TDIU11 – Föreläsning 2

Schemaläggning

Filip Strömbäck

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

Varför multiprogrammering?

Exempel: Skriv innehållet i en fil på skärmen (`cat`):

Onödigt att CPU (och resten av systemet) måste vänta. Vi nyttjar tiden genom att köra andra trådar!

Processhantering

UNIX (Linux, MacOS)

- `fork()`
- `exec(program, ...)`
- `posix_spawn(program, ...)`
- `waitpid()`

Windows NT

- `CreateProcess(program, ...)`
- `WaitForSingleObject()`

Trådhantering

UNIX (Linux, MacOS)

- `pthread_create()`
- `pthread_join()`
- `pthread_detach()`
- (`clone()`)

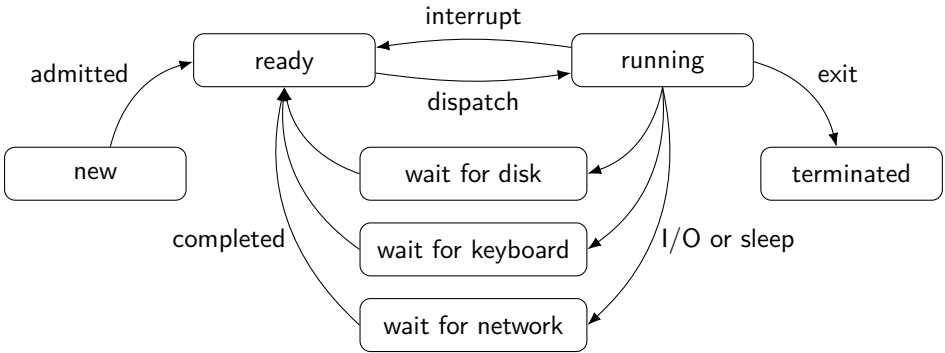
Windows NT

- `CreateThread()`
- `WaitForSingleObject()`

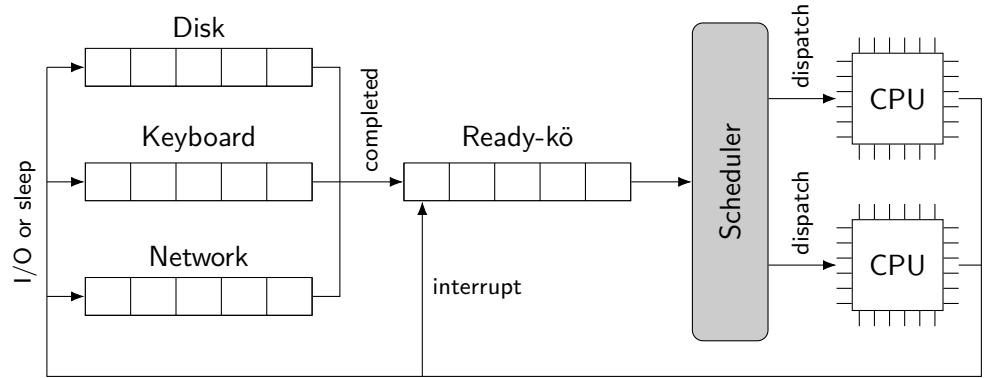
C++

- `std::thread`
- `thread.join()`
- `thread.detach()`

Trådens livstid – Tillståndsdiagram (thread/process-state diagram)



Hantering av trådar i Kernel



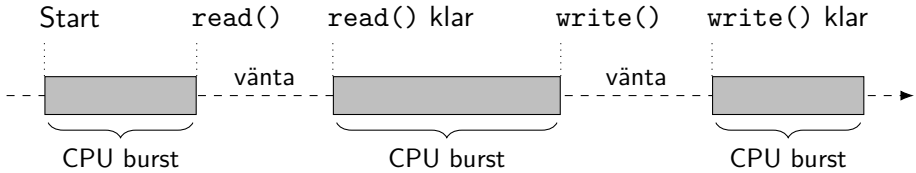
LINKÖPING UNIVERSITY

Typer av schemaläggare

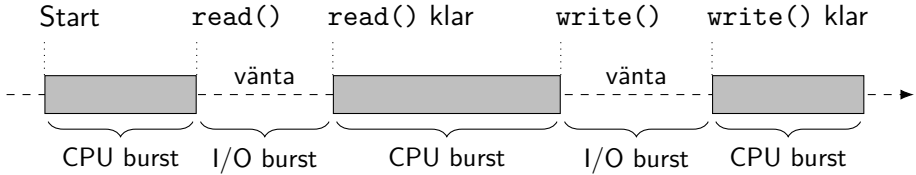
- Korttid: Scheduler
  Arbetar *under körning* av processer trådar. Ser till att CPU har något att göra så mycket som möjligt. Måste därför ta beslut snabbt.
- Långtid: Job scheduler
  Körs ibland, och beslutar vilka job som ska startas. Ser till att korttids-scheduler har ”lagom” mycket att hantera, och att systemet inte får slut på RAM.

Vad behöver schemaläggaren veta?

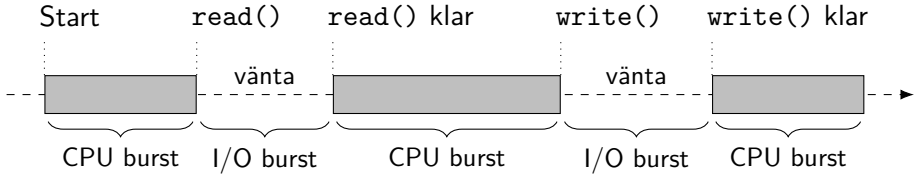
Vad behöver schemaläggaren veta?



Vad behöver schemaläggaren veta?

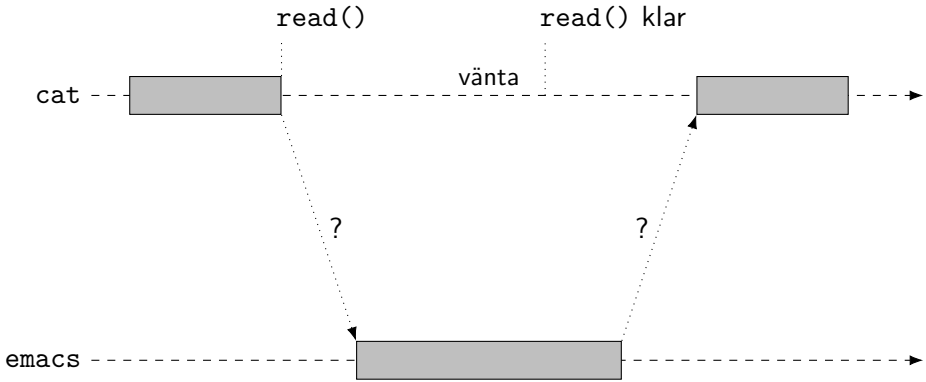


Vad behöver schemaläggaren veta?

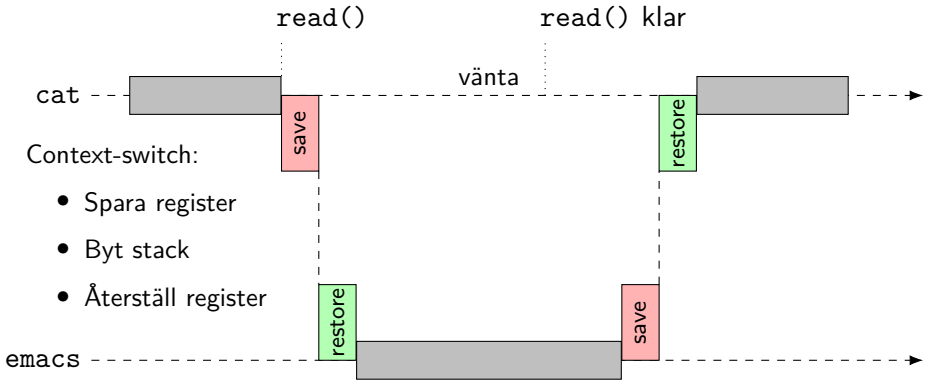


- CPU burst, följt av I/O burst
- Vi schemalägger CPU ⇒ CPU burst kan betraktas separat
- Tänk: bryr oss bara om *ready-kö*

Hur byter vi mellan olika processer?



Hur byter vi mellan olika processer?



Hur jämför vi schemaläggare?

CPU utilization:	Hur stor andel av CPU-tid används (i %)?
Throuhgput:	Hur många processer per tidsenhet blir klara?
Turnaround time:	Hur lång tid tar det för en process att bli klar?
Waiting time:	Hur lång tid har en process fått vänta ofrivilligt?
Response time:	Hur lång tid tar det för en process att reagera på indata?

Även *average* turnaround time och *average* waiting time.

LINKÖPING UNIVERSITY

FIFO (First In First Out)/FCFS (First Come First Served)

- Låt processer köra i ordningen de lades in i *ready*-kö

Exempel:

Tråd	Start	Burst
A	0	10
B	0	5
C	2	1

SJF (Shortest Job First)

- Kör det jobbet som är kortast först!
- Optimal med avseende på *average waiting time*
- Problem: vi måste kunna se in i framtiden

Exempel:

Tråd	Start	Burst
A	0	10
B	0	5
C	2	1

Prioritetsbaserad schemaläggning

- Ge varje process en prioritet!
- Problem: risk för *starvation*, kan lösas med *ageing*

Exempel:

Tråd	Start	Burst	Prio
A	0	10	3
B	0	5	1
C	2	1	2

När körs schemaläggaren?

Schemaläggaren är en bit kod som måste köras på CPU.

Vad händer om tråden/processen aldrig terminerar eller väntar?

LINKÖPING UNIVERSITY

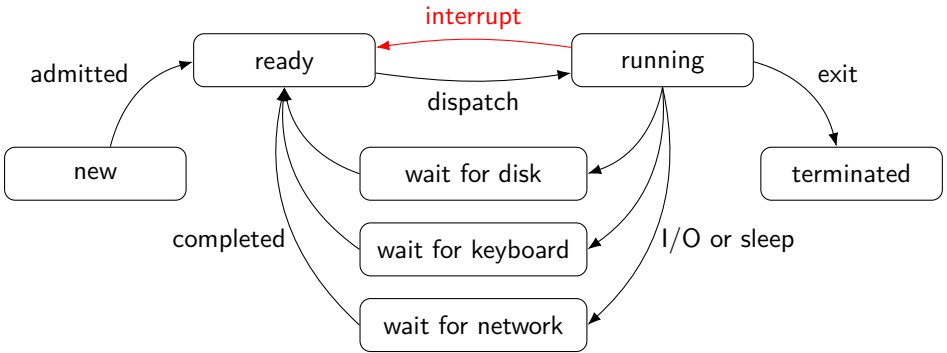
Preemption

Vi tillåter OS att avbryta körande tråd/process

- När *time-quantum* är slut, med hjälp av timeravbrott
- När andra trådar/processer blir redo (nya processer, eller om de väntat klart)

Vi får då *preemptive schedulers*

Tillståndsdiagram



Round-robin – FIFO/FCFS med preemption

- Som FIFO, men låt nästa process köra efter time-quantum är slut
- Time quantum måste vara ”lagom”
  - Litet $\Rightarrow$ mycket overhead
  - Stort $\Rightarrow$ ingen preemption

Exempel:

Tråd	Start	Burst
A	0	10
B	0	5
C	2	1

Time quantum $= 2$

Shortest remaining time first – SJF med preemption

- Som SJF, men nya trådar/processer kan avbryta körande process.

Exempel:

Tråd	Start	Burst
A	0	10
B	0	5
C	2	1

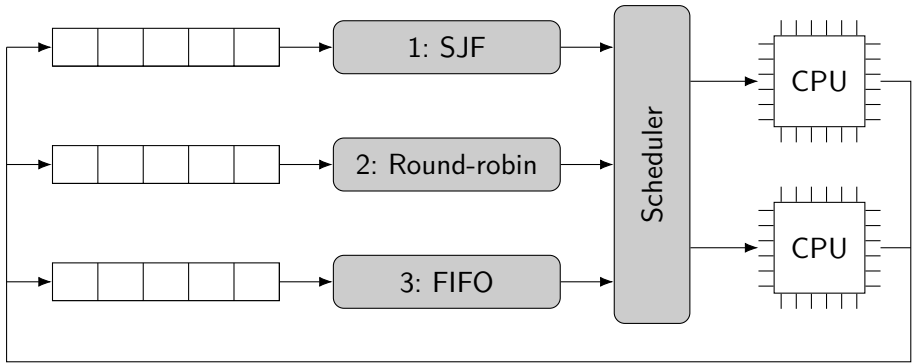
Prioritetsbaserad schemaläggning

- Som i ej-preemptive variant, men nya trådar/processer kan avbryta körande process.
- Om flera med samma prioritet, round-robin mellan dessa.

Exempel:

Tråd	Start	Burst	Prio
A	0	10	3
B	0	5	1
C	2	1	2

Multilevel scheduling



Hur gör ”riktiga” operativsystem?

- Hybridapproach, ofta prioritetsbaserad i grunden
- Preemptive med tidskvantum i storleksordning 10 ms
- Prioritet justeras dynamiskt (ex. Windows ger högre prioritet till aktiv process)
- Se artikel 1!

LINKÖPING UNIVERSITY

Exempel på redovisning

Exempelproblem baserat på schemaläggning:

Schemalägg följande processer enligt *preemptive priority scheduling*:

	Arrival	**Burst**	**Prio**
P_1	0	5	4
P_2	4	1	1
P_3	2	3	2
P_4	3	3	2

Time quantum $= 2$

Beräkna också *average waiting time*

Filip Strömbäck

www.liu.se

LINKÖPING UNIVERSITY