Kortfattade läösningsförslag till
Tentamen i IS1350, ID2206, ID2200 mfl Operativsystem
fredag 2010-10-22 kl 1400-1800

Institution: ECS/ICT

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

Rättning:
- ***Alla svar***, även på delfrågor, måste ha en kortfattad motivering för att ge poäng
- Resultat beräknas anslås senast tre arbetsveckor efter tentamen

Betygsgränser:
- Godkänt (E) garanteras från 50% av totalt antal poäng på tentamen

---

Lösningsförslag: anslås på kursens webbsida veckan efter tentamen

Frågor: (som normalt bara kräver kortfattade svar 1-10 rader)

1) Förklara vad som händer om man, inloggad som root, startar följande lilla program och hur det påverkar datorsystemet: (1p)

```
int main(int argc, char ** argv)
{ while() fork(); }
```

Svar: Processen skapar en oändlig mängd kopior av sig själv (fork()) som i sin tur skapar kopior av sig själva. Detta resulterar i att det skapas en exponentiellt ökande mängd processer. Det medför att maskinen överbelastas, att processtabellen fylls och till slut går det inte ens att logga in på maskinen.

2) Beskriv hur ett systemanrop sker på assemblernivå. (1p)

Svar: Parametrar och kod för systemanropet läggs normalt i register och kontrollen överförs till kärnan via en TRAP-instruktion

3) a) Rita en schematisk bild av adressöversättningen i ett system med sidindelning och TLB och en-nivås sidtabell. (1p)
b) Antag att minnesaccesstiden är **M**, söktiden i TLBn är **t** och att träffsannolikheten i TLBn är $\mathbf{P_1}$. Antag att vi vill byta implementation av sidtabellen till en två-nivåers sidtabell. Hur stor måste träffsannolikheten i TLBn $\mathbf{P_2}$ då vara för att vi ska få samma upplevda (genomsnittsliga) minnesaccesstid som för fallet med en en-nivås sidtabell. (2p)

Svar: a) se förläsningsanteckningarna om minneshantering
b) Effektiv minnesaccesstid med en-nivås sidtabell: $P_1t + (1-P_1)(t+M) + M$,
Med två nivåers sidtabell: $P_2t + (1-P_2)(t+2M) + M$. Sätt dessa lika och lös ut, ger: $P_2 = (1+ P_1)/2$

4) Kalle har skrivit följande funktion som han hävdar fungerar ypperligt då han testat den. Ser du några brister/ fel i koden och vad kan hända? (2p)

```
int read_and_add_positive_numbers(void)
{  int tal=1, i=0, j, vec[], sum=0;
   while(tal>0)
   {
    printf("Ge ett tal:");
    if(scanf("%d",&tal) == 1) vec[i++] = tal; // Success
    else exit(1); // Exit on failure
   }
   for(j=0; j<i; j++) sum += vec[j];
   return sum;
}
```

Svar: Problemet är att det inte reserverats något lagringsutrymme för element i vec[]. Ett ytterligare problem är att det heller inte finns någon kontroll på hur många tal som läses in. Detta program ger upphov till sönderskrivning av stacken, sk." buffer-overflow"

5) Många moderna OS har en virtuell adressrymd om 2^{64} (16 exabytes). I teorin kan man alltså köra jättelika processer men hur mycket minne kan en process i praktiken utnyttja på en given maskin? (1p)

Svar: Den största process som kan köras är: (storlek på Primärminne (RAM) **+** storlek på SWAP-arean på disken **-** den del som upptas av operativsystemet)

6) Moderna datorsystem är beroende av bra lokalitet.
a) vad betyder lokalitet? (1p)
b) beskriv vilka delar av ett modernt datorsystem (hårvara och mjukvara) som påverkas negativt av dålig lokalitet, i vilken ordning de påverkas när lokaliteten går från bra till dålig och hur de påverkar prestandan(3p)

Svar: a) Lokalitet är ett mått på vilka minnesadresser som accessas och när i tiden det sker. Man skiljer på rumslokalitet som anger hur samlade accesserna är i minnesrymden och tidslokalitet som anger hur nära i tiden man accesser till minnespositioner ligger. För att få bra prestanda ur ett datorsystem krävs både bra rums och tidslokalitet.
b) Vid dålig lokalitet påverkas cache-systemen först vilket ger sämre minnesaccesstider. När lokaliteten blir så pass dålig att att accesserna sprids över fler sidor än vad som ryms i TLBn börjar virtuellminnessystemen fungera sämre. En TLB-miss ger antagligen minst dubbel så lång minnesaccesstid som en minnesaccess där man får en TLB träff. Vid ännu sämre lokalitet och fullt primärminne för man problem med höga sidfelsfrekvenser som ger mycket pageing, då sjunker prestanda rejält och systemet kan vid sk. thrashing få extremt lite nyttigt arbete utfört.

7) I den virtuella adressrymden finns normalt tre segment som används för att lagra delar av en process data i : `Data`, `BSS` och `Heap`. Förklara vad dessa används till och varför man skiljer på dem. (2p)

Svar: Data och BSS används för variabler som har lagringsklassen static, dvs för globala variabler, static deklarerade variabler och för textkonstanter. Data innehåller data som har ett initialvärde skilt från noll medan BSS innehåller data med initialvärde noll (eller möjligen odefinierat). För data som ligger i Data-segmentet måste deras värden finnas i den exekverbara filen, medan man för BSS bara behöver spara hur många bytes som skall reserveras för BSS. Dvs genom att ha BSS kan man minska storleken på den exekverbara filen (den filen är t.ex i ELF format). Heap används för användarstyrd allokering t.ex via malloc(), free(), realloc().

8) Antag att du i ett UNIX-system vill skriva ut 20 bytes data från en buffert (char `*buf`) till både `stdout` och till en fil på en hårddisk med `fd=4`
a) hur ser systemanropen ut för utskrifterna (1p)
b) förklara vad som sker i kärnan vid utskrifterna, beskriv de viktigaste stegen, och var man hittar nödvändig information för att utskrifterna skall hamna rätt (2p)

Svar: a) write(1, buf, 20); write(4, buf, 20);
b) Kärnan tar hand om systemanropet via TRAP-handlern. Den identifierar att det handlar om ett systemanrop och kontrollerar (t.ex via register) vilket systemanrop som skall utföras. När det handlar om write() så måste man veta till vilket device man skall skriva. Det kontrolleras via fildeskriptorn som används för att indexera en tabell där man håller information från den i-node som filen/strömen förknippas med. Där finns bland annat information om device-id. Device-id används för att slå upp i drivrutinstabellen vilken drivrutin som skall användas. Ur den vektor/ struct som implementerar gränssnittet mot drivrutinen väljs den funktionspekare som pekar ut write-funktionen för drivrutinen och via den anropas funktionen med de parametrar som skicklaades med till systemanropet.

9) Hur brukar man säkerställa att en användare inte kan exekvera en fil med t.ex data eller text även om den har rättigheter i filsystemet (protection) som medger exekvering? (1p)

Svar: Genom att ha ett internt format för exekverbara filer som med stor sannolikhet identifierar att det handlar om en exekverbar fil. T.ex kan man kräva att en exekverbar fil måste inledas med ett specifikt, unikt stort heltal.

10) Beskriv kortfattat hur ett katalog (directory eller mapp) implementeras i ett UNIX-filsystem och vilken information man kan förväntas hitta i katalog för en vanlig fil (som katalogen har en hård länk till) (1p)

Svar: En katalog implementeras som en fil men med en bit satt i mode-delen av i-noden som indikarar att det är en katalog. En normal post i en katalog, en sk. hård länk, innehåller filnamn och i-nodsnummer för den i-node som implementerar filen.

11) Kalle hävdar att context switch tiden att byta mellan två kärntrådar är betydligt mindre än att byta mellan två processer - har han rätt eller fel? Förklara någorlunda utförligt varför? (2p)

Svar: Kalle har rätt. Det är betyldigt mindre data som måste bytas om man byter mellan två trådar. Då räcker det att byta stackpekare, programräknare och registerinnehåll. Medan man måste byta betydligt mer information om man byter mellan två processer som: filer, sidtabell och administrativ information om processen etc.

12) a) Vad betyder begreppet "page-out"? (1p)
b) Vad får man aldrig göra page-out på? (1p)

Svar: a) Att man flyttar en sida från primärminnet till SWAP-arean på disk
b) OS-koden som implementerar page-in

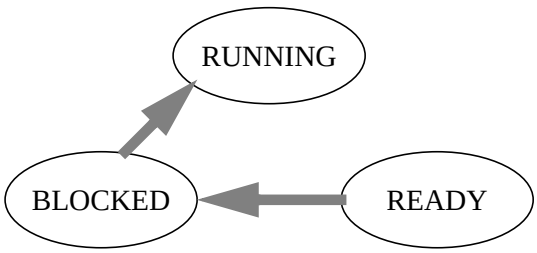
13) Schemaläggningsmetoderna på multi-core processorer måste ta hänsyn till fler faktorer än på single-core processorer. Beskriv kortfattat varför man måste ta hänsyn både till var och när man schemalägger trådarna i en flertrådad applikation (uppenbart är naturligtvis att en processorkärna måste vara ledig för att man skall kunna schemalägga något på den). (1p)

Svar: Trådarna i en applikation kommunicerar och synkroniserar normalt oftare med varandra än med andra processer och deras trådar. Därför bör de schemaläggas så att de får köra samtidigt. Ofta kan det vara fördelaktigt att köra en tråd på samma processorkärna som den körde innan eftersom man kankse kan återutnyttja cache och TLB-innehåll. Har man assymetriska accesstider till olika resurser som minne och I/O-enheter bör man också schemalägga trådarna så att de får lägtsa möjliga accesstid till de resuresr de använder.

14) Vilka av följande funktioner kan man använda inifrån kärnan på ett LINUX system (förklara varför!):
stat(), scanf(), printf(), read(), listen(), write() (1.5p)

Svar: Ingen, alla är systemanrop eller biblioteksfunktioner som använder systemanrop. Och systemanrop gör man för att gå från user-mode till kärnan och inte innifrån kärnan.

15) Processchemaläggning involverar normalt tillstånden READY, RUNNING samt BLOCKED. Förklara varför man (i princip) aldrig implementerar de gråmarkerade tillståndsövergångarna i tillståndsgrafen nedan i schemaläggningsalgoritmer? (2p)



Svar: Om en process skulle kunna gå från READY till BLOCKED så skulle det bero på att den process som exekverar gjort något som skulle blockera den process som var READY. För att upptäcka om något sådant skulle inträffa skulle den körande processen alltså behöva kontrollera om varje typ av åtgärd den utför som skulle kunna blockera en annan process har blockerat någon. Det är både oeffektivt och i praktiken omöjligt att implementera. Enklare är att låta den process som ev. blivit blockerad upptäcka det själv när den får köra.
Om en process om är BLOCKED skulle kunna gå direkt till RUNNING utan att passera READY så skulle man gå förbi schemaläggaren och den policy och mekanism den har för att välja vilken process som skall få exekvera.

16) Antag att du vill göra det möjligt att blockera vissa användare från att starta vissa program vid vissa tidpunkter. T.ex hindra barn från att starta webbläsare mitt i natten. Beskriv hur detta skulle kunna implementeras. (2p)

Svar: Alla program som körs i processer startas via ett systemanrop, t.ex exec(). Det gör att man i TRAP-handlern kan lägga till kod som kontrollerar vem som försöker starta vilket program och när detta sker. Den informationen kan kontrolleras mot den databas som innehåller mer detaljerade accessrättigheter än de som t.ex tillhandahålls via filsystemet.

17) Antag att man i ett system inför ömsesidig uteslutning för att undvika problem med race-conditions. Kan det i sig skapa några nya problem? Om det kan skapa nya problem beskriv dem i så fall kortfattat. (2p)

Svar: När man inför ömsesidig uteslutning inför man också möjligheten till synkroniseringsrelaterade problem som låsning (deadlock) och svältning (starvation). Låsning innebär att man har en grupp av processer där samtliga processer i gruppen väntar på en resurs (eller en händelse) som bara kan fås av någon annan i gruppen och att väntan uppfyller Coffmans villkor så att ingen process någonsin kommer att kunna komma vidare. Dvs att de är låsta för all framtid. Svältning innebär att en process hela tiden blir undanträngd av andra processer så att den aldrig får tillgång till de resurser den behöver för att komma vidare.

18) Vad skiljer hårda och mjuka realtidsystem? (1p)

Svar: För att ett hårt realtidsystem skall fungera korrekt måste samtliga deadlines hållas. I ett mjukt realtidsystem bör alla deadlines hållas.

19) Vad innebär det att avbrottsrutiner kan vara stackade? (1p)

Svar: Att avbrottshanteraren har (kan ha) en lista av funktioner för varje avbrottsnummer. När ett avbrott inträffar så anropas den första avbrottsfunktionen i listan av avbrott. Om den funktionen inte returnerar att den hanterat hela avbrottet anropas nästa funktion i listan osv.

20) Apple hävdar att deras OS-X är säkrare än Windows bland annat därför att alla drivrutiner är förinstallerade. Har de rätt eller fel? Förklara varför! (1p)

Svar: Drivrutiner installeras i kärnan och får på så sätt normalt rättigheter att gör allt som kan göras från kärnan. ATt installera en drivrutin som innhåller ett virus ger viruset access rakt in i kärnan. Att begärnsa möjligheten/ behovet för användaren att själv installera drivrutiner ökar alltså säkerheten.

21) Hur många sidfel får man givet ett primärminne om 4 ramar som är tomt från början, ren demand-paging och följande referenssträng: 1, 4, 2, 4, 5, 4, 8, 6, 2, 4, 3, 6, 5, 4, 6, 3
För sidutbytesalgoritmerna FIFO, LRU och OPT (1.5p)

Svar: Räkningarna måste redovisas: FIFO: 9, LRU: 9, OPT: 7

"Hur mycket vore möjligt om ingen försökte det omöjliga?"
Arne Hirdman