Lösningsförslag till Tentamen I ID2206/ID2200/IS1350 Operativsystem

(fd 2G1520/2G1504/6B2019)

Onsdagen 2011-05-25 kl14-18

Examinator: Robert Rönngren

Hjälpmedel: Inga

Besvarade frågor markeras på utdelat blad

Varje inlämnat blad skall förses med följande information:
- Namn och personnummer
- Nummer för behandlade uppgifter
- Sidnummer

ID2200: studenter som tenterar ID2200 besvarar inte frågor markerade med *
IS1350: studenter som tenterar IS1350 besvarar inte frågor markerade med ¤

Rättning: ***Alla svar, även på delfrågor, måste ha minst en kortfattad motivering för att ge poäng om inget annat angivits i tentamenstexten***
Resultat beräknas anslås inom 15 arbetsdagar efter tentamen

Betygsgränser:
Totalt antal poäng på tentamen är ID2206 31p, ID2200 26p, IS1350 26p
Godkänt (E) garanteras från 50% av den maximala poängen för respektive kurs

Lösningsförslag anslås på kursens webbsida, normalt inom 3 arbetsdagar efter tentamen

Betygsgränser:

ID2206 15.5-18 E, 18.5-21 D, 21.5-24 C, 24.5-27 B, 27.5 – A

IS1350/ID2200 13-15.5 E, 16-18.5 D, 19-21.5 C, 22-24.5 B, 25- A

Frågor:

1) Beskriv hur adressöversättning sker i ett datorsystem med ett OS som använder virtuellt sidindelat minne! 2p

   Applikationen adresserar sig i den virtuella adressrymden, dvs. från processorn kommer en virtuell adress. Den består logiskt av sidnummer och offset inom sidan. Den virtuella adressen tas omhand av MMUn som kontrollerar om sidan finns i det fysiska minnet (primärminnet) . Normalt sker detta genom en uppslagning i en TLB, men om översättningen inte finns i TLBn så slås det upp i processens sidtabell.

Finns sidan inte i det fysiska minnet så läses måste den läsas in från hårddisken till en ledig (eller frigjord) ram. När sidan finns i minnet ersätts sidnumret med ramnummer för den ram där sidan finns och primärminnet kan accessas. Kompletteras lämpligen med en bild!

2) a) Beskriv begreppet swapping! 1p
b) Varför väljer man normalt inte en större swaparea än c:a 2*RAM? 1p
a) Swapping: att hela processer flyttas till/från hårdisk och minne. Vanligt fel är att man blandar ihop swapping (flytta hela processer) och paging (flytta sidor)
b) Swaparean används för paging – dvs. att flytta ut och in sidor till/från primärminnet. Om man antar att minnesaccesserna är ganska jämnt fördelade över en process adressrymd så kan man sluta sig till at sannolikheten att hitta en sida i primärminnet är 50% om man behöver utnyttja en swaparea lika stor som primärminnet. Det skulle generera för mycket sidfel och för dålig lokalitet generellt för att prestanda skulle vara acceptabel – därav att det knapapst är ide att ha för stor swaparea.

3) Kalle håller på att designa ett nytt operativsystem, KaOS, som skall bli effektivare och snabbare än LINUX, MS-Windows mfl. Genom att det inte skall ha dynamisk adressöversättning. De lägsta 128MB i primärminnet reserveras för OS. Alla exekverbara moduler har absolutadresserad kod där lägsta adressen är 2^{27} . Beskriv de två viktigaste begränsningar med Kalles nya OS! 2p

Vad Kalle återuppfunnit är i princip en minnesmodell liknande MS-DOS. De två viktigaste begränsningarna är att: 1) det är svårt att ha fler än ett program/process aktivt (i primärminnet) i taget då alla skall laddas på samma adress, dvs. multitasking implementeras via swapping och blir väldigt långsamt; 2) man kan inte utnyttja virtuellminne utan blir begränsad till hur mycket primärminne det finns i maskinen.

4) Antag att du har en datorarkitektur utan TLB. Bör man då välja en en-nivås eller två-nivås implementation av sidtabellen? 1p

En en-nivås tabell eftersom man i varje adressöversättning från virtuell till fysisk adressrymd måste accessa sidatabellen i primärminnet. En en-nivås sidtabell ger då upphov till en upplevd minnesaccesstid som är 2M (fördubblad) medan en två-nivåers skulle ge tre dubblad minnesaccesstid, 3M Vanligt fel: att man inte insett att man INTEsöker i sidtabellen,utan att man indexerar med sidnumret

5) Beskriv vad som händer i OS:et (UNIX/LINUX) när en process gör waitpid(*PID*, &status,0) för en process *PID* som avslutats (terminerat)? 1p

OS:et returnerar den status som finns kvar i processtabellen för processen via parametern status och posten i processtabellen frigörs (dvs. det sista spåret efter processen försvinner) om inte detta gjorts tidigare i vilket fall en felkod returneras. Vanligt missförstånd: processens övriga resurser i form av minne, öppna filer etc. tar OS:et tillbaka redan när processen terminerar.

6) Vad är för- och nackdelar med att använda en fix time-slice för en process respektive en variabel? 1p

Fördelen med fix timeslice är att schemaläggningen i sig blir enklare men att man inte har lika stor flexibilitet att stödja olika typer av krav från processernas sida. Med dynamisk (individuell) timeslice kan man anpassa timeslice efter de individuella processernas behov och man får större flexibilitet att implementera den policy vad gäller processortidstilldelning man vill nå – men nackdelen är att schemaläggningen blir mer komplicerad (i.e. att beräkna storleken på timeslice) och möjligen

långsammare.Jfr hur man sätter timeslices i orginal UNIX och i LINUX. Vanliga missförstånd/fel: fix timeslice betyder inte att processen inte kan pre-emptas t.ex om den blockerar på ett systemanrop, det betyder heller inte att man undviker svältning. om man har svältning eller ej beror, om man inte kan ha oändligt lång timeslice vilket vi utesluter som orimligt, på hur man sätter/använder prioriteter

7) Under vilka förutsättningar kan man få kortare total körtid om man har två av varandra oberoende processer, en dator med en en-kärnas processor och man kör processerna pseudo-parallelt istället för först den ena och sedan den andra (i sekvens)? 1p

   Om en eller båda processerna gör I/O, till olika I/O-enheter, kan man låta den andra processen exekvera medan den första väntar på I/O. På så sätt kan I/O och beräkning överlagras vilket ger kortare total körtid (turnaround time)

8) ¤* Antag att man har flera trådar i en process. Vilka av dataareorna stack, heap, data och text måste vara privata resp. kan delas mellan trådarna? 1p

   Stacken måste vara privat för där sparas trådens privata värden för register, stackpekare och programräknare, samt aktiveringsposter för den tråd av procedurer/funktioner den anropat. Övriga areor delas normalt mellan alla trådar inom en process.

9) Varför har man i de olika standardena för C inte definierat en fix bitlängd för heltal? 1p

   *Man har bara angett minsta antal bitar för att man på **effektivast** sätt skall kunna mappa heltalstyperna mot den ordlängd som är den naturliga för den datorarkitektur man kör på. Det betyder att man bara har helt portabel kod om man inte utnyttjar heltalsvariablernas talområde mer än vad miniminivån ange. Dvs det underlättar inte portabilitet i det avseendet att det underlättar flytt av applikationer mellan olika system.*

10) Kalle har fått i uppdrag att förbättra ett distribuerat auktionssystem, Hederliga Auktions Lagret (HAL), för att det skall bli absolut rättvist i fråga om i vilken ordning buden läggs. Tanken är att kundernas bud skall sorteras efter en GPS-klockstämpel som sätts på klientsidan. Kalle tycker inte iden är bra och avråder kunden från att implementera det. Har Kalle rätt eller fel? 1p

    Man kan inte ordna händelser exakt i ett distribuerat system efter GPS-tidsstämplar. Det tillför inget gentemot att ordna buden efter hur de kommer in till servern och skulle enbart komplicera systemet utan att tillföra någon kundnytta. Att kunderna skulle kunna fejka tidsstämplar är alltså av underordnad betydelse.

11) Hur upptäcker operativsystemet att en fil är exekverbar eller ej i resp. MacOS, Unix och LINUX? 1p

    Detta är alla varianter på UNIX/LINUX och där kontrolleras det interna formatet på filen, normalt Elektronic Library Format elf, där man först i filen skall hitta ett sk. magic number, som identifierar att det är en exekverbar fil – dsv det är det identifierar att filen är exekverbar. För att få försöka exekvera filen måste rättigheterna i i-noden (x-biten) vara satt.

12) Vilka för- och nackdelar har blockindelad allokering av filer! 2p

    Fördelen med blockindelad allokering är att man slipper problem med extern fragmentering och att man enkelt kan implementera metoder för att allokera nya block till en fil resp. avallokera block. Nackdelen är att blocken för en individuell fil med tiden kan bli utspridda och därmed kan det bli långsammare att acessa filen än om blocken låg fysiskt närmare varandra.

13) *¤ I distribuerade filsystem måste man behandla konsistensproblematiken.

a) beskriv vad konsistensproblematiken är! 1p

b) beskriv hur man i AFS försökt lösa konsistensproblematiken! 1p

a) Konsistensproblematiken bottnar i att man av effektivitetsskäl ofta cachar lokala kopior av filer, filsystem eller delar av dem. Konsistensproblemet är att samordna alla lokala uppdateringar av filer och filsystem så att alla får samma bild av filsystemet.

b) I AFS har man i huvudsak vad som kallas sessionssemantik, dvs. man skriver tillbaka ändringar till en fil när den stängs. Men man kan sätta upp det så att lokala kopior av en fil invalideras när filen accessas.

14) ¤ Antag att du skall göra en trådsäker implementation av biblioteksfunktionerna för minneshantering i C och har tillgång till vanliga (binära) lås.

a) Beskriv den enklaste trådsäkra implementationen i pesudokod! 1p

b) * Vad är att föredra för en trådimplementation – en enkel implementation baserad på en first-fit lista eller en implementation baserad på ”Quick-fit” listor? 1p

a) Antag att realloc()V är implementerad baserat på malloc() och free().

```
Lock memLock;
void *malloc(int s)     void free(void *)
{                       {
   lock(memLock);           lock(memLock);
   /* the code */           /* the code */
   .....                    .....
   unlock(memLock);         unlock(memLock);
}                       }
```

b) Man bör välja en implementation där den kritiska sektionen blir så kort som möjligt – dvs den snabbaste implementationen vilket är Quick-fit. Den kan också paralleliseras genom att man så länge man inte begär mer minne från systemet via brk() kan tillåta att de olika listorna i Quick-fit, quick-fit och överflödeslista kan accessas parallelt då varje lista kan vara en egen kritisk sektion (en tråd i taget per lista). .

15) Kalle hävdar att man skulle kunna göra operativsystem betydligt säkrare om man gjorde systemanropen i `exec()`−familjen till priviligierade instruktioner som bara fick köras av root-användare. Diskutera om han har rätt eller fel? 2p

Kalle har fel. Detta ökar inte säkerheten. Eftersom en process ärver användaridentiteten från den process den skapas från (om man inte explicit sätter den via ”su”) betyder det att alla processer som skall kunna exekvera exec() måste exekvera som root när de exekverar exec. Normalt byter man inte användaridentitet under exekveringen utan det skulle betyda att alla processer exekverar som root – dvs i praktiken skulle det bara finnas en användare – root – på systemet. Detta skulle kortsluta alla normala säkerhets- och accessystem som bygger på att olika användare har olika rättigheter.

16) Vilka två grundregler bör man följa när man gör/hanterar backuper? 1p

Att göra backuper tillräckligt ofta för att inte förlora för mycket data vid en krasch
Att ha backuperna fysiskt åtskiljda från det man backat upp så att inte backupmedia och dator står tillsammans och stjäls samtidigt eller brinner upp samtidigt.

17) * Antag att du skall implementera en video-on-demand server. Vilken typ av OS bör du välja och varför – ett vanligt fleranvändar OS som WINDOWS/LINUX/UNIX eller ett dedikerat realtids OS som VxWorks/FreeRTOS/OS-E? 1p

Ett video-on-demand system måste klara: mjuk realtid, kommunikation via nätverk och hantera filsystem. Detta finns stöd för i vanliga OS medan hårda realtids OS normalt inte är bra på nätverkskommunikation och/eller filsystem och man inte har behov av att klara hård realtid så är ett vanligt OS antagligen att föredra. Dessutom kan man inte kontrollera nätverksfördröjningar vilket gör att man inte vinner något på att använda ett hårt realtids OS.

18) Antag att en process på ett UNIX/LINUX system skriver data till fönstret det är anslutet till och till en fil på hårddisken.
a) vilket systemanrop används? 1p
b) beskriv hur systemanropen går till i från det att biblioteksfunktionen anropats till det att data hamnat på skärmen respektive hårddisken. 2p

a) Systemanropet är write().
b) Man anropar egentligen en C-biblioteksfunktion som lägger parametrarna på rätt ställe för kärnan att hitta och överför kontrollen till kärnan. I kärnan når man via fildescriptorn, som man skall skriva till, den i-nod som man skall skriva till. I-noden kan antingen direkt identifiera en I/O-enhet eller så identifierar den en fil på en viss I/O enhet. I båda fallen hittar man major-device number för enheten. Det används för att slå upp rätt drivrutin i drivrutinstabellen och i den drivrutinen anropas dess write()-funktion. På så sätt kommer olika och rätt write()-funktioner att anropas för olika I/O enheter.

19) I Windows mappas operativsystemet in i en process virtuella adressrymd. Vilka för- och nackdelar medför det? 1p

Fördelen är att man enkelt kan adressera de olika funktioner/metoder som finns i OS:et. Nackdelen är att den stor del, på system med 32 bitars eller mindre adressrymd, kan OS:et ta bort upp mot 50% av adressrymden för processen – vilket kan innebära stora begränsningar för hur stora applikationer man kan köra.

20) Vad i implementationen av Windows gör det troligt att Windows skulle kunna fungera bättre för speltillämpningar än UNIX/LINUX? 1p

Praktiskt taget alla speltillämpningar idag bygger på avancerad grafik. I Windows är grafikhanteringen integrerad i kärnan med an grafik i LINUX/UNIX normalt går via användarprocesser (som klient/server). Man kan därför anta att det blir färre context-switchar i Winows och effektivare grafik.

21) Antag att du är projektledare för utvecklingen av en ny processor. Ditt hårdvaruteam säger att de antingen kan implementera en enkel TLB med 128 poster eller två separata TLBer för data respektive instruktioner om 64 poster vardera. Vilket val bör du göra? 2p

Data och instruktioner har oftast olika accessmönster, dvs. olika lokalitet. Blandar man bra och dålig lokaliteti en och samma TLB blir slutresultatet dålig lokalitet. Därför kan man anta att man totalt sett får bättre prestanda om man har separata TLB:erna för data och instruktioner. Dvs. man bör

implementera två olika TLB:er. Vanlig missuppfattning/fel: man lagrar inte instruktioner eller data i sig i TLBn, där lagras bara kopior av adressöversättningar från sida till ram