# Uitbatingssystemen



| Uitbatingssystemen | |
|---|---|
| Richting | Informatica |
| Jaar | 2BINF |

## Bespreking

Dit vak wordt pas sinds 2021-2022 door José gegeven, daarvoor was dit Benny. Het is dus nog een beetje afwachten of er verschillen gaan zijn in de examens lessen.

### Theorie

Voor de theorie van uitbatingssystemen moet je proberen de cursus vooral te begrijpen en alle beschreven technieken te kennen en de gevolgen van het gebruik ervan goed te begrijpen. Op het examen worden doordenkvragen gesteld in de trend van: vergelijk die techniek in combinatie met die andere techniek met deze techniek en geef voordelen en nadelen van beide. Dit heeft als voordeel dat je niet elk klein detail uit je cursus moet kennen, maar heeft als nadeel dat je niet gewoon de tekst in je cursus kunt reproduceren. Lees elke vraag rustig meerdere keren, schrijf eerst van elke gevraagde techniek alle informatie op die handig kan zijn bij het oplossen van de vraag en vergelijk dan pas. Schrijf je niet eerst alle informatie van beide technieken eerst op, dan heb je meer kans op het vergeten van details die erbij horen. Het mondelinge examen van organisatie is eigenlijk gewoon het opzeggen van wat je net had neergepend, er worden dus niet echt bijvragen gesteld.

### Praktijk

Enerzijds zijn er de theoretische oefeningen met pen & papier, anderzijds zijn er de drie praktijkopdrachten die niet ingediend moeten worden. De drie opdrachten i.v.m. caching, hard disks & scheduling zijn al enkele jaren hetzelfde, dus als je met een opdracht niet verder kan moet je maar eens horen bij mensen die al een credit behaald hebben voor het vak.

Op het schriftelijk praktijkexamen zijn er dan ook in totaal 4 vragen, waarvan twee theoretische oefeningen en twee inzichtsvragen i.v.m. de praktijkopdrachten (waarvan je de source code mag meenemen op het examen zelf).

## Puntenverdeling

Theorie: 10/20. Praktijk: 10/20.

## Examenvragen

### Academiejaar 2021-2022 1e zit

1. Geef drie verschillen tussen een mutliprocessing system en een multithreading system.
2. Leg de werking uit van een set associative cache. Geef aan wat de voordelen zijn, geef aan hoe 'k' gekozen wordt en hoe deze evolueert.
3. Leg de werking van dynamic partitioning uit. Geef aan wat het voordeel tov fixed partitioning en geef aan wat de nadelen zijn.
4. Leg de werking uit van een multilevel page table. Leg uit hoe de omzetting van logisch naar fysisch adres gebeurt.

5.
    1. Wat is short medium en long term scheduling. Wanneer treden deze op.
    2. Leg de werking van een multilevel feedback queue uit. Welke processen favort dit systeem? Wat zijn de voordelen en nadelen?
    3. Leg de SCAN en LOOK algoritmen uit. Geef aan wat het nadeel is en hoe dit opgelost wordt door hun Cyclic variant.
    4. Leg uit Linked allocation in de context van filesystemen. Geef ook aan wat een nadeel hiervan is. Zijn er design choices die gemaakt worden om dit tegen te gaan, zo ja welke?

## Academiejaar 2019-2020 1e zit

### Oefeningen

Bestand:INF OS JAN2020 OEF.pdf

### Theorie VM

Schriftelijk

- Memory Managment (6): Wat zijn de voornaamste kenmerken van dynamisch partitionering (denk aan de plus- en minpunten)? Wat verstaan we onder de 50 procent regel? Leg in detail uit hoe we tot deze regel zijn gekomen. Hou zou deze regel heten indien we weten dat er bij het toevoegen van een image in de helft van de gevallen een perfecte match is?
- Disk Managment (4): Bespreek de werking en voor- en nadelen van volgende algoritmes: SSTF, SCAN, C-SCAN en VSCAN(R)
- File systemen (4): Bespreek de allocatie methode die de UNIX inode hanteert, wat is de achterliggende idee van deze allocatie methode? Wat is het verschil tussen een soft en een hard link? Welke velden bevat de UNIX inode om soft en hard links te ondersteunen?

Mondeling

- Wat is Belady's anomalie?
- Wat wordt er opgeslagen in een TLB cache?
- FFS gebruikt fragmenten van 1KB, maar is toch sneller dan als je de traditionele methode zou gebruiken met blokken van 1KB. Hoe komt dit?
- Wat is spatial locality?
- Wat betekent 1||∑Uj∑Uj? Hoe optimaliseren we dit?
- Wat doet de Base value bij de UNIX-scheduler

### Theorie NM

Schriftelijk

- Caching: De pseudo LRU MRU variant, hoe werkt die bij set associative mapping? Hoeveel meest recente blokken worden niet overschreven en hoe lang kan een niet gebruikt blok overleven?
- Scheduling: 1|rjrj|∑Cj∑Cj leg uit wat hiermee bedoeld wordt. Hoe werkt nSRTN en hoeveel slechter is dit dan het optimale algoritme? Bewijs.
- Journaling: Geef een voorbeeld van garbage en inconsistent file system. Wat is het nadeel van journaling en hoe kunnen we dit deels oplossen?

Mondeling

- Hoeveel procent interne fragmentatie heb je bij het Buddy systeem en waarom?
- SCAN en CSCAN: Welke is beter en waarom?
- Hoeveel entries heeft een inverted page table?
- Welke allocatie gebruikt FAT en welke acces is beter dan gewone linked allocatie?
- Hoe gaat de UNIX scheduler starvation tegen?
- Waarom gebruiken we bij virtual memory random offsets?

## Academiejaar 2018 - 2019 1e zit

- Media:Theorie_Examen_Uitbatingssystemen_20190116_Groep_1_VM.pdf

- Media:Oefeningenexamen_Uitbatingssystemen_Januari_2019.pdf

## Academiejaar 2017 - 2018 2e zit

Media:UBS_2018_zit2.pdf

## Academiejaar 2017 - 2018

### Praktijk & Theorie

Media:US2017-2018zit1.pdf

## Academiejaar 2016 - 2017

### Praktijk & Theorie

Media:Uitbatingssytemen201627017.pdf

## Academiejaar 2015 - 2016

### Praktijk

Media:UitbatingssytemenOefeningenJanuari2015.pdf

## Academiejaar 2010 - 2011 - 2de zittijd

### Praktijk

1. Gegeven de volgende code om de cachesize (blocksize? -Bob) te berekenen:
   -

   int i = 1;

   access_cache(0);

   while(access_cache(0)) {
   i++;

   access_cache(pow(2,i));

   }

   return pow(2,i);

   -

   De cache is reeds gevuld met adressen ver boven de cache grootte. In welke van de volgende gevallen geeft deze code het juiste resultaat? In de gevallen dat de code het verkeerde resultaat berekent, wat is dit dan? LRU, FIFO telkens in combinatie met Directe Mapping, 2-way associatieve cache of 4-way associatieve cache.

2. Stel dat we deze jobs met harde of softe deadlines willen schedulen:
   A / 2 / 16 / Hard

   B / 3 / 17 / Hard

   C / 1 / 11 / Hard

   D / 2 / 7 / Hard

   E / 2 / 3 / Hard

   X / 2 / 5 / Soft

   Y / 2 / 11 / Soft

   Z / 3 / 4 / Soft

   Geef grafisch de scheduling volgorde weer opdat alle harde deadlines gehaald worden, en de maximale lateness van de soft deadlines geminimaliseerd wordt.

3. Gegeven volgende reeks van page requests: 0,1,...,511,431,0,1,...,511,332,0,1,...
   Dus in andere woorden: telkens de gehele reeks van 0 t.e.m. 511, met daartussen at random 431 of 332 ertussen.

   1. Leg uit waarom de traditionele replacement algoritmen (LRU, FIFO, Clock) hier niet efficient zijnt als er maar een gering aantal frames beschikbaar is. Hoeveel frames zou je nodig hebben om dit efficient te maken?
   2. Stel, je hebt 500 frames. Bedenk een algoritme dat efficienter is dan de traditionelen.
4. Gegeven een IO controller die gepositioneerd staat op track 26 van een harde schijf met 128 tracks (genummerd 0,1,...,127), en gegeven de volgende track requests: 26,37,100,14,88,33,99,12
   Bereken het aantal trackwissels dat de leeskop moet uitvoeren als deze reeks requests wordt behandeld via SSTF, SCAN en C-LOOK

## Academiejaar 2010 - 2011 - 1ste zittijd

### Theorie

Voormiddag

- **Virtueel Geheugen**: Bespreek de werking van Segmented FIFO en geef aan wat de voor- en nadelen zijn ten opzichten van LRU en FIFO. Staan de pages in de L2 buffer in LRU volgorde? Hoe kan je het idee van Segmented FIFO veralgemenen?
- **Scheduling**: Waarvoor wordt het Shortest-remaining-time-next (SRTN) scheduling algoritme gebruikt en hoe werkt dit precies. Geef aan voor welk scheduling probleem dit algoritme optimaal is en bewijs dit. Toon aan dat dit algoritme niet meer optimaal is indien we geen preemptie toelaten (door een tegenvoorbeeld te geven)?
- **File Systems**: Geef aan wat we precies bedoelen met "continuous allocation", wat zijn hiervan de voor- en nadelen? Wat weet je te melden over het gemiddelde aantal vrije gebieden ten opzichte van het aantal aanwezige files? Zijn er extra overwegingen die we kunnen maken bij de keuze van het placement algoritme?
- **Disk Scheduling**: Bespreek de werking en performantie van RAID 3 en 4 systemen en geef aan voor welke doeleinden deze het meest geschikt zijn. Geef ook aan hoe deze systemen presteren in geval van een diskfaling (in andere woorden: bespreek hun robuustheid)
- *Optioneel:* Zoals tijdens het jaar aangegeven mag je 1 van de bovenstaande vragen onbeantwoord laten indien je in de plaats daarvan het bewijs van de optimaliteit van het MIN algoritme kon opgeven. Bij het uitwerken van het bewijs hoef je slechts 1 ongelijkheid naar keuze uit het Lemma uit te werken.

Namiddag

- **Paging en Segmenatie**: Bespreek paging en segmentatie en geef de verschillen.
- **Scheduling**: Bespreek EDF, wat wordt er juist geoptimaliseerd? Bewijs! Geef ook een voorbeeld waar EDF niet het aantal gehaalde deadlines optimaliseert, werk een algoritme uit die dat wel doet.
- **File Systemen**:Vergelijk het FFS met het traditionele UNIX FS.

- **Disk Scheduling**: Beschrijf SSTF, (C-)SCAN, (C-)LOOK, VSCAN(R) en FSCAN. Geef ook telkens de voor- en nadelen.
- *Optioneel*: Zoals tijdens het jaar aangegeven mag je 1 van de bovenstaande vragen onbeantwoord laten indien je in de plaats daarvan het bewijs van de optimaliteit van het MIN algoritme kon opgeven. Bij het uitwerken van het bewijs hoef je slechts 1 ongelijkheid naar keuze uit het Lemma uit te werken.

Gepersonaliseerd examen

De volgende vragen werden aan een student gegeven die afwezig was tijdens de examenperiode.

- **Paging en Segmenatie**: Leg uit hoe een paging- en segmentatiesysteem werkt? Wat zijn de voornaamste verschillen? Wat verstaan we onder prepaging?
- **Scheduling**: Bespreek de werking van de traditionele UNIX scheduler. Bij welke theoretishe discipline leunt deze het dichste aan (wat zijn de verschillen)? Hoe worden I/O- en CPU-bound processen behandeld? Wat is de rol van base en nice?
- **File Systemen**: Bespreek de allocatie methode die de UNIX inode hanteert, wat is de achterliggende idee van deze allocatie methode? Wat is het verschil tussen een soft en een hard link? Welke velden bevat de UNIX inode om soft en hard links te ondersteunen?
- **Disk Scheduling**: Beschrijf SSTF, (C-)SCAN, (C-)LOOK, VSCAN(R) en FSCAN. Geef ook telkens de voor- en nadelen.
- *Optioneel:* Zoals tijdens het jaar aangegeven mag je 1 van de bovenstaande vragen onbeantwoord laten indien je in de plaats daarvan het bewijs van de optimaliteit van het MIN algoritme kon opgeven. Bij het uitwerken van het bewijs hoef je slechts 1 ongelijkheid naar keuze uit het Lemma uit te werken.

**Praktijk**

1. Beschouw de cache simulator met de zelfde interface als die van tijdens de praktijkprogrammeerlessen. Stel dat we de grootte van de cache reeds hebben bepaald en dat we nu de associativiteit van de cache willen bepalen.

   -

   int assoc = 1;

   access_cache(0);

   while(access_cache(0)) {
   access_cache(assoc*cache_size);

   assoc *= 2;

   }

   return assoc\2;

   -

   De cache is reeds gevuld met adressen ver boven de cache grootte. In welke van de volgende gevallen geeft deze code het juiste resultaat? In de gevallen dat de code het verkeerde resultaat berekent, wat is dit dan? LRU, FIFO telkens in combinatie met Directe Mapping, 2-way associatieve cache of 4-way associatieve cache.

2. Scheduling
    1. Leg je oplossing uit en hoe je er tot gekomen bent.
    2. Is je oplossing optimaal? Indien niet, hoe goed werkte ze dan wel?
    3. Kon je via aanpassingen er ook voor zorgen dat je oplossing optimaal is als we geen preemption toelaten?

3. 32-bit machine met virtuele adressen die zijn opgesplitst in 4 delen van 8, 8, 6 en 10 bits respectievelijk. We hanteren een level 3 page table systeem. Een page entry is 16 bytes.
    1. Geef de page size.
    2. Een process dat 1024 Kb geheugen vereist, hoeveel plaats nemen de page tables in in het geheugen?
    3. Een process heeft een code segment van 64 Kb, vanaf (virtueel) adres 0x10000000 tot 0x1000FFFF, een 400 Kb heap segment vanaf 0x10100000 tot 0x10163FFF en een stack segment van 10 Kb vanaf adres 0x1020000 tot 0x10203FFF. Hoeveel plaats neemt de page table van dit process in?
4. Het "sleeping barber" probleem: de kapperszaak met een wachtruimte met NN stoelen, een kappersruimte en een kappersstoel. Als er geen klanten zijn slaapt de kappper. Als een klant binnenkomt en alle stoelen in de zaal zijn vol, vertrekt de klant. Als de kapper bezig is, en er zijn stoelen beschikbaar gat die klant naar één van deze stoelen. Als de kapper slaapt, maakt de klant hem wakker. Nog extra informatie:
    - De klant gebruikt exit() als de wachtkammer vol is.
    - Als de kapper kniphaar() uitvoert mag slechts 1 klant laatKnippen() uitvoeren.

    1. Stel we lossen dit probleem op zonder concurrency. Geef een mogelijke volgorde van acties waarbij een probleem kan opduiken.
    2. Geef een oplossing voor dit probleem waarbij je gebruikt maat van binaire en counting semaforen.

## Academiejaar 2008 - 2009 - 1ste zittijd

### Praktijk

1. In het practicum heb je een harde schijf gekarakteriseerd. Leg bondig je oplossing uit om:

    1. het aantal sectoren op de buitenste track te bepalen.

    2. de VScan parameter R te bepalen. Je mag er vanuit gaan dat je de rest van de parameters reeds kent. Welke R waarden waren het moeilijkste (nauwkeurig) te bepalen en waarom?

2. In het practicum heb je voor volgende opgave een oplossing geschreven.

    Veronderstel dat we N1N1 jobs hebben met hard deadlines djdj. Daarnaast zijn er nog N2N2 jobs met soft deadlines. Het overschrijden van een hard deadline is niet toegelaten, terwijl soft deadlines beperkt mogen overschreven worden.

    Ontwerp nu een algoritme dat zowel

    - alle hard deadlines haalt

    - de maximale lateness van de soft deadlines minimaliseert, of nog $\max_{j \in N2}(C_j - d_j)$ zo klein mogelijk maakt.

    Het systeem laat preemption toe.

    1. Als je nu weet dat alle jobs met hard deadlines hun deadline kunnen halen, beschrijf dan bondig je algoritme om dit probleem op te lossen.

    2. Werkt je oplossing optimaal? Zo ja, verklaar. Zo neen, welk(e) alternatief(/ven) heb je gebruikt voor je tot je finale oplossing gekomen bent?

3. We willen het volgende algoritme gebruiken, om een schedulingprobleem op te lossen, waarbij de taken een deadline hebben en er zijn precedence voorwaarden.

    - Laat JJ de verzameling taken zijn die nog uitgevoerd moeten worden.
    - Kies de taak jj met de kleinste deadline uit de verzameling JJ.
    - Als er in JJ nog taken zitten die voor jj moeten uitgevoerd worden, volgens de precedence voorwaarden, voer dan al deze taken uit, zodat aan de precedence voorwaarden voldaan is. Wanneer hierbij nog keuze is tussen verschillende uitvoeringsvolgordes, wordt eerst de taak met de kleinste deadline gekozen.
    - Voer taak jj uit.
    - Herhaal zolang niet alle taken zijn uitgevoerd.

    1. Is dit algoritme optimaal wanneer de maximale mate waarin de deadlines overschreden worden maxj(Cj−dj)maxj(Cj−dj) zo klein mogelijk moeten zijn? Zo neen, geef een tegenvoorbeeld. Zo ja, verklaar.
    2. Wat verandert er aan je antwoord, als we in het bovenstaande algoritme "*Wanneer hierbij nog keuze is tussen verschillende uitvoeringsvolgordes, wordt eerst de taak met de kleinste deadline gekozen.*" vervangen door "*Wanneer hierbij nog keuze is tussen verschillende uitvoeringsvolgordes, wordt een willekeurige taak gekozen*"?

4. Beschouw de volgende oplossing voor het readers/writers probleem, welke een variant is van de oplossing op p125 in de cursus. Met welk van de volgende twee concurrency issues geeft deze code een probleem: mutual exclusion of deadlock? Geef een uitvoeringsvolgorde waarbij dit probleem effectief optreedt.

```
procedure reader:
  wait(x);
   rcount := rcount+1;
   if rcount = 1 then wait(wsem);
  signal(x);
  READ
  wait(x);
   rcount := rcount-1;
   if rcount = 0 then signal(wsem);
  signal(x);

procedure writer:
  wait(y);
   wcount := wcount+1;
   if wcount = 1 then wait(x);
  signal(y);
  wait(wsem);
    WRITE;
  signal(wsem);
  wait(y);
   wcount := wcount-1;
   if wcount = 0 then signal(x);
  signal(y);
```

## Academiejaar 2007 - 2008 - 1ste zittijd

*(Alle vragen van 2e bachelor, 3e bachelor en 1e master zijn samengevoegd aangezien dit vak dit academiejaar een schakelvak was voor alle jaren vanaf 2de bachelor)*

### Theorie uitbatingssystemen

1. Leg uit hoe je de associativiteit van de L1 cache kan bepalen uit de cache benchmarking test die we in de cursus besproken hebben? (Maak eventueel een schets ter illustratie).

2. Beschouw een virtueel geheugen systeem dat gebruik maakt van multilevel page tables en dat is uitgerust met een L1 en een TLB cache. Welke fouten en algoritmen kunnne er dan allemaal voorkomen en in werking treden bij het vertalen van een virtueel adres?
3. Voor welk scheduling probleem is Earliest Deadline First (EDF) optimaal? Toon dit aan.
4. In de cursus is er het MESI protocol besproken voor cache coherency. Hoe zou MSI dan werken? wat zijn de voor- en nadelen ten opzichte van MESI?

**Praktijk uitbatingssystemen**

1. Beschouw de cache simulator met de volgende functie zoals gebruikt in het practicum.
   `int access_cache(unsigned long InputAddres);`
   Stel dat we volgende code gebruiken om de cache grootte te bepalen:

   > int i = 0; access_cache(0); while(access_cache(0)){ i++; access_cache(i); } return i;

   In welke van volgende zes gevallen zal deze code de juiste cache grootte kunnen achterhalen, wetende dat de cache reeds gevuld is met adressen ver boven de cache grootte? Verklaar waarom (niet). Voor de gevallen waarin de code niet werkt, welk resultaat krijg je dan wel?

   > Direct mapping & 2-way associative mapping & 4-way associative mapping
   >
   > LRU & &
   > FIFO & &

2. In het practicum van Disk Benchmarking heb je de VSCAN parameter R, gelegen tussen 0 en 1, van een aantal disks proberen te achterhalen.

   1. Leg bondig uit hoe je oplossing werkt, je mag de (random) rotational delay hierbij buiten beschouwing laten.

   2. Leg uit hoe je getracht hebt hierbij het effect van de random rotational delay weg te filteren. Als je verschillende methodes uitgeprobeerd hebt, welke gaf dan het snelste resultaat?

3. Een round robin scheduler moet 2 taken schedulen. De lengte van taak 1 is m, die van taak 2 is n met m>n. Gegeven een quantum q > m en een process switch time t ( die enkel nodig is als er ook effectief van taak wordt gewisseld).

   1. Voor welke volgorde is mC1 + n C2 minimaal? (Ci = completion time i)?

   2. Voor welke waarde(n) van t zorgen alle volgordes voor een minimale mC1 + nC2?

4. Stel dat we een file van 512000 bytes willen lezen van een harde schijf met RPM = 10000, elke sector bevat 512 bytes, er staan 320 sectoren per track (geen zone bit recording) en het aantal tracks is zeer groot. Als er van track moet gewisseld worden, kan je de tijd hiervoor goed benaderen met:

   1ms+10msaantal tracks te wisselenaantal tracks in totaal1ms+10msaantaltrackstewisselenaantaltracksintotaal Je mag veronderstellen dat de disk head zich reeds bevindt boven de track van het eerste block. Bereken nu de gemiddelde tijd om deze file te lezen (de corresponderende directory entry staat reeds in het RAM geheugen) indien:

   1. er gebruik gemaakt wordt van continous allocatie.

   2. er gebruik gemaakt wordt van linked allocatie zonder clustering, waarbij een blok 1 sector aan data kan bevatten. De file is aangemaakt op een moment dat het filesysteem reeds intensief gebruikt werd.

   3. Het free space algoritme leverede telkens een blok op dat op een willekeurige plaats op de harde schijf staat. Zo is de kans klein dat 2 blokken na elkaar staan op de schijf zeer klein.

**Theorie Aanvullingen uitbatingssystemen**

Bij het beantwoorden van eventuele ja/nee vragen wordt steeds verwacht dat je ook een woordje uitleg voorziet.

1. **REAL-TIME SCHEDULING**: Bespreek de werking van het algoritme van Moore en Hodgson dat tracht het aantal taken dat zijn deadline niet haalt te minimaliseren. Geef de exacte voorwaarden waaronder dit algoritme optimaal is.
2. **DEADLOCK PREVENTIE**: Wanneer wordt de toestand van een systeem *safe* of *unsafe* genoemd? Wat impliceert dit met betrekking tot deadlocks? Hoe controleert het Banker's algoritme of een toestand safe is (wat is de complexiteit)? Hoe voorkomt het Banker's algoritme hiermee deadlocks?
3. **FILE SYSTEMEN**: Welke allocatie methoden wordt er door de File Allocation Table (FAT) gebruikt? Welke voor- en nadelen biedt deze methode (voor onder andere de snelheid waarmee *sequential* en *random* kunnen plaatsvinden)? Welke vorm van free-space management vinden we bij de FAT terug?

#### Praktijk Aanvullingen uitbatingssystem

1. Beschouw de cache simulator met de volgende functie zoals gebruikt in het practicum.
   `int acces_cache(unsigned long memoryAddress);`
   Stel dat we de grootte van de cache reeds hebben bepaald en dat we nu de associativiteit van de cache willen bepalen met de volgende code:

   > int associativity = 1; access_cache(0); while (access_cache(0)) { access_cache(associativity * cache_size); associativity *= 2; } return associativity / 2;

   Veronderstel dat de cache reeds is gevuld met adressen ver boven de cachegrootte. In welke van de volgende 6 gevallen geeft deze code het juiste resultaat? Verklaar waarom (niet). In de gevallen dat de code het foute resultaat berekent, welk resultaat is dit dan?

   > Direct mapping & 2-way associative mapping & 4-way associative mapping
   >
   > LRU & &
   > FIFO & &

2. In het practicum "Scheduling" heb je het volgende probleem bekeken:

   > Veronderstel dat we N1N1 jobs hebben met hard deadlines djdj (met hun gezamenlijke load kleiner dan of gelijk aan 1). Daarnaast zijn er nog N2N2 jobs met soft deadlines. Het overschrijden van een hard deadline is niet toegelaten, terwijl soft deadlines beperkt mogen overschreden worden.
   >
   > Ontwerp nu een algoritme dat:
   >
   > - Alle hard deadlines haalt
   > - De maximale lateness van de soft deadlines minimaliseert, of nog maxj∈N2(Cj−dj)maxj∈N2(Cj−dj) zo klein mogelijk maakt
   >
   > Het systeem laat preemption toe.

   1. Leg bondig je oplossing uit en leg uit hoe je tot die oplossing bent gekomen.
   2. Is je oplossing optimaal? Indien niet, hoe goed werkt ze dan wel?

### Academiejaar 2007 - 2008 - 2de zittijd

1. Leg het buddy systeem uit. Hoeveel interne fragmentatie treedt er op bij gebruik van dit systeem? Kunnen we ook met fibonaccigetallen werken?
2. Leg het Clock algoritme uit. Bespreek de voor- en nadelen m.b.t. LRU.
3. Leg Rate Monotonic Scheduling uit. Wat is het verschil met EDF? Welk algoritme geeft de beste resultaten? Waarom wordt het mindere optimale algoritme toch gebruikt?
4. Leg de UNIX iNode uit. Wat is een soft link en een hard link? Welke velden gebruikt de inode om de links te implementeren?

## Academiejaar 2006 - 2007 - 1ste zittijd

### Theorie

1. We know we can use first,next, worst and best fit in memory. How and when would you use these placement algorithms on hard drives ?
2. How is an instruction being searched using a translation lookaside buffer (TLB)?
3. Suppose you have 15 hard disks and you wish to use about 30% of it for redundant data such as error detections code. The system has to use a RAID 5-1, how would your setup be like.
4. Suppose you have a cache for your memory transfer using direct mapping. The first r bits are being used to select the line in the cache on which the block of data should be placed. what would be the consequences if you used the middle r bits to select the line in the cache ?
5. Suppose you modify the round robin scheduling so that when a process is placed in the processor to run and isn't interrupted by an IO-event before a timeout occurs, the next time the process enters the processors, it can occupy the processor twice as long as before. However, when a IO-event does occur, the next time the process enters the processor, the time it can occupy the processor without timing out doesn't change. What would be the result of that ? Compare it with the multilevel feedback queue scheduling algorithm.