erasmus

HOGESCHOOL BRUSSEL

IT Essentials

Deel III: Operating Systems

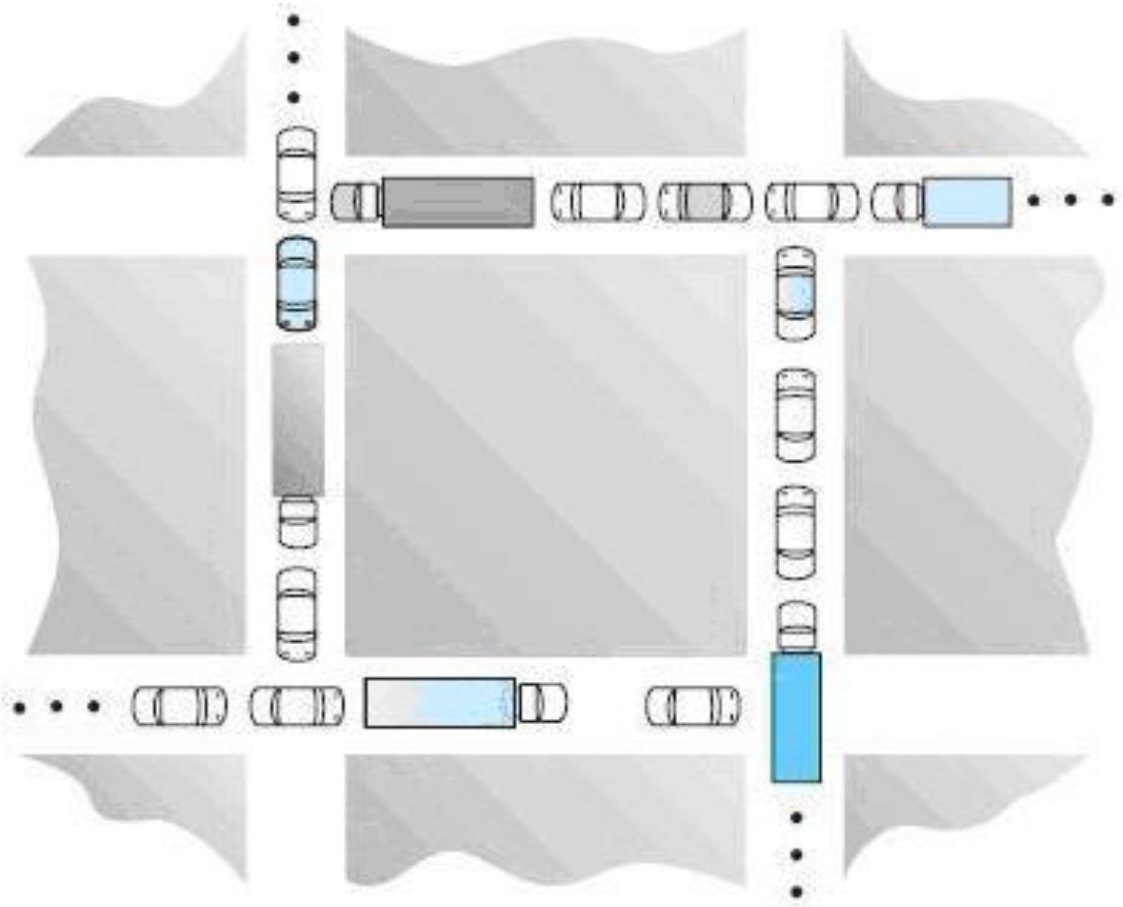
6: Deadlocks

INHOUD

- Algemene introductie
- Bronnen
- Deadlocks introductie
- Deadlock detectie en oplossing
- Deadlocks vermijden
- Deadlocks voorkomen
- Starvation (uithongering)

erasmus
HOGESCHOOL BRUSSEL

ALGEMENE INTRODUCTIE



erasmus
HOGESCHOOL BRUSSEL

ALGEMENE INTRODUCTIE

erasmus
HOGESCHOOL BRUSSEL

ALGEMENE INTRODUCTIE

- Veel processen hebben exclusieve toegang nodig tot 1 of meerdere bronnen
- Voorbeeld: scannen en printen
  - Process A: blokkeert de scanner en vraagt de printer
  - Process B: blokkeert de printer en vraagt de scanner

  = Deadlock
- Ook op netwerk of database

erasmus HOGESCHOOL BRUSSEL

BRONNEN

- Deadlock zowel op HW als SW bronnen
- Bronnen moeten vastgenomen worden, gebruikt en dan terug vrijgelaten

erasmus
HOGESCHOOL BRUSSEL

BRONNEN

- Wegneembare versus niet-wegneembare bronnen
  - (preemptive /non-preemptive)
  - Wegneembaar zonder schadelijke gevolgen: voorbeeld RAM
    - Kan weg- en teruggeswapt worden: page fault is kostelijk maar onschadelijk
  - Niet-wegneembaar: BluRay writer (in het proces van het schrijven)

erasmus
HOGESCHOOL BRUSSEL

BRONNEN

- Wegneembare versus niet-wegneembare bronnen
  - Deadlocks alleen bij niet-wegneembare bronnen
  - Bij wegneembare bronnen: resources verschuiven zodat 1 process zijn taak kan afwerken en alle resources terug vrijmaakt

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS INTRODUCTIE

Een set van processen is in deadlock als elk process aan het wachten is op een gebeurtenis dat enkel één van de andere processen kan veroorzaken

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS INTRODUCTIE

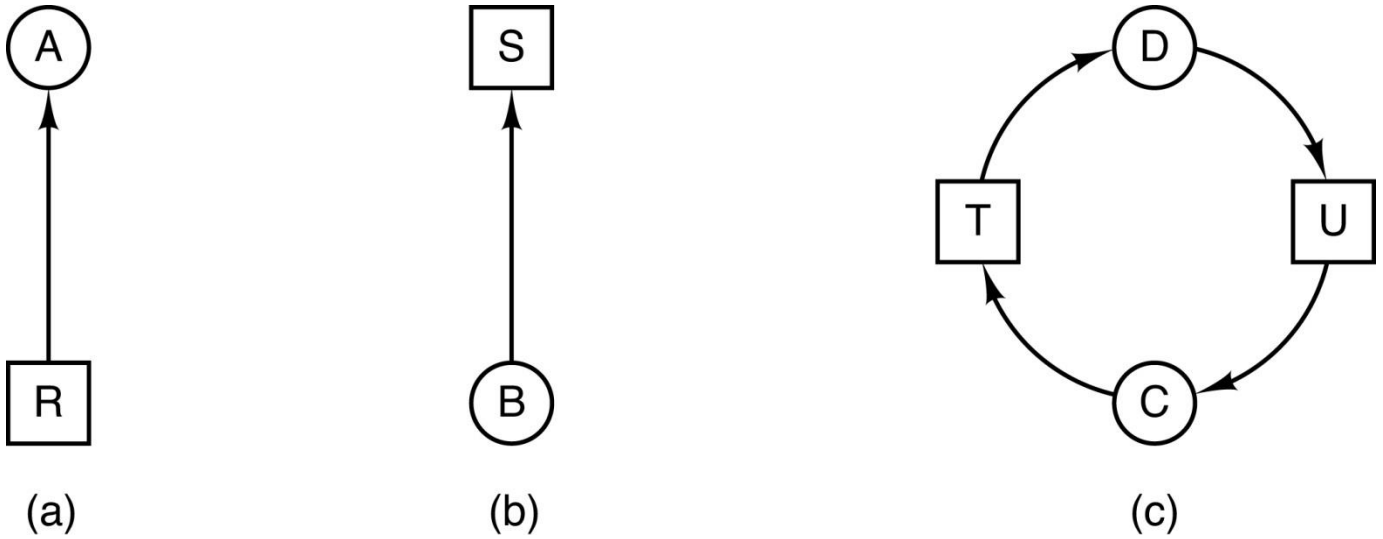
Voorwaarden voor deadlock

1. Wederzijdse uitsluiting van bron (bron is toegekend aan juist 1 process)
2. Vasthouden en wachten (Processen met bronnen mogen nieuwe bronnen aanvragen)
3. Resources zijn niet wegneembaar (non-preemptive)
4. Cirkelvormige wacht

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS INTRODUCTIE

- Deadlock modellen
  - (a): bron R is gelockt door proces A
  - (b): bron S is gevraagd door proces B (in wacht)
  - (c): Cirkelvormige deadlock



A

Request R
Request S
Release R
Release S

(a)

B

Request S
Request T
Release S
Release T

(b)

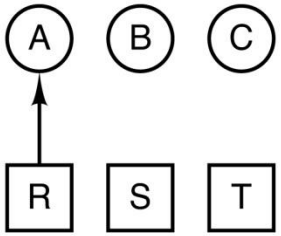
C

Request T
Request R
Release T
Release R

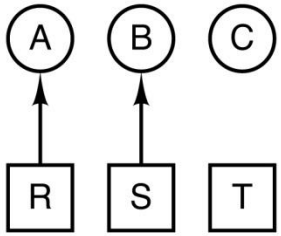
(c)

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
   deadlock

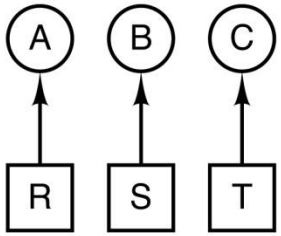
(d)



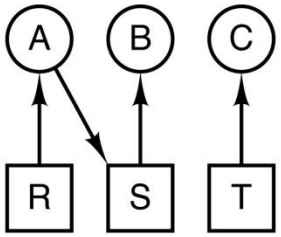
(e)



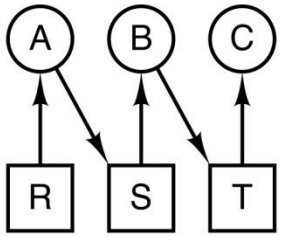
(f)



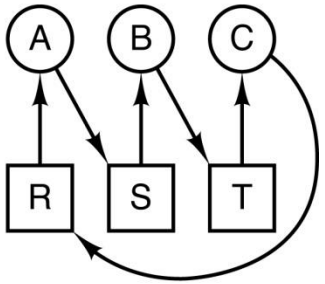
(g)



(h)



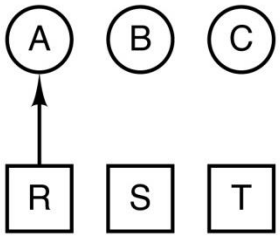
(i)



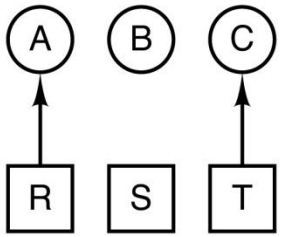
(j)

1. A requests R
2. C requests T
3. A requests S
4. C requests R
5. A releases R
6. A releases S
   no deadlock

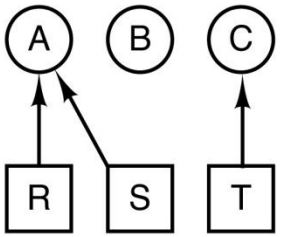
(k)



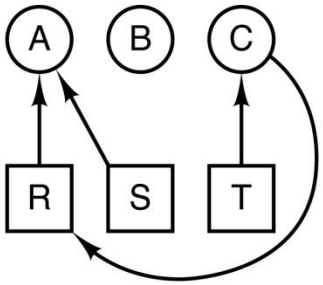
(l)



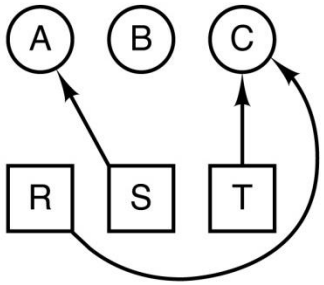
(m)



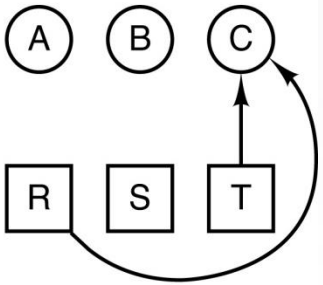
(n)



(o)



(p)



(q)

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS INTRODUCTIE

- 4 deadlock strategieën
  - Struisvogelpolitiek
  - Deadlock detectie en oplossing
  - Deadlocks ontwijken via zorgvuldige bronnenallocatie
  - Deadlock preventie door 1 van de 4 voorwaarden te vermijden

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

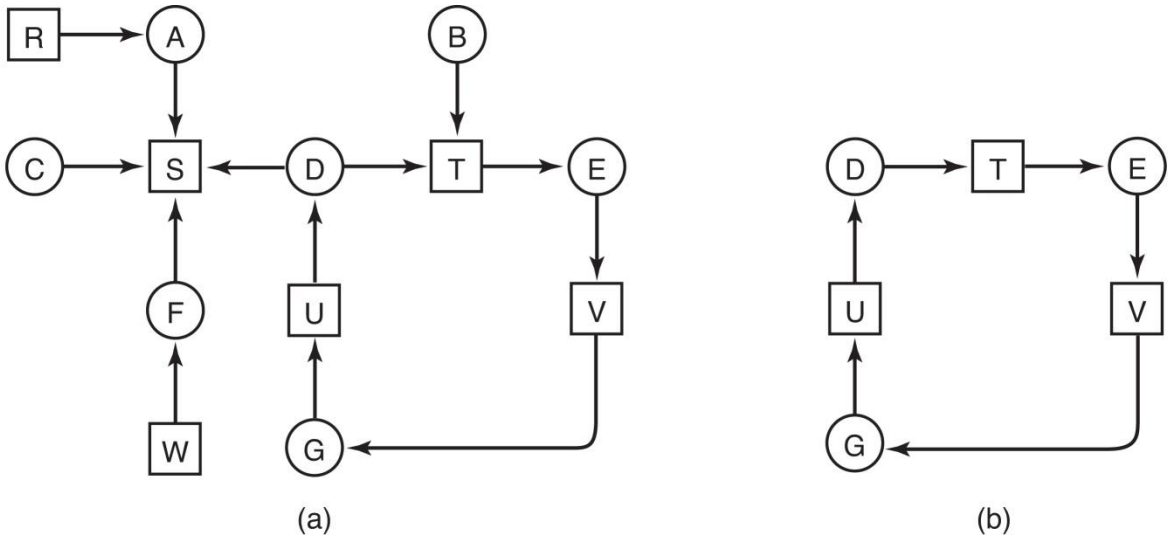
Deadlock detectie

- Met 1 bron per klasse (vb 1 printer)
  - voorbeeld met 7 processen (van A tot G) en 6 bronnen (R tot W)
    - Process A heeft R en wil S
    - Process B heeft niets maar wil T
    - Process C heeft niets maar wil S
    - Process D heeft U en wil S en T
    - Process E heeft T en wil V
    - Process F heeft W en wil S
    - Process G heeft V en wil U

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Is er een deadlock? Makkelijk via modellen (grafiek)



DEADLOCK DETECTIE EN OPLOSSING

Via een algoritme:

1. vertrekken vanuit elke node.
2. alle pijlen volgen tot er niet meer verder kan gegaan worden. Eventueel teruggaan tot eerder knooppunt en terug pijlen volgen.
3. Oplijsten van alle nodes
4. Kijken of er een node 2 keer voorkomt: DEADLOCK

Andere (betere) algoritmes bestaan

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock detectie

- Met meerdere bronnen per klasse
- Nodig:
  - Aanwezige bron vector
  - (nog) beschikbare bron vector
  - Huidige toegekende bronnen matrix
  - Gevraagde bronnen matrix

erasmus
HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock detectie

- Met meerdere bronnen per klasse
- werkwijze:
  1. process aflopen dat zou kunnen verder lopen tot eind door de 2 matrixen te vergelijken (kleiner dan of gelijk aan): markeren
  2. 1 gevonden: bronnen vrijgeven en volgend process zoeken
  3. Indien niet gemarkeerde processen overblijven: deadlock

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock detectie

- Met meerdere bronnen per klasse
- Voorbeeld:

	Tape drives	Plotters	Scanners	Blu-rays
E =	4	2	3	1

	Tape drives	Plotters	Scanners	Blu-rays
A =	2	1	0	0

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock detectie

- Wanneer detectie uitvoeren?
  - Bij elk bronverzoek:
    - CPU intensief
  - Om de zoveel minuten
    - Mogelijks tijdsverlies
  - Bij laag CPU verbruik
    - Deadlocks kunnen dit veroorzaken

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock oplossing

- Wegnemen van de bronnen
  - Zeer moeilijk uitvoerbaar en sterk afhankelijk van de bron
  - Vereist meestal interventie van de gebruiker
  - Voorbeeld: printer: stapel reeds afgeprinte pagina's apart leggen, andere batch printen, en oorspronkelijke stapel terugleggen

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock oplossing

- Rollback
  - Processen moeten checkpoints maken van status memory en resource gebruik
  - Bij deadlock: processen terugrollen tot eerdere checkpoint waarbij de resource nog niet in gebruik was
  - Processen terug opstarten
  - Output is verloren
  - Kans op opnieuw deadlock, maar weinig waarschijnlijk omdat volgorde niet vastligt en dus mogelijks anders zal verlopen

erasmus HOGESCHOOL BRUSSEL

DEADLOCK DETECTIE EN OPLOSSING

Deadlock oplossing

- Kill process
  - Een process kiezen in de deadlock en killen
  - Hopelijk komt de deadlock vrij, zo niet, nog een process killen
  - Simpele maar zeer brute methode

- Bij alle oplossingen steeds het detectie-algoritme terug laten lopen

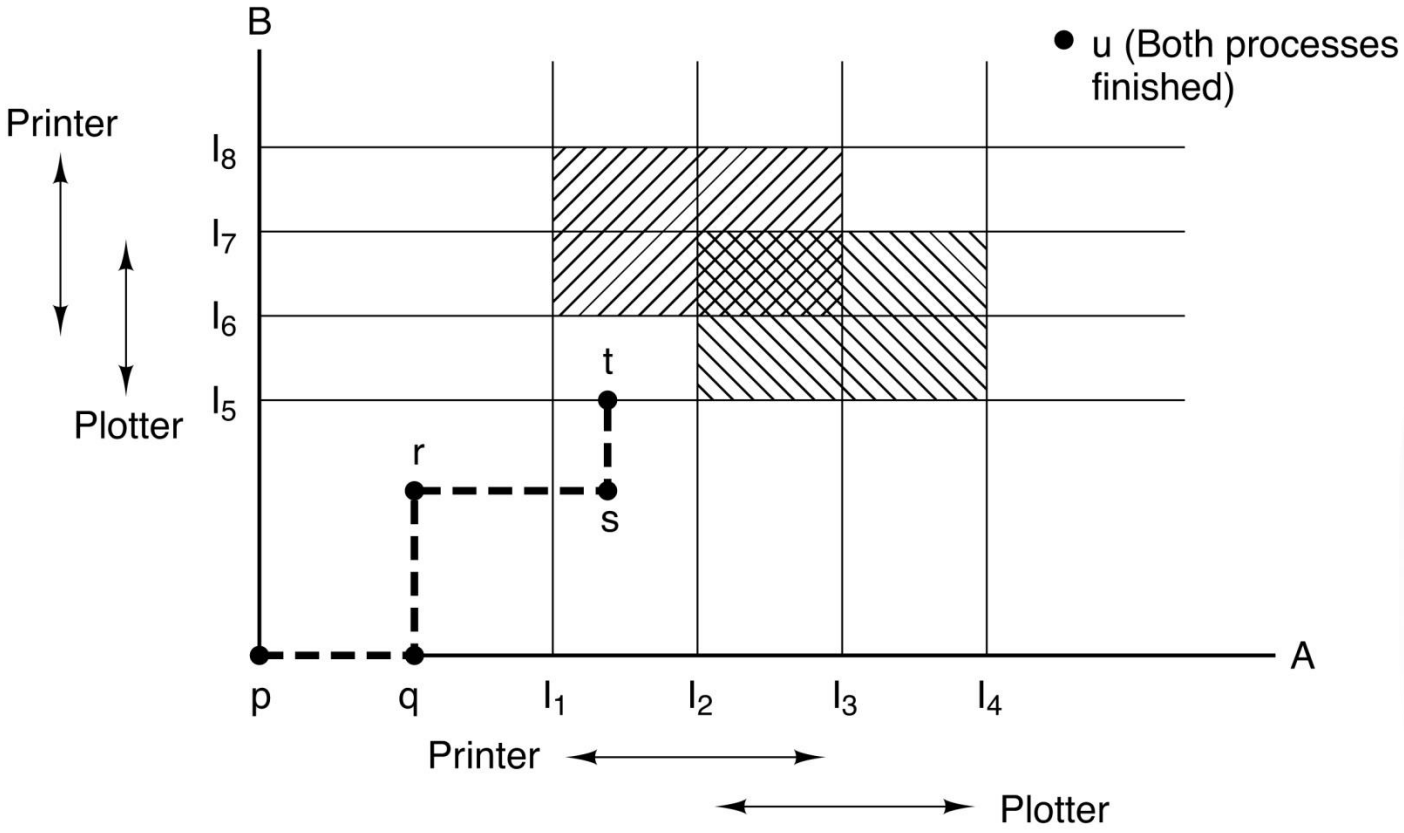
erasmus HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN

- Evaluatie maken of het toekennen van een bron veilig is
- Alle benodigde resources bij een proces moeten op voorhand gekend zijn (!)

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN



DEADLOCKS ONTWIJKEN

- Werken met veilige en onveilige status ((un)safe state)
- Voorbeeld safe state (a) met 1 bron klasse

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1
(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5
(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0
(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7
(e)

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN

- Voorbeeld unsafe state (a) met 1 bron klasse

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2
(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0
(c)

	Has	Max
A	4	9
B	Đ	Đ
C	2	7

Free: 4
(d)

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN

- Let wel: unsafe wil niet zeggen deadlock maar wel geen garantie op ontwijken van deadlock
  - Mogelijks zelfs geen deadlock: A zou vroegtijdig een resource kunnen vrijgeven voor het zelf helemaal klaar is

erasmus HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN

- Bankiersalgoritme van Dijkstra
  - Enkel leningen toekennen met de zekerheid dat ze kunnen terugbetaald worden (safe state)
  - Gebaseerd op deadlock detectie

DEADLOCKS ONTWIJKEN

- Bankiersalgoritme van Dijkstra
  - Met enkele bron

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

(c)

erasmus HOGESCHOOL BRUSSEL

- Met meerdere bronnen

Process	Tape drives	Plotters	Printers	Blu-rays
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Printers	Blu-rays
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still assigned

E = (6342)
P = (5322)
A = (1020)

1. Zoek naar een process waarvan elke waarde in gevraagde bronnen ≤A

   Geen? Deadlock
2. Indien wel (D), markeer process als afgelopen en geef alle resources vrij (tel bij A)
3. Herhaal 1 en 2 tot alle processen zijn beëindigd (safe) of 1 of meerdere niet voorbij 2 geraken (unsafe)

DEADLOCKS ONTWIJKEN

- Bankiersalgoritme van Dijkstra
  - Met meerdere bronnen

    B vraagt extra printer: safe (D kan nog verder) dus ok

    E vraagt ook nog een printer: niet safe want mogelijke deadlock: E krijgt de printer niet

    => Enkel resource toekennen als de status safe blijft

erasmus HOGESCHOOL BRUSSEL

DEADLOCKS ONTWIJKEN

- Niet altijd mogelijk om op voorhand alle nodige resources te weten voor een process => deadlocks ontwijken (preventie)

$\Rightarrow$ Aanval op 1 van de 4 deadlock voorwaarden

$\Rightarrow$ Deadlocks voorkomen

erasmus
HOGESCHOOL BRUSSEL

DEADLOCKS VOORKOMEN

- Aanval op wederzijdse uitsluiting van bron voorwaarde
  - Zeer moeilijk toe te passen
  - 2 processen tegelijk laten printen is niet mogelijk
    - Werken met spooler: slechts 1 process (printer daemon) heeft toegang tot de resource
    - Spooling best volledig afwerken anders mogelijks lang wachten op process voor 2de deel van document
    - 2 processen vullen spooling space volledig zonder klaar te geraken: deadlock op spooling space

erasmus HOGESCHOOL BRUSSEL

DEADLOCKS VOORKOMEN

- Aanval op vasthouden en wachten voorwaarde
  - Processen verplichten van alle benodigde bronnen voor executie aan te vragen
    - Processen weten niet op voorhand welke bronnen nodig zullen zijn (zie ook Bankiersalgoritme)
    - Inefficiënt: mogelijk bronnen aanvragen die pas veel later nodig zijn: bronnen worden de hele tijd vastgehouden

erasmus HOGESCHOOL BRUSSEL

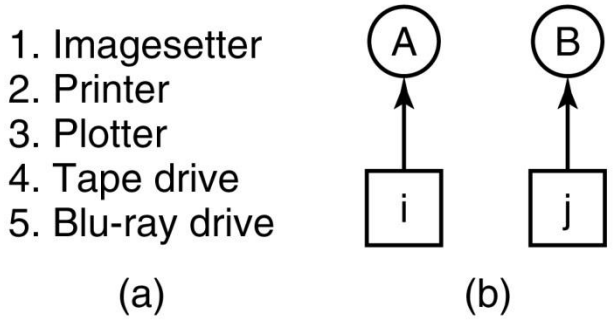
DEADLOCKS VOORKOMEN

- Aanval op bronnen zijn niet wegneembaar (preemption) voorwaarde
  - Moeilijk toe te passen: printer in het midden van een taak onderbreken
  - Werken met een printer daemon (virtualisatie) mèt voldoende disk space
  - Niet altijd mogelijk: records in een database moeten gelockt worden (non preemption is noodzakelijk)

erasmus HOGESCHOOL BRUSSEL

DEADLOCKS VOORKOMEN

- Aanval op cirkelvormige wacht voorwaarde
  - Rangorde maken van bronnen: enkel hogere bronnen mogen aangevraagd worden



  - A mag j aanvragen maar B mag i niet aanvragen: deadlock niet mogelijk

STARVATION

In een dynamisch systeem is een constante vraag naar bronnen

Algoritme vereist voor het toekennen van de bronnen (vergelijkbaar met process scheduling algoritme)

Voorbeeld: drukke printer met shortest job first

- Grote file blijft oneindig wachten door steeds andere kleinere opdrachten

= **Starvation**

Op te lossen door ander algoritme te gebruiken zoals First come, first served

erasmus
HOGESCHOOL BRUSSEL