Lösningsförslag till tentamen i ID2200, ID2206, IS1350 Operativsystem **(svarsförlsagen är ibland bara skissade - dvs inte fullständiga för full poäng)**

(fd 2G1520, 2G1504, 2G1113, 6B2019)
Tisdagen 2009-06-01 kl 1400-1800

Examinator: Robert Rönngren

Hjälpmedel: Inga
Tentamensfrågorna behöver inte återlämnas efter avslutad tentamen.

Ange på omslaget vilken kurs du tenterar och vilken termin du läste kursen.

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

ID2200, 2G1504
- Studenter som tenterar ID2200, 2G1504 besvarar inte frågor markerade med *****

IS1350, 6B2019
- Studenter som tenterar IS1350, 6B2019 besvarar inte frågor markerade med ¤

Rättning:
- ***OBS!!!! Alla svar*, även på delfrågor, måste ha åtminstonde en kortfattad motivering för att ge poäng**
- Resultat beräknas finnas rapporterat i LADOK och tentorna tillgängliga via STEX senast 17/6

Betygsgränser:
- Godkänt (E) garanteras från 50% av maximalt poängantal

Lösningsförslag: anslås på kursens webbsida efter tentamen (antagligen inte förrän 3-4/6)

Frågor:

1) Kalle flyttar ett program skrivet i ANSI-standard C från en maskin till en annan där bägge stöder ANSI-C och POSIX. Programmet använder enbart POSIX systemanrop, gör bara enkel I/O mot terminalen och använder inte mycket minne. Han kompilerar om programmet, som går igenom kompileringen utan felmeddelanden. När han skall exekvera programmet beter det sig felaktigt. Vilka är de troligaste orsakerna till problemen om det kraschar med segmentation fault resp. ger felaktiga svar på beräkningar? (2p)

Segmentation fault beror antagligen på att det finns något fel i den användarstyrda minnesallokeringen vilket kan visa sig genom att olika kompilatorer lägger ut data på olika sätt i minnet och eller att implementationerna av biblioteksfunktionerna malloc/free mfl skiljer sig åt. Beräkningsfel kan bero på att man har olika talområden, i.e. olika antal bitar, i implementationen av olika typer av tal. Little- och big endian kan man avskriva som felkällor eftersom kompilatorn hanterar detta så att alla lokala användningar av variabler blir korrekta. Det enda som skulle kunna bli fel med little/big endian är vid nätverkskommunikation - men det gör programmet inte.

2) Vilka är de grundläggande skälen till att man infört/har processer i praktiskt taget alla moderna operativsystem? (1.5p)

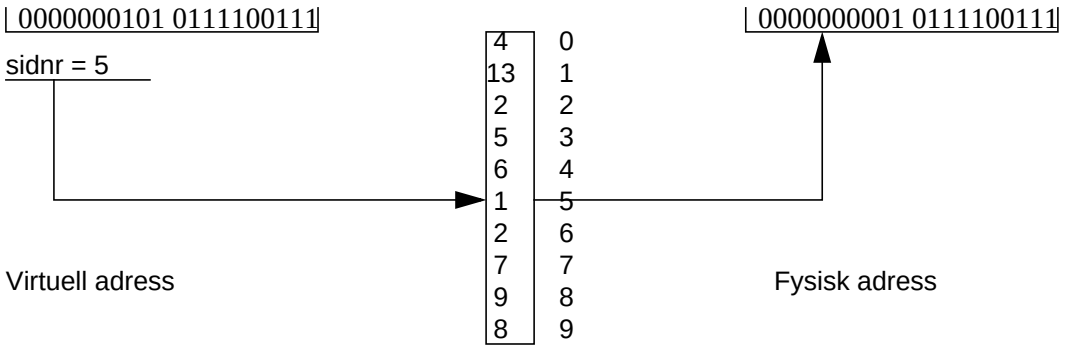
Tre av följande skäl ger full poäng:
- Att kunna öka utnyttjandegraden i datorsystemet genom att kunna överlappa beräkningar och I/O
- Få en bättre programmeringsmodell som avspeglar att mycket i ett datorsystem kan och bör ske som isolerade aktiviteter som går (pseudo)parallelt
- Enkelt kunna separera och tillåta flera samtidiga användare av ett datorsystem
- Kunna isolera fel till enskilda processer och på så sätt få robustare, mer feltoleranta system

3) Relokerbar kod, länkare och laddare är begrepp de flesta känner till. Men vad är skälet till att de behövs? (1p)

Flyttbar kod (relokerbar kod) dvs kod som inte innehåller absoluta adresser behövs om man inte känner till exakt var en kodmodul skall laddas i minnet redan när den skapas. Två viktiga skäl till detta är: Om man vill kunna kombinera olika kodmoduler, t.ex olika biblioteksfunktioner, i ett och samma program - då kan man inte utnyttja absolutadresserad kod utan måste kunna länka ihop koden och reda ut hur olika delar av koden, adresserna, hamnar i förhållande till varandra - detta kräver en länkare. Man kan heller inte använda abolutadresserad kod om

koden skall kunna ladas in på godtyckliga eller olika adresser i minnet som kan ske vid virtuellminneshantering. Laddaren behövs för att kunna ladda in koden i minnet.

4) Antag en fysisk adressrymd om 2^{20} och att man har virtuellminne med sidhantering där man använder en sidstorlek på 1K. Till vilken fysisk adress översätts följande (binära) virtuella adress: 00000001010111100111 givet följande sidtabell: [4,13,2,5,6,1,2,7,9,8] Indexering sker som i C. Svara Binärt! (1p)



5) Givet ett primärminne med 4 ramar hur många sidfel får man för följande referenssträng vid ren demand paging för sidutbytesalgoritmerna FIFO, LRU och OPT? Referenssträng: 1,3,7,8,2,6,1,7,9,4,2,3,6 (1.5p)

```
FIFO 1 3 7 8 2 6 1 7 9 4 2 3 6
     x x x x x x x x x x x x x  13 sidfel
     1 3 7 8 2 6 1 7 9 4 2 3 6
       1 3 7 8 2 6 1 7 9 4 2 3
         1 3 7 8 2 6 1 7 9 4 2
           1 3 7 8 2 6 1 7 9 4

LRU  1 3 7 8 2 6 1 7 9 4 2 3 6
     x x x x x x x x x x x x x  13 sidfel
     1 3 7 8 2 6 1 7 9 4 2 3 6
       1 3 7 8 2 6 1 7 9 4 2 3
         1 3 7 8 2 6 1 7 9 4 2
           1 3 7 8 2 6 1 7 9 4

OPT  1 3 7 8 2 6 1 7 9 4 2 3 6
     x x x x x x     x x   x      9 sidfel
     1 1 1 1 1 1 7 2 2 2 6 6 6 ►Exakt inbördes ordning mellan sidor som inte kommer
       3 7 7 7 7 2 6 6 6 2 3 3   att användas någon gång i framtiden är inte viktig
         3 3 2 2 6 7 7 7 7 7 7
           8 3 6 1 1 9 4 4 4 4
```

6) ¤ Förklara på vilken/vilka nivå(er) (kernel/user) som det är lämpligt att implementera server respektive klient i ett distribuerat filsystem! Om det finns olika alternativ jämför dem! (1p)

Under antagandet att vi vill undvika onödiga context-byten, onödiga kopiering och att man vill hålla kärnan så ren som möjligt samtidigt som det ibland kan öka säkerheten att ha saker i kärnan istället för på användarnivå så kan man resonera på följande vis:
En användare av ett filsystem vill helst att ett ev. distribuerat filsystem skall vara osynligt. Man accesar ett filsystem via systemanrop. Alltså tas anropen hand om i kärnan. Befinner man sig i kärnan är det enklast och effektivast att hantera klientdelen där. Dessutom skulle man kunna få problem om man skulle försöka anropa en klientdel i användraläge som i sin tur kanske försökte göra systemanrop för att accessa en fil genom systemanrop.... Normalt implementeras alltså klientdelen i kärnan.
Eftersom kommunikationen mellan klient och server går vi ett nätverk, normalt IP, så kommer en begäran till servern som ett paket til en port. På serversidan finns två huvudval: Att implementera serverdelen i kärnan som t.ex i NFS. Fördelen då är att man slipper extra context-byten som sker om man skickar data vidare till en server implementerad som en användarprocess, men man tvingas å andra sidan implementera servern i kärnan vilket är lite svårare. Om man istället implementerar servern i användarläge så kan servern bli ett helt vanligt anvädraprogram som utnyttjar systemanrop och inte behöver bry sig om detaljer i implementeringen av det lokala filsystemet den utnyttjar.

Summering: Klienten implementeras normalt i kärnan. Serverna kan implementera antingen i kärnan (NFS) eller i användarläge (AFS)

7) Förklara hur implementation av ett filsystem med indexerad allokering via i-noder fungerar! (1p)
Hur implementeras directoryn i ett filsystem med indexerad alokering baserad på i-noder? (1p)

Se föreläsningsanteckningarna! Kort så är en i-nod en datastruktur som innåller information om filen som: länkräknare, typ (directory, fil), ägare, skapare, rättigheter, tidpunkter för skapande/modifiering/senaste access, och pekare till diskblock där filens innehåll lagras. När en fil skapas allokeras en i-nod till filen och en hård länk skapas från det directory där filenm skapas.

Ett directory implementeras som en fil, dvs med en i-nod. Men i i-noden flaggas att det är ett directory. Till ett directory får det bara finnas en hård länk. Innehållet i directoryt består i sin enklaste form av översättningar <namn -> i-nodsnummer> för directoryn och filer till vilka man har hårda länkar, och <namn -> sökväg> för mjuka länkar. Formatet på posterna i ett directory är filsystemsberoende.

8) Vilken information måste finnas i monteringspunkten för ett monterat filsystem? (1p)

Att det är ett monterat filsystem, typ för det monterade filsystemet, pekare till drivrutiner för filsystemet, vilka fysiska enheter filsystet finns på

9) Förklara för var och en av följande: användarkommandot `kill`, biblioteksfunktionen `kill()` och systemanropet `kill()`, om/varför det kan finnas en eller flera instanser/implementationer av dem i ett system och hur man i så fall vet vilken version som används! (3p)

Användarkommando: Implementeras antingen direkt i kommandotolken eller som en exekverbar fil som finns i sökvägen. Det kan alltså finnas fler än en implementation i ett och samma system.
Biblioteksfunktion: gränssnitt för att underlätta för användaren att göra systemanrop (som sker i assembler). Kan finnas flera av då det kan finnas flera bibliotek för ett och samma språk och då det kan finnas olika biblioteksfunktioner för olika språk.
Systemanrop: om vi definerar det som det gränssnitt kärnan har mot omvärlden finns bara en instans

10) I många hårddiskenheter finns en buffert/cache. Beskriv två vanliga användningar för den och hur de påverkar prestandan! (2p)

Som mellanlagring av data som lästs från disken men inte hunnit skickas över bussarna, som mellanlagring för data som skickats för skrivning till disken men ännu inte hunnit skrivas fysiskt till disken, för att mellanlagra och omordna läs- och skrivbegäran för att implementera en mer effektiv diskarmsschemaläggning, för att implementera read-ahead av data från disken. Read-ahead och mellanlagring gör att man slipper att direkt synkronisera överföring av data till/från med bussarna - vilket i sig skulle vara mycket svårt - och sparar på så sätt tid i systemet.

11) Kalle föreslår att man genom att modifiera en schemaläggningen för processer på en modern PC för att öka prestandan. Det han föreslår är att om det finns flera körklara processer så skall man alltid välja i) procesen med högst prioritet och ii) om det finns flera processer med samma prioritet skall den väljas som exekverade senast.
a) Kommer detta att öka prestandan? Varför/varför inte? (1p)
b) Vad finns för övriga nackdelar/fördelar med iden? (1p)

Prestandan skulel antagligen öka då man kan utnyttja cachear, TLB mm bättre i systemet. Däremot skulle man antagligen få ett orättvist system med svältning inte bara mellan prioriteter utan även inom en prioritet.

12) I ett modernt datorsystem med stöd för sidindelat virtuellminne påverkas prestandan i hög grad av processernas lokalitet.
a) Vilka typer av lokalitet finns? Förklara dem! (1p)
b) Hur påverkas prestandan av dålig lokalitet? (2p)

i) Rums- och Tidslokalitet. Rumslokalitet innebär att accesserna sker till en begränsad delmämngd av den data som kan accessas, normalt till närliggande adresser i den fysiska eller virtuella adressrymden. Tidslokalitet innebär att eaccesserna sker inom begränsad tid. För att få bra lokalitet i praktiken krävs oftast både bra rums- och tidslokalitet.
ii) vid dålig lokalitet påverkas först cachear - vilket leder till att minnessacesstiden går från att vara mindre än för att accessa primärminnet till att vara i storleksordningen av att accessa primärminnet. Nästa steg är att man får

fler TLB-missar vilket leder till minst fördubblad minnessaccesstid gentemot en access med en TLB träff eftersom sidtabellen i minnet måste accessas. Nästa steg är att sidfelsfrekvensen ökar vilket kan ge drastiskt försämrad prestanda och i värsta fall får man thrashing vilket innebär att väldigt lite reelt arbete blir utfört.

13) a) Förklara begreppet "working set" i samband med sidindelat virtuellminne! (1p)
b) Hur kan man använda begreppet för att avgöra vilka/hur många processer som skall tillåtas vara aktiva (dvs in-swappade) samtidigt? (1p)

Working-set är de sidor en proces använder inder en begränsad tid.
Om summan av processernas working-sets är större an det totala antalet tillgängliga sidor i system får man problem med hög sidfelsfrekvens. Alltså bör inte fler processer få swappas in än att summan av deras working-set ryms i minnet för att prestanda skall bli acceptabel.

14) Jämför trådar i användarläge, trådar i kärnläge respektive processer med avseende på: i) möjlighet att bygga feltoleranta system, ii) portabilitet (2p)

Fel isoleras normalt till en process. Fel i en tråd påverkar oftast hela processen och alla trådar i processen. Därför är det oftast enklare att bygga feltoleranta system baserat på processser.

Att portera tillämpningar som strikt använder standarder för systemanrop och trådar/processr är enkelt att göra mellan system som stödjer standarden. Vill man flytta en trådapplikation till ett system som inte stöder en standard för trådar är det enklare om man har ett system för trådar i användarläge, där run-time systemet för trådarna är byggt på en standard som målsystemet stöder.

15) I distribuerade system väljer man oftast att inte adressera en server/klient med dess PID utan man använder IP-adress och portnummer. Varför? (1p)

IP-adressen är knuten till ett interface och portnumren kan standardiseras. PID kan däremot variera och en proces får PID tilldelat dynasmiskt vid skapandet och processen kan normalt inte räkna med att få ett visst PID. Så för att kunna identifiera en tjänst behövs välkända id som IP-adress och portnummer.

16) a) Beskriv "dining philosophers" problemet! (0.5p)
b) Ge två olika lösningar till problemet och bevisa att de undviker baklås (deadlock) (2p)

I dining philosophersproblemet finns ett runt bord med ett antal sittplatser. I mitten på bordet står en skål med spagetti och mellan varje par av sittplatser ligger en gaffel. En filosof som vill äta sätter sig på en ledig plats och plockar upp bägge gafflarna som finns bredvid sittplatsen innan han/hon kan börja äta. En filosof äter aldrig oändligt länge. En filosof lämnar inte sin plats innan han/hon kunnat ta bägge gafflarna och ätit sig mätt.

För att visa att man ionte har deadlock måste man kunna visa att något av Coffmans villkor inte gäller/kan uppfyllas. Några förslag: Inför en premption via hovmästare som kan ta gafflar från filosofer och ge till andra så att någon får två gafflar och kan äta - bryter villkoret om no-preemption. Låt bara N-1 filosofer sitta till bords samtidigt - bryter villkoret om cirkulär väntan.

17) a) Förklara begreppet live-lock! (1p)
b) Om Kalle och Stina vill ringa till varandra finns då någon risk för att de ska råka hamna i live-lock? Vad gör att risken finns/inte finns? (1p)

live-lock: två eller flera processer synkroniserar med varandra på så sätt att de inte hamnar i dead-lock men att de heller inte kommer ur synkroniseringen.

ja man kan koppla upp samtal mot varandra så att bägge upprepade gånger får upptaget-ton. I praktiken varar den här typen av live-lock inte i oändlighet eftersom det är osannolikt att båda kommer att ringa varandra så väl synkroniserat som krävs.

18) Du vill kontrollera om ett program, som du inte har tillgång till källkoden till, kan vara en trojan eller innehålla ett virus och vad det i så fall gör. Vilka standardverktyg/systemprogramvara kan då vara lämpliga att använda i ett UNIX/LINUX system? Och hur kan de vara användbara? (det handlar alltså inte om anti-virus programvara eller barndväggar och vi antar att du kör det på en maskin utan kontakt med omvärlden där ev bieffekter av viruset/trojanen kan försummas). (2p)

Med en disassembler kan man få fram en assemblerkod från den exekverbara filen - vilket gör det enklare att se vad programmet gör.

Med TRUSS/STRACE kan man se vilka systemanrop som görs och parametrarna till dessa. Eftersom allt intressant som kan orsakas av en vanlig användarprocess i ett datorsystem går via systemanrop kan man på så sätt se vad programmet/processen gör.

19) Vilka av följande filtyper kan dölja/innehålla ett virus?
    i) ASCII-text
    ii) postscript
    iii) MS-Word (1.5p)

källkodsvirus
PS - scripting språk som interpreteras - kan innehålla virus
macro-virus

20) * Vad är det som gör att man i många sammanhang ofta föredrar Rate Monotonic Scheduling före Shortest Job First som schemaläggningsstrategi för realtidsystem trots att SJF kan schemalägga större laster? (1p)

RMS bygger på fix prioriter vilket drastiskt förenklar schemaläggningen

21) a) Var implementeras användargrafiken (t.ex fönsterhantering) i Microsofts OS som t.ex WINDOWS XP respektive i LINUX? (1p)
    b) Vad har strategin/strategierna för för- och nackdelar? (1p)

a) MS - i kärnan, UNIX/LINUX - som användarprocesser
b) MS- ger en de facto standard, kan ge bättre prestanda, låter inte användaren välja gränssnitt på ett enkelt sätt, mindre flexibelt
UNIX/LINUX - kräver att man enas om standarder, kan ge sämre prestanda (fler context-sitchar), mer flexibelt

"Det krävs ett helt nytt sätt atttänka för att lösa de problem vi skapat med det gamla sättet att tänka"
Albert Einstein