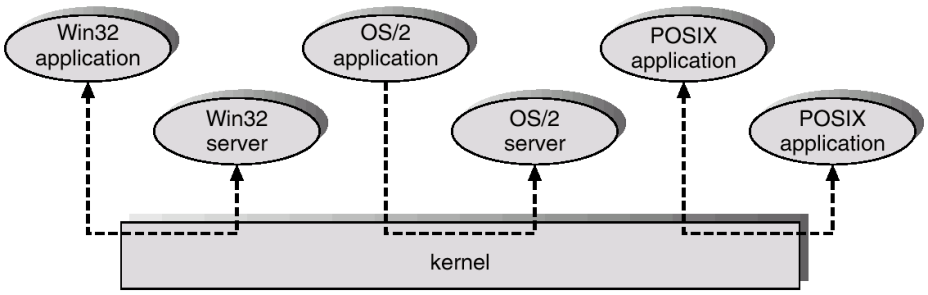


De laatste architectuur betreft de architectuur van een microkern. In een microkern probeert men de code die in kernmode moet uitgevoerd worden tot het absolute minimum te beperken. In de praktijk zal dit gaan over de implementatie van de contextwisseling en de basis synchronisatieprimitieven. Talrijke andere subsystemen worden uitgevoerd in gebruikersmode. Deze subsystemen communiceren dan met elkaar via de microkern.

Een dergelijke kern kan zeer betrouwbaar gemaakt worden omdat er **minder kritische code** is. De grootte van de kern is zo klein dat formele bewijzen tot de mogelijkheden behoren. Een microkern is ook **gemakkelijk porteerbaar** en **gemakkelijk uitbreidbaar**. Het nadeel is wel dat een dergelijke architectuur vertragend werkt omdat een systeemoproep doorgaans communicatie zal veroorzaken met verschillende subsystemen – communicatie die steeds via de microkern moet verlopen met de bijhorende (dure) overgang tussen protectiedomeinen.

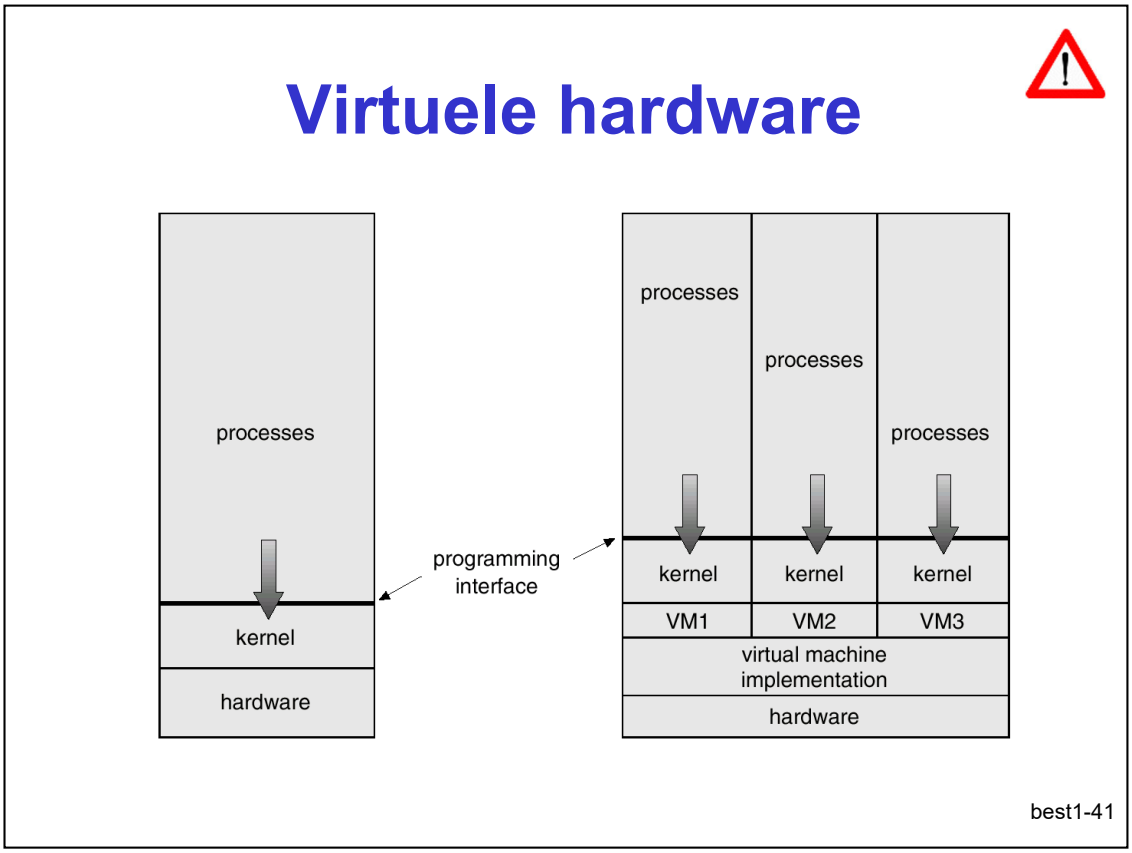
Mach is het eerste besturingssysteem dat opgebouwd werd rond een microkern.

Windows NT Client-Server Structure

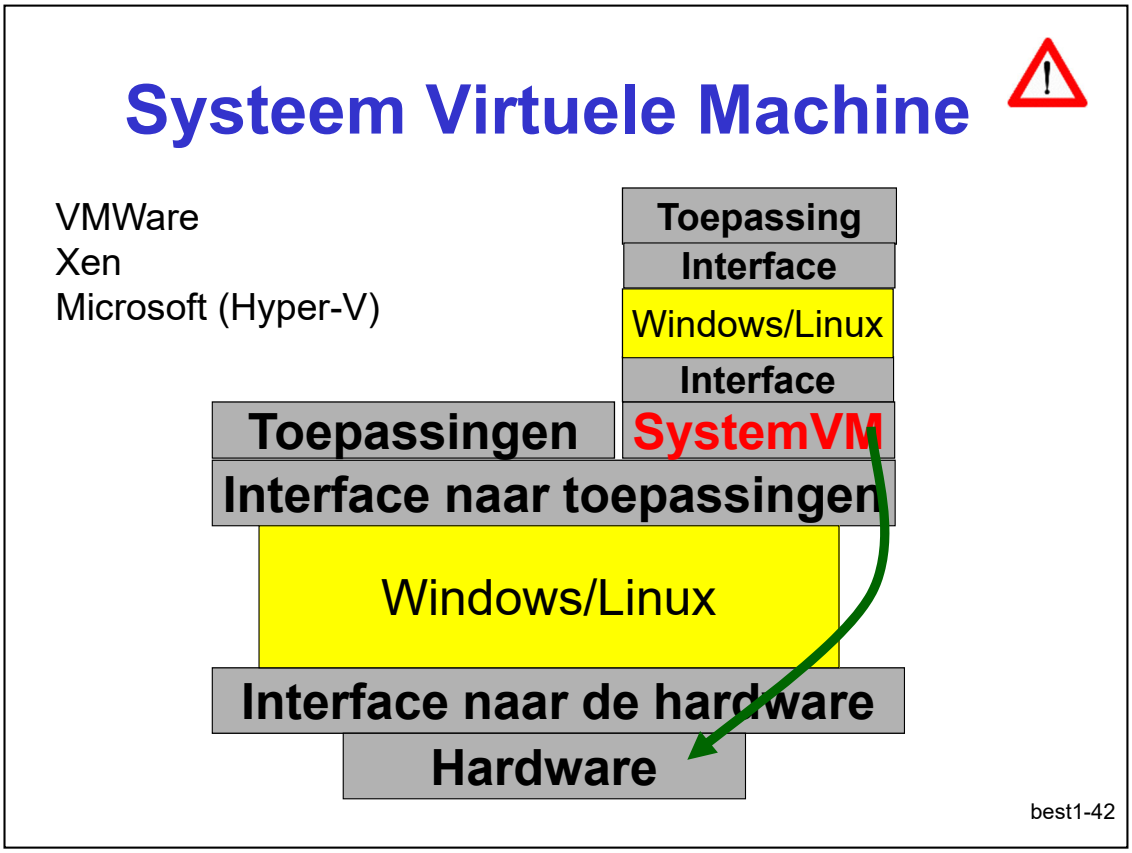


best1-40

Ook de oorspronkelijke Windows NT architectuur was een microkern. Naderhand is men daarvan afgestapt om prestatieredenen en heeft men het volledige grafische subsysteem geïntegreerd in de kern omdat het anders te traag werd voor de spelletjesmarkt.



In de regel werkt een besturingssysteem rechtstreeks met de onderliggende hardware. Mits de nodige softwareaanpassingen is het echter ook mogelijk om een besturingssysteem te laten werken op virtuele hardware. Dit noemt men het virtualiseren van de hardware. De virtualisatiesoftware biedt als het ware de onderliggende hardware opnieuw aan – en soms zelfs meer dan eens. In de bovenstaande figuur wordt de hardware driemaal aangeboden door de virtuele machines VM1, VM2 en VM3. Bovenop elk van deze drie virtuele machines kan dan een besturingssysteem geïnstalleerd worden (het kunnen zelfs drie verschillende zijn). Bovenop elk van deze besturingssystemen kunnen dan de bijhorende applicaties uitgevoerd worden. Virtualisering is een techniek die reeds decennia lang in de wereld van de mainframes gebruikt wordt en pas nadien ingang gevonden heeft in de serverwereld – inclusief de nodige hardwareondersteuning.



Een systeem virtuele machine is een programma dat een hardwareplatform implementeert. De schijf van de virtuele hardware wordt geïmplementeerd als een (groot) bestand op de schijf van het onderliggend systeem. Op die virtuele schijf kan dan een besturingssysteem geïnstalleerd worden. Dat besturingssysteem is zich niet bewust van het feit dat het met virtuele hardware werkt. Het grote verschil is wel dat alle geprivilegieerde instructies van het besturingssysteem nu in gebruikersmode uitgevoerd worden, waardoor ze protectiefouten veroorzaken. Die fouten worden door de virtuele machine opgevangen en doorgegeven aan het onderliggende besturingssysteem voor verdere afhandeling. Dit werkt uiteraard vertragend, maar de voordelen overtreffen vele malen dit nadeel.

Deze oplossing laat toe om bovenop een Linux platform een Windowsplatform te draaien – inclusief alle Windowssofware, en vice versa.

Overzicht

- Wat is een besturingssysteem
- Een korte geschiedenis
- Soorten besturingssystemen
- Architectuur van besturingssystemen
- Concepten in besturingssystemen

best1-43

Concepten in besturingssystemen

- Abstrahering
- Virtualisering
- Onderscheid tussen Mechanisme & Beleid
- Onderscheid tussen Interface en Implementatie

best1-44

Een aantal belangrijke concepten uit de informatica zijn ontstaan in het domein van de besturingssystemen. Deze worden hierna besproken.

Abstrahering

- Hardware-elementen worden geabstraheerd door software-elementen:
  - Sectoren op schijf worden bestanden
  - Input/output apparaat wordt software device (met bestandsinterface)
  - Systeemparameters worden bestanden (/proc bestandensysteem)

best1-45

Bij abstrahering worden fysieke hardware-elementen vervangen door software-elementen. Randapparaten worden voorgesteld door zgn devices die zich in een aantal gevallen als bestand gedragen, sectoren op een schijf worden geabstraheerd tot bestanden, enz.

Abstracties zijn een heel stuk gemakkelijker om mee te werken en bovendien zijn ze ook minder hardwareafhankelijk. Dit laatste maakt de ontwikkeling van platformonafhankelijke software een stuk eenvoudiger.

Virtualisering

- Het geheugen wordt opgedeeld in verschillende gescheiden (virtuele) adresruimten, gebruikt door processen
- De rekentijd wordt verdeeld tussen de verschillende draden die kunnen beschouwd worden als virtuele cpu's
- Randapparaten worden gevirtualiseerd (o.a. door spooling)

best1-46

Naast abstrahering hebben we virtualisering van de hardware, dit wil zeggen dat één fysiek hardware-element kan gebruikt worden als verschillende – onafhankelijke – virtuele hardware-elementen. Voorbeelden hiervan zijn de verschillende virtuele adresruimten die één fysieke adresruimte delen, de verdeling van de processortijd over verschillende processen, en virtuele randapparaten die via spooling ter beschikking gesteld worden.

Mechanisme & Beleid

- Mechanisme definieert de manier waarop iets gebeurt
- Het beleid bepaalt hoe er gebruik gemaakt wordt van het mechanisme
- Bv. Protectiemechanisme en het gebruik ervan, planningsmechanisme en de parameters om het af te stellen

- **Mechanisme is vast**, **beleid kan aangepast worden**

best1-47

Besturingssystemen worden ingezet op miljoenen plaatsen die elk hun eigen specifieke randvoorwaarden opleggen en moeten op deze situatie dus flexibel kunnen inspelen. In een poging om die flexibiliteit op een structurele manier te implementeren heeft men het onderscheid gecreëerd tussen mechanisme en beleid.

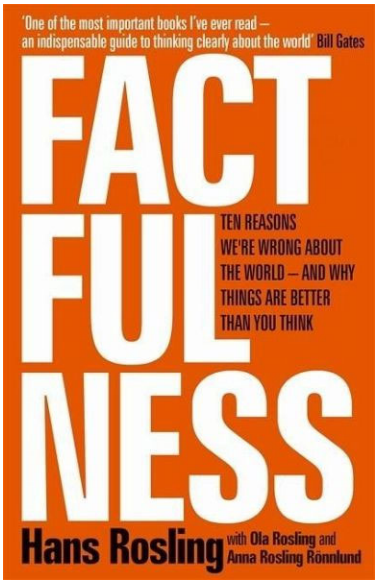
Zo heeft men voor de afscherming van bestanden een protectiemechanisme gecreëerd waarbij men per bestand rechten kan toekennen aan bepaalde gebruikers. De manier waarop men van dit mechanisme gebruik maakt wordt het beleid genoemd, en dit is de verantwoordelijkheid van de lokale systeembeheerders, niet van de ontwikkelaars van het besturingssysteem: zij hebben voor het mechanisme gezorgd. Andere voorbeelden zijn de prioriteiten van de procesplanner, de regels van de firewall, enz.

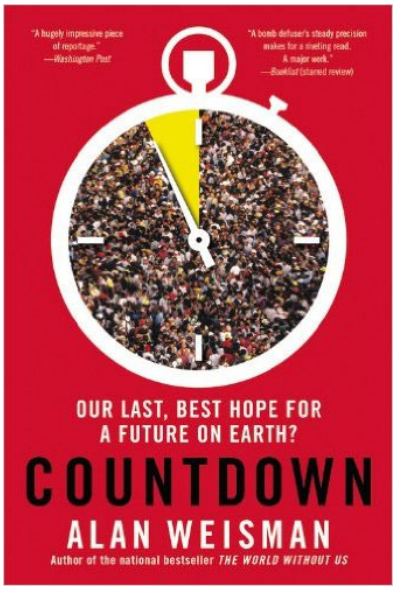
Interface en Implementatie

- Bestandsinterface gekoppeld aan diverse onderliggende bestandensystemen
- Win32 interface, bovenop diverse implementaties ervan

best1-48

Tegenwoordig vinden wij het heel gewoon dat een bibliotheek een gestandaardiseerde API aanbiedt aan de programmeur. Het concept van interface en implementatie hebben we ook te danken aan de besturingssystemen. Het was essentieel om besturingssystemen het kunnen upgraden zonder de applicaties te moeten herschrijven.





best1-49

best1-49