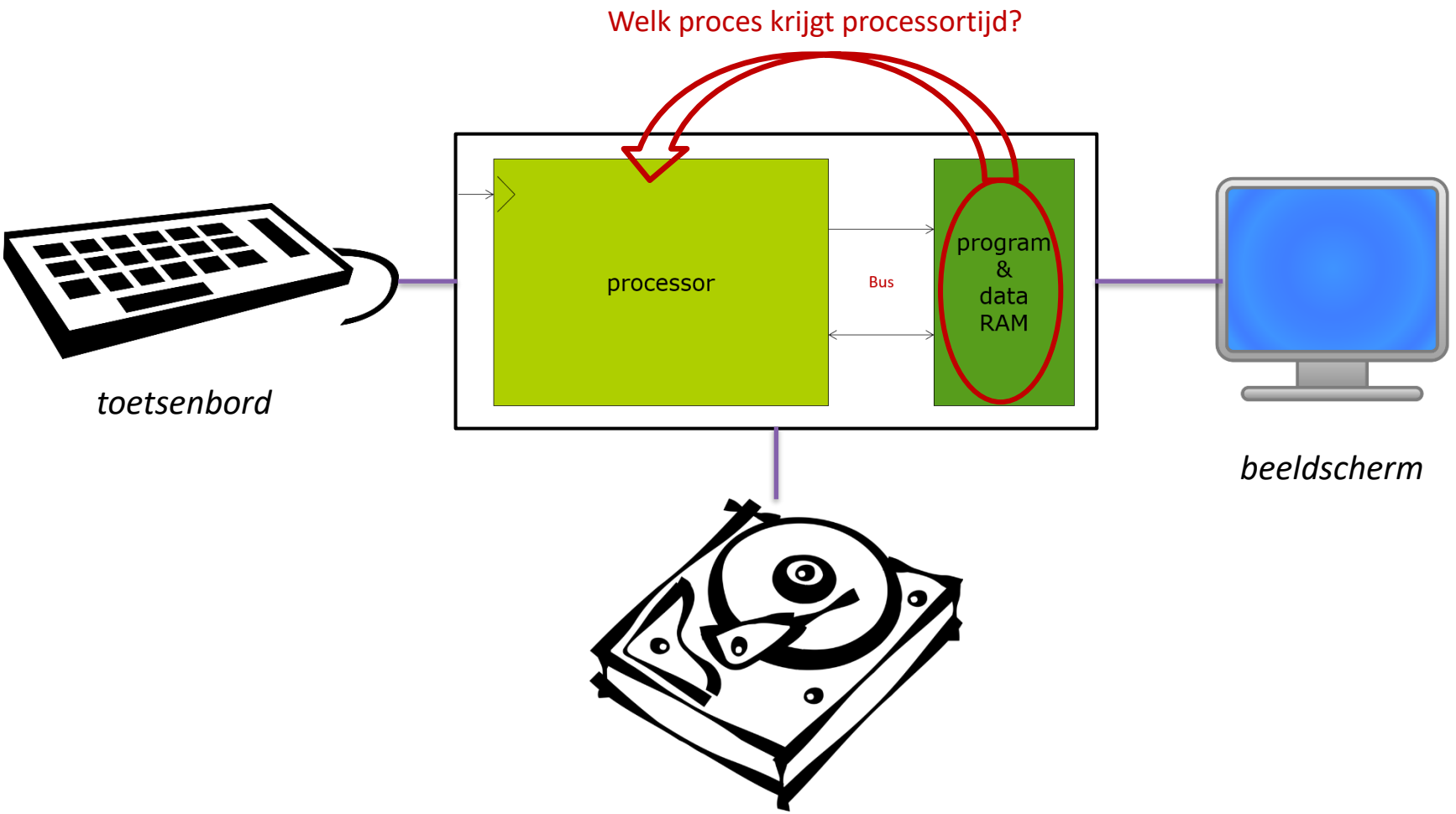
Computersystems 2 Theorie

6. Procesbeheer

KdG Karel de Grote Hogeschool

25/11/2024

Procesbeheer

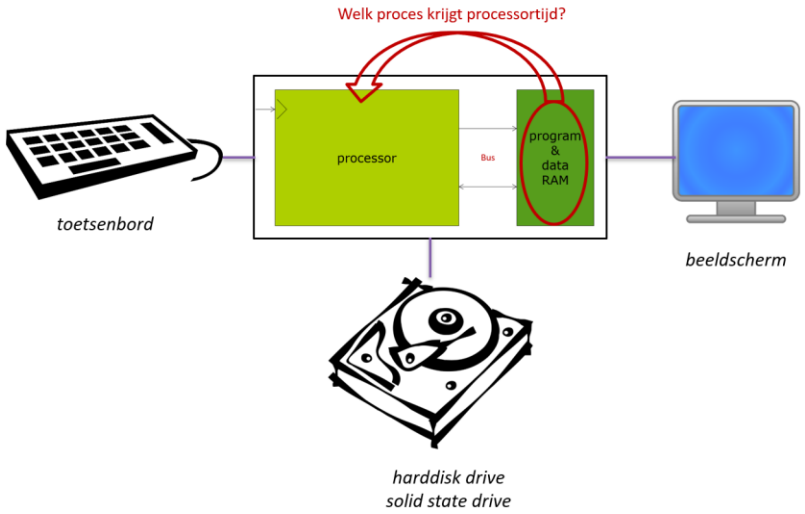


Inhoud

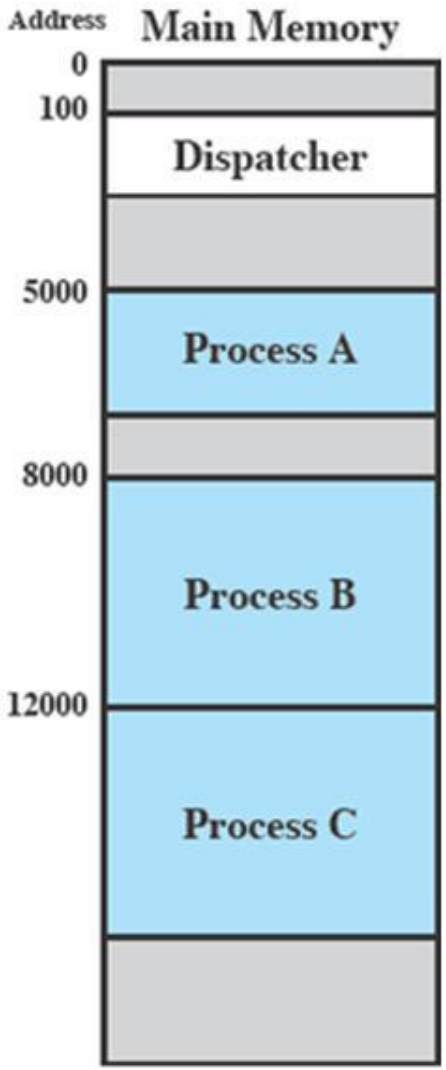
- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Procesbeheer

- Multiprocessing
  - Computer runt verschillende processen “tergelijkertijd”
- Multi-user
  - Verschillende gebruikers werken “tergelijkertijd” op de computer



Sporen (traces)



- 3 processen in het geheugen
- dispatcher is deel van het OS
- dispatcher heet soms 'scheduler'

Traces gezien vanuit de processen

5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011
(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Traces gezien vanuit de processor

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Timeout
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O Request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
----------	Timeout
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Timeout
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Timeout

100 = Starting address of dispatcher program

Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Scheduling in de praktijk

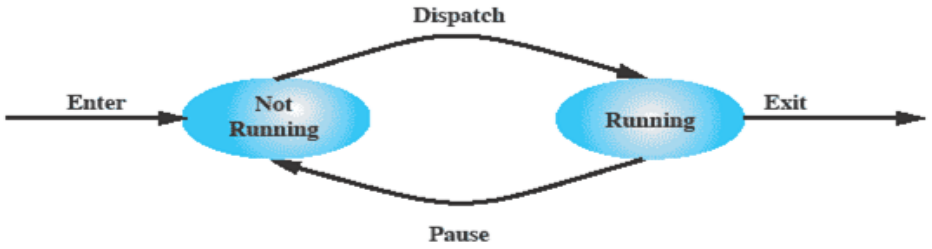
- in Linux en Mac OS-X
  - xload
  - top
  - ps
  - pstree
- in Windows
  - Taakbeheer
  - PowerShell: Get-Process

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Procestoestanden

- Een proces bestaat uit:
  - code
  - data, stack
  - toestand (context)



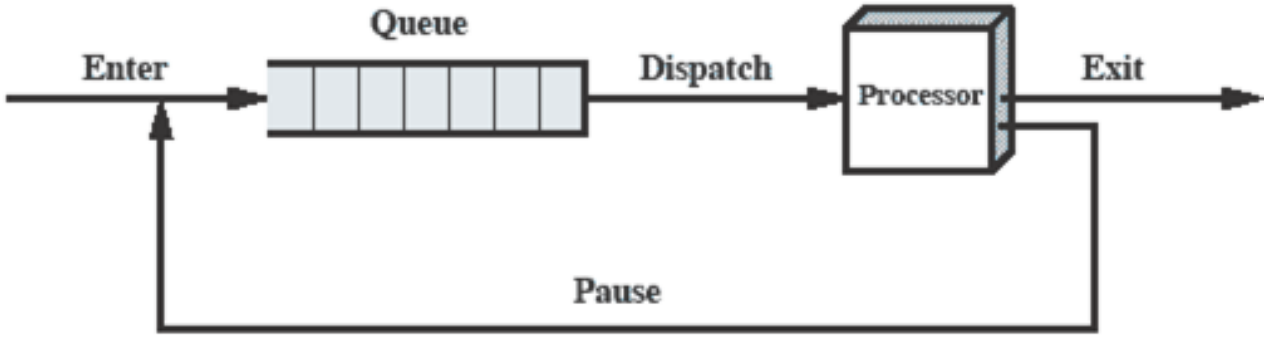
- Een proces moet minstens 2 toestanden hebben:
  - running
  - not-running
- OS moet de toestand van een proces (context) bewaren en omwisselen voor volgend proces (=context switch)

Procestoestanden

- context wordt opgeslagen in PCB
- Process Control Block = datastructuur
  - welk was de laatst uitgevoerde instructie?
  - wat waren de waarden van de registers?
  - hoe was het geheugen geconfigureerd (paging, segmenting, virtual memory)
  - wat is de id van dit proces?
  - welke resources gebruikt dit proces (open files, netwerkverbindingen, ...)?
  - heeft dit process kind-processen?
  - starttijd, processortijd, ...
  - ...

Procestoestanden

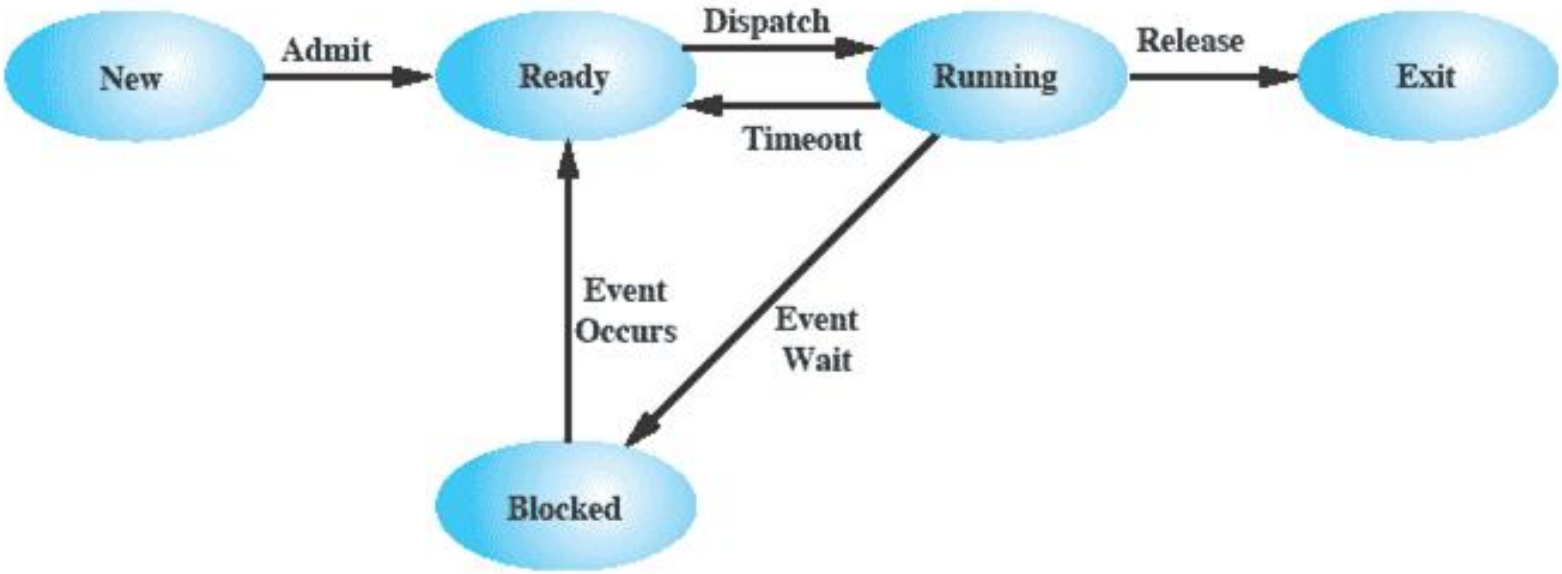
- OS heeft een queue van PCB's



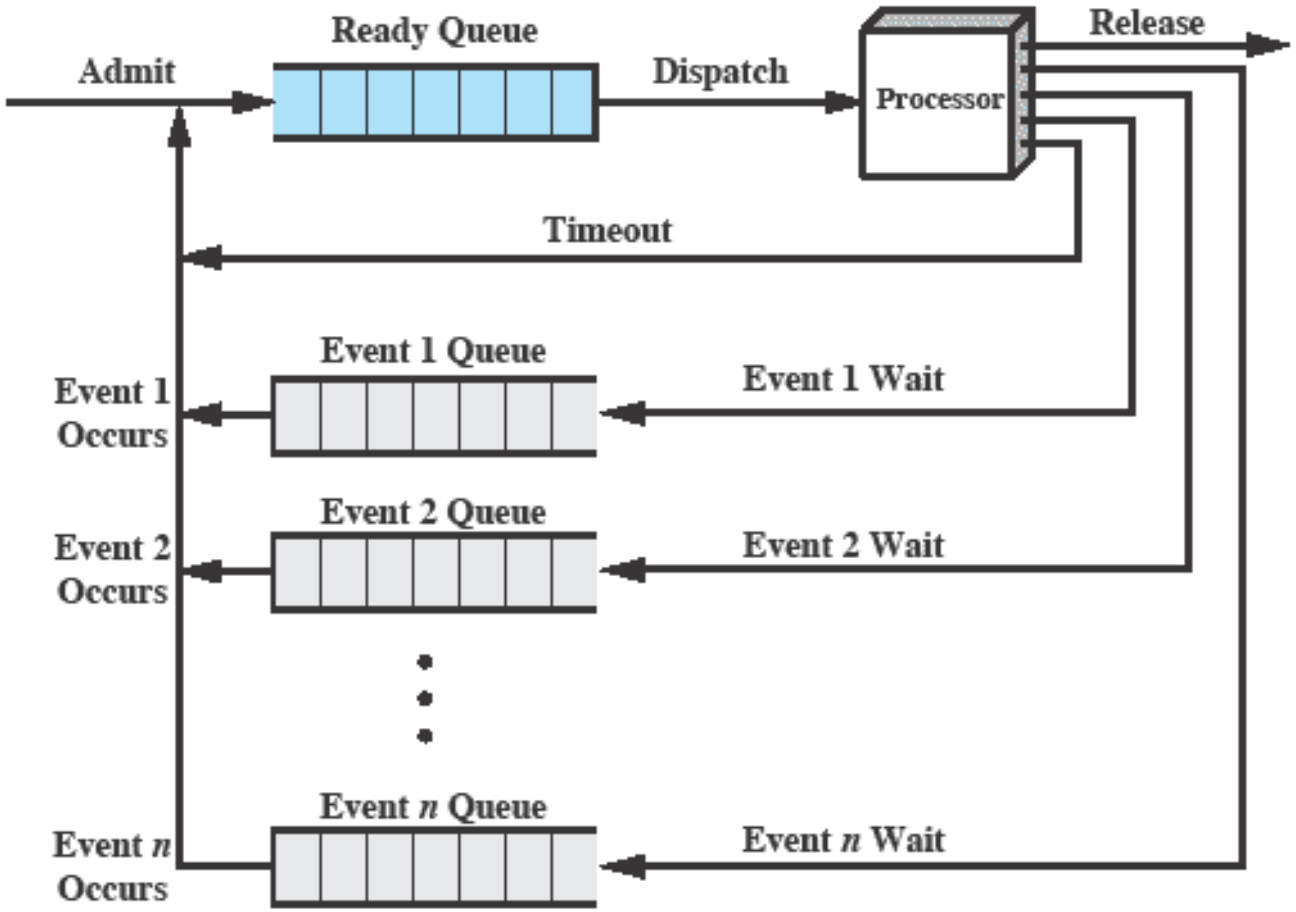
Procestoestanden

- 2 toestanden is niet genoeg: geen verschil tussen wachten op i/o of gepauzeerd
- dus: 3 toestanden
  - ready
  - running
  - blocked
- veel OS-en voegen er nog 2 toestanden bij:
  - new
  - exit

Procestoestanden



Queues in het OS



Linux procestoestanden

1. Met welk commando kan je de processen en hun procestoestand zien?

2. Welke toestanden bestaan?

3. Wat is idle time? Met welk commando kan je deze zien?

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Nieuw proces starten in Linux

Opstarten nieuw programma: 2 system calls:

1. int fork()
   - huidige proces wordt gekopieerd
     - code segment
     - data en stack segment
     - PCB
   - huidige proces kan verder lopen
   - nieuw 'kind' proces wordt opgestart in gekopieerde versie
   - kindproces krijgt 0 als return-value
   - ouderproces krijgt process-id als return value
2. exec("executable")
   - vervangt code segment met nieuwe code nieuw programma
   - reset stack en data segment

Nieuw proces in bash

bash_$ xload

1. bash doet *fork* om child proces aan te maken
2. daarna *exec* om in het child proces de xload code te laden

Nieuw proces starten in Linux

```
#include <stdio.h>
#include <unistd.h>

void doe_child(int i) {
 printf("start van proces %d\n",i);
 int t;
 for(t=0; t<5; t++) {
   sleep(1);
   printf("proces %d: tel=%d\n",i,t);
 }
}
```

```
int main() {
 int i;
 for(i=0; i<10; i++) {
   int f = fork();
   if (f==0) {
     doe_child(i);
     return 0;
   }
 }
 sleep(3);
 execl("/bin/ps","ps", "-f",
   NULL);
 printf("Niet uitgeprint!");
}
```

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Context switch

- non pre-emptive
  - OS onderbreekt de applicatie niet
  - Applicatie runt tot return of tot deze moet wachten (bv. op I/O)
  - Voordeel: heel eenvoudig
  - Nadeel: ?
  - Voorbeelden: Windows 3.11, Mac OS9, Oberon, ...
- pre-emptive
  - OS onderbreekt proces om een ander proces Running te maken, bv. na time slice
  - Voorbeelden: huidige Windows en Linux

Pre-emptive context switch

- veroorzaakt door interrupt
- ga naar 'kernel mode'
- save registers en config (seg/page table) in PCB
- zet PCB in juiste queue
- zoek nieuw proces om te starten
- bijwerken PCB nieuw proces (toestand Running)
- laad registers uit nieuwe PCB
- zet timer voor volgende context switch
- ga naar 'user mode'
- jmp juiste adres

Online and batch job scheduling

- Online scheduling
  - Jobs runnen in foreground
  - Interactie met user
- Batch scheduling
  - Jobs runnen in background
  - Geen interactie met user
  - Gebruikt voor jobs die lange run time hebben
    - Material Requirement Planning voor een fabriek
    - Trainen van Deep Learning modellen

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Scheduling strategieën

- bij context-switch: "kies nieuw proces"...
- verschillende strategieën mogelijk
- doel:
  - eerlijk verdelen van processortijd over processen
  - "uithongering" beletten
  - weinig overhead veroorzaken
  - prioriteiten van processen respecteren

Scheduling strategieën

- ieder proces heeft volgende waarden (in PCB):
  - **w** = tijd doorgebracht in het systeem voor wachten (tot nu toe)
  - **e** = tijd besteed aan uitvoering (tot nu toe)
  - **s** = totale uitvoeringstijd die nodig is om het proces af te ronden (inclusief e) (=schatting!)

First come first served (FCFS)

- kies proces met maximale w
- dit is het 'oudste proces'
- non pre-emptive: wacht gewoon tot i/o request of einde
- voordeel: lange processen worden sneller doorlopen (in vgl. met 2 volgende algoritmen)
- nadeel: korte processen moeten soms lang wachten
- niet voor online scheduling

Shortest Process Next (SPN)

- kies het proces met de kleinste s
- non pre-emptive
- voordeel: snellere response-tijd
- nadelen:
  - je moet weten hoe lang een proces nodig heeft
  - mogelijkheid tot starvation (uithongering)
- niet voor online scheduling

Shortest remaining Time (SRT)

- kies proces met kleinste (s-e)
- pre-emptive: als er een nieuw proces bij komt met kleinere (s-e), dan wordt huidige proces onderbroken
- nadeel: zoals vorige
- niet voor online scheduling

Highest Response Ratio Next (HRRN)

- zoek proces met hoogste (w+s)/s
- non pre-emptive
- Response ratio = (w+s)/s
  - initieel gelijk aan 1
- voordeel: korte processen hebben grote RR en worden snel afgewerkt maar lange processen die al lang wachten krijgen steeds hogere RR
- nadeel: s moet nog steeds geschat worden
- niet voor online scheduling

Round-Robin

- meest voorkomende scheduling in moderne OS'en
- queue van processen: neem steeds de volgende
- pre-emptive
  - ieder proces krijgt evenveel tijd (time-slice)
  - proces wordt onderbroken door interrupt
- voordeel: goede response-tijd, rechtvaardige behandeling
- nadeel: i/o gebonden processen krijgen minder tijd

Ubuntu timeslice

1. Wat is de grootteorde van de timeslice bij Linux?

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Scheduling in Linux

- pre-emptive, round-robin
- verschillende prioriteiten
  - ieder proces prioriteit/nice-value
  - nieuw proces start default met PR 20 (NI 0)
    - PR = NI + 20
  - kan aangepast worden met nice (of renice)
    - NI: -20 tot +19
    - PR: 0 (hoog) tot 39 (laag)
  - Hoe lager PR of NI, hoe hoger de prioriteit

Slurm (optional, niet te kennen)

- Simple Linux Utility for Resource Management
- Open source batch job scheduler
- Used by many supercomputers
- Allocating exclusive/non-exclusive access to computer nodes
  - cat submit.sh

```
#!/bin/bash
#SBATCH --job-name=test
#SBATCH --output=test.out
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=100
srun testprogram
```

  - sbatch submit.sh
  - squeue
- https://drtailor.medium.com/how-to-setup-slurm-on-ubuntu-20-04-for-single-node-work-scheduling-6cc909574365
- https://support.ceci-hpc.be/doc/_contents/QuickStart/SubmittingJobs/SlurmTutorial.html

OPDRACHT

Inhoud

- Dispatcher/scheduler
- Toestanden van een proces
- Opstarten van processen in Linux
- (Non) pre-emptive context switch
- Scheduling strategieën
- Linux scheduling
- Herhalingsvragen

Voorbeelden examenvragen

- Bespreek de delen van een proces: stack, data, code, PCB
- Leg de 5 toestanden van een proces uit en geef uitleg bij de overgangen.
- Wat is een PCB?
- Wat is idle time?
- Wat gebeurt er tijdens een pre-emptive context switch?
- Wat is het verschil tussen pre-emptive en non pre-emptive scheduling?
- Wat is een context-switch?
- Wat is scheduling?
- Hoe start een proces in Linux? Verklaar de fork en exec system calls.
- Wat doet de volgende C code?
- Bespreek de scheduling strategie X (wat/hoe, voordeel, nadeel)
- Is scheduling strategie X pre-emptive of niet, waarom?
- Wat is uithongering (starvation)?