Processer och Processkontroll

Ett centralt begrepp när vi diskuterar operativsystem (OS) är process. OS uppgift är att samordna exekveringen av de processer som för ögonblicket körs. En process skapas t ex när ett program startas. Processen är den aktivitet som sker när ett program körs.
Så en process aktivitet kan spåras utifrån de instruktioner som exekveras.
Det är ibland svårt att läsa texten när processer och processorn används i samma mening.
Om något verkat konstigt så läs noga vad som står

Operativsystemets Uppgifter

- Samordna exekveringen av flera processer för att uppnå maximalt processorutnyttjande, men med rimlig responstid
- Tilldela resurser till processer
- Understödja “Interprocess Communication” och ge användaren möjlighet att skapa processer

Mittuniversitetet
MID SWEDEN UNIVERSITY

Process

- Kallas även *task*
- Exekvering av ett individuellt program, d v s en följd av instruktioner
- Följden kallas *trace*
  - Den följd av instruktioner som exekveras

Mittuniversitetet
MID SWEDEN UNIVERSITY

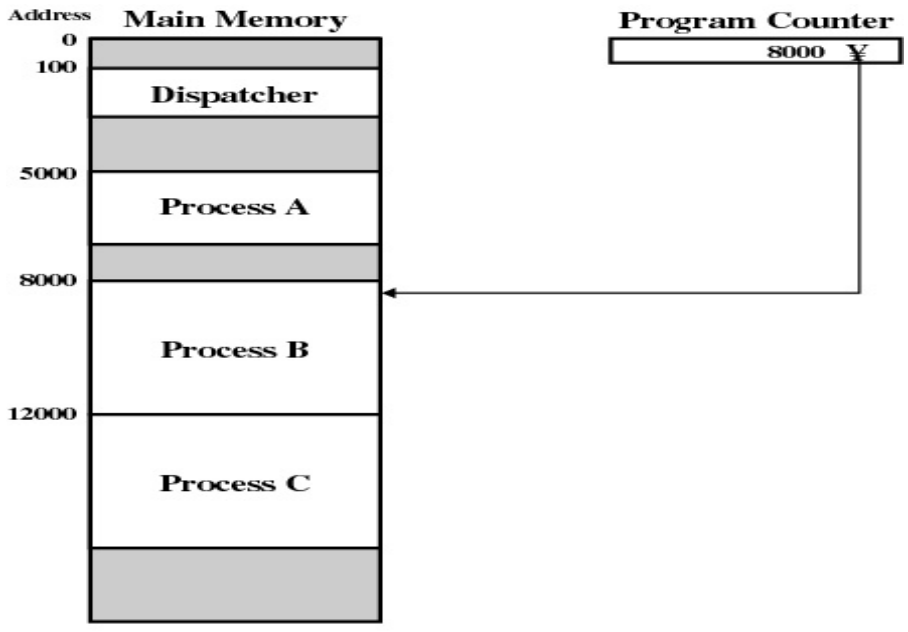


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

Dispatcher

Bild 3: visar hur det i minnet är inläst ”Dispatcher” och tre processer A, B, C.

Dispatchern är den del av OS som ser till att alla processer får tillgång till processorn.

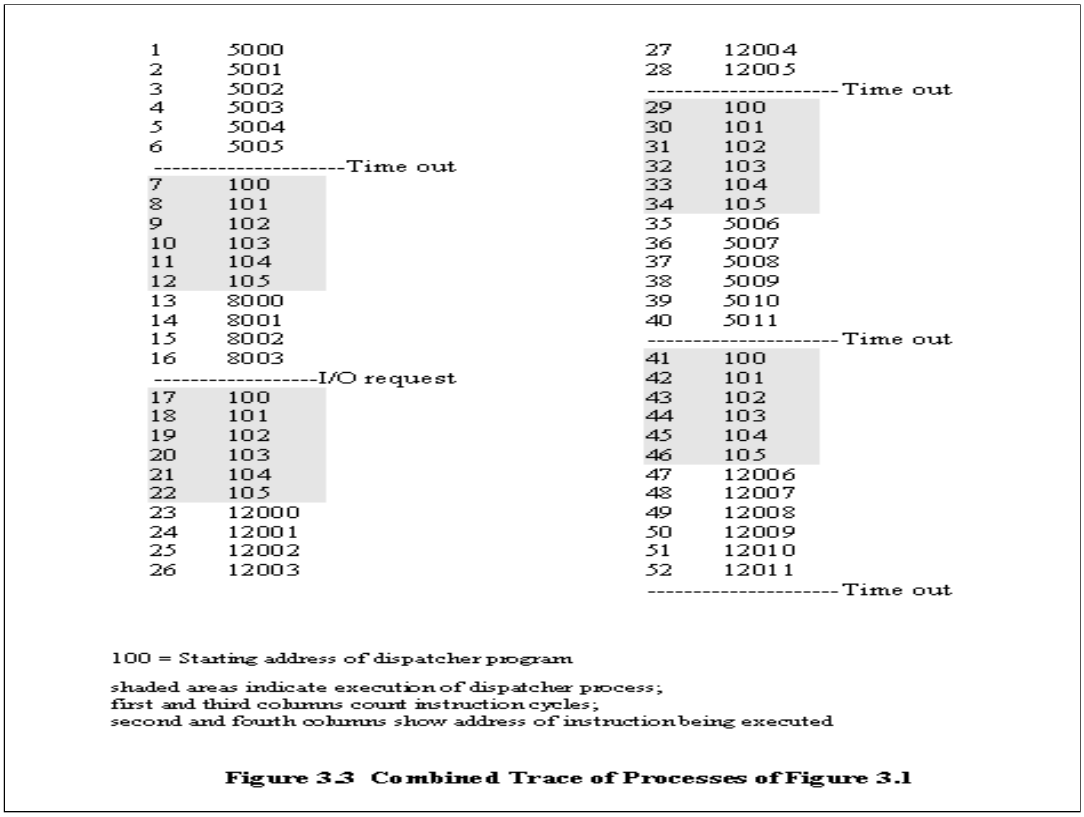
I PC ligger adressen till starten av B.

Så antagligen har processen B just laddats och ska börja exekvera i processorn

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.2 Traces of Processes of Figure 3.1



1	5000	27	12004
2	5001	28	12005
3	5002	----------	Time out
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------	Time out	32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------	Time out
16	8003	41	100
----------	I/O request	42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------	Time out

100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

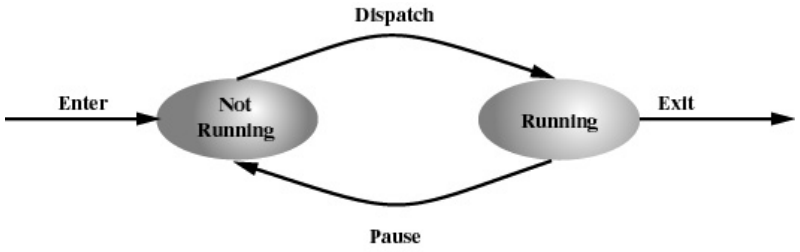
Figure 3.3 Combined Trace of Processes of Figure 3.1

Preemptive multi-tasking

Bild 6: Vid preemptive multi-tasking får alla processer en begränsad exekveringstid i processorn innan OS avbryter denna och nästa får fortsätta.
En exekverande process kan alltså avbrytas av att tiden är slut (time out) eller av att ett I/O-anrop sker från processen (ex vis diskaccess).
Eftersom I/O-anrop är c:a 1000 ggr långsammare än minnesaccess så avbryts processen och nästa får ta vid tills I/O-anropet är klart.
Vid time out eller I/O-anrop exekveras dispatcherns kod som sköter context switch och väljer nästa process från en kö.
En process kan i den enklaste modellen befinna sig i ett av två tillstånd, Running/Not running.

Processmodeller: två tillstånd

- Process befinner sig i ett av två tillstånd
  - *Running*
  - *Not-running*



(a) State transition diagram

Mittuniversitetet
MID SWEDEN UNIVERSITY

Skapandet av en process

Bild 7: visar hur en process skapas och får tillståndet *not running*.

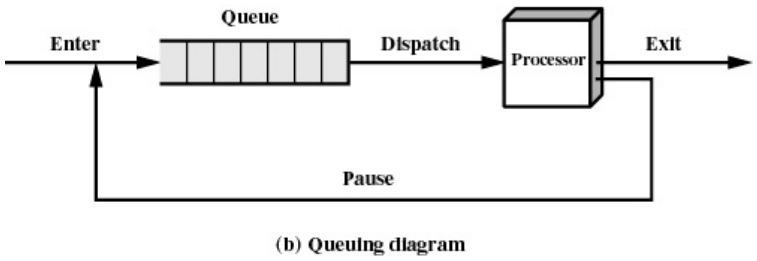
Om den väljas av dispatchern blir den *running*.

Ett avbrott, time out eller I/O flyttar tillbaka den till not running.

När den har exekverat färdigt så blir dess tillstånd *exit*,

processen existerar inte längre.

Processer som är *Not-Running* ligger i en kö



(b) Queuing diagram

Mittuniversitetet
MID SWEDEN UNIVERSITY

Bild 8: den process som väntar på att få exekvera placeras i en kö som dispatchern plockar ur.

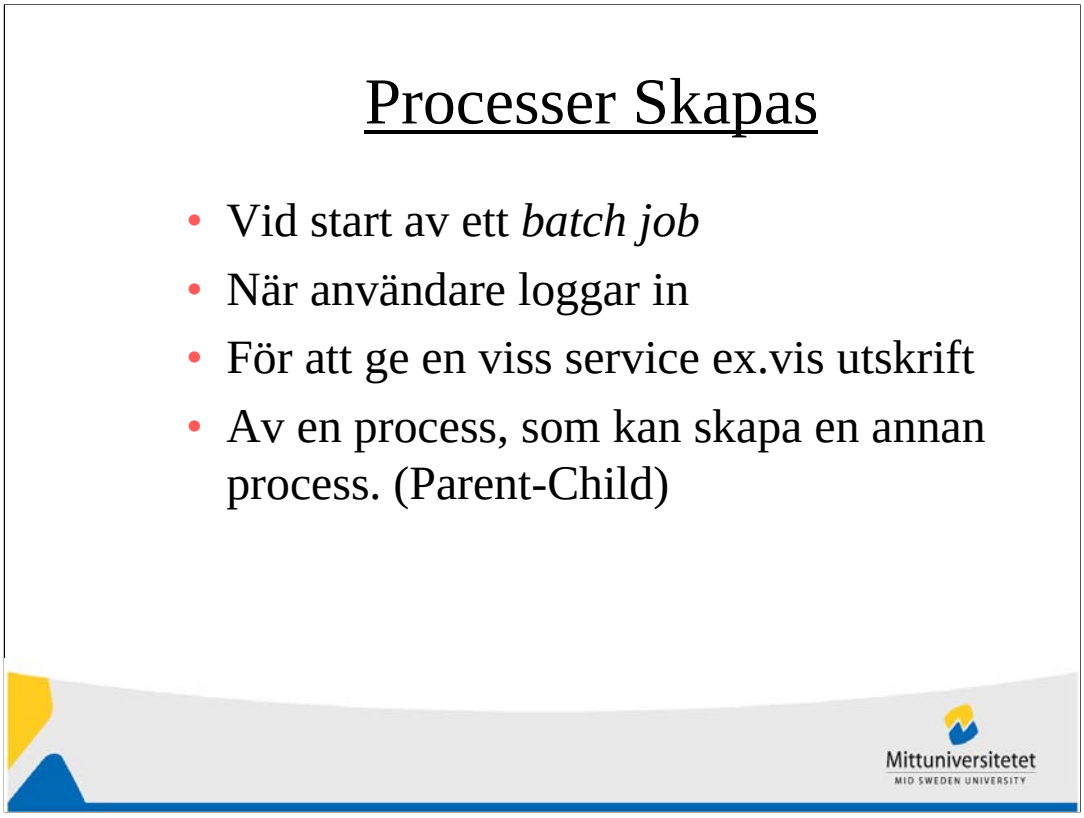


Bild 9: Processer skapas vid ett antal tillfällen:

• Ett batchjobb (ett antal instruktioner). Ex vis bat-filer körs.

• När jag som användare loggar in startas ett antal processen för att hantera min resursaccess

• En process kan själv starta egna processer (dessa blir child-process till den parentprocess som startade den. I UNIX finns från början endast en parent-process som sedan ger upphov till alla efterföljande).

• Olika typer av services kan starta som egna processer.

Processer avslutas

- När ett *Batch job* innehåller en *Halt* instruktion
- När användare loggar ut
- När en applikation avslutas
- Vid olika fel

Avslutande av processer

Bild 10-12: Processer avslutas när:

• ett batchjobb är slut.

• Inloggningsprocesserna avslutas naturligtvis när jag loggar ut.

• Den applikation som startade processen avslutas.

• Vid olika fel, då OS går in och avslutar processen.

• En parent-process kan avsluta sina child-processer.

• En användare kan också döda processer (kill).

Anledningar Till Processavslut

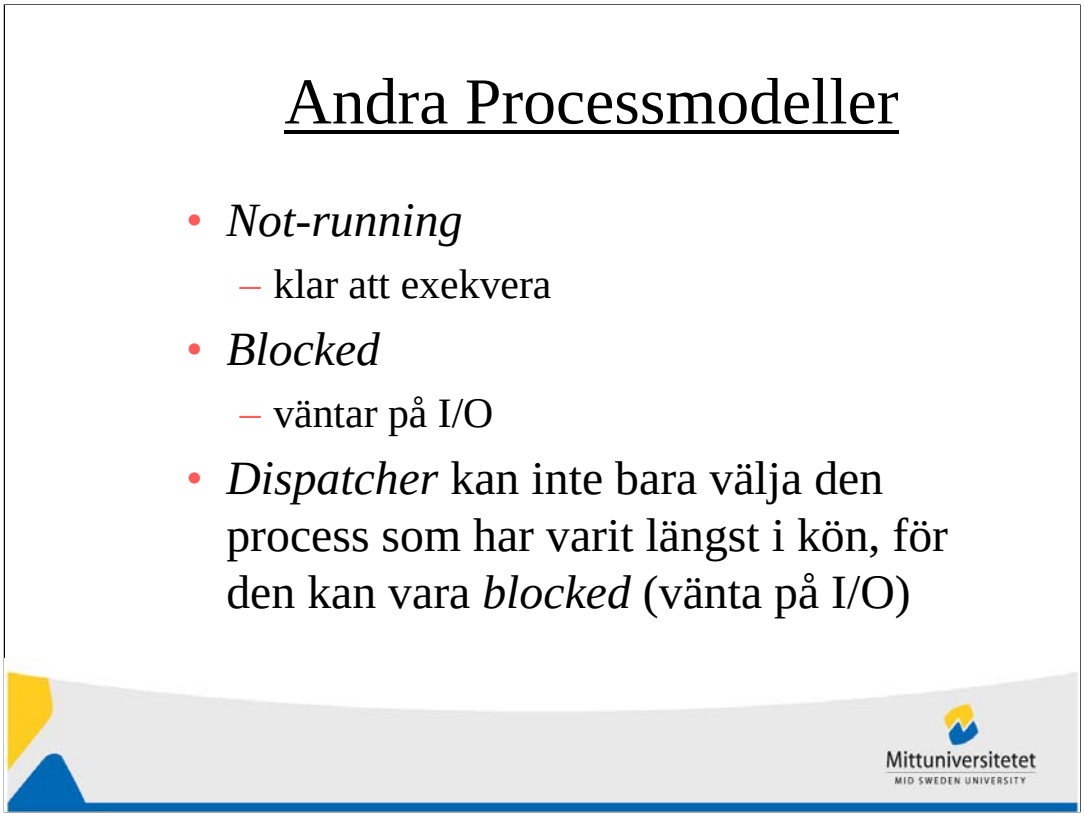
- Färdigexekverad
- Timeravbrott, total processtid
- Processen kräver mer än tillgängligt minne
- Processen försöker accessa otillåtet minne
- Skyddsfel
  - exempel skrivning till skrivskyddad fil
- Aritmetiskt fel
- Timeravbrott
  - processen har väntat på en händelse längre än specificerad maxtid

Mittuniversitetet
MID SWEDEN UNIVERSITY

Anledningar Till Processavslut

- I/O-fel
- Ogiltig instruktion
  - Ex.vis försöker exekvera data
- Instruktion reserverad för operativsystemet
- Felaktigt data: typ, initiering
- Operativsystemet avslutar
  - ex.vis vid *deadlock*
- *Parent* avslutas vilket gör att *child*-processen avslutas
- *Parent* avslutar *child*-process

Mittuniversitetet
MID SWEDEN UNIVERSITY



Fler processtillstånd behövs i modellen

Bild 13-: En fullständigare modell över processens tillstånd behövs.

Se till att ni förstår allt i bild 15.

Modell med fem tillstånd

- *Running*
- *Ready*
- *Blocked*
- *New*
- *Exit*

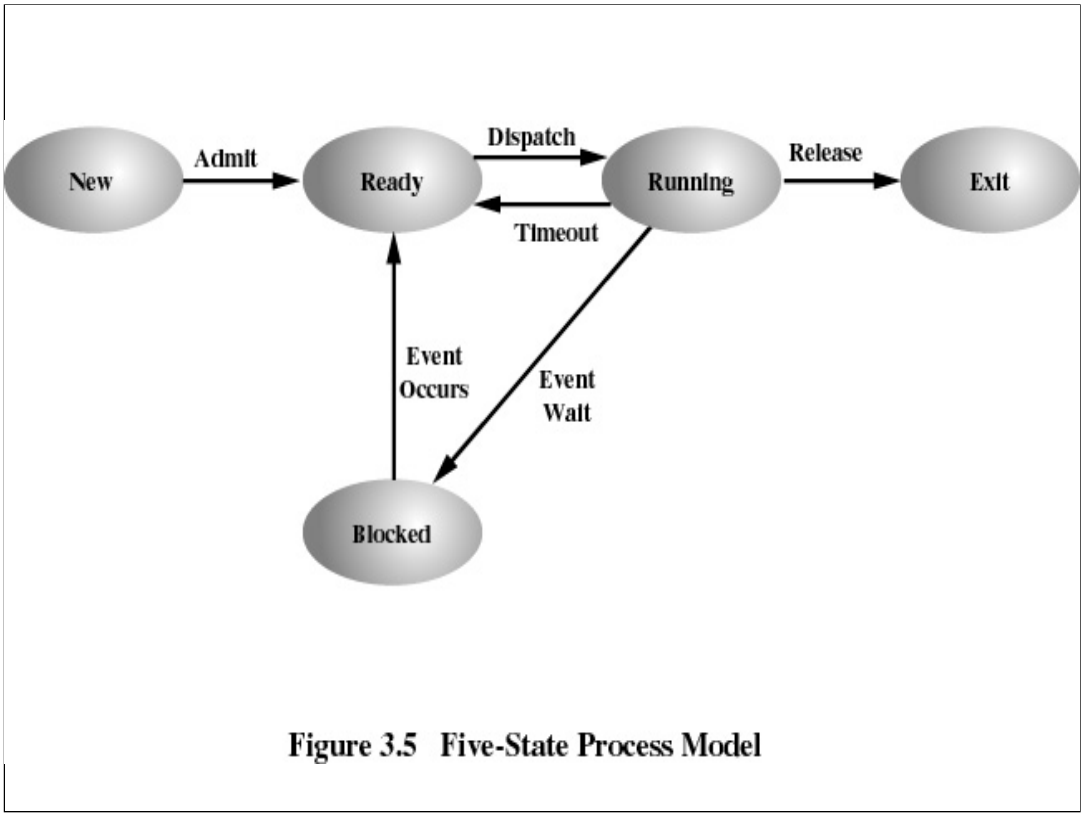


Figure 3.5 Five-State Process Model

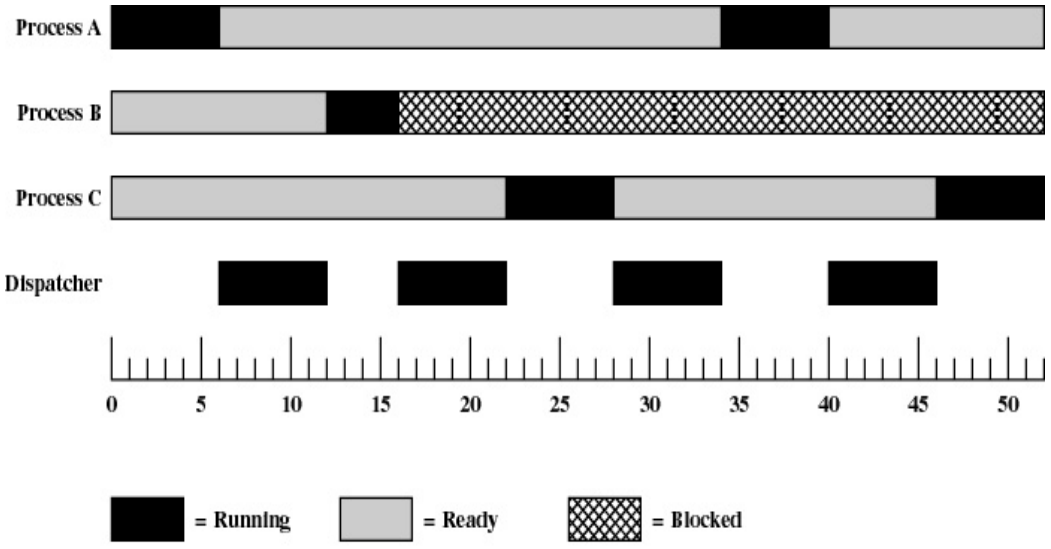


Figure 3.6 Process States for Trace of Figure 3.3

Vad är den troligaste orsaken till att B blir blocked?

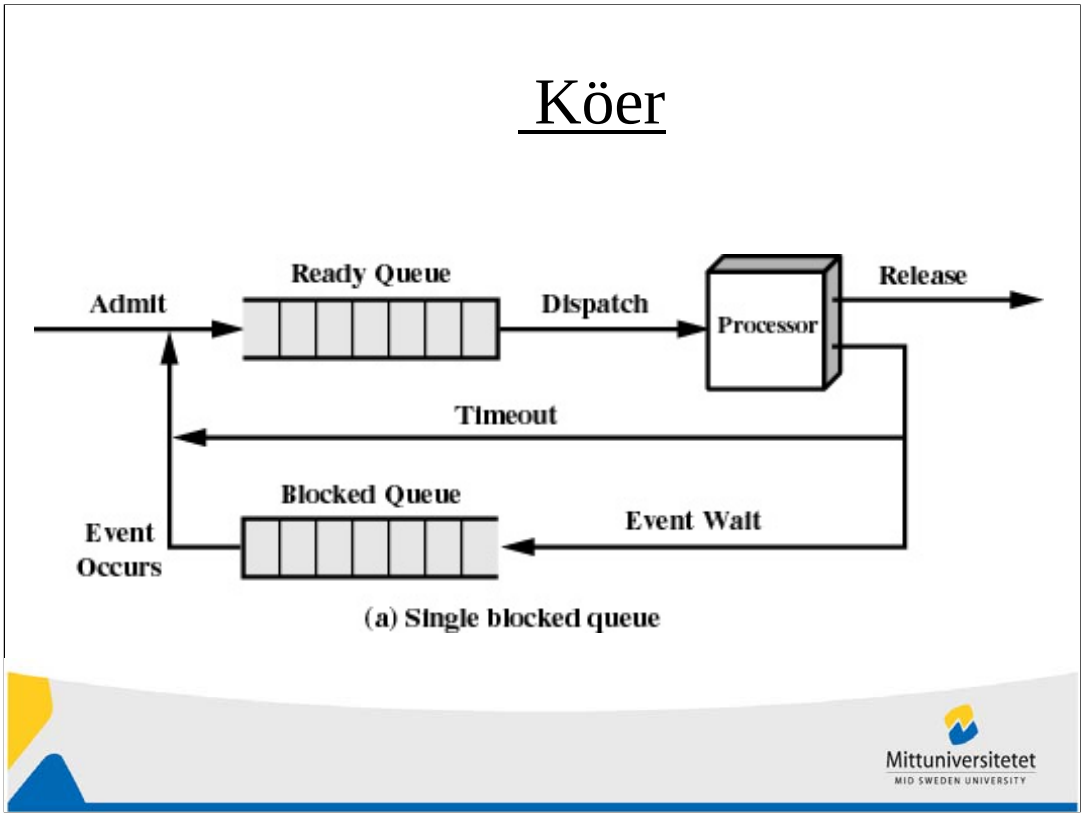
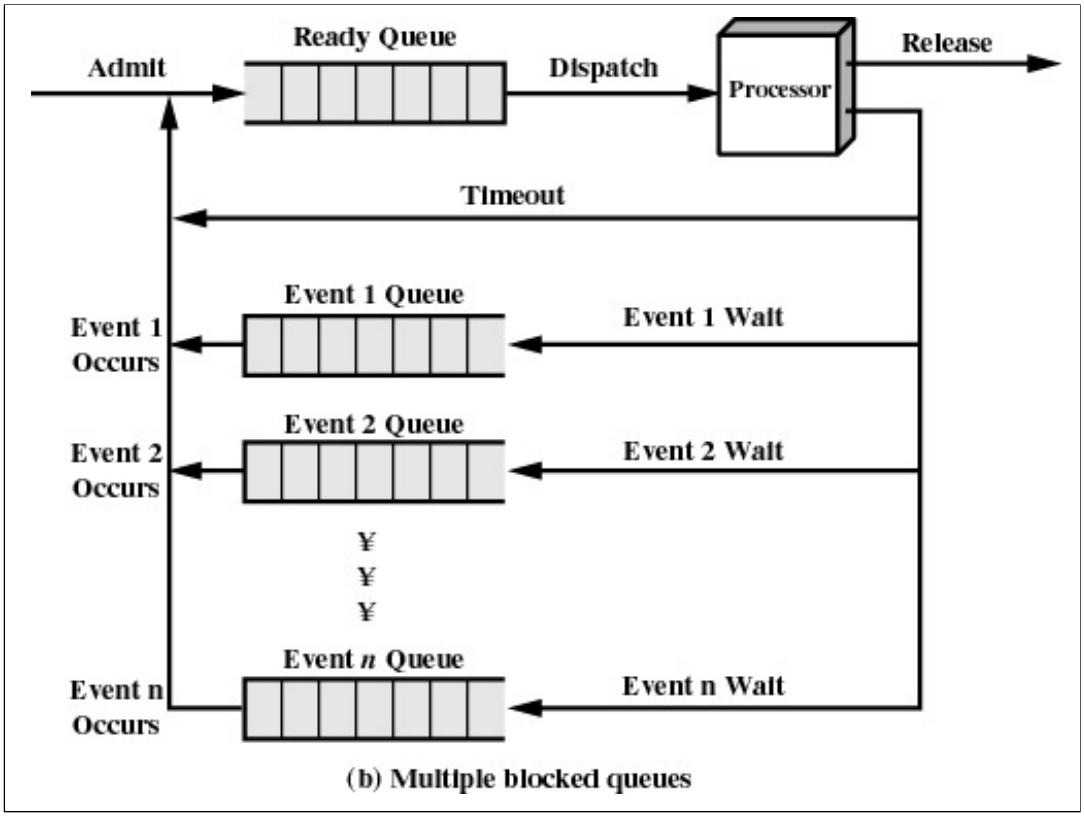
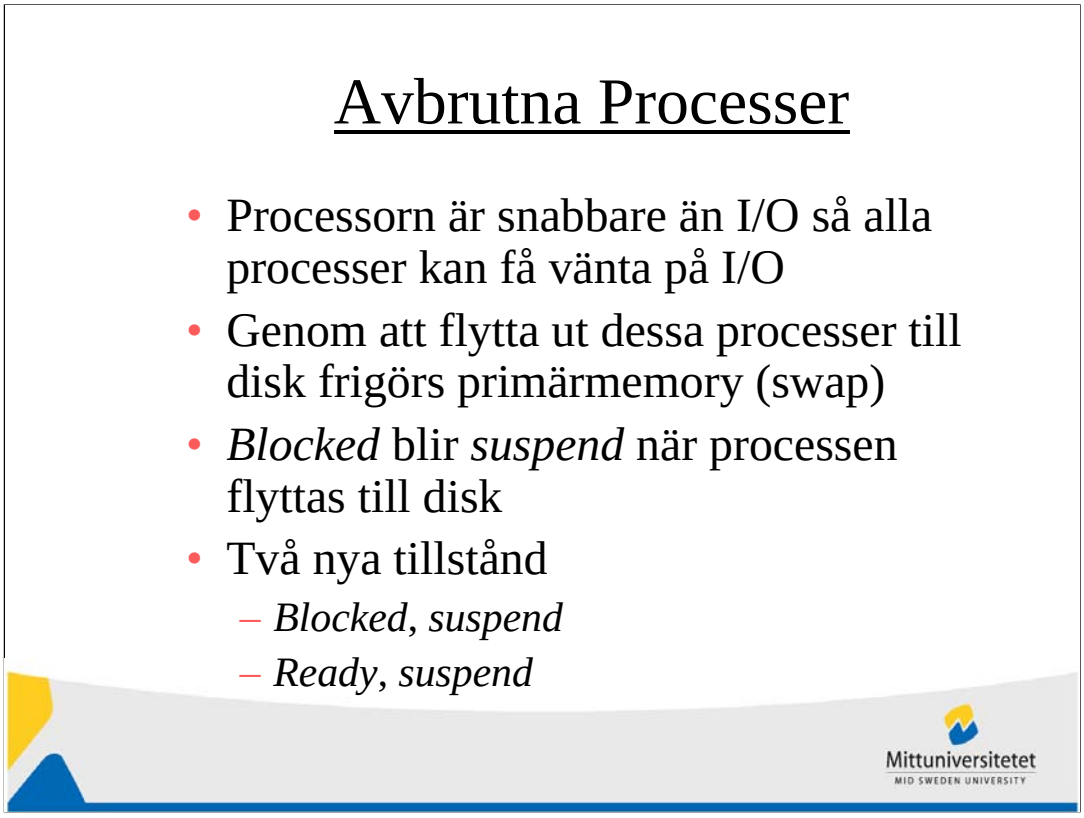


Bild 17-18: För att göra modellen fullständigare kompletterar vi med flera köer. *Blocked Queue* behövs eftersom det kan vara flera processer som är avbrutna och väntar på samma resurs (ex hårddisk).



(b) Multiple blocked queues



Swappning av processer

Bild19-22: Med begränsad mängd primärminne kanske inte alla processer kunde läsas in samtidigt, men det kan hända att samtliga inlästa processer är i blocked mode. Då finns ingen process som exekveras. Vi ökar effektiviteten genom att ge möjlighet att tillfälligt flytta ut inlästa processer till hårddisk.

När OS behöver mer minne än som finns ledigt swappas ett antal sidor ut och den nya processens sidor läggs i dessa frames. Fler processer kan då läsas in och exekveras. Detta sätt att flytta ut delar av eller hela processer kallas swapping.

Två nya tillstånd behövs:

• Blocked/Suspend där processen är avbruten pga. I/O-anrop och dessutom utswappad

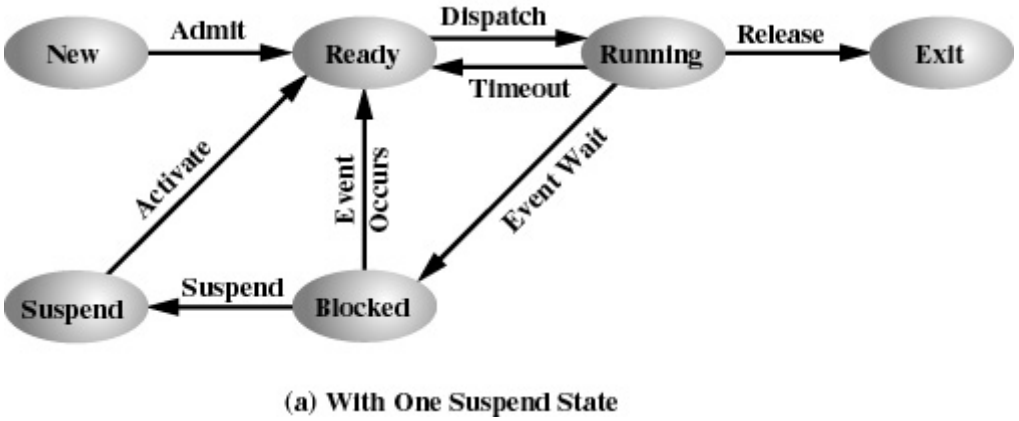
• Ready/Suspend där processens I/O-anrop är klart med processen fortfarande är utswappad. Den kan ju inte köras igen innan den har swappats in i minnet igen.

Bild 21 är också viktig att förstå allt i.

Ett antal processer fyller hela primärminnet. Alla kan vara upptagna med I/O

För att öka antalet processer utan att öka minnet så används swapping och ett extra tillstånd Suspended

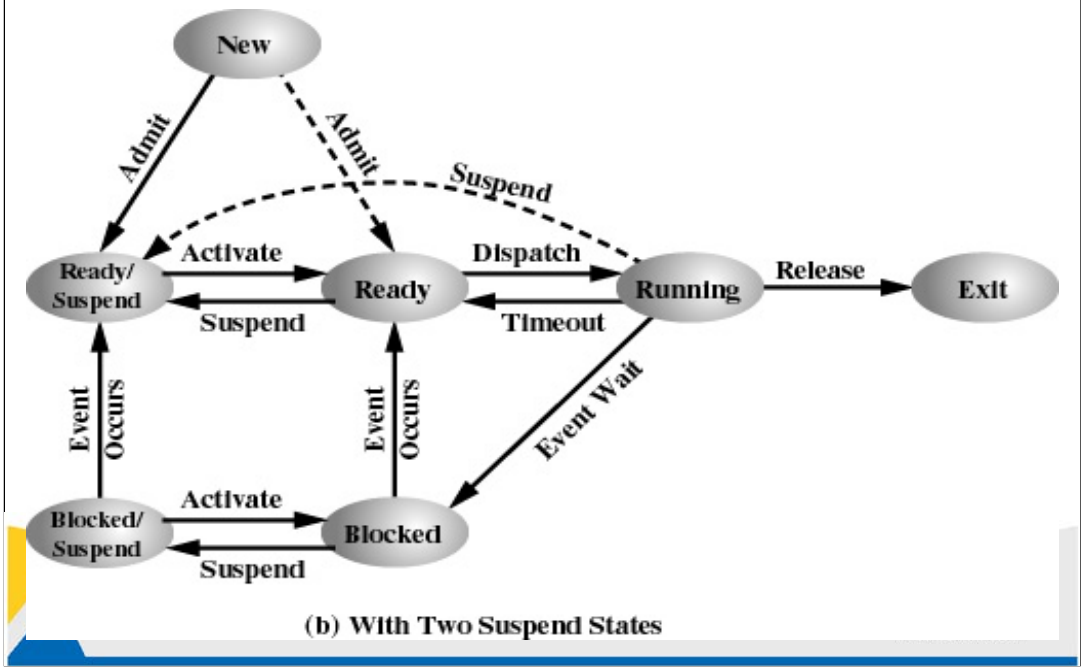
Ett *Suspend*



(a) With One Suspend State

Mittuniversitetet
MID SWEDEN UNIVERSITY

Två *Suspend*



(b) With Two Suspend States

När Behöver En Process Swappas

Swapping	The operating system needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The operating system may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.
Parent process request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents.

Processer och resurser

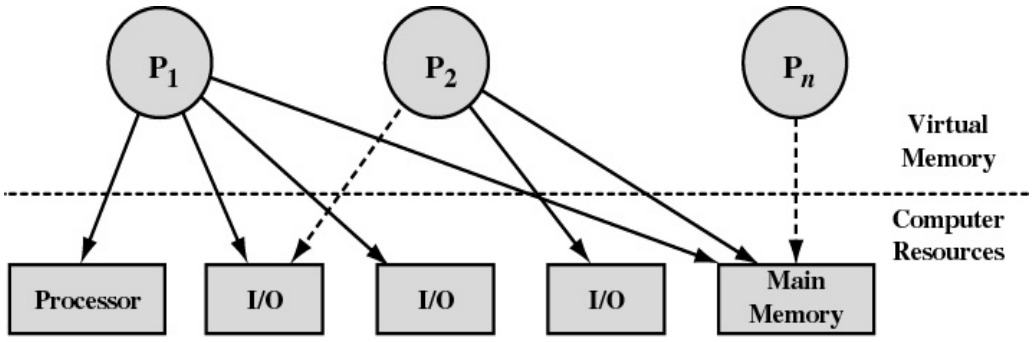


Figure 3.9 Processes and Resources (resource allocation at one snapshot in time)

Resurser tilldelade en process

Bild 23: När en process startas önskar den en viss mängd resurser. OS bestämmer hur mycket som
inklusive processorn.

Varför är det streckade linjer för Pn till minnet och P2 till I/O?

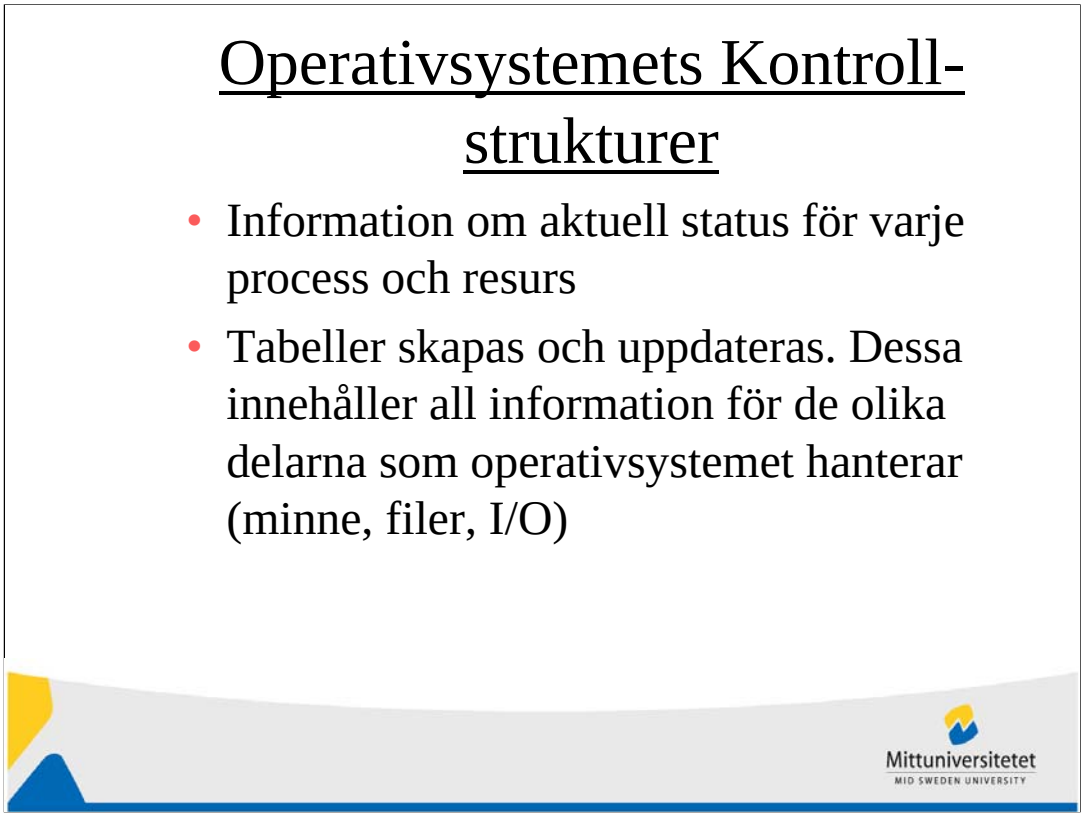


Bild 24-: för att hålla reda på vilka resurser som är tilldelade vilken process samt aktuell status för processerna krävs ett antal kontrollstrukturer, i form av tabeller.

Minnestabeller

- Tilldelning av primärminne till process
- Tilldelning av sekundärminne
- Skyddsfunktioner för access till delade minnesadresser
- Information som behövs för att hantera virtuellt minne

Mittuniversitetet
MID SWEDEN UNIVERSITY

I/O-tabeller

- I/O-enheter är tillgänglig/upptagen
- Status på I/O-operation
- Adresser i primärminnet som används som databuffert vid I/O-hantering

Mittuniversitetet
MID SWEDEN UNIVERSITY

Filtabeller

- Existerande filer
- Plats i sekundärminnet
- Aktuell status
- Attribut
- Ibland hanteras denna information av ett filsystem

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processtabell

- Var processen ligger (i minnet)
- Attribut som behövs för process-hantering
  - Process ID
  - Processtatus
  - Plats i minnet

Processlokalisation

- Processen består av ett antal program som ska exekveras
  - Plats för lokala och globala variabler
  - Definierade konstanter
  - Stack
- Processkontrollblock
  - Samling av attribut för processen
- *Process image*
  - Samtliga program, data, stack, och attribut som hör till processen

Mittuniversitetet
MID SWEDEN UNIVERSITY

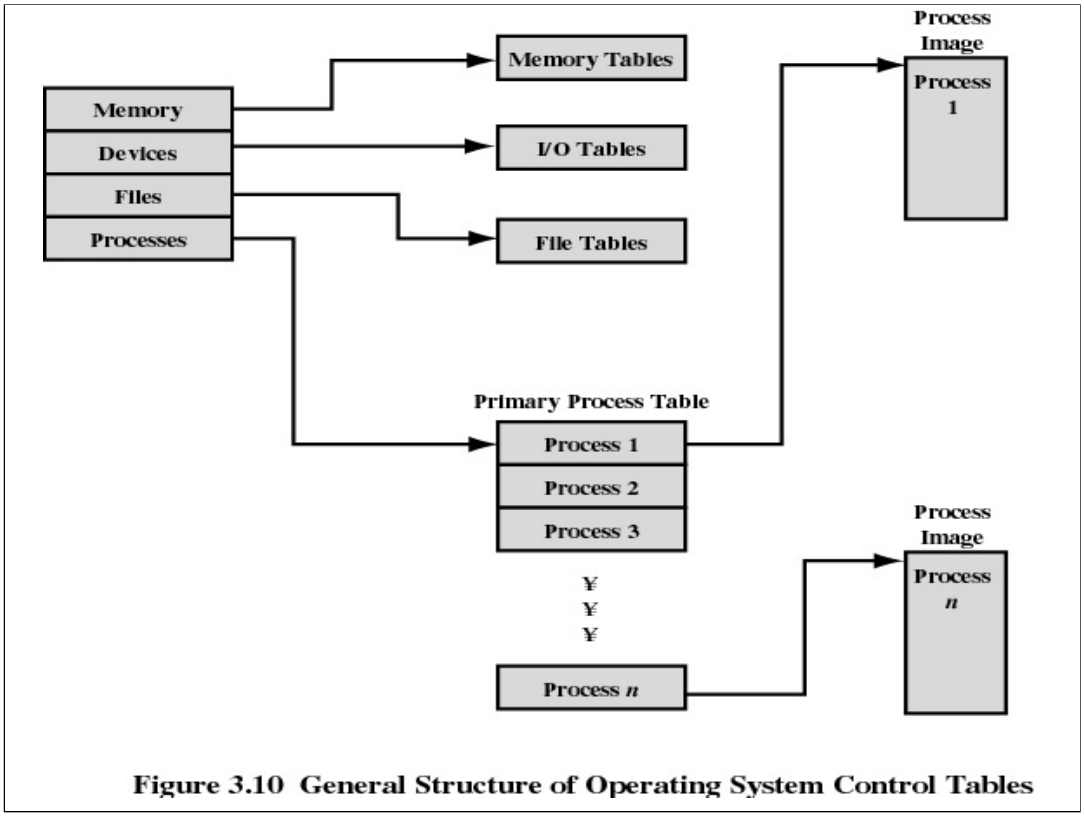
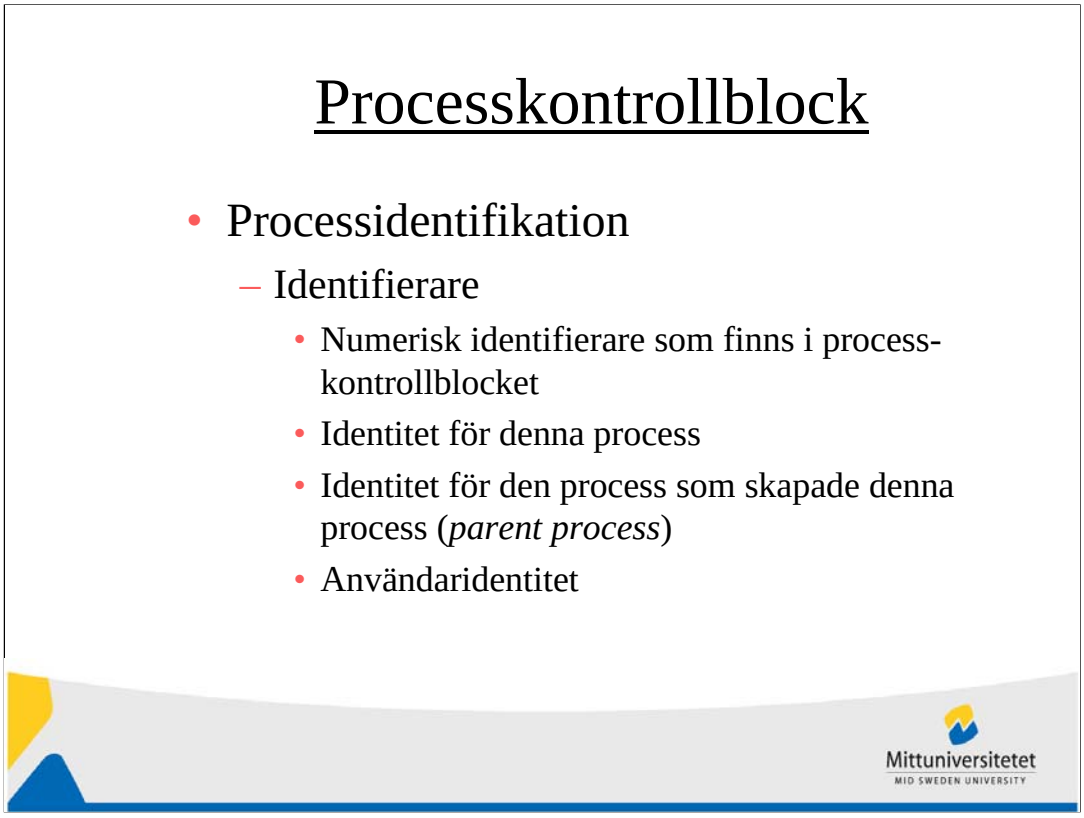


Figure 3.10 General Structure of Operating System Control Tables



Process Control Block

Bild 31-: Processkontrollblocket (Process Control Block) samlar all information om en process. Varje process har en unik processID (pid).

Pid tilldelas av OS vid skapandet av processen.

Flera processer kan samarbete kring en uppgift och behöver då kunna kommunicera med varandra detta kallas IPC (Inter Process Communication) Mer i senare kapitel.

Processkontrollblock

- Information om aktuell processorstatus
  - Ett register, som är åtkomligt för användare
    - åtkomligt för användare genom kod som körs i processorn.
    - Det kan finnas mellan 8 och 12 sådana register, men vissa RISC-processorer kan ha över 100.

Processkontrollblock

- Processor Statusinformation
  - Kontroll och statusregisters

    Det finns ett stort antal register som används för kontroll-funktioner för processorns operationer. Bland annat:

    - •Programmräknare *(PC)*: Håller addressen till den instruktion som ska hämtas nästa gång
    - •Resultatregister: Resultatert oav den senaste aritmetiska eller logisk operation (ex. tecken, noll, carry, lika, spill)

      •Statusinformation: Innehåller flaggor för interrupt enabled/disabled, execution mode

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processkontrollblock

- Processorns Statusinformation
  - Stackpekare
    - Varje process har en eller flera stackar, last-in-first-out (LIFO) system, knutna till sig. En stack används för att spara parametrar och retur-adresser for procedurer och systemanrop. Stack-pekaren pekar till toppen på stacken.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processkontrollblock

- Processkontrollinformation
  - Schedulering och Statusinformation. Denna information används av operativsystemet vid schedulering. Innehåller bl.a. information om:
    - •Processtatus: visar om processen är klar för att scheduleras (ex. running, ready, waiting, halted).
    - •Prioritet: En eller flea fält används för att beskriva scheduleringsprioritet för processen. I vissa system, behövs flera värden (ex. default, current, highest-allowable)
    - •Schedulingsinformation: Beror på använd schedulings-algoritm. Ex.vis. Processens väntetid och senast processtid.
    - •Händelser: Id för den händelse som processen väntar på innan den kan återupptas (*resumed*)

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processkontrollblock

- Processkontrollinformation
  - Datastruktur

    En process kan vara länkad till andra processer i en kö, ring eller annan struktur. T.ex. alla processer som väntar i samma prioritetsnivå kan be ligga i en kö. En process kan befinna sig i en relation som *parent-child* med en annan process. Processkontrollblocket kan innehålla pekare till andra processer som stödjer dessa strukturer.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processkontrollblock

- Processkontrollinformation
  - *Interprocess Communication*
    - Olika flaggor, signaler, och meddelanden kan användas för kommunikation mellan två oberoende processer. En del eller all denna information kan hanteras i process-kontrollblocket.
  - Processrättigheter
    - Processen ges rättigheter till bl.a. Tillåtna minnesareor, tillåtna instruktioner. Dessutom kan rättigheter ges att använda systemfunktioner och services.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processkontrollblock

- Processkontrollinformation
  - Minneshantering
    - Denna del kan innehålla pekare till segment och/eller sidtabller som visar det virtuella minnet som är tilldelat denna process.
  - Resursers ägare och användning
    - En resurs kontrolleras/ägs av den process, T.ex. öppnade filer. Hur processen använt processorn och andra resurser tidigare. Denna information kan användas av scheduleraren.

Mittuniversitetet
MID SWEDEN UNIVERSITY

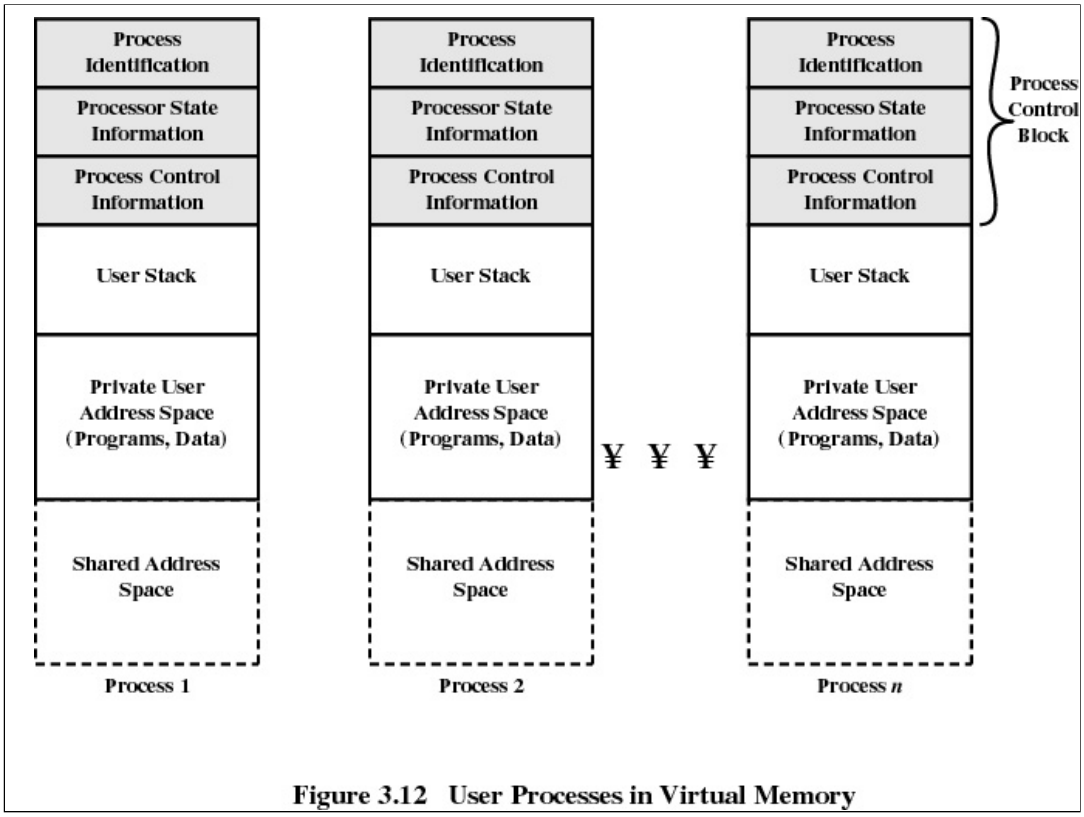
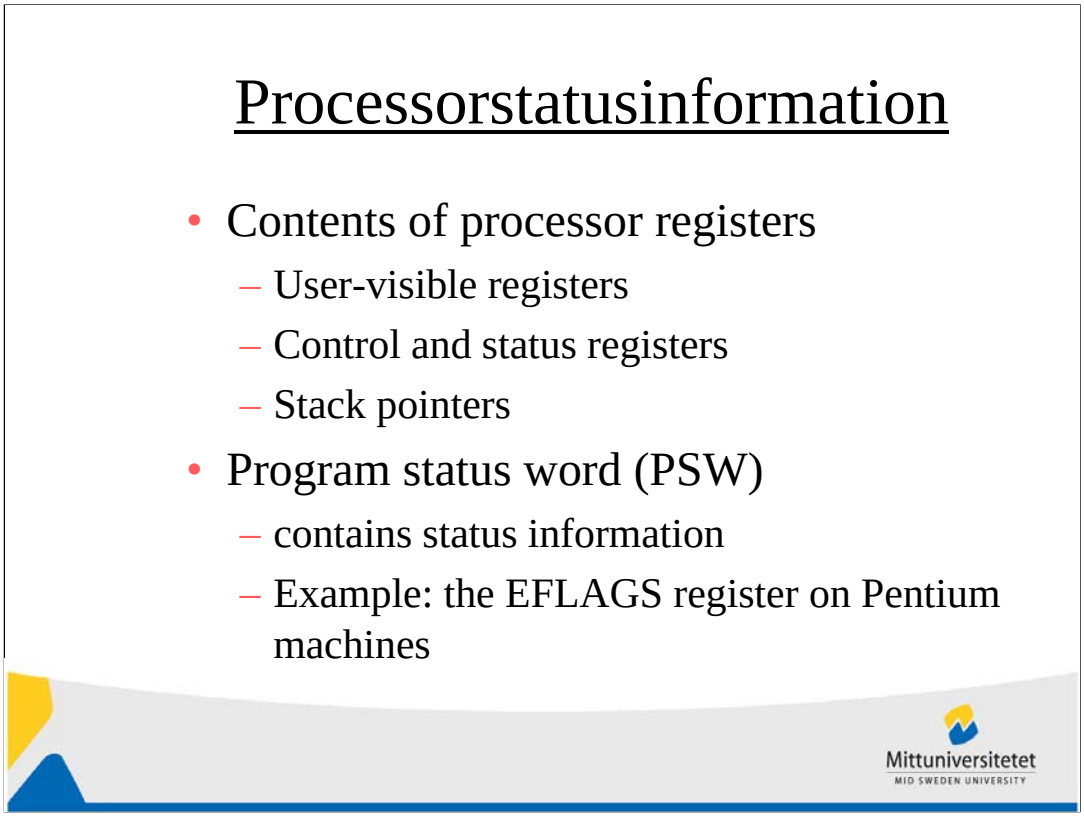


Figure 3.12 User Processes in Virtual Memory

Virtuellt minne

Bild 39: Med Virtuellt minne (mer senare) upplever varje process att den har tillgång till hela adressområdet själv. Processen delar in minnet i flera separerade delar:
PCB, Stack, Privat minnesområde (endast för den egna processen,
som kan bestå av fler delar: kod, data…)
och ev. ett delat minnesområde för kommunikation och delat data med andra processer.



Context switch

Bild 40-41: När en process avbryts för att en annan ska exekvera måste ju processens status sparas dvs. samtliga register och PSW.

Pentium II EFLAGS Register

ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

Figure 3.11 Pentium II EFLAGS Register

Mittuniversitetet
MID SWEDEN UNIVERSITY

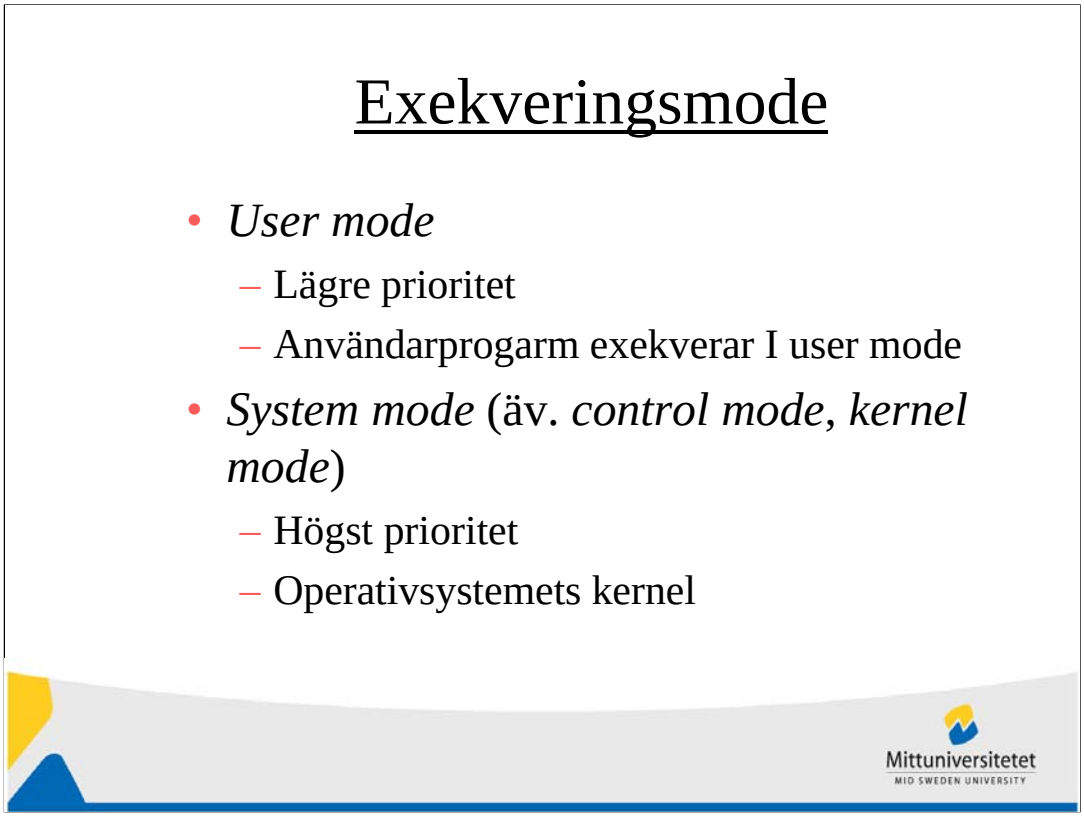


Bild 42: Det finns två olika typer av exekveringsmode *User mode* och *Kernel mode*. Kernel mode har högst prioritet för att OS ska kunna få tillbaka kontrollen över processorn även om en process försöker ta alla resurser eller låser sig

Skapande Av Process

- Tilldela en unik processidentifierare
- Allokera minne till processen
- Initiera processkontrollblock
- Skapa de länkar som behövs
  - Ex. lägg till processen i den länkade lista som används för schedulingskön
- Skapa och uppdatera andra datastrukturer
  - Ex. upprätta debiteringsfil

Mittuniversitetet
MID SWEDEN UNIVERSITY

Processwitchning

- Vid klockinterrupt
  - Processorn har använt hela sin timeslice
- I/O-interrupt
- Fel vid minnesaccess
  - Minnesarean som ska användas ligger i virtuellt minne och måste först flyttas in i primärminnet

Mittuniversitetet
MID SWEDEN UNIVERSITY

Bild 44-: Visar när en process-switchning sker:

• • Processen har använt sitt tidskvanta för denna gång,

• • I/O-interrupt (processen begärde ex vis läsning från disk och blev blockerad, när disken har läst önskat data och lagt det i buffert skickas ett interrupt som säger att det är klart. Processorn kan nu flytta data dit det ska.
Processen är klar för readykön igen.)

• • Fel vid minnesaccess, egentligen **Pagefault** dvs den sida som skulle användas fanns inte i primärminnet, utan var utswappad och måste läsas in.

• • Trap, ex vis processen försökte skriva/läsa i område reserverat för kerneln

• • Systemanrop, processen innehöll t ex filaccess

• •

Processwitchning

- Trap
  - Ett fel har inträffat
  - Kan göra att process flyttas till *Exit*
- Systemanrop
  - Ex. öppna fil

Mittuniversitetet
MID SWEDEN UNIVERSITY

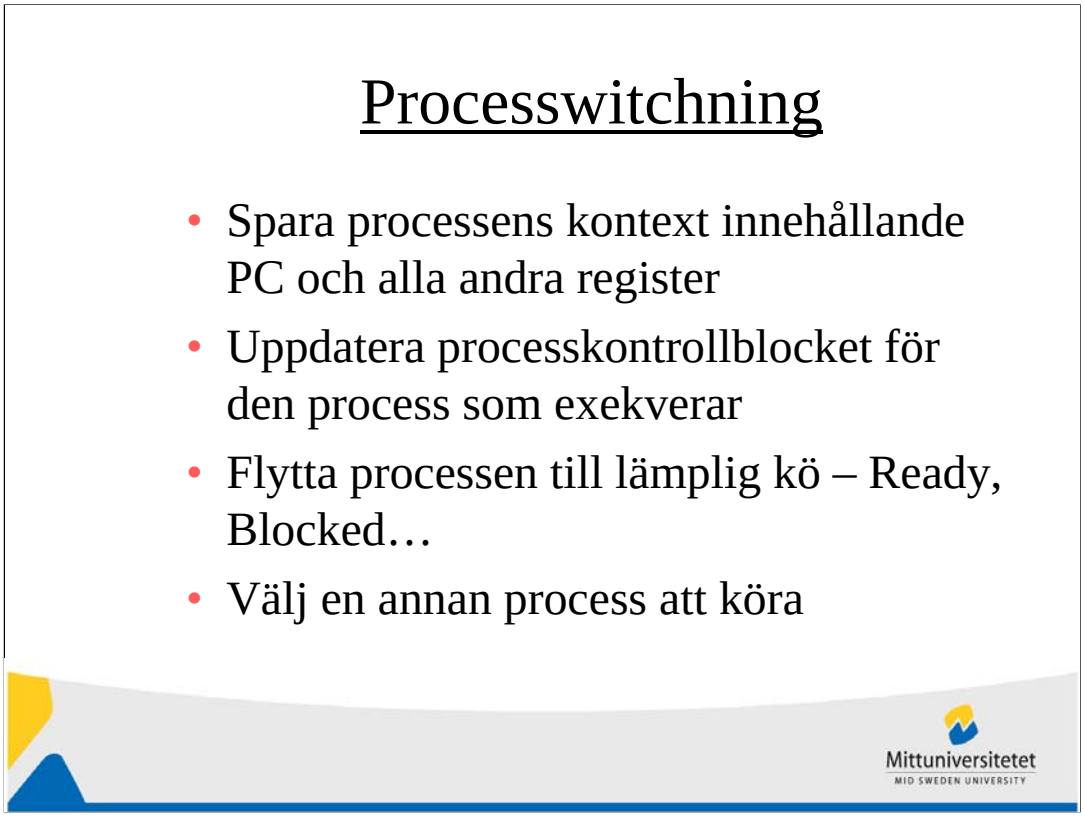


Bild 46-47: Viktigt att kunna hur processswitchningen går till

Processwitchning

- Uppdatera processkontrollblocket för den process som ska köras
- Uppdatera datastrukturen för minneshanteringen
- Återställ kontext för vald process

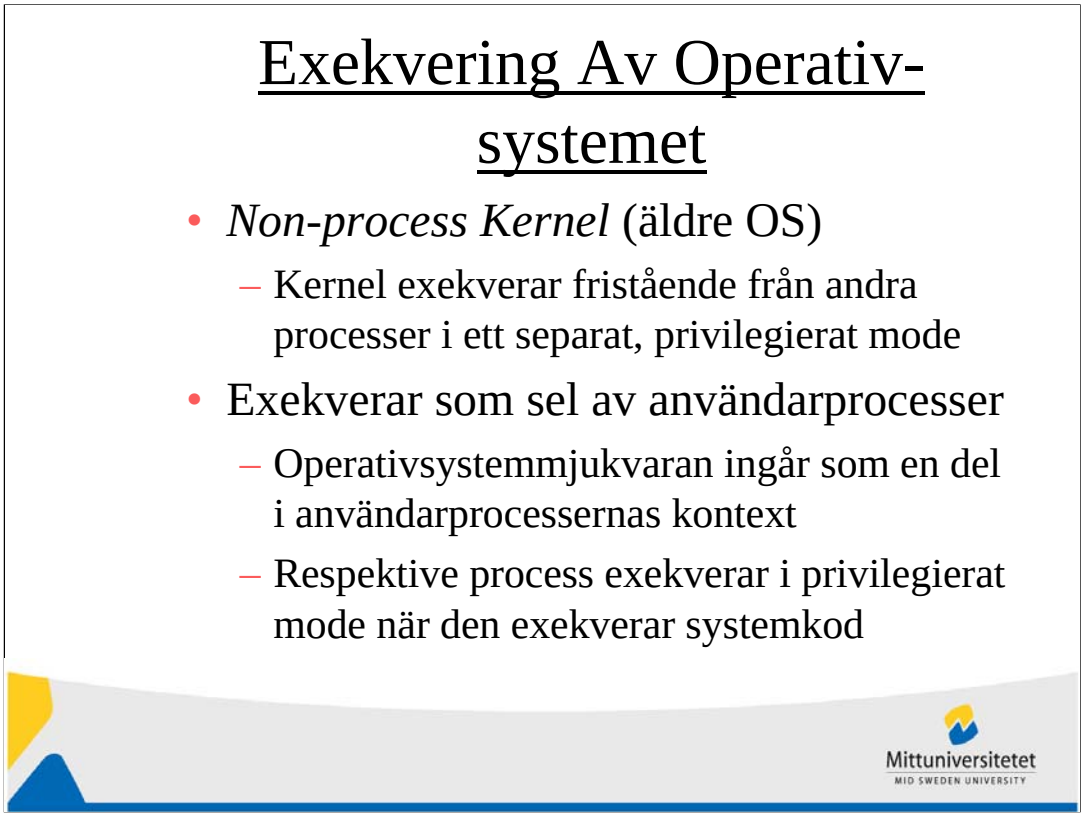


Bild 48-: OS själv kan exekveras fristående från processer,
men i nyare OS exekverar processen själv i kernel mode när den exekverar systemkod.

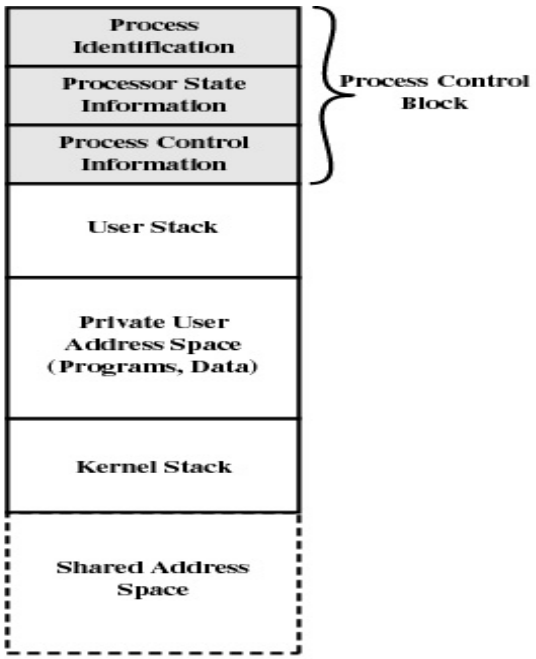


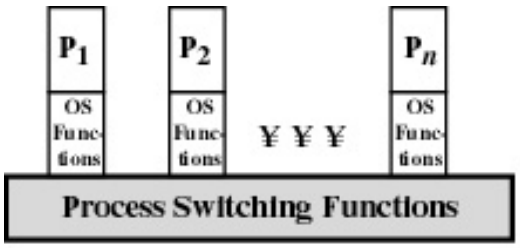
Figure 3.15 Process Image: Operating System Executes Within User Space

Exekvering Av Operativ-systemet

- Processbaserat operativsystem
  - De flesta kernelfunktionerna består av separata processer
  - Praktiskt i multiprocessor och distribuerade miljöer

UNIX SVR4 Process Management

- De mesta av operativsystemet exekverar som en del av användarprocesser



(b) OS functions execute within user processes

Mittuniversitetet
MID SWEDEN UNIVERSITY

UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.

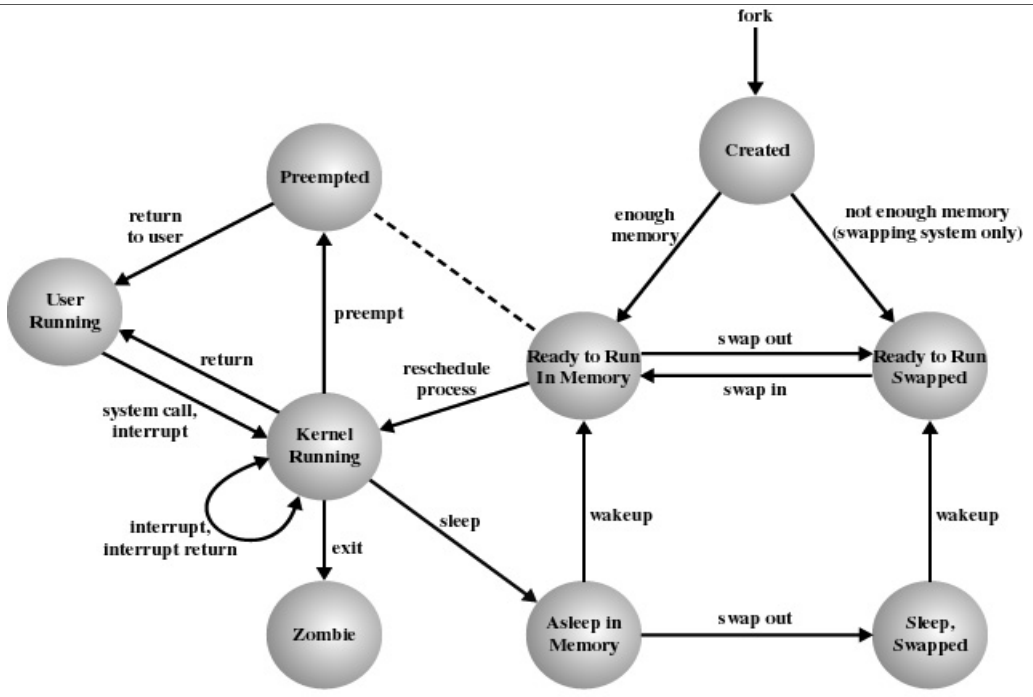


Figure 3.16 UNIX Process State Transition Diagram