Hoofdstuk 3: Processen: Beschrijving en Besturing

- Wat is een proces ?
- Waarom processen ?
- Wat moet het OS ervoor doen ?
- Is het OS zelf een proces ?

Wat is een proces ?

Een **proces** is

- een **programma in uitvoering**
- een **rij instructies (trace) uitgevoerd in een bepaalde context**

proces $\neq$ programma

proces wordt uitgevoerd door CPU; zonder CPU, geen proces

Latijn: processus = voortgang

Van Dale: proces = werking in haar voortgang beschouwd; procedé in uitvoering

Alle multiprogramming OSs gebouwd rond notie proces!

Gebruiker ziet alleen processen en werkt in omgeving opgebouwd uit processen:

- de shell is een process
- een programma dat je uitvoert is een proces
- je www browser is een proces
- . . .

Centrale idee:
processen zijn onafhankelijk en kunnen elkaar niet direct beïnvloeden

elk proces heeft eigen stuk geheugen

Enkele van de 92 Processen Vanochtend op Mijn PC

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  76   0 -   609 -      ?        00:00:04 init
1 S     0     2     1  0  94  19 -     0 -      ?        00:00:00 ksoftirqd/0
5 S     0     3     1  0  65 -10 -     0 -      ?        00:00:01 events/0
1 S     0     4     3  0  66 -10 -     0 -      ?        00:00:00 khelper
1 S     0     5     3  0  75 -10 -     0 -      ?        00:00:00 kacpid
1 S     0    27     3  0  65 -10 -     0 -      ?        00:00:00 kblockd/0
1 S     0    28     1  0  75   0 -     0 -      ?        00:00:00 khubd
1 S     0    39     3  0  75   0 -     0 -      ?        00:00:03 pdflush
...
0 T  4005 32525 32407  0  76   0 -  3370 finish pts/2    00:00:02 xpdf
0 S  4005   325 32407  0  76   0 -  3388 -      pts/2    00:00:00 xpdf
0 R  4005   379 32407  0  76   0 -   675 -      pts/2    00:00:00 ps
```

- **Uniprogramming**
  er bestaat altijd maar een proces tegelijk
  MS-DOS
- **Multiprogramming** (multi-tasking)
  meerdere processen tegelijk
  UNIX, Windows-NT

Waarom multiprogramming ?

- Benut meerdere CPU's op multiprocessor machine
- Het houdt (dure) CPU bezig tijdens I/O
- Het houdt (dure/ongeduldige) gebruiker bezig
- Handig voor organisatie software
  Daarom later ook: meerdere threads in één proces

Meerdere CPU's: echte multiprogramming; **true concurrency**

1 CPU kan maar één proces tegelijk uitvoeren
Multiprogramming toch mogelijk door **interleaving**

Hiervoor is nodig:

- switchen tussen processen
- administratie van contexts van processen
- afschermen van processen van elkaar

OS creëert illusie van meerdere CPU's met meerdere geheugens

OS administratie voor multiprogramming

- Welke processen zijn er ?
- Welk proces draait nu ?
- Wat is de **context** van ieder proces ?
  - de program counter
  - alle registers in de CPU
  - de locatie in geheugen
  - geopende files
  - parent/child processen, etc

Een proces bestaat uit programmacode en bijbehorende data plus een **process control block**. Bevat alle informatie om een lopend proces te onderbreken en later weer te herstarten alsof er niets is gebeurd.

Process Control Block

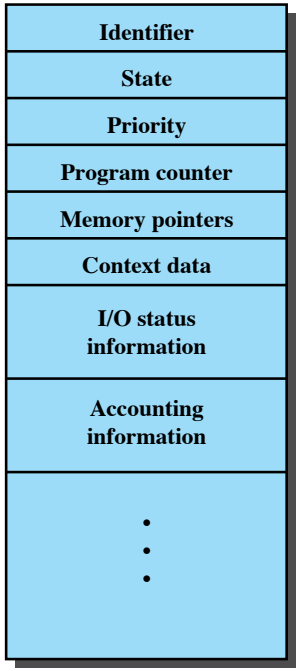


Figure 3.1 Simplified Process Control Block

Trace: Lijst Instructies Uitgevoerd Door Proces

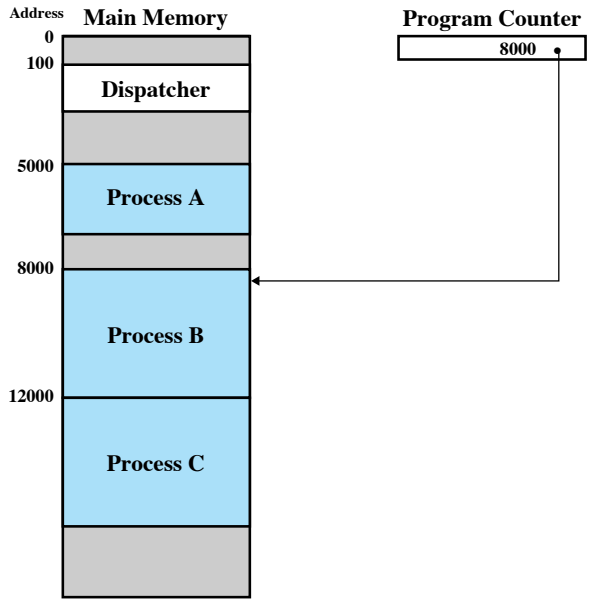
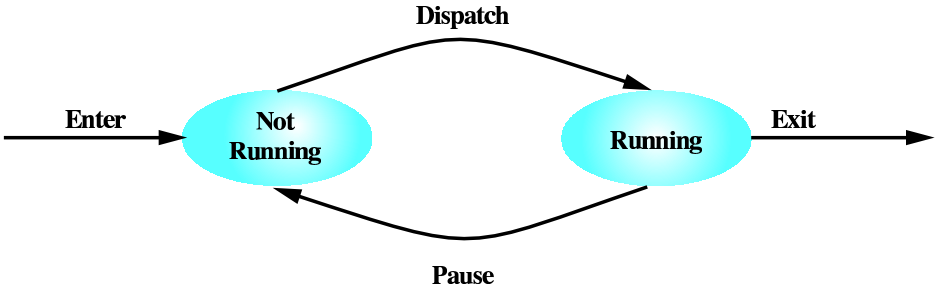
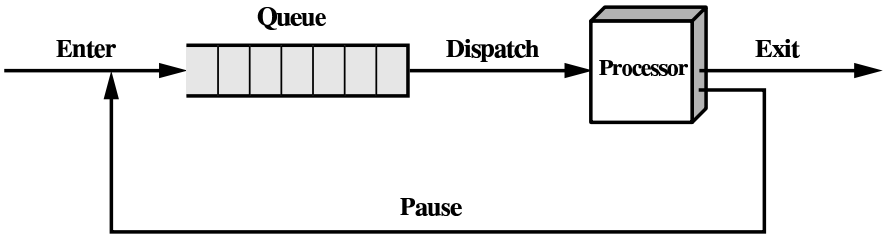


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

Procestoestanden



(a) State transition diagram



(b) Queuing diagram

Figure 3.5 Two-State Process Model

Geblokkeerde Processen

Blocked proces kan niet verder tot een event gebeurt

Proces wacht bijvoorbeeld op:

- I/O operatie
- ander proces
- . . .

Extra state **blocked**. Waarom?

De meeste processen op je PC zijn meestal geblokkeerd!

Procestoestanden (vervolg)

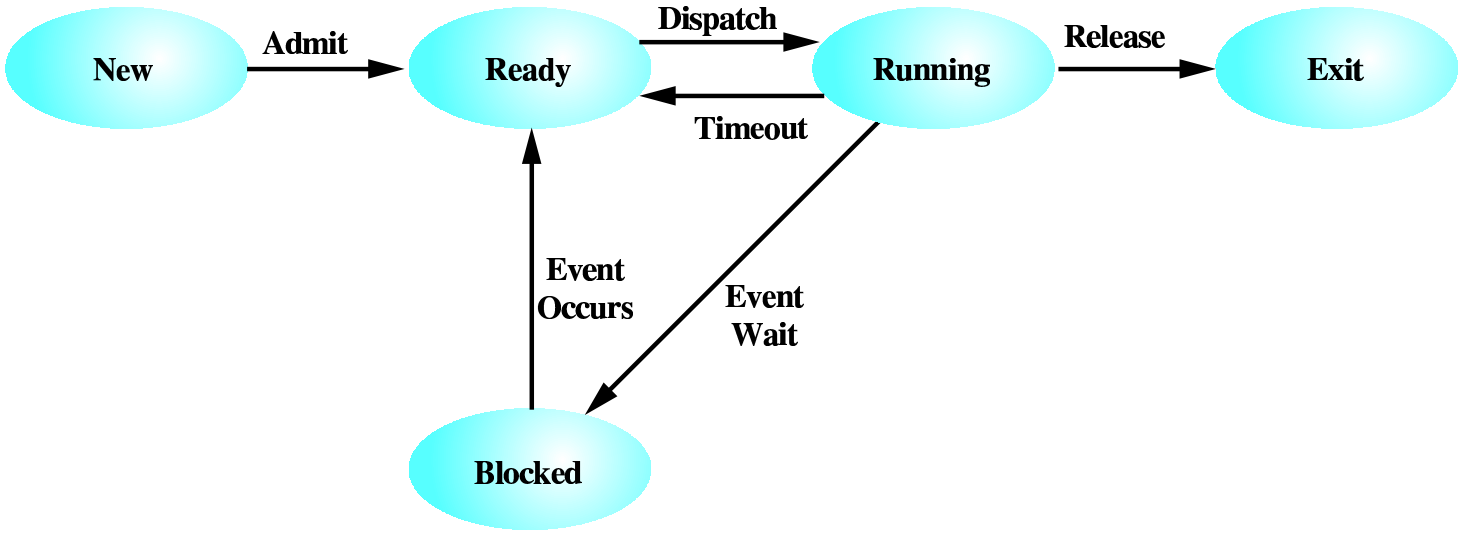


Figure 3.6 Five-State Process Model

Queuing Model for 5 Toestanden

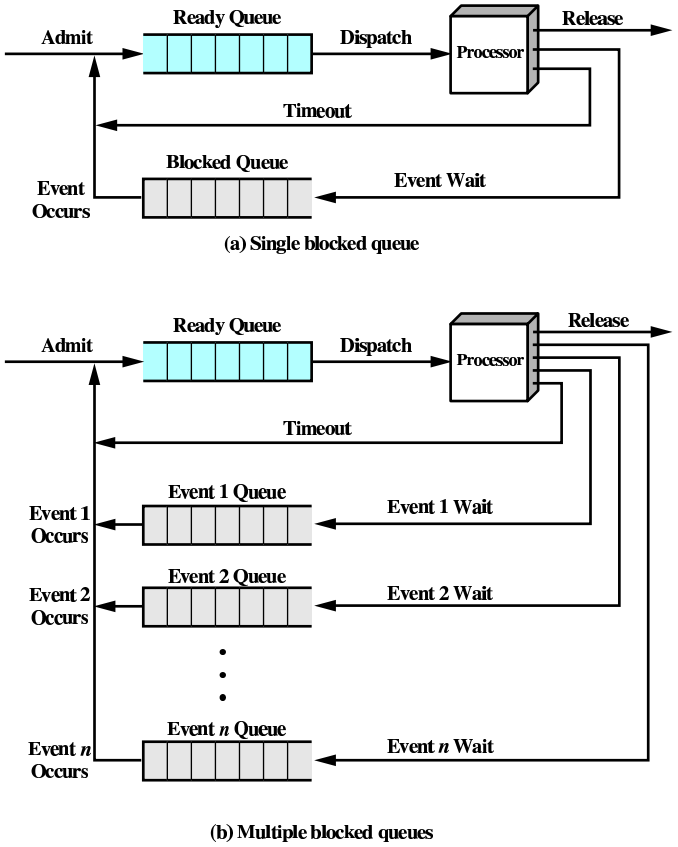


Figure 3.8 Queuing Model for Figure 3.6

Suspended processen

Wat als er niet genoeg geheugen is voor alle processen ?

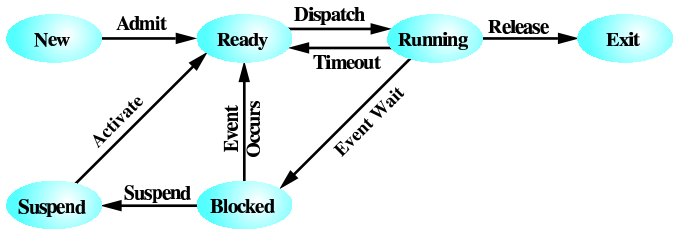
Suspend een proces, dwz **swap** het naar de disk

Extra toestand : **suspended**

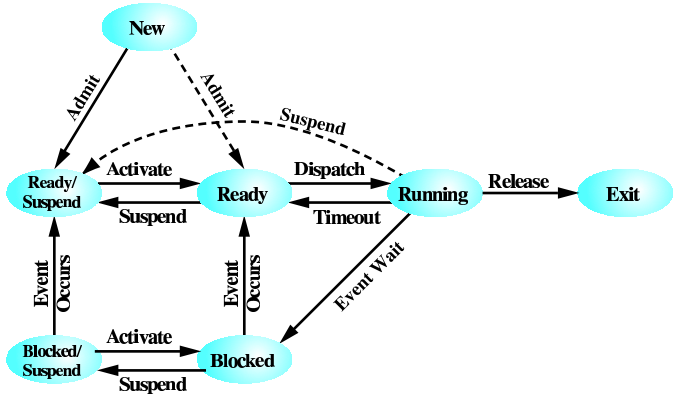
Een suspended programama can ready of blocked zijn

Extra toestanden: **ready suspended** en **blocked suspended**

Toevoegen Suspend Toestanden



(a) With One Suspend State



(b) With Two Suspend States

Figure 3.9 Process State Transition Diagram with Suspend States

OS Administratie voor Processen

Voor elk proces is er een **Process Control Block (PCB)**:

- *Process Identification*:
  uniek proces id number, parent process, user, . . .
- *Process State Information*:
  registers, Process Status Word (PSW), **mode**, stack pointer, . . .
- *Process Control Information*:
  proces state (ready, blocked, . . .), event waar het op wacht, **geheugen**, resource ownership (bv. geopende files), prioriteit . . .

Zie Sectie 3.3.

Modes en Mode Switches

Er zijn **twee executie modes**

- **user** mode
- **kernel** mode (of system/control/supervisor mode)
  met speciale priviliges, namelijk
  - uitvoeren bepaalde instructies (bijv I/O)
  - toegang tot hele geheugen

Aangegeven met bit in de hardware: een bit in het PSW in CPU
Hardware-controle op privileges

Mode switch als een user proces een system call doet of bij interrupt
System call *lijkt* een gewone procedure aanroep

Kernel = dat deel van het OS dat in kernel mode draait

Process switches (context switches)

Proces switch telkens als een ander proces de CPU krijgt

Dit kost tijd!
De hele context (alle CPU registers) van oude process opslaan
de hele context van nieuwe process ophalen

Een process switch duurt langer dan een mode switch

Normale Levensloop van Proces

1. Proces loopt tot de eerstvolgende
   - system call
   - interrupt (clock, I/O, memory fault)
2. Dan een **mode switch**: OS neemt over
   Bewaar deel van PCB
3. Dan *mogelijk* een **process switch**
   Zoja, bewaar hele PCB, zet PCB achterin rij, kies volgende proces uit
4. Terug naar 1

Is OS Zelf een Proces ?

Verschillende mogelijkheden:

- Nee: **Nonprocess kernel**
  OS heeft eigen geheugen & stack, maar executie van kernel code is geen onderdeel van een 'proces'
  Voordeel: simpel. Nadeel: **te** *simpel*
- Nee: **OS-activiteit onderdeel van een user proces**
  Voordeel: weinig context switches, dus efficiënt
- Ja: **OS-activiteit in aparte (high-priority) processen**
  Voordeel: organisatie, gebruik meerdere processoren
  Eventueel ipv system calls: message passing naar een OS proces

Meestal mengeling van laatste twee. Altijd een paar pure OS processen: daemons. Verder veel OS activiteit als onderdeel van user proces.

Central begrippen

- **proces**
- **proces toestand**
- **proces context**
- **proces control block (PCB)**
- **proces switch**
- **kernel vs user mode**
- **mode switch by system call/interrupt**
  system call *lijkt* een gewone procedure aanroep
- **(delen van) OS als losse processen en/of onderdeel van user processen.**