Scheduling Theorie opdrachten – CAR ESAD

Thomas Kooi
Michiel Beuvink

1. Beschrijf het fundamentele verschil tussen de twee \hoofd-policies".

Fair scheduling policies:
Hier worden jobsgescheduled/ingepland op basis van capaciteit / aantal verschillende jobs. *Hier krijgen elke job de zelfde kans om gescheduled te worden.*

Priority scheduling policies:
Hoe belangrijker hoe meer tijd er voor de job beschikbaar wordt gesteld OF hoe sneller die wordt ingepland.

2. Beschrijf van elke “hoofd-policies" tenminste twee varianten, uitgezonderd het “LeastLaxity" of “minimum/LeastSlack" algoritme.

Fair schedulingpolicies:

- **FIFO scheduling**
  First in first out. De eerste job/taak die arriveert wordt als eerste gepland.
  Voordelen:
    - Stable. Alle jobs zullen op een moment in tijd aan de buurt komen. Er kan berekend worden of een job lang duurt. Er is geen sprake van starvation.

  Nadelen:
    - Niet repsonsive. Zodra er een job toegevoegd word moet er eerst gewacht worden tot alle voorgaande jobs uitgevoerd zijn.
    - Niet optimaal. Er wordt niet gekeken of een andere job ondertussen ook kan runnen. De eerste job wordt ingepland en uitgevoerd, daarna de tweede pas.
    - Niet robuust. Als een taak er erg lang overdoet, zal de rest hierop wachten.

  Geschikt:
    - Processen waar gegarandeerd alle jobs uitgevoerd moeten worden maar er geen belang is aan de tijdigheid hiervan.

  Ongeschikt:
    - Processen waar er een belang is aan het moment waarop een taak wordt uitgevoerd: Bijvoorbeeld alles zo snel mogelijk.

- **RotatingStaircase deadline**.
  Bij RotatingStaircase deadline worden taken op prioriteit gesorteerd. Elke prioriteit krijgt een bepaald quotum aan CPU tijd. Wanneer één van de taken over het quotum gaat worden alle taken in prioriteit verlaagd met het daarbij behorende quotum. Alle taken met de zelfde prioriteit worden onderling gescheduled met de round-robin scheduling.

  Voordelen:
    - Robust. Doordat een taak na het overschrijden van het quotum een lagere prioriteit krijgt zullen taken met een lagere prioriteit vanzelf een hogere prioriteit hebben.
    - Stable. Er kan een eind tijd van te voren worden bepaald.

  Nadelen:
    - Niet optimaal.

  Geschikt:
    - Processen waar geen starvation mag voorkomen en taken tijdig moeten worden ingepland.

  Ongeschikt:
    - Embedded systems.

Priority schedulingpolicies:

- **Shortest Job First**.
  De job die het dichtste bij zijn deadline is wordt het eerste gepland.
  Voordelen:
  - Robust. Er hoeft niet te worden gewacht op de langste job.

  Nadelen:
  - Niet responsive. Een nieuwe job is niet perse de kortste. Het kan dus zijn dat deze heel lang wordt uitgesteld tot deze wordt ingepland.
  - Niet optimaal. Er word alleen gekeken naar de kortste job. Deze wordt eerst ingepland. Er worden geen jobs gelijktijdig bij gepland.

  Geschikt:
  - Niks, er zijn geen gevallen te bedenken dat het nuttig is om altijd als eerst de kortste taak te plannen.

  Ongeschikt:
  - Alles.

- **Fixed-priority pre-emptivescheduling**
  Een taak krijgt op basis van prioriteit een beetje CPU-tijd waarna die weer onderbroken word. Vervolgens krijgt de taak met dan de hoogste prioriteit een beetje CPU-tijd. Zoals de naam al zegt is er sprake van een fixed-priority die dus niet verandert.
  Voordelen:
  - Robust. Taken worden opgesplitst in verschillende stukken, die worden uitgevoerd. Hierdoor worden jobs wel uitgevoerd ook als er een taak van een job er erg lang over doet.
  - Responsive. Ieder moment kan een taak met een hoge prioriteit de CPU-tijd krijgen door de verdeling in kleine stukken.
  - Stable.

  Nadelen:
  - -

  Geschikt:
  - Embedded systems. Real-Time Operating systems.

  Ongeschikt:
  - Processen waar alle taken uitgevoerd moeten worden.