Tim Mallezie 2de bach. III

Architectuur van besturingssystemen: Vraag A1.

Procesbeheer: *interrupts en uitvoering van besturingssystemen*.

a) Beschrijf de werking en de bedoeling van interrupts:
   - Hoe deze door de processorhardware en het besturingssysteem worden verwerkt?
   - Waarom ze de efficiëntie van I/O bewerkingen verbeteren? (Geef hierbij eerst een beschrijving van het mechanisme van I/O bewerkingen!)
   - ·Welke benaderingen men kan volgen indien tijdens het verwerken van een interrupt een andere interrupt optreedt?

b) Bespreek drie alternatieven waarop de relatie tussen het besturingssysteem tot processen kan worden benaderd. Bestaat het besturingssysteem zelf uit processen?

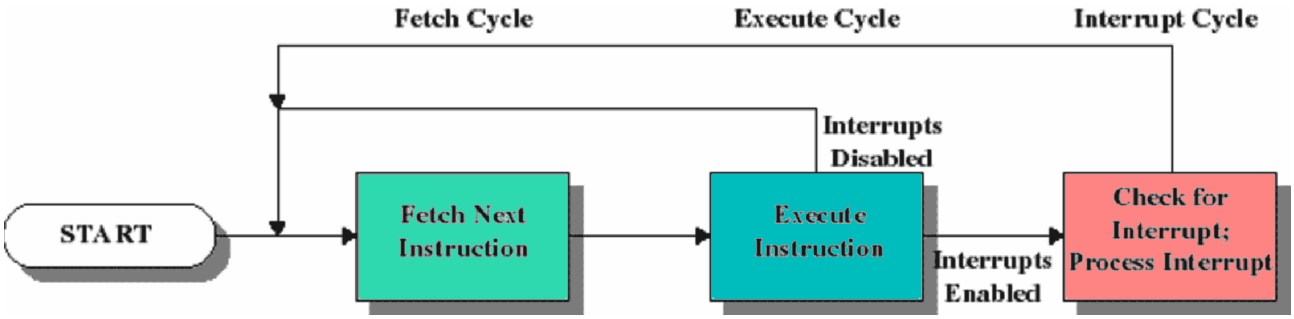
c) Welke alternatieven worden gevolgd in UNIX, Linux en Windows NT?

d) Kan men in deze besturingssystemen gebruik maken van threads? Zo ja, welk implementatiemodel volgen deze threads?

Werking en bedoeling van interrupts:

Een interrupt is een hardwaremechanisme dat een extern apparaat in staat stelt om de CPU een signaal te sturen wanneer de I/O operatie voltooid is.

Een interrupt wordt gegenereerd door een I/O controller (niet door de administrator), door een signaal op de IRQ (Interrupt Request Line) lijn te zetten.



Verwerking door de CPU:
In de interruptcyclus controleren of er interrupts zijn opgetreden door het interruptsignaal te bekijken.
Er zijn geen interrupts:
=> De CPU gaat verder met de opvraagcyclus.
Er zijn wel wachtende interrupts:
=> De CPU onderbreekt huidige programma, slaat informatie op voor het hervatten van het programma. De inhoud van alle registers en het adres van de volgende instructie in het hoofdprogramma, worden op de stack geplaatst. De programmateller wordt bijgewerkt zodat hij naar het begin van de interruptroutine verwijst.

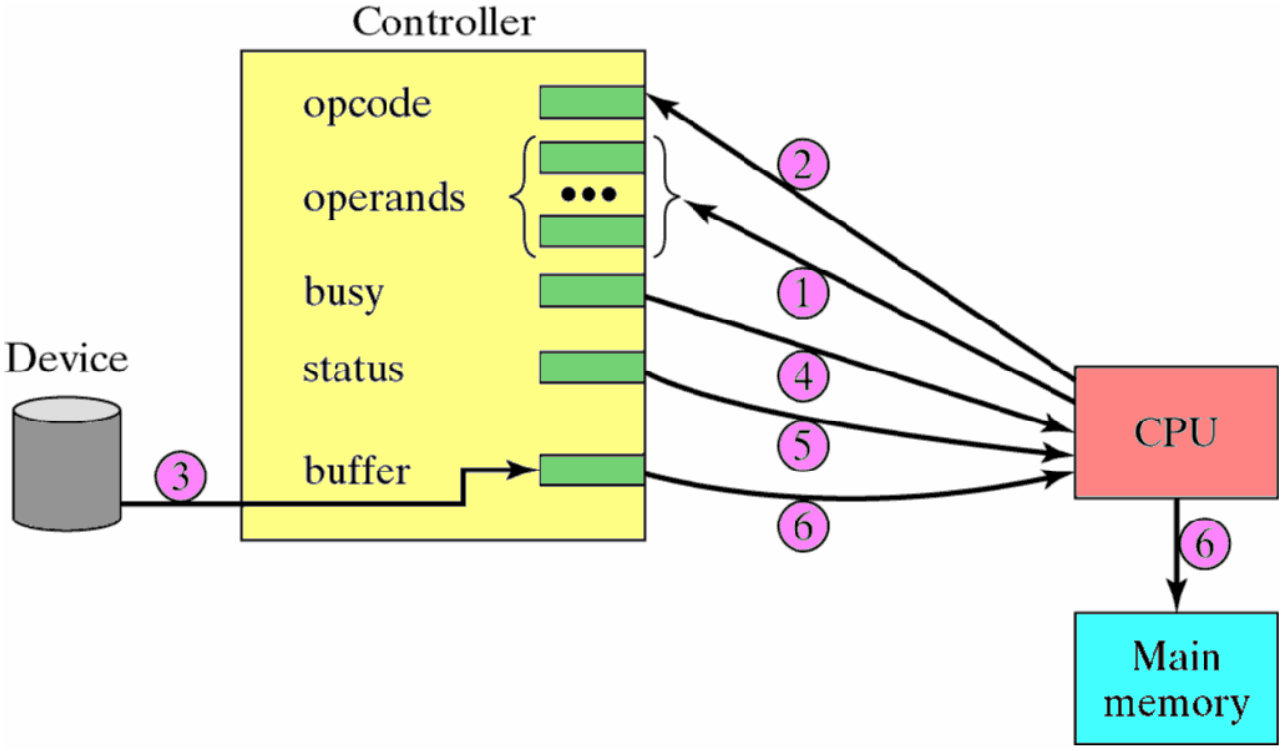
Architectuur van besturingssystemen: Vraag A1.

Verwerking door het besturingssysteem:

De verwerking is afhankelijk van de computerarchitectuur en het besturingssysteem. Soms is er één enkele routine voor de interruptafhandeling, of er is een specifieke routine voor elk type interrupt.
Soms is er één interruptlijn per I/O controller, soms is er één interruptlijn per voor meerdere controllers, deze moeten zich dan kunnen identificeren bij het genereren van een interrupt.

De meeste architecturen maken gebruik van een interruptvector, een tabel van verwijzingen naar gespecialiseerde routines voor specifieke interruptafhandeling. De routine voor de interrut wordt hierbij indirect via de tabel aangeroepen. Zo moeten niet alle apparaten afgescand worden, om te zien welk apparaat de interrupt gegenereerd heeft.

Mechanisme van I/O bewerkingen:



Om een I/O-operatie te starten, laadt de CPU gegevens in de I/O-controller. Hierdoor weet de controller, welke gegevens moeten opgehaald worden. Bij een uitvoer, van hoofdgeheugen naar I/O-apparaat, moeten ook de buffers gevuld worden. (1) en (2).
De controller interageert dan met het apparaat, om de data in of uit de buffer te schrijven. (3).
Bij geprogrammeerde I/O moet het gebruikersprogramma nu een wachtlus uitvoeren, om te zien of de actie is uitgevoerd. (4) en (5).
In elke wachtlus gaan 3 instructiecycli verloren:

- Het apparaatregister lezen.
- Waarde van signaleringsbit bepalen.
- Voorwaardelijke spronginstructie uitvoeren.

Na de wachtcycli kan nu de buffer naar het hoofdgeheugen geschreven worden, of is de buffer vrij voor een nieuwe I/O-operatie. (6).

Deze synchrone benadering houdt de processor telkens op wanneer een I/O-aanvraag wordt verwerkt.

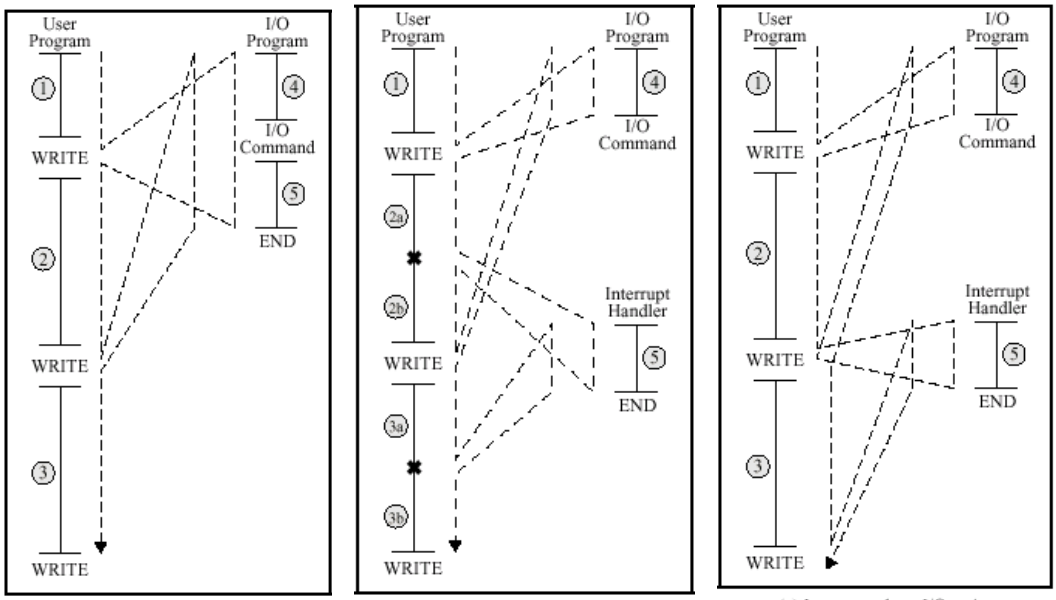
Een I/O-bewerking bestaat uit 3 stappen:

- Een voorbereidende sectie.
- De feitelijke I/O-overdracht.
- Voltooiing van de opdracht.

De feitelijke I/O-overdracht kan enkele duizenden processorcycli in beslag nemen, dit verspilt processorgebruik.

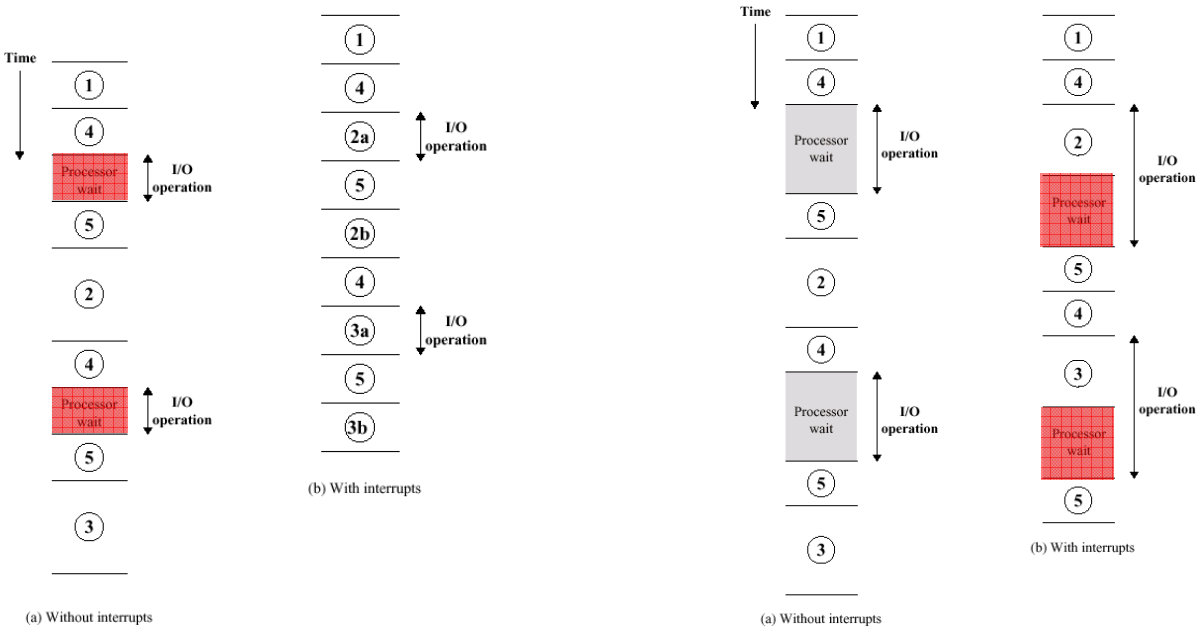
<u>Interrupts en I/O-bewerkingen.</u>

Met interrupts kan de processor andere instructies uitvoeren terwijl een I/O-bewerking bezig is.



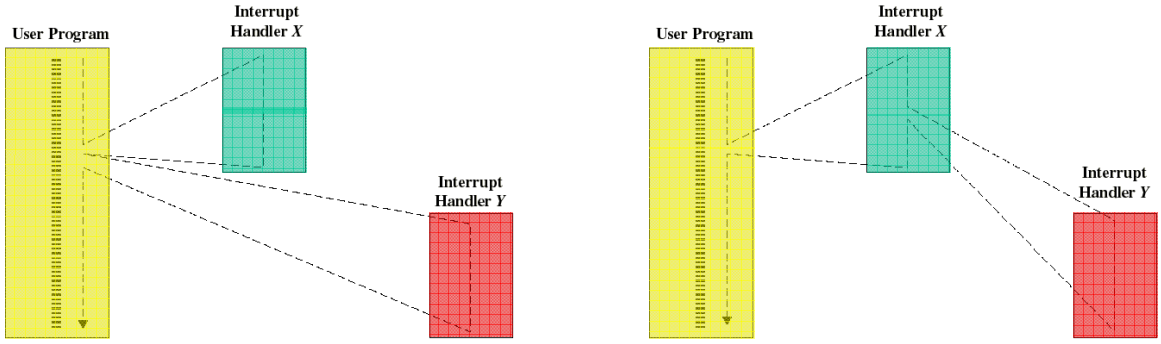
Nadat de I/O-opdracht is gegeven, wordt de besturing teruggegeven aan het hoofdprogramma. De I/O-bewerking wordt gelijktijdig uitgevoerd met het hoofdprogramma, als de bewerking voltooid is wordt een interrupt naar het hoofdprogramma doorgestuurd, die de I/O-opdracht voltooit, en dan doorgaat met het hoofdprogramma.
Hierdoor ontstaat overhead, door het verwerken van de interrupt, maar de processor verspilt geen tijd met wachten op afhandeling van de I/O-bewerking. Toch kan het toch voorvallen dat de processor toch moet wachten, maar dan nog is er verbetering van efficiëntie.

Architectuur van besturingssystemen: Vraag A1.



Een interrupt gedurende een interrupt:

Er zijn twee mogelijke manieren om dit op te lossen. Een eerste benadering is om interrupts te blokkeren, als een interrupt verwerkt wordt. De interrupts worden in sequentiële volgorde behandeld.



In een tweede benadering worden prioriteiten ingevoerd. Routines voor interruptafhandeling kunnen onderbroken worden door interrupts met hogere prioriteit.

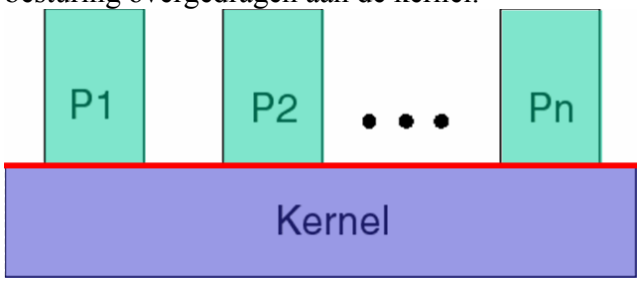
Drie alternatieven waarop de relatie tussen het besturingssysteem tot processen kan worden benaderd. Het besturingssysteem als proces.

Kernel zonder processen:

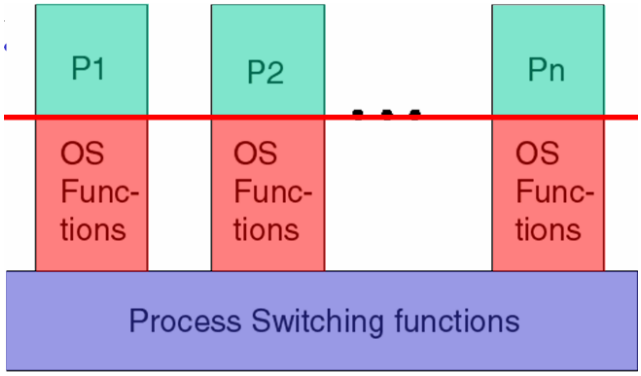
Dit wordt vooral gebruikt in oudere besturingssystemen.
De kernel heeft een eigen geheugengebied en een eigen systeemstack.
Processen worden enkel toegepast op gebruikersprogramma's.
De kernel werkt als een afzonderlijke entiteit, in een geprivilegieerde kernelmodus.

Architectuur van besturingssystemen: Vraag A1.

Bij een interrupt, trap of systeemaanroep wordt het huidige actieve proces opgeslagen en de besturing overgedragen aan de kernel.

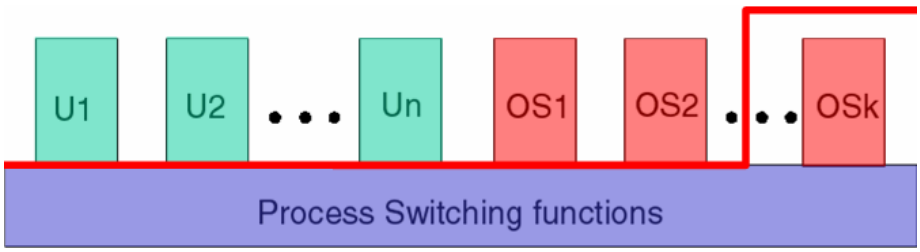


<u>Besturingssysteem is een verzameling van systeem aanroepen.</u>



De software van het besturingssysteem wordt uitgevoerd in de context van een gebruikersproces. De procesbeelden bevatten ook programma-, gegevens- en stackgebieden voor kernelprogramma's.
Bij een interrupt, trap of systeemaanroep wordt een moduswisseling uitgevoerd, en een routine voor het besturingssysteem uitgevoerd. Er dient niet altijd een proceswisseling plaats te vinden. Indien een proceswisseling noodzakelijk is, wordt binnen het proces de context opgeslagen en de besturing overgedragen aan een routine voor proceswisseling buiten alle processen om. Binnen één proces kunnen zowel gebruikersprogramma's als besturingssysteemfuncties uitgevoerd worden, door de verschillende modi, kan de gebruiker de functie van het besturingssysteem niet storen.

<u>Microkernelbenadering.</u>



Besturingssysteemfuncties worden gestructureerd als aparte processen, uitgevoerd in kernelmodus.

In deze benadering zijn veel meer proceswisselingen met de nodige overhead noodzakelijk.
Het besturingssysteem kan wel beschouwd worden als een verzameling verschillende modules, met onderling eenvoudig interfaces.
De processen kunnen met aangepaste prioriteit verweven worden met andere processen.
Sommige processen hoeven niet noodzakelijk in de kernelmodus uitgevoerd worden.
Een deel van de besturingssysteemprocessen kan toegewezen worden aan specifieke processoren.

Windows NT:

Windows NT is gemodelleerd volgens de microkernelarchitectuur. NT bestaat uit een microkernel en een aantal modules. Het verschil met een zuivere microkernelarchitectuur is dat enkele modules in de kernelmodus worden uitgevoerd.

UNIX en Linux:

In UNIX en Linux wordt alle software van het besturingssysteem uitgevoerd in de context van een gebruikersproces. Het besturingssysteem wordt dus beschouwd als een verzameling van systeemaanroepen.

Gebruik van threads in besturingssystemen:

UNIX (klassiek):

De klassieke UNIX versies ondersteunen geen multithreading, elk proces komt met één thread overeen.

Solaris:

Solaris laat in tegenstelling tot de klassieke UNIX varianten, een zeer intensief gebruik van threads toe.

Solaris gebruikt net zoals Windows NT en Linux een combinatie van user-level en kernel-level threads.