Schedulering med en processor

Mittuniversitetet
MID SWEDEN UNIVERSITY

Lektion 6, Schedulering med en processor.

Kapitel 9 (sid. 393-422)

Ni har ju redan lärt er vad schedulering är för något. Nu ska vi gå lite djupare in i det och detta kapitel handlar om schedulering med en processor.

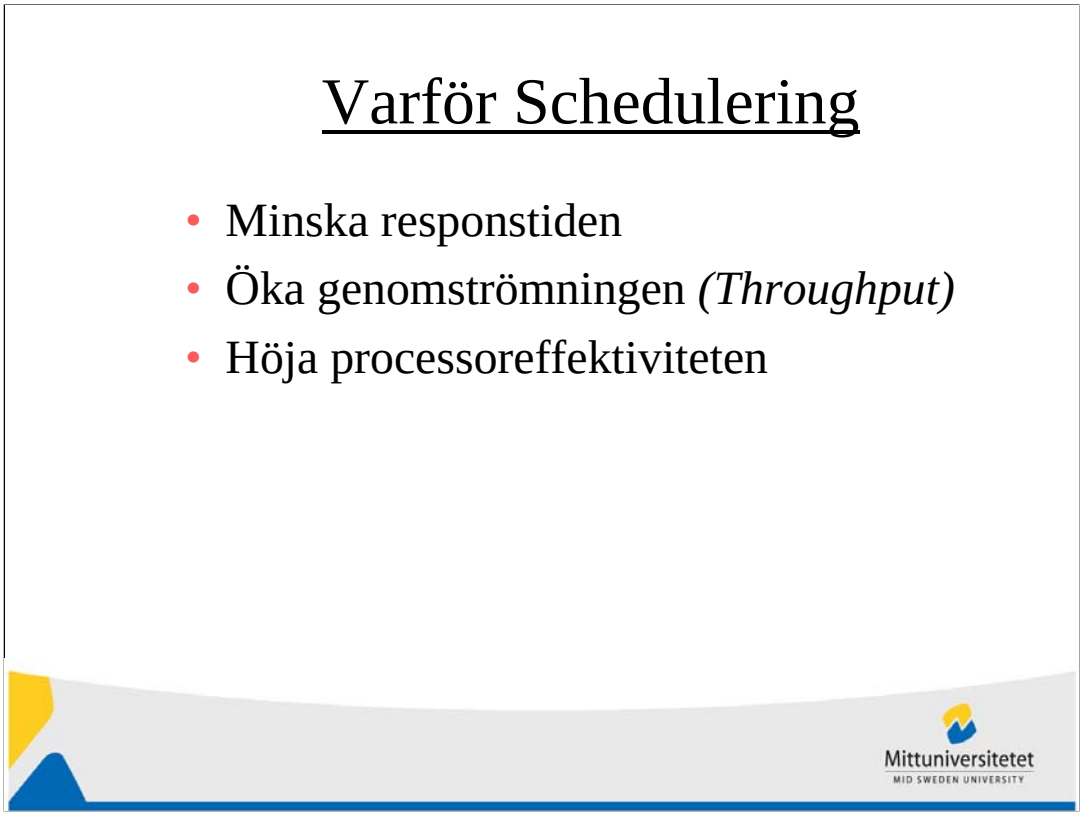


Bild 2: Varför är schedulering viktigt? Jo den syftar till att hantera ett par uppgifter som ställer olika krav:

• • Minska responstiden, dvs. se till att alla processer inom rimlig tid får en chans att exekvera. Så att jag får resultat på t ex en inmatning.

• • Öka genomströmningen (*troughput*), dvs. se till att så mycket arbete som möjligt utförs. Jämför med troughput inom datakommunikation.

• • Effektiv användning av processorn. Minskad responstid kan ju stå i konflikt med ökad genomströmning.

Olika Typer Av Schedulering

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

Mittuniversitetet
MID SWEDEN UNIVERSITY

Schedulering på olika tidsbas

Bild 3-4: Boken använder fyra olika typer/nivåer av schedulering. Dessa är kopplade till den processtatusmodell som visas i Figure 3.8b sidan 123:

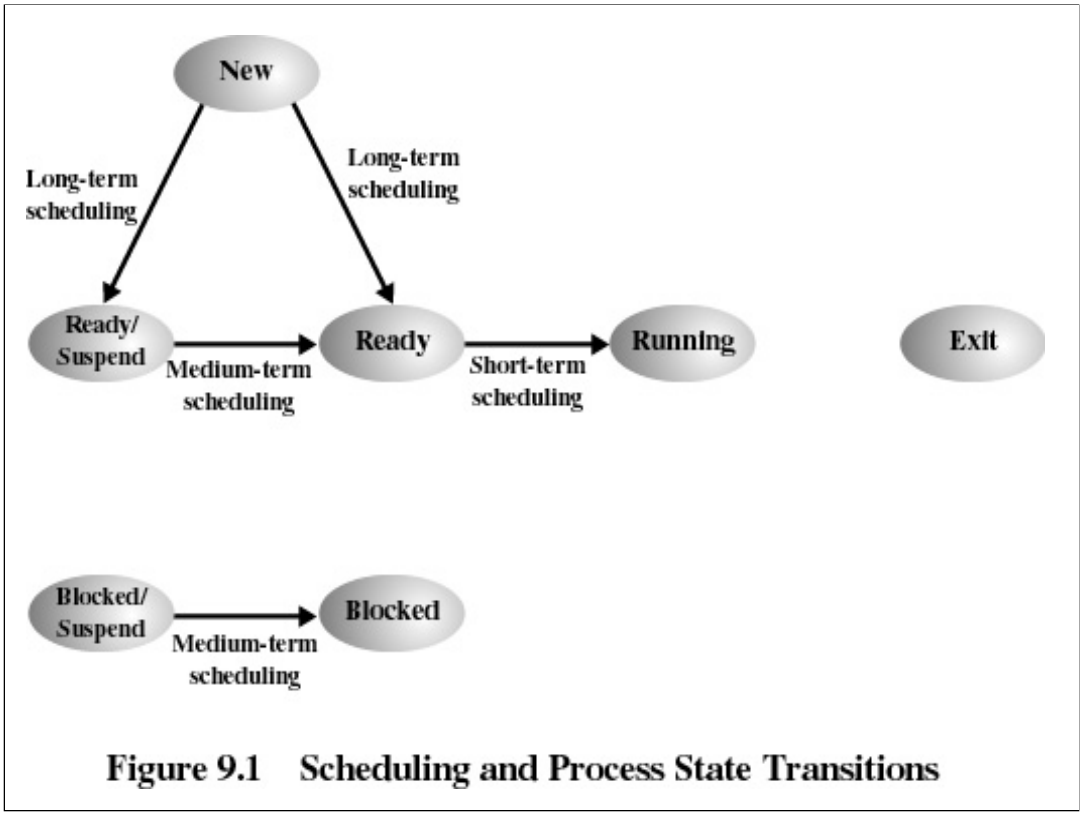


Figure 9.1 Scheduling and Process State Transitions

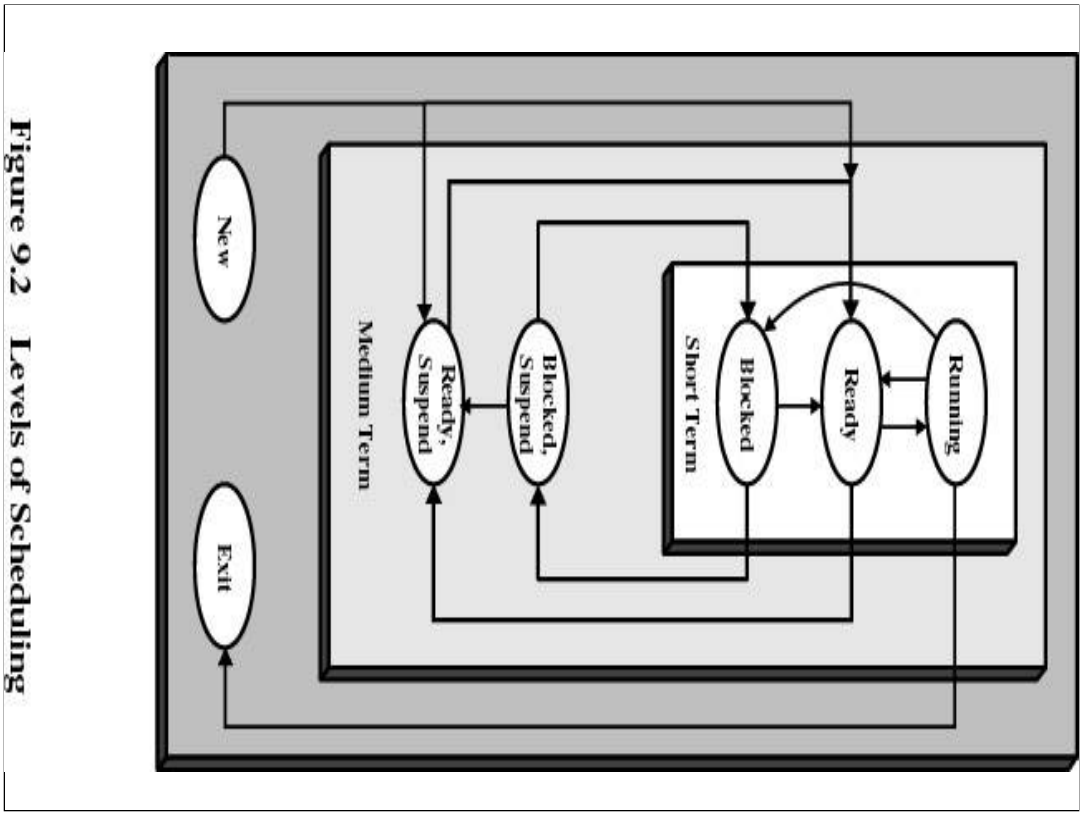
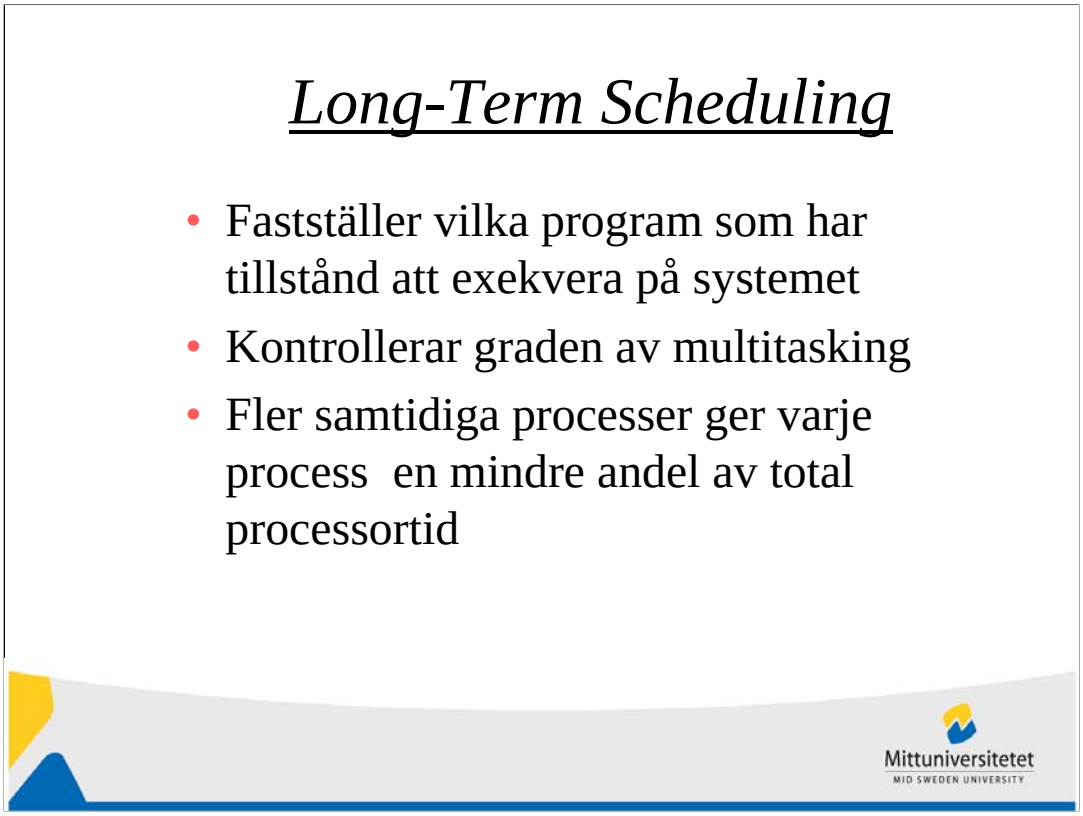


Figure 9.2 Levels of Scheduling

Bild 5: visar hur man har ritat om Figure 3.8b i en modell där scheduleringsnivåerna ligger som skal.

Sammanfattningen: scheduleringen ska hantera köer på ett sådant sätt att kötiden minimeras och systemet blir så effektivt som möjligt.



• • *Bil d 6: Long-Term scheduling*, som hör ihop med *Admit*. Vilka nya processer ska plockas in i ready-poolen. Här bestäms graden av multiprogrammering.

Long-Term scheduling

Alla program som startas hamnar ju först i en kö för Long-Term schedulering. Programmet blir ju en process först när det tas in och läggs i kön för Short-Term schedulering. Vissa system kan plocka in processer i kön för Medium-Term schedulering genom att lägga dom på disk. Som om de vore utswappade.

Kön kan behandlas first-come-first-served dvs. i tur och ordning som programmen startats, men processerna kan också väjas utifrån prioritet, förväntad exekveringstid, interaktivitet eller I/O-intensitet. Effektivast är att försöka ha en mix av processer som är I/O-intensiva och de som är använder processorn mycket. Då finns det ju alltid en processorintensiv process att släppa in om andra är avbrutna av I/O-hantering

Medium-Term Scheduling

- En del av swappfunktionen
- Är nödvändig för att hantera multitasking

Mittuniversitetet
MID SWEDEN UNIVERSITY

• • *Bil d 7: Medium-Term scheduling*, som hör ihop med *Activate/Suspend*. Dvs. swapp-funktionen.

Medium-Term scheduling

Dvs. swapping har vi ju diskuterat ganska ingående i samband med minneshanteringen.

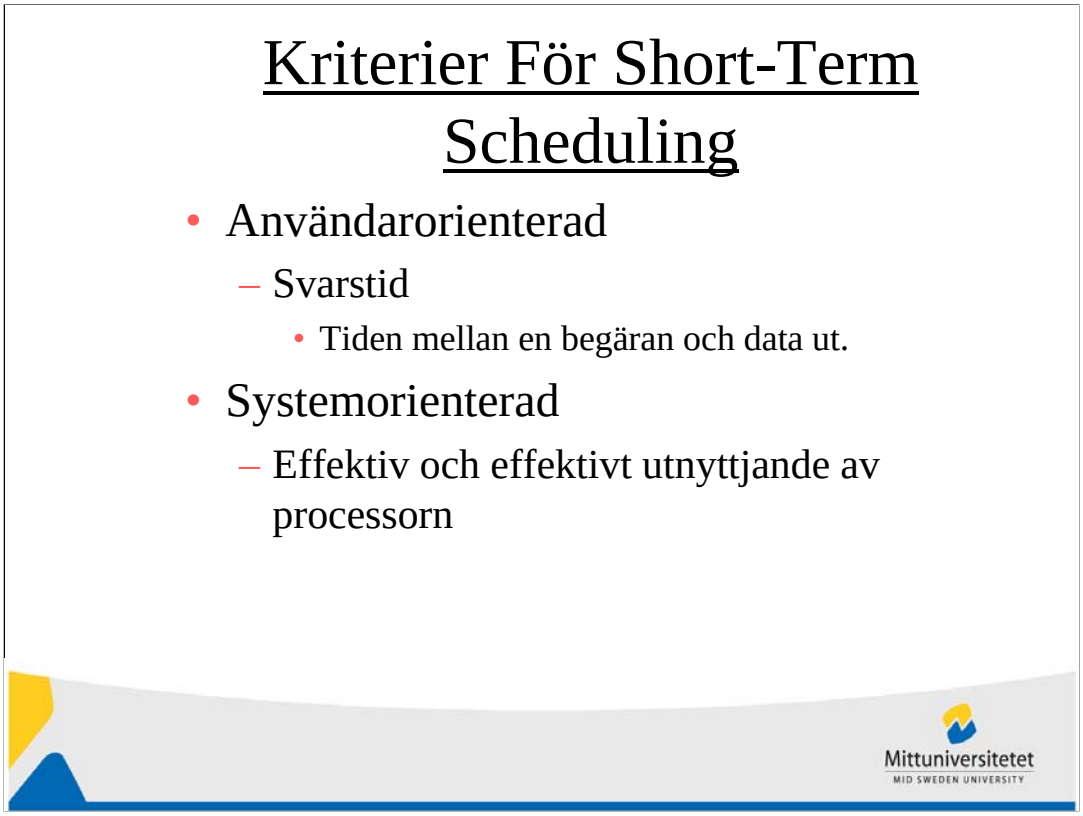
Short-Term Scheduling

- *Dispatcher*, funktionen som väljer vilken process som ska exekveras
- Den del som arbetar oftast
- Anropas vid olika händelser
  - Klockinterrupt
  - I/O-interrupt
  - Operativsystemsanrop
  - Signaler

Mittuniversitetet
MID SWEDEN UNIVERSITY

• • *Bil d 8: Short-Term Scheduling*, som hör ihop med *Dispath/Timeout*. Vilken process ska få exekvera nu?

• • (*I/O scheduling* hantering av köerna till I/O-enheterna. Handlar egentligen inte om schedulering av processorn.)



Scheduling algorithms

Resten av detta kapitel om schedulering behandlar Short-Term scheduling dvs. dispatherns arbete.

Bild 9-10: visar de konflikter som uppstår med short-term schedulering. Vi vill ha korta svarstider och hög trough-put. Problemet är att korta svarstider kräver liten time-slice för varje process, men vid liten time-slice så minskar trough-put pga. många context-switch.

För en ensam modern arbetsstation med relativt snabb processor blir inte scheduleringen kritisk. Däremot för en hårt belastad server.

Se även Table 9.2 i boken som visar olika kriterier för schedulering. Se till att ni förstår alla begrepp innan ni går vidare.

Kriterier För Short-Term Scheduling

- Prestandarelaterat
  - Kvantitativ
  - Baserad på ex. vis. svarstid och genomströmning
- Ej restandarelaterat
  - Kvalitativa
  - Förutsägbarhet

Mittuniversitetet
MID SWEDEN UNIVERSITY

Prioritering

- Scheduleringen kommer alltid att välja den process som har högst prioritet
- Använder flera lästköer för hantering av anrop med olika prioritet
- Lågt prioriterade processser kanske aldrig får exekvera *(starvation)*
  - Lösning: tillåt en process att ändra sin prioritet utifrån hur länge det var sedan den exekverades senast

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processprioritet

Bild 12: (prioritering) ska komma före *Bild 11:* (Figure 9.3) och visar att det behövs köer på alla nivåer i scheduleringen eftersom vi kan ha flera processer väntande på samma ställe.

Bild 11: för att se till att viktiga processer (interaktiva processer) hanteras före andra (ex

utskrift i bakgrunden) så tilldelas processerna olika prioritet. Vi kan då enligt Bild 13 behöva flera köer, en för varje prioritetsnivå. Ett problem med prioritet är risken för att någon process aldrig får exekvera (*starvation*).

Resten av kapitlet behandlar olika scheduleringsstrategier för att uppfylla de kriterier jag valt (utifrån vad datorn arbetar som, Webbserver/mailserver etc).

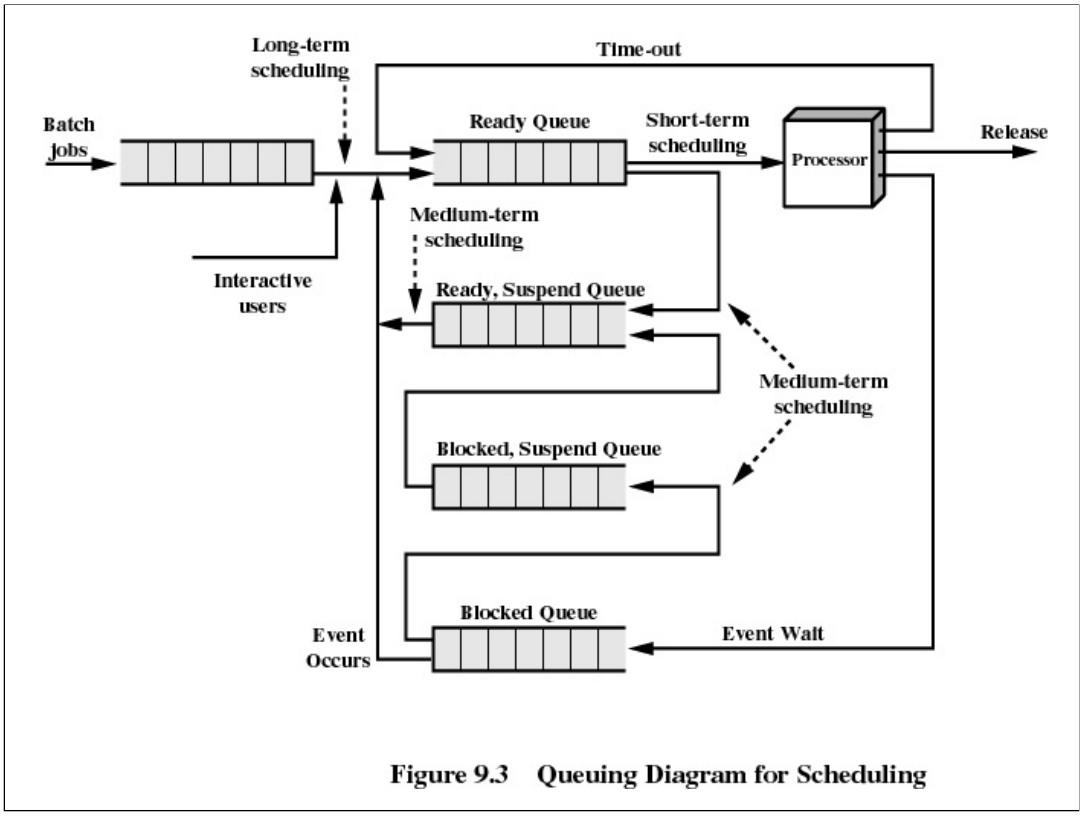


Figure 9.3 Queuing Diagram for Scheduling

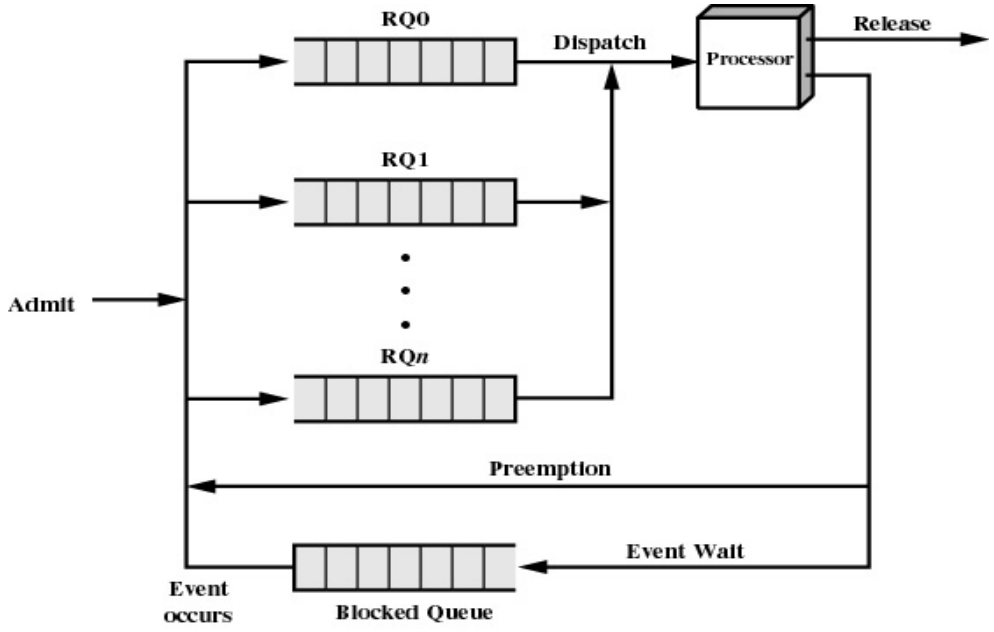


Figure 9.4 Priority Queuing

Scheduleringsstrategier

- *Nonpreemptive*
  - Den process som exekverar kommer att få fortsätta tills den är klar eller blockeras p.g.a. I/O-anrop
- *Preemptive*
  - Exekverande process kan bli avbruten och överförd till *Ready state* av operativsystemet (timesharing)
  - Ger en bättre service till alla. Eftersom en process inte kan lägga beslag på processorn lång tid

Mittuniversitetet
MID SWEDEN UNIVERSITY

***Preemtive* / *non preemtive* schedulering**

Bild 14: visar den viktigaste skillnaden mellan *preemtive* och *non preemtive* schedulering. Vid preemtive kan en process avbrytas av OS när dess time-slice är slut.

Scheduleringsalgoritmer

Detta avsnitt börjar på sidan 400, ”Alternative Scheduling Policies”.

Börja med at läsa/lära om respektive strategi innan ni försöker sätta er in i alla tabeller och figurer i boken.

Exempel På Schedulering

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

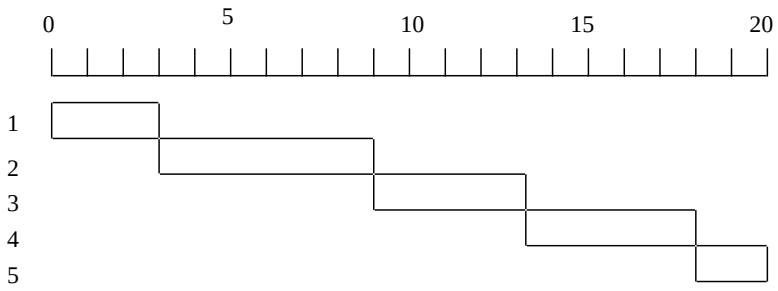
Mittuniversitetet
MID SWEDEN UNIVERSITY

Bild 15: visar förutsättningen för resonemangen i de följande bilderna.

En process *arrival time* är alltså inte när den början exekvera utan när processen kommer till readykön. *Service time* är den totala tid som processorn behöver använda processorn innan den är klar. Ingen av processerna i exemplet har något I/O-anrop så de kör tills de är klara. Om de inte blir preemtived.

Samtliga strategier (policies) rör short-term schedulering, utan prioritering (utom den sista)

First-Come-First-Served (FCFS)



- Varje process placeras i Readykön
- När exekverande process slutar så kommer nästa process the i Readykön att väljas (äldsta)

Mittuniversitetet
MID SWEDEN UNIVERSITY

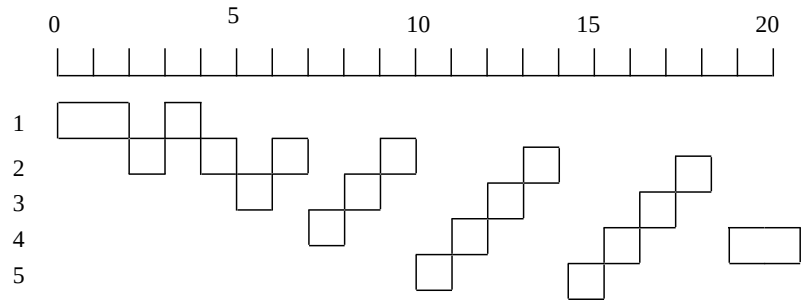
First-come-first-served

Bild 16-17: beskriver FCFS. Varje process placeras i kön efterhand som de kommer och får exekvera tills de är klara eller har ett I/O-anrop (non preemtive).

First-Come-First-Served (FCFS)

- Trots att en process skulle kräva liten processortid kan den få vänta länge
- Favoriserar processer som endast använder CPU
  - I/O-intensiva processer får vänta tills de andra är klara

Round-Robin



- Använder timerkontrollerad preemption
- En viss maximal tid är avsatt som en process får exekvera utan att avbrytas

Mittuniversitetet
MID SWEDEN UNIVERSITY

Round-Robin

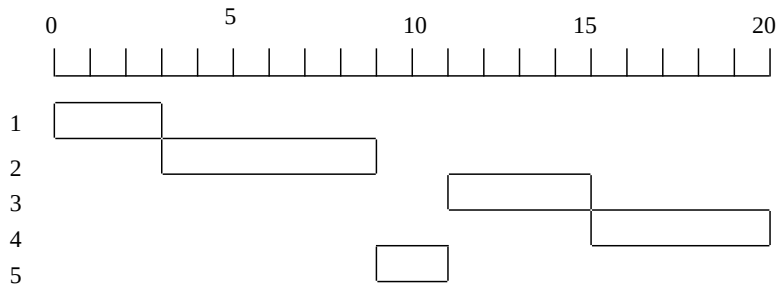
Bild 18-19: beskriver Round-Robin (preemtive). När tiden är slut för en process tas nästa in. Alla kommer i tur och ordning att få exekvera.

Round-Robin

- Klockinterrupt genereras med jämna mellanrum
- När ett interrupt kommer så placeras exekverande process i readykön och nästa process i kön får exekvera
- Kallas också *time slicing*

Mittuniversitetet
MID SWEDEN UNIVERSITY

Shortest Process Next



- *Nonpreemptive*
- Processen kortast förväntad exekveringstid väljs
- Gör att korta processer prioriteras

Mittuniversitetet
MID SWEDEN UNIVERSITY

Shortest Process Next

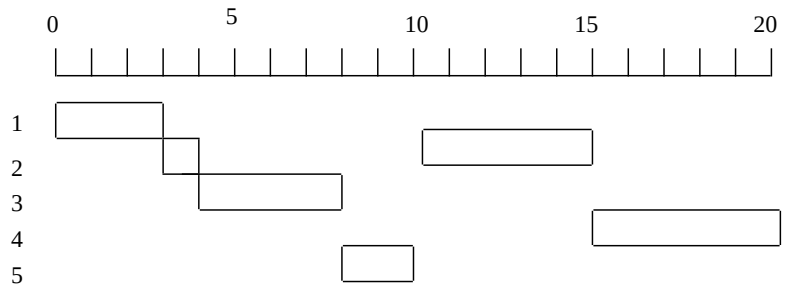
Bild 20-21: FCFS tar ju inte hänsyn till vilken process som tar minst tid att exekvera, en kort interaktiv process kan få vänta medan ett stort beräkningsjobb körs. För att öka throughput kan varianter tänkas.

SPN försöker göra just det. Problemet är att om vi har många ”korta” processer som hela tiden kommer, kan långa råka ut för starvation.

Shortest Process Next

- Svårare att förutsäga totaltid för långa processer
- Om förväntad exekveringstid för en process inte stämmer kan operativ-systemet avbryta den
- Möjlighet att långa processer råkar ut för *starvation*

Shortest Remaining Time



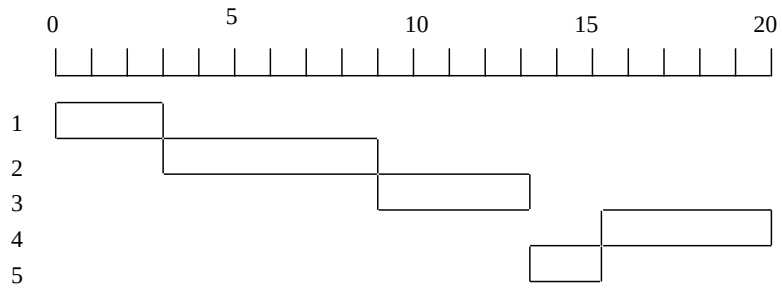
- *Preemptive* variant av *shortest process next*
- Kvarvarande exekveringstid måste uppskattas

Mittuniversitetet
MID SWEDEN UNIVERSITY

Shortest Remaining Time

Bild 22: En variant till denna är då SRT som är preemtive *”at arrival”* dvs. om en ny process kommer som har kortare beräknat tid än den som för ögonblicket exekverar (detta måste uppskattas) så avbryts den exekverande processen.

Highest Response Ratio Next (HRRN)



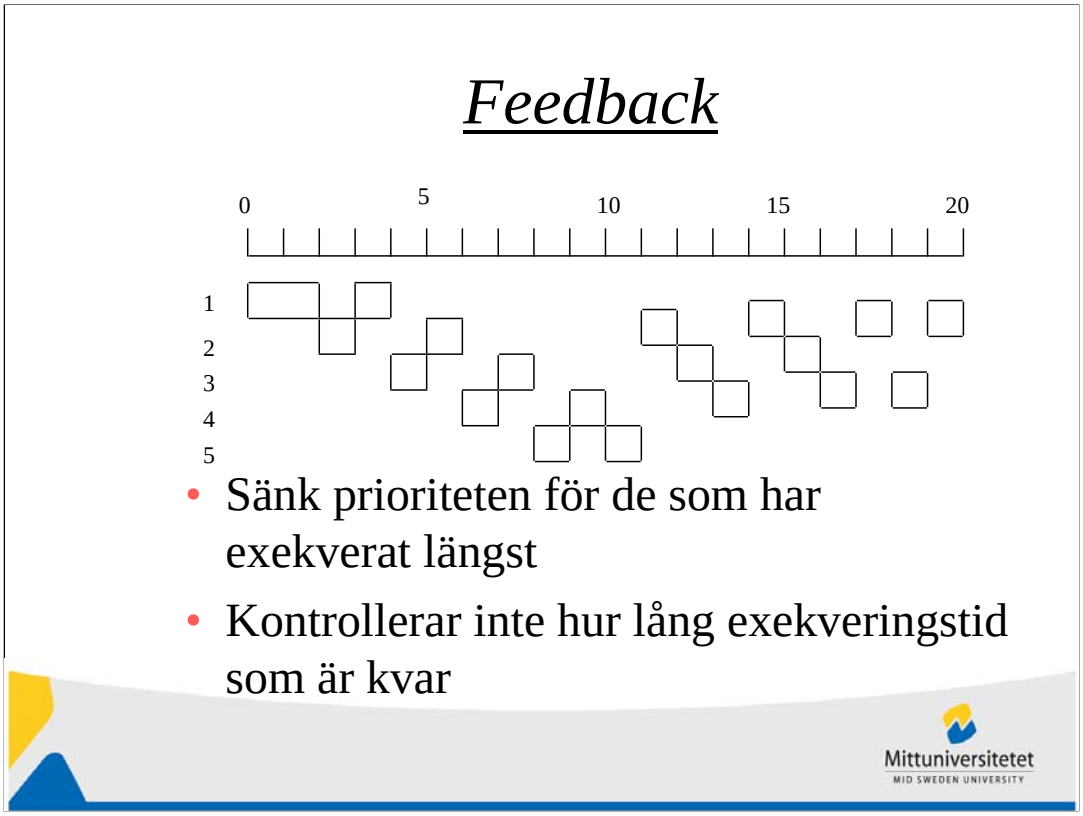
- Välj nästa process med minsta

$$\frac{\text{Väntad tid + förväntad exekveringstid}}{\text{förväntad exekveringstid}}$$

Mittuniversitetet
MID SWEDEN UNIVERSITY

Highest Response Ratio Next

Bild 23: HRRN kommer att välja den process som har väntat längst i förhållande tills sin förväntade totala exekveringstid. Långa processer kan få vänta längre och korta snabbas på.



Feedback

Bild 24-25: Om vi inte vet eller kan beräkna den förväntade totala exekveringstiden så kommer SPN, SRT och HRRN inte att kunna användas. Vi använder då en feedback-metod där processens tidigare aktiviteter får avgöra scheduleringen.

Feedback-strategin är preemtive och använder prioritetsköer. En process som blir avbruten får sin prioritet satt utifrån hur länge den har exekverat (hur många gånger). Nya korta processer favoriseras framför äldre, längre. Internt i varje prioritetskö används enkel FCFS.

Även här finns risk för att stora processer råkar ut för starvation, om det hela tiden kommer nya korta. Ett sätt att klara det är att öka längden på timeslice för processer med lägre prioritet.

Som ni ser så är inte schedulering lätt, heller.

Jämförelse mellan algoritmerna

Nu är det dags att ta itu med *Table* 9.3 sid. 401 som jämför de olika strategierna samt sidorna 402 och 403. Är allt solklart så gå vidare till sidorna 414-419. Ni behöver inte kunna rita upp några diagram, men kunna läsa och tolka.

Schedulering av processer med trådar

För att göra allt ännu svårare så kan ju en applikation bestå av flera processer och varje process av flera trådar (trådar är delar av en process som kan scheduleras fristående). Som användare har jag ju inte någon nytta av ett en enskild tråd i min applikation blir klar tidigt, om de andra blir hängande. Scheduleringen måste ta hänsyn till detta och se till att hela applikationen förs framåt.

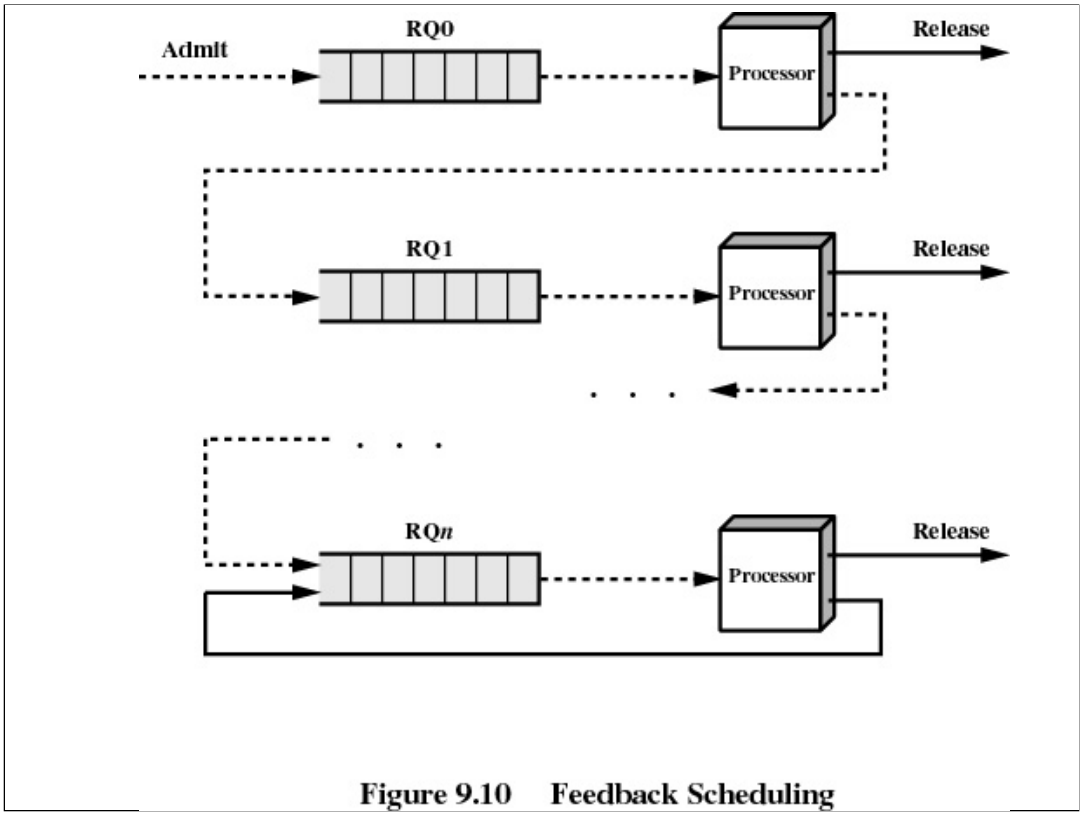


Figure 9.10 Feedback Scheduling

Fair-Share Scheduling

- Användarnas applikationer körs som en samling av processer (trådar)
- Användaren är intresserad av applikationens prestanda, inte enskilda trådars
- Schedulering måste baseras på hela applikationen

Mittuniversitetet
MID SWEDEN UNIVERSITY

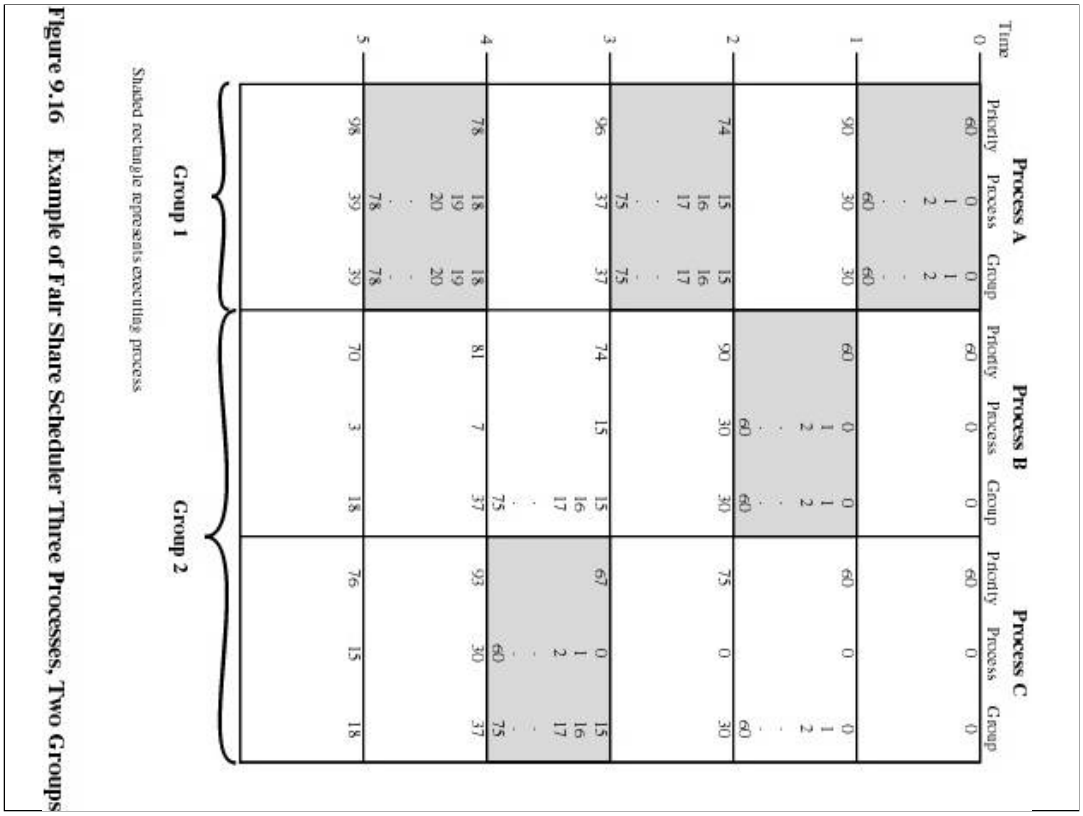


Figure 9.16 Example of Fair Share Scheduler Three Processes, Two Groups

Scheduleringen kan dessutom behöva ta hänsyn till att olika användare/grupper av användare, ska ha olika prioritet. Här används *Fair-Share Schedulering* fastna inte i Bild 27.

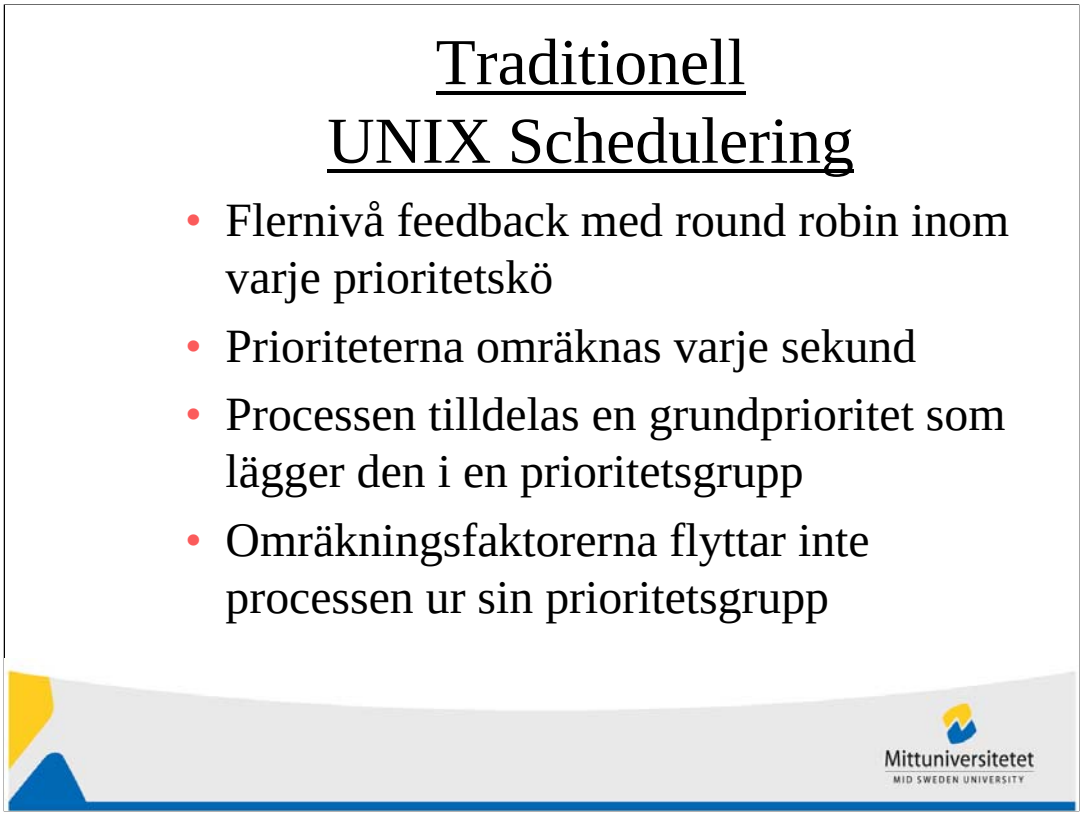


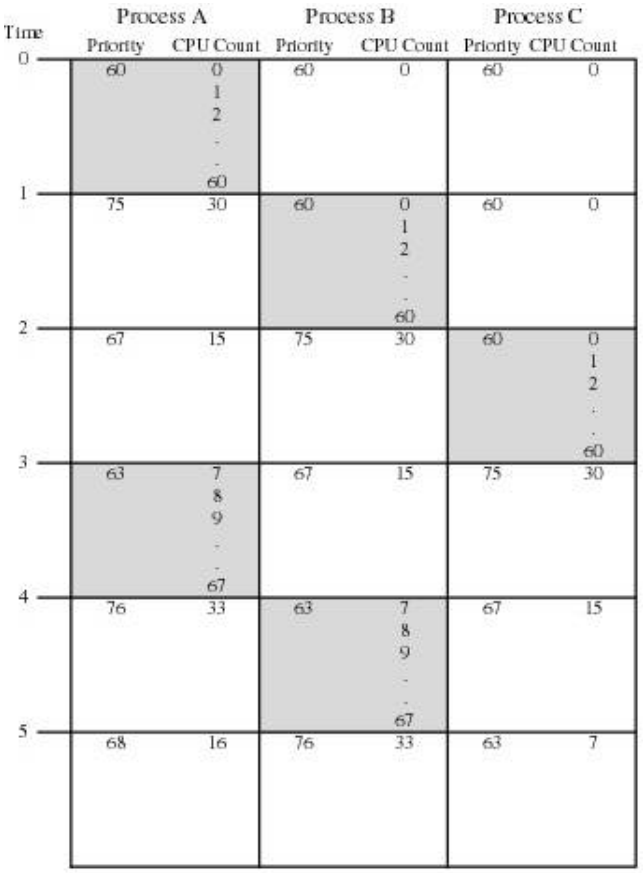
Bild 28-30: Raskt vidare till ett praktiskt exempel, UNIX-schedulering. Intresserade läser i boken.

Windows 2000 använder: preemtive schedulering, olika prioritetsnivåer med round-robin inom varje nivå och för vissa nivåer dynamisk prioritetshantering beroende på trådarnas aktivitet. Sidan 465ff i boken för de som vill läsa mera.

Prioritetsgrupper

- Från hög till låg
  - *Swapper*
  - Blockvis I/O-access
  - Filhantering
  - Teckenbaserade I/O-enheter
  - Användarprocesser

Mittuniversitetet
MID SWEDEN UNIVERSITY



Shaded rectangle represents executing process

Figure 9.17 Example of Traditional UNIX Process Scheduling