Lösningsförslag till tentamen i IS1350/ID2200/ID2206 Operativsystem

måndag 2013-01-07 kl 1400-1800 (GMT+1)

Skolan för Informations och Kommunikationsteknik

Examinator: Robert Rönngren

Hjälpmedel: Inga. Mobiltelefoner skal vara avstängda och får inte finnas vid skrivplatsen.
Tentamensfrågorna behöver inte återlämnas efter avslutad tentamen.

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

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

Rättning:
- ***Alla svar*, även på delfrågor, måste ha åtminstone en kortfattad motivering för att ge poäng**
- Resultat beräknas anslås inom 3 arbetsveckor
- Frågor markerade med ¤ ingår inte i tentamen för kursen ID2200
- Frågor markerade med * ingår inte i tentamen för kursen IS1350

Betygsgränser:
- Godkänt (E) garanteras från 50% av det totala poäng antalet för respektive kurs

Lösningsförslag: anslås på kursens webbsida efter tentamen

Frågor:

1) a) Varför bör man skilja på mekanism och policy vid implementation av operativsystem? (1p)
b) Ge ett exempel på hur man i praktiken skiljer på mekanism och policy i någon viktig del av ett modernt operativsystem (1p)

a) (se boken sid161-162) Det man vinner på att separera mekanism (att systemet kan utföra en tjänst) från policyn (hur man använder tjänsten) är att man får ett mer flexibelt system som har bredare användningsområde.
b) Typiska exempel på hur man separerar mekanism och policy är t.ex. i schemaläggning. Om man har en pre-emptiv shemaläggning baserad på prioriteter så är mekanismen att schemaläggaren alltid ser till att den körklara process som har högst prioritet också är den som får exekvera. Användaren tillåts själv bestämma hur han/hon sätter prioriteter på sina processer och kan därmed också bestämma policyn, t.ex att en dag ge hög prioritet åt beräkningstunga jobb och kanske en annan dag (på helgen) ge hög prioritet åt ett spel. I förlängningen kan man också tillåta en användare att inte bara bestämma policyn genom att sätta värden utan också kanske genom att tillhandahålla funktioner som bestämmer policyn. Ett exempel kan vara sorteringsfunktionen qsort(3C) där användaren tillhandahåller en funktion som används för att jämföra två element som skall sorteras, Då kan sorteringspolicyn enkelt ändras från till exempel stigande till fallande ordning genom att byta funktion som skall användas vid sorteringen mellan "<" och ">" och qsort(3C) kan användas för att sortera godtyckliga typer av element. Ett annat exempel kan vara att man i system med virtuellt sidindelat minne bör kunna byta policies både för hur adressöversättningen sker via olika implementationer av sidtabellen och policy för sidutbyte genom att ha olika algoritmer för sidutbyte.

2) Förklara begreppen "process" och "tråd" och relationen mellan dem. (1p)

(sid 38-40, 95-117) En process är ett program under exekvering. Processer skapas genom systemanrop, är kända av OS:et och upptar en plats i processtabellen, dvs har ett unikt processID. Till en process är knutet resurser i form av minnesareor (text, data, BSS, heap och stack), registeruppsättning, programräknare, öppna filer. En process har minst en exekveringsväg aktiv, dvs en programräknare och en stack på vilken aktiveringsposter för anropade funktioner läggs.
En tråd är en exekveringsväg genom en process, dvs en tråd har en egen stack och registeruppsättning inkluderat PC. En process kan innehålla en eller flera trådar. Trådar kan antingen implementeras så att de är kända och schemaläggs av kärnan, sk. kärntrådar, eller på användarnivå i form av biblioteksrutiner i vilket fall kärnan inte känner till dem.
För system som inte har kärntrådar är processen schemaläggningsenhet. Har man kärntrådar blir processen ofta reducerad till att vara en kontext för trådarna med för trådarna gemensamma resurser.

3) Vissa operativsystem använder variabel timeslice eftersom man tycker det ger fördelar. Kalle hävdar att man kan nå samma effekt med fix timeslice och prioriteter. Resonera kring om Kalle har rätt eller fel! (1p)

Timeslice och prioritet är två olika dimensioner av schemaläggning. Timeslice bestämmer hur länge en tråd/ process maximalt får exekvera varje gång den schemaläggs medan prioriteten bestämmer vilken tråd/process som får exekvera. Dessa interagerar om man har preemptiv schemaläggning då en tråd/process som blir körklar kan komma att avbryta en lägre prioriterad dito i förtid. Frågan är om man kan nå samma effekt med enbart prioriteter? Kalle har i det här fallet i delvis rätt. Genom att använda variabla prioriter kan man inom ett givet tidsintervall låta processer/trådar exekvera i princip lika länge som med variabla timeslices genom att schemalägga dem ett antal ggr med ett kortare tidinterval. Nackdelen är att det kan leda till fler context switches.

4) Beskriv kortfattat vilket hårdvarustöd som är lämpligt att en datorarkitektur har om man skall kunna implementera sidindelat virtuellminne effektivt och hur dessa enheter fungerar. (2p)

För att systemet skall bli effektivt och nå de mål man vill nå med virtuellt minne gäller oftast:
1) Normalt vill man utöka den virtuella minnesrymden jämfört med den fysiska (primärminnet) vilket kräver en hårddisk eller annat sekundärminne
2) För att snabba upp överföringen av sidor till och från sekundärminne och nå rimliga prestanda krävs normalt möjlighet att göra överföringen mha DMA
3) För att snabba upp översättningen av virtuella adresser till fysiska behövs en Memory Management Unit (MMU)
4) För att nå rimlig minnesaccesstid behövs också en Translation Lookaside Buffer TLB i MMUn. TLBn cachear de mest frekvent använda adressöversättningarna från <PID, sidnr> till <ramnr) och används av MMUn för att kunna slå upp dessa snabbare än om man tvingas göra uppslagningen via sidtabellen

5) Jämför fördelar och nackdelar med att implementera en sidtabell med datastrukturerna två-nivåers eller hashad inverterad. (1p)

Två-nivåers tabell: Fördelar: ger (algoritmiskt) en konstant accesstid där man i normalfallet hittar information om en sida med två minnesreferenser. Eftersom varje process normalt har en egen sidtabell om den implementeras som en två-nivåerstabell så kan den också användas för att hålla information om var ev. ut-pageade sidor finns på SWAP arean. Nackdel: är att den skalar dåligt (minnesutnyttjandet för tabellen i sig kan bli stort) om man har stora virtuella adressrymder.

Hashad inverterad sidtabell: Fördelar: eftersom storleken är proportionerlig mot primärminnet så är dess storlek oberoende av storleken på den virtuella adressrymden, dvs. man kan hantera mycket stora virtuella adressrymder utan problem. Nackdelar: Något mer komplex implementation. Även om accesstiden, som är proportionerlig mot. antal sidor som mappar mot samma hashvärde i genomsnitt bör vara lågt (nära 1) så kan man drabbas av att flera sidor mappar till samma hashvärde och accesstiden i enskilda fall blir längre (dvs. sämre värstafalls prestanda jämfört med en två-nivåers tabell). Eftersom tabellen bara mappar sidor som finns i minnet krävs separata datastrukturer för att hålla reda på var sidor som är ut-pageade till swap arean finns.

6) Ge en enkel tumregel för hur stor SWAP (backing store) area man bör reservera på en persondator med ett OS som har virtuellt sidindelat minne. Förklara också vad som ligger bakom tumregeln. (1p)

Normalt rekommenderas att man inte har större SWAP-area än 1-2 ggr storleken på primärminnet. Skälet är helt enkelt att om man har större SWAP-area än så, så kommer man, om processerna utnyttjar hela SWAP-arean och dessa inte är relativt få till antalet och de inte uppvisar extremt god lokalitet, att få relativt många sidfel och riskera att hamna i thrashing-beteenden och därmed få väldigt dålig prestanda.

7) Beräkna antalet sidfel givet ren demand paging, 4 ramar och följande referenssträng:
4, 2, 8, 3, 1, 4, 3, 5, 1, 6, 8, 4, 6, 4, 3
för sidutbytesalgoritmerna:
i) LRU (0.5p)
ii) FIFO (0.5p)
iii) NRU (0.5p)

i) 11 sidfel (beräkningar måste redovisas)
ii) 10 sidfel (beräkningar måste redovisas)
iii) Korrekt svar: Går ej att beräkna med given information då algoritmen utnyttjar information om när en sida modifierats, samt när referensbiten nollställs.

8) Antag att vi har ett Unix/LINUX filsystem. Beskriv vilken information som skapas i katalogen respektive i i-noden för filen om man:
i) skapar en ny fil i katalogen (1p)
ii) skapar en hård länk till en existerande fil (1p)
iii) skapar en mjuk länk till en existerande fil (1p)

i) en ny i-nod allokeras för filen. i-noden innehåller all information om filen utom det namn den är känd under i den katalog där den skapades. Bland annat finns och uppdateras information i i-noden om ägare, skapare, tider för skapande och senaste modifiering, accessrättigheter för ägare/grupp/andra, var datablocken finns på disk och hårdlänkräknaren sätts till ett. I katalogen skapas en post (entry) som kopplar filnamn (i katalogen) till i-nodsnummer.
ii) I katalogen skapas en post (entry) som kopplar filnamnet (i katalogen) till den existerande i-nodens nummer. I i-noden räknas hårdlänkräknaren upp med ett.
iii) I katalogen skapas en post (antingen i katalogen själv eller som en separat liten fil) där namnet på den mjuka länken kopplas till sökvägen till filen/katalogen som pekas ut av länken. I i-noden för det som pekas ut förändras inget.

9) Som användare på KTHs LINUX/Unix system har du dina filer i filsystemet AFS. Beskriv de viktigaste fördelarna det ger för användarna jämfört med om man bara använt UFS. (1p)

Eftersom AFS är ett distribuerat filsystem blir dina filer tillgängliga (på ett uniformt sätt) från alla system i världen som är uppkopplade mot AFS. I AFS finns möjligheter för användarna att själva administrera rättigheter för kataloger och att själva skapa nya grupper av användare i filsystemet och ge dessa rättigheter på ett betydligt mer utvecklat sätt än i UFS.

10) Om man baserar all I/O på DMA behöver man då ha stöd i OS:et för avbrott som genereras från I/O-systemet? (1p)

Ja, DMA använder avbrott för att signalera när DMA överföringar är klara. Många enheter som klockor och tangentbord kan inte heller hanteras på ett bra sätt med DMA utan behöver stöd för avbrott.

11) Beskriv relationen/kopplingarna mellan systemanrop och drivrutiner. (1p)

Många systemanrop implementeras/utförs av drivrutiner. Detta kan konkretiseras genom ett exempel: Antag att man vill skriva något till en fildeskriptor genom systemanropet write(2). När systemanropet utförs skickas bland annat fildeskriptorn som parameter till anropet. När koden för biblioteksfunktionen som implementerar systemanropet genererar TRAP till kärnan kommer TRAP-handlern i kärnan att upptäcka att det är ett systemanrop och att det är systemanropet write(2) som skall utföras. Enheten som det skall skrivas till identifieras genom att hämta device-id från den i-nod som pekas ut av fildeskriptorn. Device-id används för att slå upp den datastruktur i devicedriver-tabellen som implementerar devicedrivern för enheten. I den datastrukturen finns en pekare till en funktion som implementerar systemanropet write på enheten. Den funktionen anropas och när den returnerar returneras returvärdet till den process som startade systemanropet.

12) ¤ a) Beskriv hur en stackad drivrutin fungerar! (1p)
¤ b) Vilka fördelar får man genom stackade drivrutiner jämfört med "icke"-stackade drivrutiner. (1p)

En stackad drivrutin består av flera funktioner som anropas i sekvens för att tillsammans utföra det drivrutinen skall göra. Fördelen med stackade drivrutiner är att man kan återanvända kod (funktioner) mellan drivrutiner, t.ex kan man ha en gemensam funktion för att föra över data över en buss. Man kan också, om det finns stöd för det i OS:et, lägga till nya funktioner i kedjan för att nå ny funktionalitet och/eller övervaka vad som händer i systemet. Drivrutiner kan vara stackade både på den egentliga drivrutinssidan och i de eventuella avbrottsrutiner som drivrutinen använder.

13) * Vilka är det två viktigaste saker en schemaläggare bör ta hänsyn till när man schemalägger fler-trådade applikationer på en maskin med flera processorkärnor? (var noga med att förklara vad man förväntar sig vinna för båda fallen). (2p)

Schemaläggaren måste ta hänsyn till både var (på vilken kärna) och när (i tiden) en tråd schemaläggs. Generellt bör schemaläggaren sträva after att schemalägga en tråd på samma kärna som senast för att utnyttja information i cachear och TLB:er samt om möjligt schemalägga tråden nära resurser den utnyttjar (om accesstiden till dessa inte är uniform) som minnesbankar och I/O-enheter.Genom detta kan man nå kortare exekveringstid. På samma sätt bör schemaläggaren sträva efter att schemalägga trådar som kommunicerar med varandra så att dessa får exekvera parallellt i tiden för att minska väntetider i kommunikationen.

14) ¤ I realtidssystem föredrar man ofta att använda schemaläggningsmetoder där man använder fasta prioriteter för jobben. Förklara fördelar samt vilka nackdelar det har jämfört med schemaläggnings -metoder baserat på variabla prioriteter. (1p)

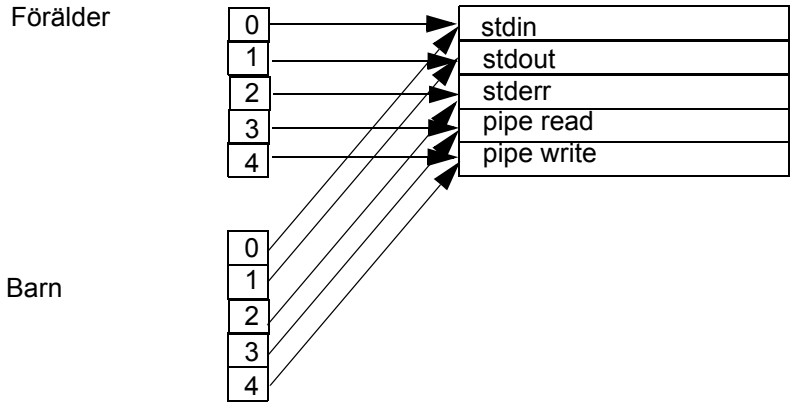
Fördelen med att använda fasta prioriter, som t.ex i Rate Monotonic Scheduling, är att schemaläggningen blir enklare då man slipper beräkna om prioriteten varje gång ett jobb blir körklart (och ev. också för alla andra jobb i systemet). Nackdelen är att man i det generella fallet normalt inte garanterat kan schemalägga system med lika hög last som om man har variabla prioriteter (jämför t.ex Earliest Deadline First mot RMS på en-processor system)

15) a) Förklara begreppen "nödvändigt" respektive "tillräckligt" villkor. (1p)
b) Är Coffmans villkor för baklås (deadlock) nödvändiga eller tillräckliga? (1p)

a) (grundläggande matematisk/logisk baskunskap...) Ett nödvändigt villkor X för A måste vara uppfyllt för att A skall gälla, men att X är uppfyllt betyder inte (implicerar inte) att A gäller (dvs uttryckt som implikation X <- A). Om ett tillräckligt villkor Y för A är uppfyllt så gäller A (Y -> A). Man kan inte dra den omvända slutsatsen, dvs att om Y inte gäller så gäller inte A. Ett villkor kan vara antingen nödvändigt eller tillräckligt eller båda delarna.
b) I det allmänna fallet är Coffmans villkor nödvändiga för att deadlock skall kunna uppstå (dvs. bryter man ett av villkoren kan deadlock inte uppstå). För specialfallet att det bara finns en enhet av varje resurstyp så är Coffmans villkor både nödvändiga och tillräckliga, dvs om i ett sådant fall alla Coffmans villkor är uppfyllda så har vi också ett deadlock.

16) * Rita upp en figur över öppna fildeskriptorer (korrekt numrerade) och vad de pekar ut vid punkten "HÄR" i nedanstående program! (antag att alla systemanrop lyckas) (1p)

```
int main(int argc, char **argv)kan deadlock inte uppstå
{
  int pipa[2];
  pipe(pipa);
  if(fork() == 0){ /* HÄR */ do_stuff1();}
  else {/* HÄR */ do_stuff2();}
```




```
}
```

17) Kalle föreslår att man skall förenkla synkroniseringen i pipes genom att förändra den så att om en process försöker läsa från en tom pipe så skall filslut (EOF) returneras direkt. Kalle hävdar att det skulle förenkla användningen genom att många onödiga deadlocks (baklås) skulle undvikas. Diskutera om det är en bra eller dålig idé! (1p)

Att en pipe är tom när man försöker läsa från den betyder inte att den som har skrivrättigheter till pipen aldrig mer kommer att skriva till pipen. Slutsatsen är att Kalles idé innebär att man inte längre skulle kunna synkronisera mellan processer/trådar på avsett sätt och är därför dålig.

18) a) Beskriv hur antivirusprogramvara baserad på "integritetskontroll" fungerar. (1p)
b) Jämför antivirusprogramvara baserad på integritetskontroll med vanliga virusscanners. Vad är deras styrkor? Kan de kombineras för att nå ökad säkerhet? (1p)

a) Den fungerar så att man utgår från att den (exekverbara) fil man vill skydda är fri från virus. Man beräknar en checksumma för filen som sparas krypterat i systemet. Närhelst man sedan vill exekvera filen beräknar man dess checksumma och kontrollerar mot den sparade. Om dessa är desamma är det mycket hög sannolikhet att filen inte infekteras av något virus.
b) Om vi säkert vet att de filer vi beräknar checksummor för och sparar på disk inte är infekterade så ger antagligen integritetskontroll högre säkerhet (lägre sannolikhet att man startar ett virusinfekterat program) än virusscanners som bara kan leta efter en begränsad mängd virussignaturer. Nackdelen med integritetskontroll är att man måste kunna lita på källan eftersom de inte kan hitta virus i infekterade filer och att det kan bli krångligare att hantera programuppdateringar. Fördelen med virusscanners är att man inte behöver lita på källan.

Dessa kan kombineras på olika sätt. T.ex skulle man kunna an vända integritetskontroll i de fall man kan lita på källan och virusscanners i övriga fall. Man kan också tänka sig att om en virusscanner kontrollerat en fil och inte hitta virus så kan man beräkna en checksumma och spara i systemet så att filen kan kontrolleras med integritetskontroll då man vill exekvera den inte behöver kontrolleras av virusscannern igen förrän dess att den antingen modifierats eller då man uppdaterat databasen med virussignaturer för virusscannern.

19) Du har fått i uppdrag att implementera en mekanism som möjliggör att man kan blockera enstaka eller grupper av användare från att starta vissa utvalda applikationer på en dator under vissa delar av dygnet. Beskriv hur man kan implementera en sådan mekanism integrerad med operativsystemet. (2p)

Eftersom allt som en vanlig användare kan göra på en dator går via systemanropsinterfacet är det rimligt att man implementerar kontrollerna där. Dvs att man i traphandlern som hanterar systenmanrop kontrollerar vilken användare som försöker göra vad. Det gör det inte bara möjligt att blockera start av olika program via exec() men också möjligt att kontrollera vad man t.ex försöker göra inom ett program som en webbläsare.

20) ¤ Beskriv vilka åtgärder man kan förvänta sig att man implementerat för att stödja mjuk realtid på ett bra sätt i ett vanligt operativsystem för persondatorer? (1p)

Men vill normalt ha stöd för:
- att man kan sätta fixa, höga prioriteter på realtidsprocesserna som inte sänks av t.ex. aging mekanismer
- att realtidsprocesser kan avbryta andra processer som exekverar i kärnan. Vilket betyder att kärnan kan preemptas för att undvika "dispatch latency", dvs. att realtidsprocesserna inte tvingas vänta på att andra processer som exekverar i kärnan skall bli klara med det.

21) Pekare i C ses ibland som ett säkerhetsproblem. Beskriv kortfattat vad konsekvenserna skulle bli för den som implementerar operativsystemskod om man tog bort möjligheten att ha pekare till funktioner i C. (1p)

Konsekvenserna skulle bli att det skulle vara svårt (omöjligt) att implementera avbrotts- och Trap-hanterings interfacen som de görs idag. Och det skulle bli svårt att implementera drivrutiner då man inte längre kan anropa olika funktioner som utför samma sak anonymt (utan att känna till funktionsnamnet) via pekare. Det senare skulle också kunna leda till problem med namnkollisioner i kärnan. Dvs det finns många nackdelar.

22) *Antag att du skall implementera ett system där en process övervakar ett antal barnprocesser där syftet är att upptäcka om barnprocesserna förändrar status.
i) Beskriv fördelar och nackdelar för implementationer baserade på pollning resp. signalhantering! (1p)
ii) Vad bör man speciellt tänka på om implementationen baseras på pollning? (1p)

i) Fördelar med pollning är att man slipper hantera asynkrona förlopp i programmet, man kan själv styra när man vill polla och man slipper problem med att systemanrop ev. kan bli avbrutna av signaler. En av de främsta nackdelarna är att man kan få fördröjningar innan man upptäcker statusförändringarna om man gör andra saker än bara pollar.
Fördelar med signalhantering är att man så omedelbart man kan, kan få besked från OS:et att status för en barnprocess ändrats. Nackdelarna är att programmet blir asynkront och att man kan behöva hantera avbrutna systemanrop/biblioteksfunktioner.
ii) Man bör se till att polla tillräckligt ofta för att hinna upptäcka statusförändringar inom givna tidsramar. Man bör inte polla genom att gå igenom en lista med PID för de barnprocesser man övervakar utan man bör lägga de processer man övervakar i en grupp och polla så att OS:et meddelar statusförändringar som skett för någon process i gruppen.

"Den som inte hoppas på att vinna, den har redan förlorat"
José Joaquin Olmedo