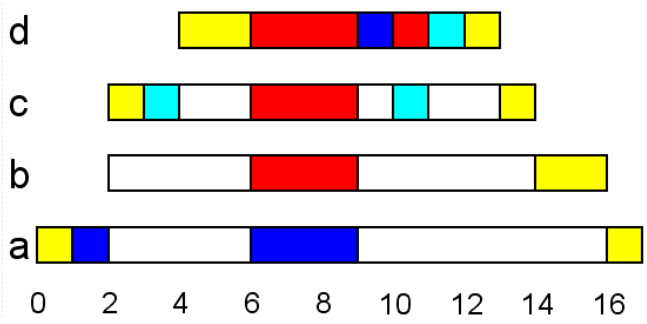
Real-Time Systems (RTSYST)

Week 7

DE HAAGSE HOGESCHOOL

Priority inheritance voorbeeld



- Executing
- Executing with Q locked
- Executing with V locked
- Preempted
- Blocked

taak	prio	execution	release time
d	4	EEQVE	4
c	3	EVVE	2
b	2	EE	2
a	1	EQQQQE	0

170

Blocking Priority inheritance

- De tijd dat een taak blocked kan zijn is nu beperkt.

$$B_i = \sum_{k=1}^{K} usage(k,i)C(k)$$

- ***B*** = maximum blocking time
- ***K*** = aantal gebruikte resources
- ***usage(k, i)*** = boolean functie
  - 1 als er een taak is met prioriteit lager dan P_i en een taak met prioriteit hoger of gelijk aan P_i (dit kan taak ***i*** zelf zijn) die de resource ***k*** delen.
- ***C(k)*** = maximale tijd dat resource ***k*** gebruikt wordt.

DE HAAGSE HOGESCHOOL 171

Blocking Response time analyse

$$R_i = C_i + B_i + I_i$$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$w_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

DE HAAGSE HOGESCHOOL 172

Blocking Priority inheritance

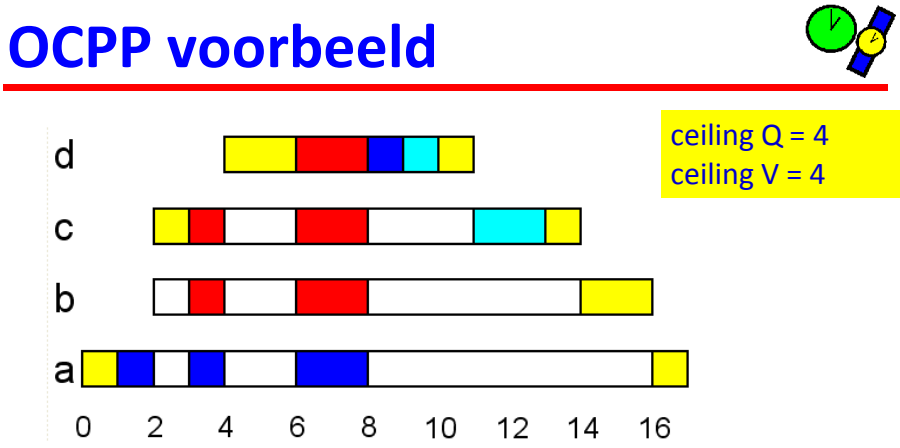
- Priority inheritance kan niet eenvoudig geïmplementeerd worden!
  - Bij semaphoren en condition variabelen is vaak niet te achterhalen op wie je wacht (wie de sem_post of pthread_cond_signal zal geven)!

  - Bij message passing is het vaak niet te achterhalen op wie je wacht (wie de mq_send zal doen waarop de mq_receive moet wachten)!
- Alternatief: priority ceiling

FPS-DMPO Priority ceiling

- Original Ceiling Priority Protocol OCPP:
  - Elke taak heeft een static priority.
  - Elke resource heeft een ceiling priority = maximale prioriteit van de processen die deze resource delen.
  - Een taak die een taak met een hogere prioriteit blocked erft de hogere prioriteit.
  - Een taak kan alleen een resource gebruiken als zijn prioriteit hoger is dan de ceiling van alle door andere taken gebruikte resources.
- Voordelen:
  - Proces kan maar 1x blocked worden.
  - Deadlock is niet mogelijk.
  - Mutual exclusive gaat altijd “van zelf” goed.
  - Transitive blocking is niet mogelijk.

OCPP voorbeeld



ceiling Q = 4
ceiling V = 4

- Executing
- Executing with Q locked
- Executing with V locked
- Preempted
- Blocked

taak	prio	execution	release time
d	4	EEQVE	4
c	3	EVVE	2
b	2	EE	2
a	1	EQQQQE	0

175

OCPP Blocking

- Eerste resource kan altijd worden gebruikt.
- Tweede resource kan alleen worden gebruikt als er geen taak is met een hogere prioriteit die beide resources gebruikt.
- De tijd dat een taak blocked kan zijn wordt nu:

$$B_i = \max_{k=1}^{K} usage(k,i)C(k)$$

- Impementatie is nog steeds net zo moeilijk!
- Alternatief: Immediate CPP.

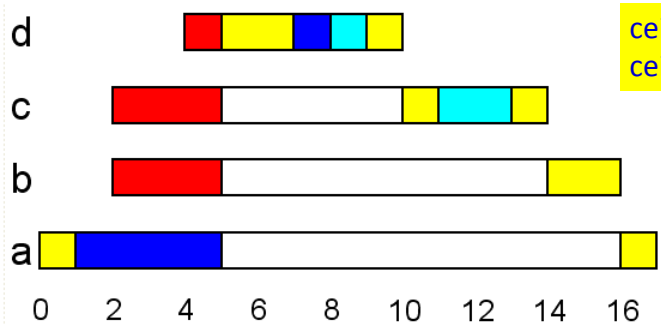
DE HAAGSE HOGESCHOOL 176

FPS-DMPO Priority ceiling

- Immediate Ceiling Priority Protocol ICPP:
  - Elke taak heeft een static priority.
  - Elke resource heeft een ceiling priority = maximale prioriteit van de processen die deze resource delen.
  - Als een taak een resource gebruikt krijgt die taak de ceiling prioriteit van de resource.
- Voordelen:
  - Proces kan maar 1x blocked worden.
  - Deadlock is niet mogelijk.
  - Mutual exclusive gaat altijd “van zelf” goed.
  - Transitive blocking is niet mogelijk.
  - Ten opzichte van OCPP:
    - Eenvoudiger te implementeren.
    - Minder taak wisselingen.

DE HAAGSE HOGESCHOOL 177

ICPP voorbeeld



ceiling Q = 4
ceiling V = 4

Executing
Executing with Q locked
Executing with V locked
Preempted
Blocked

taak	prio	execution	release time
d	4	EEQVE	4
c	3	EVVE	2
b	2	EE	2
a	1	EQQQQE	0

178

POSIX Priority Based Scheduling

- FIFO
  - Een thread blijft running totdat de thread blocked of preempted wordt.
  - Een thread die preempted wordt komt vooraan de readyqueue te staan.
- Round-Robbin
  - Een thread blijft running totdat de thread blocked of preempted wordt of totdat de time-slice is verstreken.
  - Een thread die preempted wordt komt vooraan de readyqueue te staan.
  - Een thread waarvan de time-slice is verstreken komt achter de threads met gelijke prioriteit op de readyqueue te staan.
- Sporadic Server
- Other

DE HAAGSE HOGESCHOOL 179

Priority Protect Protocol =ICPP

- int pthread_mutexattr_getprotocol(const thread_mutexattr_t* attr, int* protocol);
- int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr, int protocol);
  - mogelijke waarden voor protocol:
    - PTHREAD_PRIO_NONE
    - PTHREAD_PRIO_INHERIT
    - PTHREAD_PRIO_PROTECT

Bij een mutex weet je wel wie de pthread_mutex_unlock gaat doen!

POSIX naam voor ICPP

- int pthread_mutexattr_getprioceiling(const pthread_mutexattr_t* attr, int* prioceiling);
- int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr, int prioceiling);

DE HAAGSE HOGESCHOOL 180

Huiswerk

- Boek: H11
  - 11.8 en 11.9 bestuderen.
  - Opgaven 5 en 6 maken.

Tentamenopgave

- Een programma bestaat uit 5 threads T1 t/m T5. Deze threads gebruiken 4 gedeelde resources R1 t/m R4. Elke resource is mutual exclusive beveiligd met een mutex M1 t/m M4. Deze mutexen gebruiken het **priority inheritance protocol**. Als een thread een resource toegewezen heeft gekregen dan kan deze thread de resource voor een bepaalde maximale tijd bezetten. In de onderstaande tabel staat *k* voor het nummer van de resource en *C(k)* voor de maximale tijd dat resource *k* bezet kan worden.

k	*C(k)*
1	20
2	15
3	10
4	5

Tentamenopgave

- In de volgende tabel staat aangegeven welke resources elke thread gebruikt. In de onderstaande tabel staat *i* voor het nummer van de thread, *T(i)* voor de periodetijd van thread *i* en *C(i)* voor de maximale executietijd van thread *i*. Gegeven is dat de deadline van elke taak gelijk is aan zijn periodetijd.

i	*T(i)*	*C(i)*	Gebruikt
1	100	50	R1, R3
2	200	45	R2, R3, R4
3	350	20	R2
4	300	15	-
5	400	25	R1

Alle gegeven tijden zijn in ms.

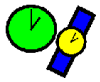
Tentamenopgave

- A. (5)
  Bepaal de prioriteit van de 5 processen als RMPA (**Rate** Monotonic Priority Assignment) wordt gebruikt. De hoogste prioriteit is 5 en de laagste prioriteit is 1.

- B. (15)
  Bereken voor elke thread *i* de blocking time *B(i)*.

- C. (15)
  Bereken voor thread 1, 3 en 5 of de deadline wordt gehaald en geef, indien de deadline wordt gehaald, de response tijd *Ri*.

Tentamenopgave



k	C(k)
1	20
2	15
3	10
4	5

i	T(i)	C(i)	Gebruikt
1	100	50	R1, R3
2	200	45	R2, R3, R4
3	350	20	R2
4	300	15	-
5	400	25	R1

i	P(i)	T(i)	C(i)	Gebruikt
1	5	100	50	R1, R3
2	4	200	45	R2, R3, R4
4	3	300	15	-
3	2	350	20	R2
5	1	400	25	R1

DE HAAGSE HOGESCHOOL 185