a) Först skrivs följande ut på skärmen:

1: date
2: who
3: ls
4: pwd
Make Your choice please...

Sedan väntar ekvipset på att användaren skall mata in sitt val. (vilket lagras i variabeln choice).

Om användaren väljer ("enter"):

1. så utförs kommandot logout som loggar ut användaren.

2. så utförs kommandot vm med '*' som parameter, detta raderar samtliga filer i aktuellt bibliotek

3. så utförs kommandot 'who am i' som skriver ut användarens inloggnings-namn etc. på skärmen

4. så utförs 'ls -l' som listar aktuellt katalogs innehåll i "långt" format.

Vid alla andra val skrivs texten
What ?!?
på skärmen

4

b)
close(1); stäng std out
open(utfilnam, O_WRONLY); öppna fil (vars creat? namn finns i strängen utfilnamn) för skrivning. Filen tar då den plats som frigjorts av close & blir således standard output.

2

c) int fildes[2];
pipe(fildes); skapa rör ledning

Här skapas sedan två barn med fork.
I barn 1 (före pipen):
close(1); stäng std out
dup(fildes[1]); kopiera in ena pipe-anden på dess plats
close(fildes[0]); stäng "överflödiga" filer
close(fildes[1]); exec(...)
I barn 2 (efter pipen):
close(0); stäng std in
dup(fildes[0]); kopiera andra pipe-änden på dess plats.
close(fildes[0])
close(fildes[1])
exec(...);

000304

c) fourth

close(fildes[0]);
close (fildes[1]);
exec*(...);

stäng alla oanvända filer

i forkloop;

close(fildes[0]);
close (fildes[1]); — " —

3

a) En tråd är en lättviktsprocess - den har samma kod & data som "huvud processen och "systemtrådarna" men har egen programräknare och stack.

En tråd kan vara en (av flera alternativa) exekveringsväg i ett program. Om en tråd blockeras (t.ex. I/O wait) kan övriga fortsätta.

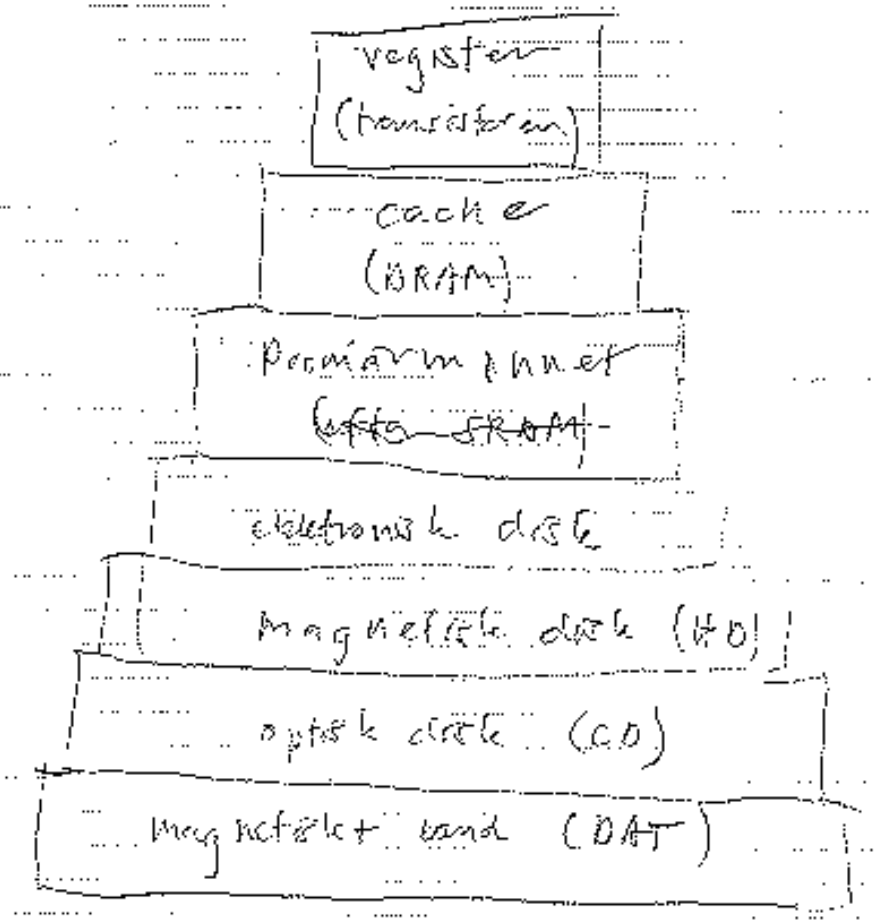
Det går snabbare att växla mellan trådar i en process än mellan processer (liten context-switch). 2

b) Mekanismer beskriver hur något skall göras; t.ex att en timer med jämna mellanrum skall ge oss kontrollen.

Policies anger vad som skall göras; t.ex hur ofta timern genererar avbrott. 2

Mekanismer skall vara generella medan policies ändras från en tid till en annan eller mellan olika implementeringar.

c)



Desto högre upp i hierarkin desto snabbare & dyrare blir minnena. 2

d) Ett system där flera processer kan exekvera "samtidigt" genom att bli tilldelade en liten tidskvantum åt gången tills de är klara 1

e) Ett system där en användare alltid är "påloggad" och kan ge sina kommandon direkt utan att dessa direkt går i förväg kända tex batchjobb eller switchar. Ingen interaktion används 1

f) Att I/O-enheter lånar bussen av CPU:n och själva överför data direkt till primärminnet 1

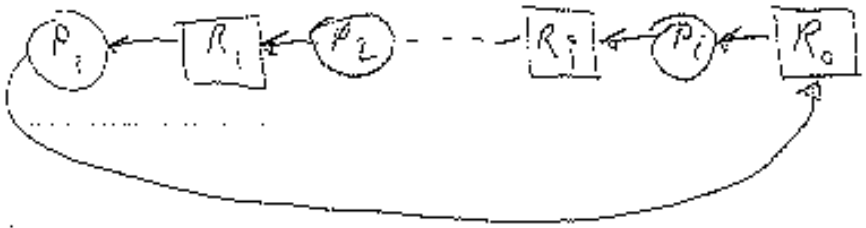
g) Att CPU:n kan försättas i olika moder såsom Supervisor/system/kernel vilka är. Genom att låta vissa instruktioner och åtkomst av vissa minnesområden vara begränsade till kernel mode kan man undvika skydd. 1

a) • Ömsesidig uteslutning – varje resurs kan bara hållas av en process.

• "Hold and wait" – en process som tilldelats en resurs kan fortsätta att begära fler.

• No – preemption – En process kan bara "bli av med" en resurs genom att frivilligt släppa den.

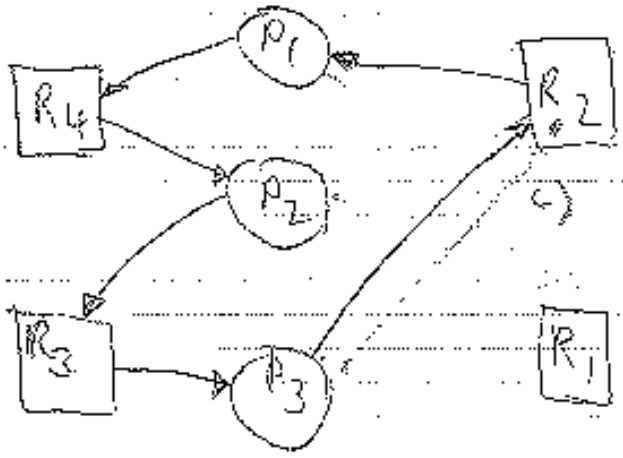
• Cirkulär väntan – En process håller en resurs som en annan process väntar på. Den första väntar även på en resurs som hålls av den andra. (Ofta med fler steg emellan...)



4

000304

b) RAG



3

Eftersom vi har en cykel i grafen så finns ingen möjlig exekveringsväg och vi har ett deadlock

c) Medan P2 äger R2 så kan P3's begäran uppfyllas och följande exekvering är möjlig

P3 får (en av 2) R2
P3 exekverar (och släpper då R2)
P2 får R3
P2 exekverar (och släpper då R4)
P1 får R4
P1 exekverar

3

a) med 12 bitar för adressering inom sida bjuds möjlighet att adressera

$2^{12} = 2^{10} \cdot 2^2 = 4 \cdot 2^{10} = 4$ kB

b) Med 12-bitars sidnummer kan vi ha 4096 st sidor/process. Varje process kan således ha maximalt

4 kB · 4096 = 16 MB adressrum.

($2^{24} = 16$ MB)

c) Varje post är 12+1+1+1 = 16 bitar = 2 bytes

Med 4096 sidor blir det 8192 bytes

Dvs. 8 kB

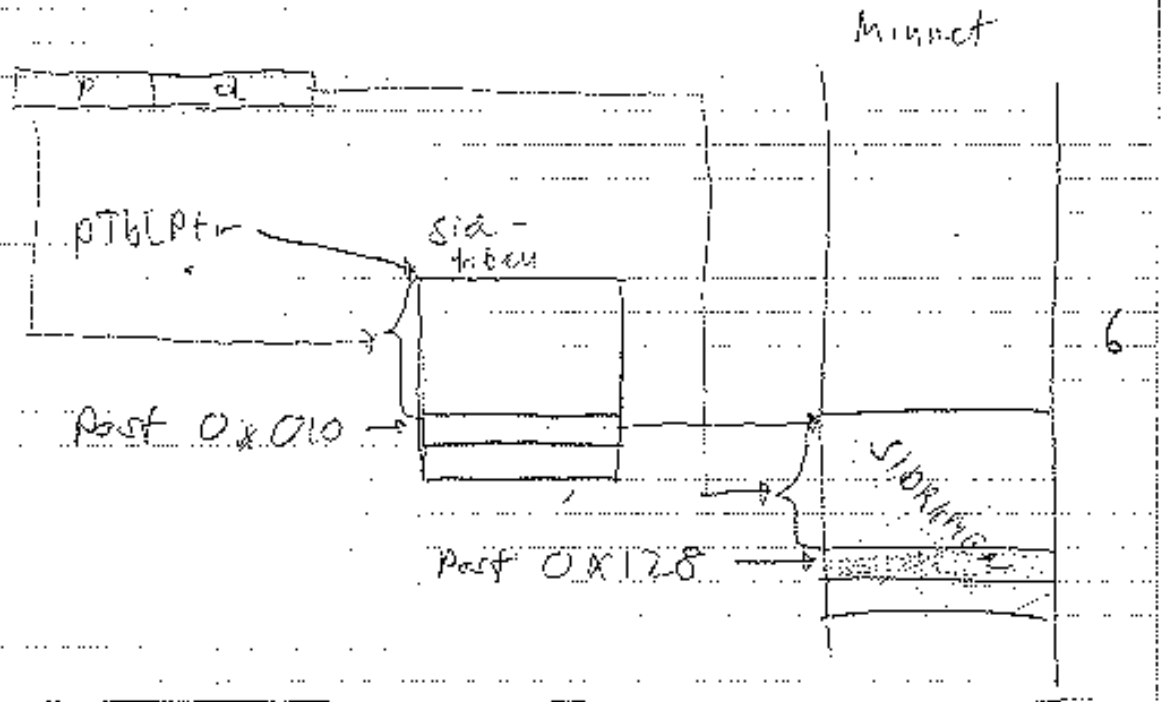
d) Adressen delas upp enligt Page: 0x010

Förskjutning 0x125

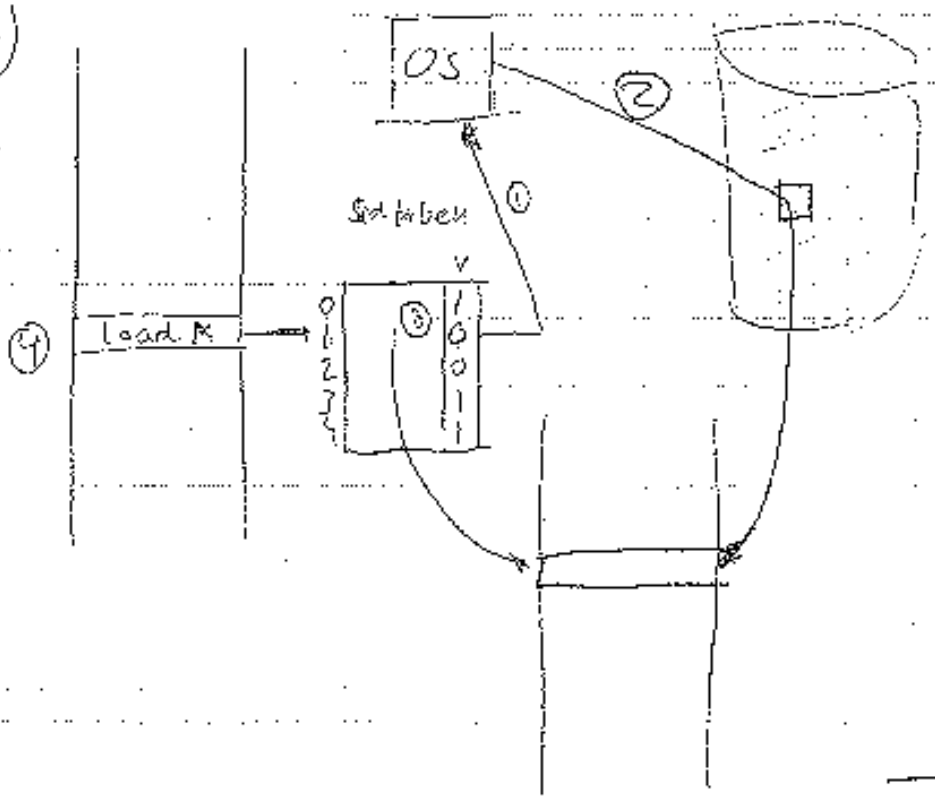
Vilket används enl. fig på nästa sida.

- Om V-flaggan är noll genereras ett sidfel (se uppg. e)
- Om p-flaggan är satt OCH vi försöker skriva till sidan genereras en TRAP till OS.

d) forts. (Systemet anses inte ha någon TLB)



e)



Om en uppslagning i sidtabellen hittar en V-bit som är noll så

① Genereras en TRAP- till OS som kontrollerar om det är en giltig referens för processen (genom att se i en intern tabell i processkontrollblocket)

Om referensen är giltig åtgärdas ledig sidram, av genom att swappa ut någon annan sida, * . 10

② Därefter swappas aktuell sida in från disk till den allokerade sidramen

③ Sidtabellen uppdateras & V-biten sätts till ett.

④ Instruktionen som orsakade sidfelet återstartas.

* I första hand en med R-bit noll och i andra hand en med M-bit noll. Dessa kan ej ändras och behöver således ej swappas ut.

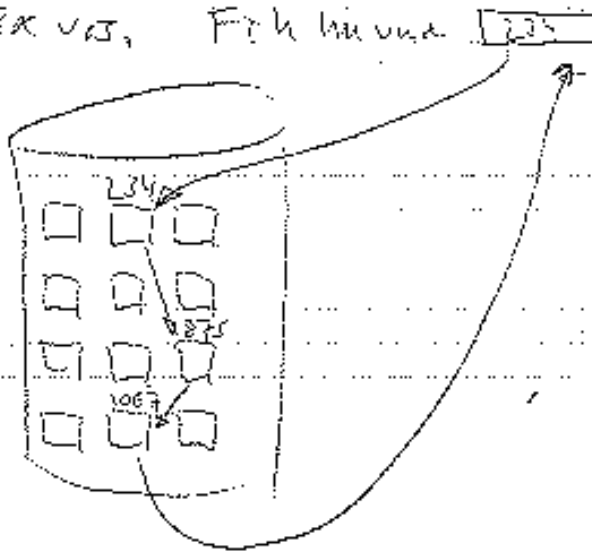
ING- OCH SJÖBEFÄLS-SKOLAN

000304

10 | 12 | 5

I ett filsystem med länkad lista finns i slutet på varje disk block några bytes med en pekare till filens fortsatt unig

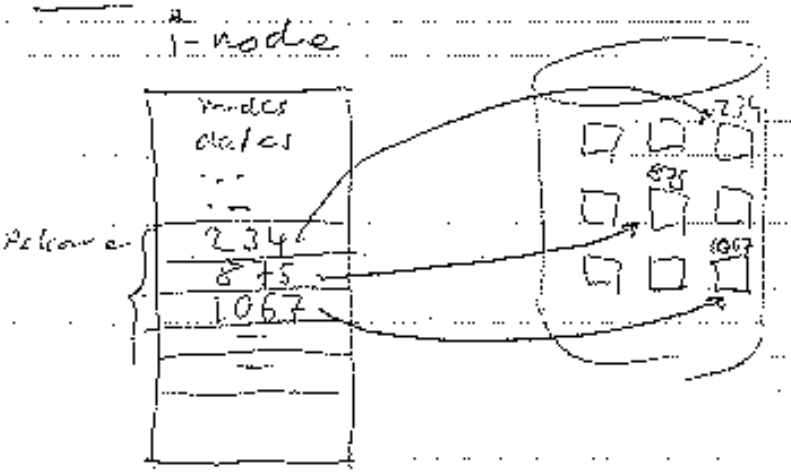
Ex vis, Fil huvud



Pekare	: block
234	→ 875
875	→ 1067
1067	→ -1

I ett traditionellt filsystem finns alla pekare i filhuvudet (i-noden) (för stora filer finns pekare i ut pekade block ... second-level ...)

Ex vis



ING- OCH SJÖBEFÄLS SKOLAN VID OTH

000304

För och Nackdelar

Länkad lista spar direkt regnme som att endast nödvändiga poleme laqras. Du måste däremot läsa filen sekventiell och kan inget direkt åtkomst. Dessutom är det omfattligt, (om du skall e tappa en poleme).

Indexerad allokering sköter utrymmet genom att allokera plats för poleme som av små filer inte används. Fördelen är att du kan direkt åtkomst var som helst i filen.

Slutligen systemet primärt används för att du själv kan välja ett lämpligt filsystem.