Tim Mallezie 2de bach. III

Architectuur van besturingssystemen: Vraag A2.

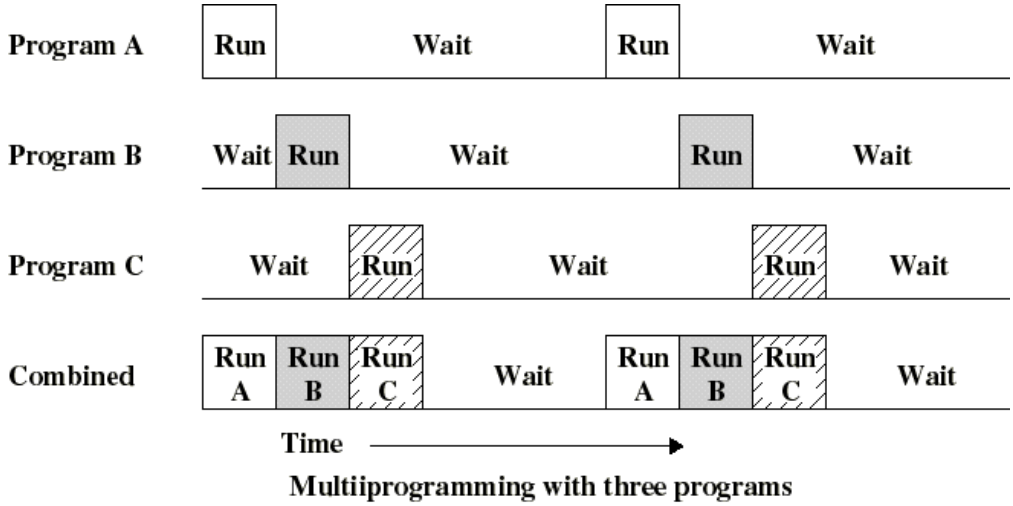
Procesbeheer: *kenmerken van moderne besturingssystemen.*

1. Bespreek de (drie) meest typische *kenmerken van moderne besturingssystemen*.
2. In hoeverre beantwoorden UNIX, Linux en Windows NT hieraan? Geef hierbij de belangrijkste elementen van de detailstructuur van deze besturingssystemen.

Kenmerken van moderne besturingssystemen:

1) Multitasking en multithreading:

Het CPU-gebruik kan verbeterd worden, door programma's niet sequentieel, maar tegelijkertijd in het hoofdgeheugen te brengen, dit wordt Multitasking of multiprogramming genoemd. De volgorde van verwerking wordt bepaald door hun relatieve prioriteit, en of ze al dan niet wachten op I/O. Als een programma onderbroken wordt door een interrupt kan de routine voor interruptafhandeling na afhandeling beslissen om de controle terug te geven aan het onderbroken programma of aan een ander programma. Door meerdere programma's met elkaar te verweven (interleaving) kan men de globale doorvoersnelheid verhogen. In systemen met Multitasking blijft elk individueel programma de indruk wekken dat het als enige van de systeembronnen gebruikmaakt. Het besturingssysteem moet instaan voor de transparantie van gedeeld gebruik van processortijd, geheugen en andere bronnen.



Multiiprogramming with three programs

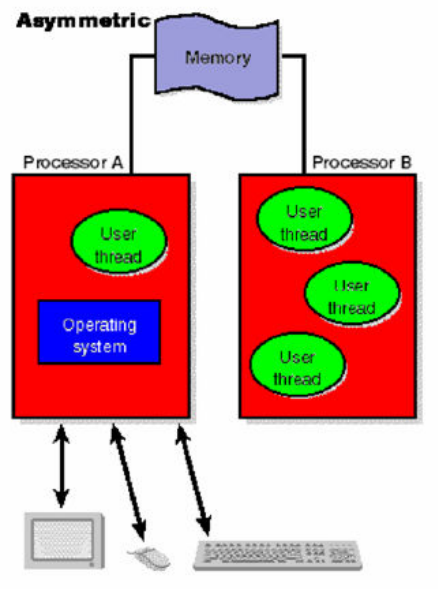
Een meer geavanceerde vorm van multitasking is multithreading.
Multithreading is niet hetzelfde als multiprocessing, dat betekent dat het systeem meerdere processen bevat.
Er is een verschil tussen processen en threads, een proces is een eenheid voor de eigendom van bronnen, terwijl een thread een eenheid is voor de verdeling van processorinstructies.
Binnen elk proces kunnen meerdere gelijktijdige threads gebruikt worden voor de uitvoering van de instructies. Alle threads delen alle bronnen die aan het proces zijn toegewezen.
Alle threads binnen een proces zijn evenwaardig.
Net zoals processen kunnen threads zich in verschillende toestanden bevinden. De besturingssysteem scheduler maakt geen onderscheid of threads al dan niet tot hetzelfde proces behoren. Het wisselen van threads vraagt aanzienlijk minder overhead dan het wisselen van proces.

Tim Mallezie 2de bach. III

Architectuur van besturingssystemen: Vraag A2.

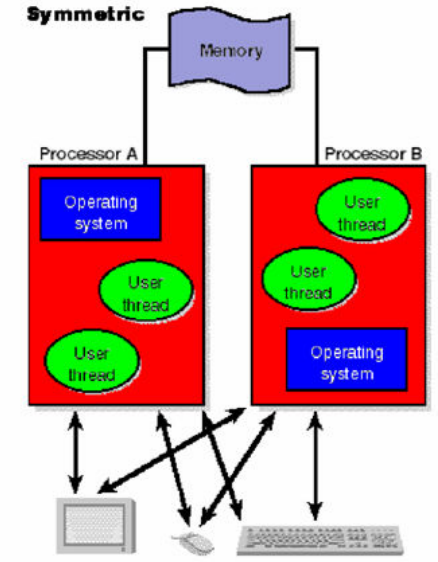
2) Ondersteuning van symmetrische multiprocessing:

Asymmetrische multiprocessing:



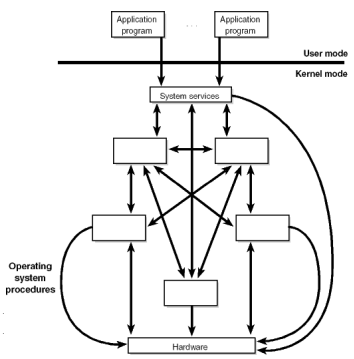
Bij asymmetrische multiprocessing wordt het besturingssysteem altijd uitgevoerd op een bepaalde master processor. De master is verantwoordelijk voor scheduling, en heeft dus volledige controle over het volledige geheugen en de andere bronnen. Heeft de andere processor een dienst nodig, dan moet die een verzoek sturen naar de master en wachten totdat de dienst is uitgevoerd. Deze manier vereist weinig aanpassing aan een besturingssysteem dat reeds multitasking voor één processor mogelijk maakt. De master wordt echter de bottleneck voor het volledige systeem.

Symmetrische multiprocessing:



Bij symmetrische multiprocessing kan de kernel worden uitgevoerd op elke processor.
De kernel kan worden opgebouwd worden als meerdere processen of threads, die parallel kunnen worden uitgevoerd, of elke processor kan een kopie van het volledige besturingssysteem uitvoeren. Op deze manier kan de scheduling op elke processor worden uitgevoerd.
Deze mogelijkheid stelt hogere eisen aan het besturingssysteem. Het biedt echter wel ruime mogelijkheden op het gebied van beschikbaarheid en stapsgewijze uitbreidingsmogelijkheden.
Systemen met meerder processoren geven uiteraard ook betere prestaties.

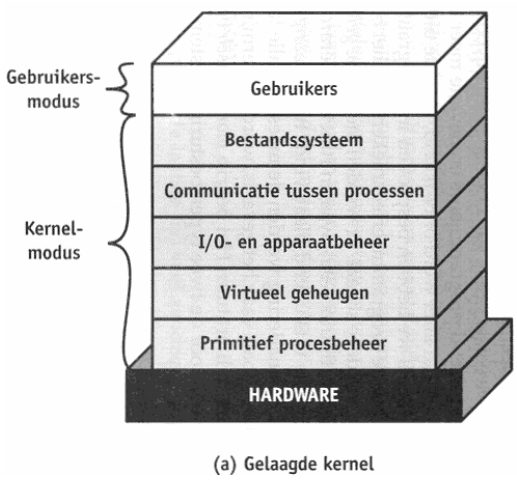
3) Modulair ontwerp:



De meeste besturingssystemen hadden tot voor kort één kolossale monolithische kernel. De volledige kernel wordt uitgevoerd in dezelfde gedeelde geheugenruimte, zonder restricties voor toegang tot de hardware.
Het voordeel van dergelijk systemen is dat ze zeer efficiënt zijn, een besturingssysteem moet echter net als alle andere software

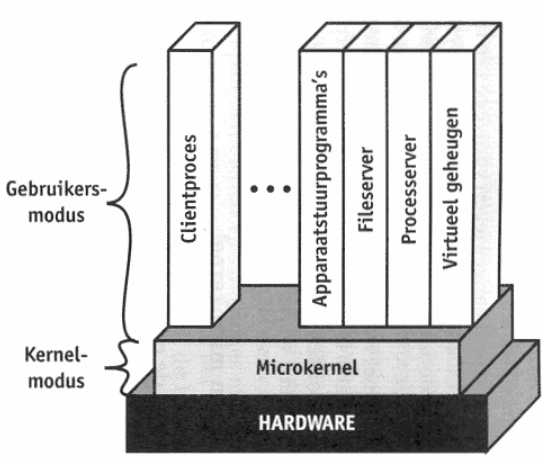
regelmatig aangepast worden. Dit leidt tot de noodzaak van een meer modulair ontwerp.

Een systeem kan op verschillende manieren in modules worden ingedeeld. Eén van de mogelijk manieren is om duidelijk hiërarchisch gescheiden lagen in te voeren. Elk niveau is verantwoordelijk voor een deel van de functies van het besturingssysteem. Veranderingen op één niveau hebben in principe geen impact op de andere niveaus.

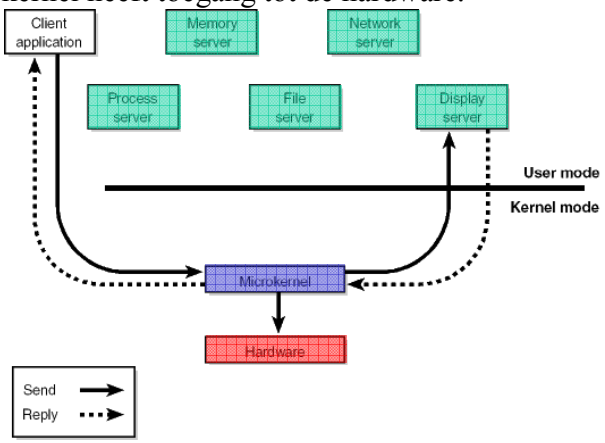


(a) Gelaagde kernel

Er werd een theoretisch model voor een hiërarchisch besturingssysteem opgesteld dat uit 13 niveaus bestaat.
Deze benadering heeft echter verschillende tekortkomingen, de volledige kernel blijft draaien in de kernelmodus. Elke laag heeft dus toegang tot de hardware. Bij besturingssystemen blijkt ook dat men de diverse modules niet in een zodanig specifieke volgorde kan stapelen dat van een strikte gelaagdheid sprake kan zijn. Tussen diverse, niet noodzakelijk aangrenzende lagen blijven een groot aantal noodzakelijke interacties nodig.
Op niveau van een globaal besturingssysteem vindt men dan ook geen geslaagde implementatie van dit model terug. Het principe van gelaagdheid wordt wel succesvol op bepaalde deelmodules toegepast.
In een ander ontwerp, de microkernelarchitectuur, ook wel client/server-model genoemd, worden slechts enkele essentiële functies van het besturingssysteem toegewezen aan de kernel. De kernel wordt zo klein mogelijk gehouden. Andere diensten worden verzorgd door processen (servers). Serverprocessen werken onderling samen door berichten door te geven via de kernel. Enkel de kernel heeft toegang tot de hardware.



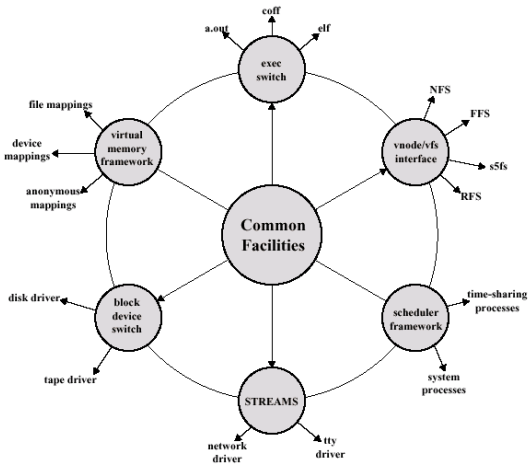
(b) Microkernel



De microkernel benadering maakt een meer modulaire ontwikkeling mogelijk. Ook kan men een meer gereduceerde implementatie maken met beperkte functionaliteit.
Het gebruik van objectgeoriënteerde technieken biedt mogelijkheden om de kernel uit te bereiden.
Alle hardware afhankelijkheid bevindt zich in de microkernel.
Deze benadering is ook geschikt voor een gedistribueerde omgeving. Dit is een cluster van van afzonderlijke computers met gedeelde secundaire opslag.

Architectuur van besturingssystemen: Vraag A2.

Microkernel benaderingen blijken in de praktijk echter te weinig performant.
In de praktijk is men dan ook afgeschaft van een doorgedreven gelaagde of microkernel benadering.
In veel moderne systemen werkt men nu met verschillende modules, maar men is afgestapt van het definiëren van lagen, en de communicatie ertussen. Men zorgt er voor dat elke module instaat voor een specifieke functionaliteit, en dan implementeert men deze module op een monolithische manier. Er is een hoofdmodule, de kernel, die zorgt voor de basisdiensten, en de specifieke modules worden geladen als om die functionaliteit gevraagd wordt. Het voordeel is dat toegelaten wordt dat de verschillende modules met elkaar kunnen communiceren. De hoofdmodule kan ook met de andere modules communiceren, en weet hoe deze geladen moeten worden. Een mooi voorbeeld van deze aanpak is Solaris.



Implementatie in specifieke besturingssystemen:

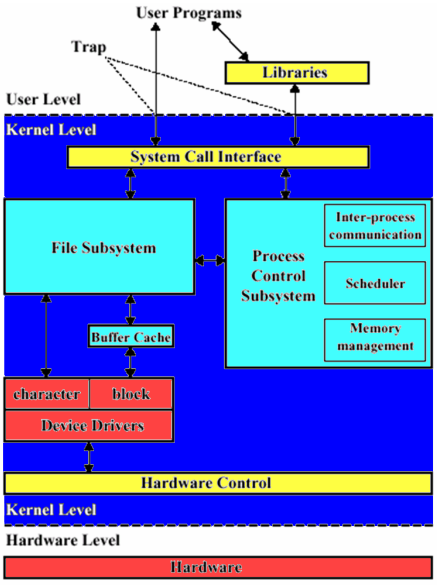
Multitasking en multithreading: UNIX.

UNIX ondersteunt geen multithreading.

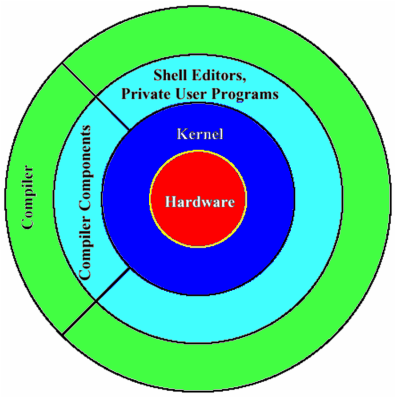
Ondersteuning van asymmetrische multiprocessing: UNIX.

Er is in UNIX niets voorzien om gegevenstoegang te beschermen tegen gelijktijdige toegang door meerdere processors. Asymmetrische multiprocessing is dus niet (goed) ondersteund in UNIX.

Modulair ontwerp: UNIX.



UNIX bestaat zowel uit een kernel, die de hardware isoleert van de gebruiker, als uit gebruikersdiensten zoals de shell, andere interfacesoftware en de componenten van de C-compiler.



Architectuur van besturingssystemen: Vraag A2.

De kernel van UNIX is weinig modulair opgebouwd. Dit komt door de beperkte hardware mogelijkheden ten tijde van de klassieke UNIX. Het UNIX systeem bestaat uit twee interfaces, één et de gebruiker en één met de hardware. Verder bestaat het systeem uit twee hoofdonderdelen, een systeem voor procesbeheer, en een systeem voor bestandsbeheer en I/O. Het procesbeheerblok is verantwoordelijk voor de geheugenbeheer, scheduling en communicatie tussen processen. In het I/O systeem op de lagere lagen, bevindt zich de machineafhankelijke code, dit maakt het makkelijk om UNIX naar nieuwe machines over te brengen. De recente uitbreidingen bevinden zich vooral in dit niveau.

Multitasking en multithreading: Solaris en Linux.

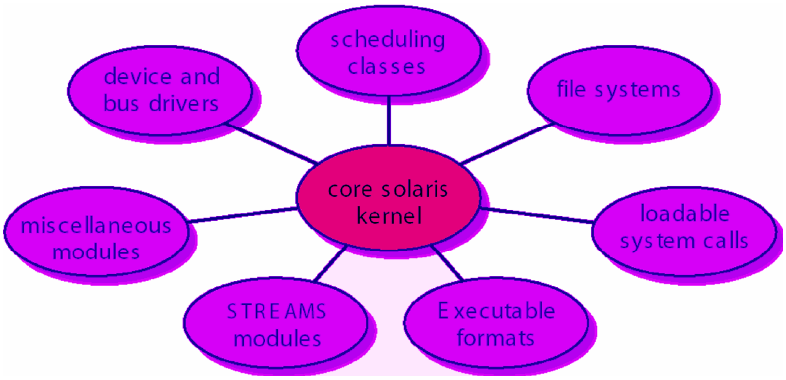
Solaris laat intensief gebruik van threads toe. Het ondersteunt processen, user-level threads en kernel-level threads.

Ook Linux ondersteunt het gebruik van threads.

Ondersteuning van asymmetrische multiprocessing: Linux en Solaris.

Sommige Linux distributies ondersteunen asymmetrische multiprocessing.

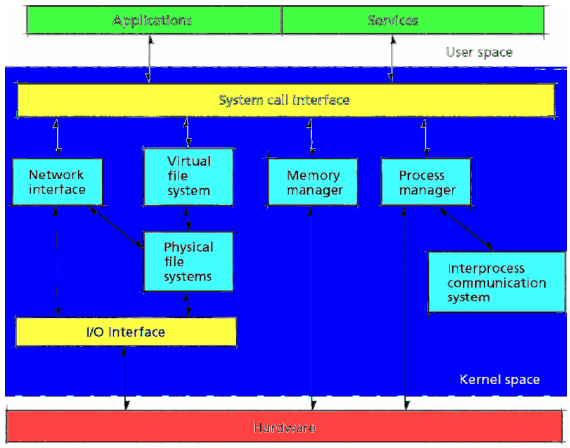
Modulair ontwerp: Linux en Solaris.



Solaris werkt met een kleine kern die zorgt voor de diensten noodzakelijk voor andere modules.

Ook Linux heeft een meer modulaire kernel, ondanks dat het ontwerp lijkt op de monolithische UNIX kernel. Dit zorgt voor een grote performantie.

De kernel kan echter dynamisch kernelmodules laden en linken aan een actieve kernel en weer verwijderen. De meeste modules implementeren een stuurprogramma, bestandssysteem of een netwerkprotocol. Dit laadt toe om Linux systemen met een minimale kernel op te starten. Ook is het niet nodig om de volledige kernel te compileren als nieuwe functionaliteit moet worden toegevoegd.



Multithreading en multitasking: Windows NT.

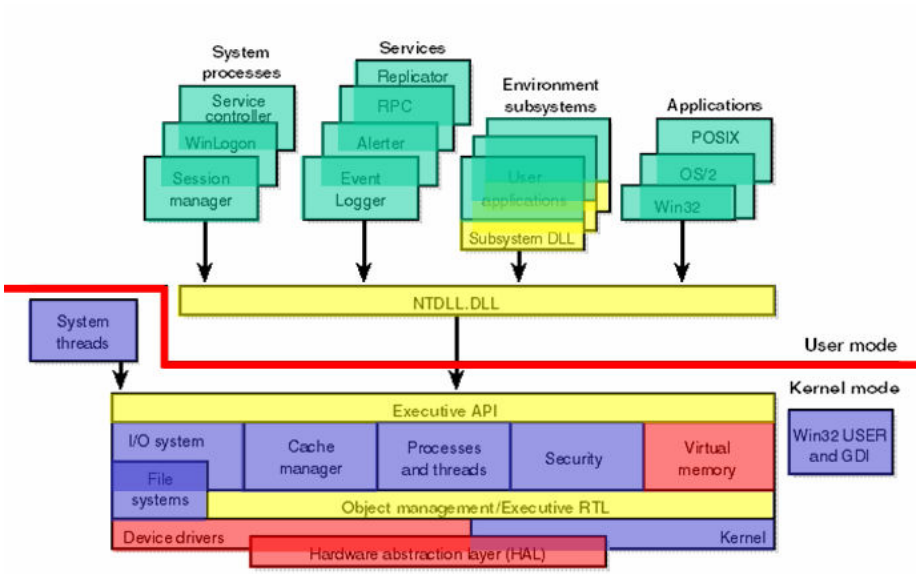
Windows NT verdeelt zijn processen in verschillende types. Sommige subsystemen, zoals Win32 en OS/2 ondersteunen threads, andere, zoals Win16 en POSIX echter niet.

Asymmetrische multiprocessing: Windows NT.

Windows NT ondersteunt asymmetrische multiprocessing.

Modulair ontwerp: Windows NT.

Windows NT bestaat uit een microkernel en enkele extra modules. Elke module heeft een gestandaardiseerde interface, die zowel door de andere NT modules als door gebruikersprocessen kan aangesproken worden. Elke module kan onafhankelijk van andere modules worden vervangen. Een aantal modules worden, in tegenstelling tot een zuiver microkernelarchitectuur, om performantieredenen in de kernelmodus uitgevoerd. Deze modules samen met de microkerne worde de Executive genoemd.



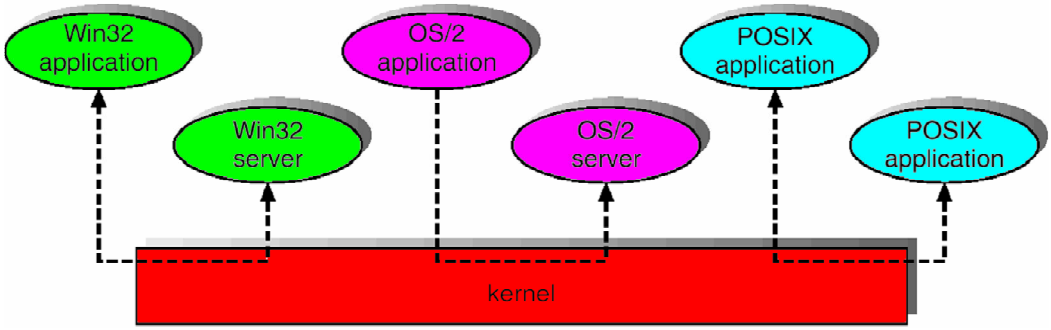
De Hardware Abstraction Layer (HAL) voert een vertaling uit tussen algemene opdrachten en instructie eigen aan de processor. De hogere niveaus zijn eerder afhankelijk van de HAL dan van de specifieke hardware.

De (micro)kernel is de meest essentiële component, hij i altijd in het geheugen geladen en kan nooit door andere componenten onderbroken worden. De microkernel zorgt voor scheduling en synchronisatie, handelt traps en interrupts af en zorgt voor herstel na stroomonderbreking.

De executive diensten zijn de modules die in de kernelmodus worden uitgevoerd.
De I/O-manager is gelaagd opgebouwd.
De Cache Manager beheert de schijfcache, in NT is dat een taak van het besturingssysteem.
De Security Reference Monitor is verantwoordelijk voor het controleren van toegang.
De Object Manager maakt en verwijdert objecten, terwijl de Process Manager proces- en threadobjecten beheert.
De Virtual Memory Manager voert ondermeer vertaling uit tussen virtuele en fysieke geheugenadressen, en bevat dus ook hardwareafhankelijke code.
Ook de Windows-module en de grafische module behoren in de recente versies tot de Executive.

Architectuur van besturingssystemen: Vraag A2.

De omgevingssubsystemen en de afschermingssubsystemen zijn de belangrijkste modules die in gebruikersmodus worden uitgevoerd.
De afschermingssubsystemen leveren beveiligingsfuncties.
De omgevingssubsystemen zorgen voor interactie met een gebruiker, en voor een Application Programming Interface set. NT ondersteunt hierdoor toepassingen die voor verschillende besturingssystemen zijn geschreven.
Met elk programma komt altijd slechts één omgevingssubsysteem overeen. Men kan nooit een systeemaanroep uitvoeren naar een API-routine in een andere omgeving.



Als een toepassing opstart dan wordt het juiste omgevingssubsysteem dynamisch opgestart.

Elk omgevingssubsysteem wordt uitgevoerd als een server proces. De aanvraag naar één van de diensten van de server processen gebeurt door het versturen van berichten, die dor de Executive naar de juiste server worden gestuurd. De server stuurt de antwoorden opnieuw via de Executive terug naar de cliënt. Dit mechanisme wordt Local Procedure Call genoemd.

NT heeft objectgeoriënteerde principes verwerkt in het ontwerp van de microkernel. Objecten worden gebruikt wanneer gegevensstructuren binnen de Executive worden geopend voor toeging in de gebruikersruimte, of wanneer de toegang tot gegevens wordt gedeeld of beveiligd.