Operativsystem ID2200/06

omtentamen

2017-04-10 14:00-18:00

Namn: _____

Instruktioner

- Du får, förutom skrivmateriel, endast ha med dig en egenhändigt handskriven A4 med anteckningar. Mobiler etc skall lämnas till tentamensvakterna.
- Svaren skall lämnas på dessa sidor, använd det utrymme som finns under varje uppgift för att skriva ner ditt svar.
- Svar skall skrivas på svenska eller engelska.
- Du skall lämna in hela denna tentamen.
- Inga ytterligare sidor skall lämnas in.

Versioner

Denna tentamen gäller för flera olika omgångar av kurserna ID2200/06. Beroende på vilken kursomgång du följer så skall olika delar av tentamensfrågorna besvaras.

För omtentander i ID2200 gäller följande:

- Registrerade för tentamen på 6hp, VT16 och HT16: besvara frågorna 1-9, inte fråga 10.
- Registrerade för tentamen på 3.8 hp, dvs före VT16: besvara frågorna 1-8, inte 9-10.
- För de som är registrearde för tentamen på 3.8hp men som ännu inte har lab-momentet avklarat kan man besvara även fråga 9 och då få det momentet tillgodoräknat. Fråga 9 hanteras separat så få poäng på fråga 9 kompensears inte av flera poäng i övriga delar.

För omtentander i ID2206 gäller följande:

- Registrerade för tentamen på 6hp, HT16: besvara frågorna 1-9, inte 10.

- Registrerade för tentamen på 4.5hp, före HT16: besvara frårna 1-8 och fråga 10
- För de som är registrearde för tentamen på 4.5hp men som ännu inte har lab-momentet avklarat kan man besvara även fråga 9 och då få det momentet delvis tillgodoräknat. Fråga 9 hanteras separat så få poäng på fråga 9 kompenseras inte av flera poäng i övriga delar.

Betyg för 6hp

Tentamen har ett antal uppgifter där några är lite svårare än andra. De svårare uppgifterna är markerade med en stjärna, *poäng**, och ger poäng för de högre betygen. Vi delar alltså upp tentamen i grundpoäng och högre poäng. Se först och främst till att klara grundpoängen innan du ger dig i kast med de högre poängen.

Notera att det av de 40 grundpoängen räknas bara som högst 36 och, att högre poäng inte kompenserar för avsaknad av grundpoäng. Gränserna för betyg är som följer:

- Fx: 21 grundpoäng
- E: 23 grundpoäng
- D: 28 grundpoäng
- C: 32 grundpoäng
- B: 36 grundpoäng och 12 högre poäng
- A: 36 grundpoäng och 18 högre poäng

Gränserna kan komma att justeras nedåt men inte uppåt.

Gränsen för E är för tentamen på 4.5hp 18 poäng och för 3.8hp tentamen 16 poäng.

Erhållna poäng

Skriv inte här, detta är för rättningen.

Uppgift	1	2	3	4	5	6	7	8	9
Max G/H	4/0	2/2	4/2	4/2	4/2	4/2	4/2	2/2	12/8
G/H									

Totalt antal poäng:

Namn:_____ Persnr:_____

1 Operativsystem

1.1 vad händer här? [2 poäng]

Om vi ger komandona nedan, efter varandra, i ett *shell*; vad kommer resultat att vara?

```
> mkdir foo
> cd foo
> echo "hello hello" > tomat.txt
> mkdir ../bar
> ln tomat.txt ../bar/gurka.txt
> rm tomat.txt
> cd ../
> wc -w bar/gurka.txt
```

1.2 kommandon i ett shell [2 poäng]

Ge en kort beskrivning av vad kommandona nedan gör.

- `cat`
- `less`
- `ln`
- `mv`

Namn:_____ Persnr:_____

2 Processer

2.1 vad är var? [2 poäng]

I koden nedan har vi allokerat upp två arrayer; vilka arrayer och i vilka segment återfinns de: globalt, stack eller heap?

```
#include <stdio.h>
#include <stdlib.h>

int x = 43;

int h[] = {1,2,3,4};

int *foo(int *a, int s) {

  int *r = malloc(s * sizeof(int));

  for(int i = 0; i < s; i++) {
     r[i] = a[i] + x;
  }
  return r;
}

int main() {

  int *c = foo(h, 4);

  printf("%d \n", c[3]);

  return 0;
}
```

Namn:_____ Persnr:_____

2.2 privilegierade instruktioner [2 poäng*]

I en x86-arkitektur så är vissa instruktioner priviligerade och endast möjliga att utföra i *Ring-0*. Som exempel kan vi ta: HLT (halt), LIDT (load interupt descriptor table), MOV CR (skriver till Control Register). Hur använder vi denna egenskap för att kunna implementera ett operativsystem på ett effektivt sätt?

3 Schemaläggning

3.1 shortest time-to-completion first [2 poäng]

Schemaläggaren "shortest time-to-completion first" är en optimal schemaläggare; vad är det den optimerar och varför är den i praktiken inte användbar?

3.2 reaktionstiden [2 poäng]

När vi vill minska reaktionstiden så kan vill vi helst kunna avbryta jobba även om de inte är klara. Om vi gör detta så har vi en parameter som vi kan välja, genom att anpassa denna så kan vi förbättra reaktionstiden. Vad är det för parameter som vi kan sätta? Hur skall vi förändra den och vilka oönskade konsekvenser kan det få?

Namn:_____ Persnr:_____

3.3 realtidsschemaläggare [2 poäng*]

I realtidsschemaläggning så beskriver vi jobb med tre värden som brukar betecknas: e, d och p. Vad står dessa parametrar för (du behöver inte komma ihåg vilken som är vilken men skall kunna ange egenskaperna som beskriver ett jobb).

- e:
- d:
- p:

4 Virtuellt minne

4.1 paging [2 poäng]

Antag att vi har en virtuell adress som består av ett 22-bitars sidnummer och en 12-bitars offset. Vi har en fysisk adressrymd på 32 bitar, kodar ett ramnummer i 20 bitar och element i en sidomvandlingstabell är fyra bytes stort. Hur stor är en sida och hur stor skulle en komplett omvandlingstabell behöva vara?

4.2 omvänd sidtabell [2 poäng]

Om vi har ett fysiskt minne som är betydligt mindre än de virtuella minnen som operativsystemet erbjuder kan det vara idé att implementera en så kallad omvänd sidtabell. Hur fungerar en omvänd sidtabell och vad ger den för fördel?

Namn:_____ Persnr:_____

4.3 bounds [2 poäng*]

När vi implementerar ett segmenterat minne så vill vi gärna ha ett *bounds-värde* som säger hur stort segmentet är. Om vi inte har det så riskera vi att adressera utanför segmentet. När vi implementerar ett sidat minne (*paging*) så behövs inget bounds-värde? Varför behöver vi inte det? Vad är det som hindrar oss från att adressera utanför sidan?

Om det nu är något som hindrar oss från att göra fel, är det något som kostar? Vad kostar det?

5 Minneshantering

5.1 tältsemester [2 poäng]

När jag rest runt och tältat på olika campingplatser i Sverige (det har jag inte) så har jag märkt att campingplatser har helt olika system för var man skall slå upp sitt tält. En del säger att man skall slå upp det i närheten av de andra, men se till att man lämnar ett lucka på fyra meter till nästa tält. Andra platser har rutat in ett område i lika stor a rutor och säger att jag kan slå upp mitt tält i ruta 17.

Vad är problemet med de två olika strategierna och vad har detta med operativsystem att göra?

Namn:_____ Persnr:_____

5.2 statisk reallokering [2 poäng]

Istället för att implementera dynamisk minneshantering så kan man göra en statisk reallokering av program när de läggs in i minnet. Vad innebär statisk reallokering, vad är det som måste göras när ett program läggs in i minnet? Ge ett exempel.

5.3 kod, data, stack... [2 poäng*]

Vid implementering av segmenterat minne är en lösning att till exempel låta de översta bitarna i en virtuell adress avgöra vilket segment som skall användas. En annan lösning är att helt enkelt ha separata segment för kod, data och stack. Om vi skulle ha dessa tre segment, hur skulle då processorn avgöra vilket segment som skall användas när vi gör en operation mot minnet?

Namn:_____ Persnr:_____

6 Flertrådad programmering

6.1 count [2 poäng]

Vad kommer skrivas ut om vi exekverar proceduren `hello()` nedan samtidigt i två trådar? Motivera svaret.

```
int loop = 10;

void *hello() {
  int count = 0;

  for(int i = 0; i < loop; i++) {
      count++;
  }
  printf("the count is %d\n", count);
}
```

6.2 men varför [2 poäng]

Om vi har en multicore-cpu så är det klart en fördel att arbeta med flera trådar eftersom vi då kan utnyttja beräkningskraften bättre. Om vi nu sitter på en maskin med enbart en kärna så är det väl rätt meningslöst att dela upp sitt program i flera trådar, eller? Kan ett program som är uppdelat i trådar exekvera snabbare även om vi bara har en kärna att köra på? Motivera.

Namn:_____ Persnr:_____

6.3 gröna trådar [2 poäng*]

En del operativsystem erbjuder s.k. *gröna trådar* dvs trådar som är implementerade med hjälp av biblioteksfunktioner och som hanteras av användarprocessen. Nämn en fördel och en nackdel med gröna trådar.

7 Filsystem och lagring

7.1 en helt vanlig HDD [2 poäng]

Om en hårddisk har en medelsöktid (*average seek time*) på 10 ms, en rotationshastighet på 7200 rpm (*rounds per minute*) och har en läshastighet på 200 MiB/s. Vad är då medeltiden för att läsa en slumpvald sektor på 4 KiB?

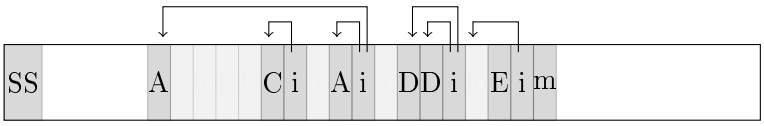
Namn:_____ Persnr:_____

7.2 sektorer till filsystem [2 poäng]

En hårddisk består av ett antal sektorer. Beskriv en enkel implementation av ett filsystem och hur filsystemets datastrukturer kan lagras på en hårddisk.

7.3 loggbaserat filsystem [2 poäng*]

Nedan ser du en schematisk bild av ett loggbaserat filsystem. Om systemet nu börjar få ont om plats så kommer det att försöka skapa plats. Hur kan mer plats skapas och hur kommer systemet att se ut efter det att mer utrymme tillgängliggjorts?



Namn:_____ Persnr:_____

8 Virtualisering

8.1 varför [2 poäng]

Beskriv ett viktigt skäl till varför man vill köra flera virtualiserade operativsystem istället för ett enda.

8.2 kernel/user mode [2 poäng*]

Antag att vi har en hypervisor som kör i *kernel mode* och det virtualiserade systemet kör i *user mode* för att skydda hypervisorn. Varför kan vi inte låta det virtualiserade operativsystemet ligga kvar i *user space* när en av dess processer kör?

Namn:_____ Persnr:_____

9 Implementering

9.1 stacken [2 poäng]

Nedan ser vi ett program som skriver ut innehållet på stacken.

```
void zot(unsigned long *stop, int a1, int a2, int a3, int a4, int a5, int a6) {
  unsigned long r = 0x456;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){
    printf("%p      0x%lx\n", i, *i);
  }
}

int main() {
  unsigned long p = 0x123;

  zot(&p,1,2,3,4,5,6);
 back:
  printf("  back: %p \n", &&back);
  return 0;
}
```

Vid en körning får vi följande utskrift. Beskriv de värden som pekas ut med pilar (<--).

```
0x7ffeb3331f58      0x456
0x7ffeb3331f60      0x7ffeb3331f60  <-- ??
0x7ffeb3331f68      0x3a7dbfad7df4b100
0x7ffeb3331f70      0x7ffeb3331fa0
0x7ffeb3331f78      0x400663    <-- ??
0x7ffeb3331f80      0x6   <-- ??
0x7ffeb3331f88      0x4004a0
0x7ffeb3331f90      0x123
  back: 0x400667
```

Namn:_____ Persnr:_____

9.2 sbrk() [2 poäng]

Proceduren `malloc()` är en biblioteksrutiner som använder sig av systemanropet `sbrk()` för att allokera minne. Vad är fördelen för en process om den använder `malloc()` istället för att anropa `sbrk()` direkt?

9.3 pipes [2 poäng]

Om vi har två processer, en producent och en konsument, som kommunicerar via en s.k. *pipe*. Hur kan vi då förhindra att producenten skickar mer information än vad konsumenten kan ta emot och därmed får systemet att krascha?

9.4 tmpfs [2 poäng]

Vad gör kommandot nedan och varför skulle vi vilja göra det? Vad är nackdelen?

```
> sudo mount -t tmpfs tmpfs ./tmp
```

Namn:_____ Persnr:_____

9.5 byt kontext [2 poäng]

Programmet nedan leker med sitt eget s.k. *kontext*. Vad innehåller strukturen `ucontext_t` och vad blir resultatet när vi kör programmet?

```
#include <stdlib.h>
#include <stdio.h>
#include <ucontext.h>

int main() {

  int done = 0;
  ucontext_t one;
  ucontext_t two;

  getcontext(&one);

  printf("hello %d\n", done);

  if(!done) {
    done = 1;
    swapcontext(&two, &one);
  }
  return 0;
}
```

9.6 läshastighet [2 poäng]

Hur stor är skillnaden i läshastighet om vi jämför att läsa från från minnet med att läsa från en fil på en roterande hårddisk (första läsningen av filen)?

- $10ns$ vs $1\mu s$
- $10ns$ vs $10ms$
- $100ns$ vs $10\mu s$
- $1\mu s$ vs $1ms$

Namn:_____ Persnr:_____

9.7 ett huvud och en fotnot [2 poäng*]

Vid implementation av minnesallokering så är det vanligt att man ha ett gömt *huvud* placerat alldeles innan den minnesarea som man delar ut. I detta huvud kan man bland annat skriva hur stor arean är så att det blir enklare att ta hand om arean när vi gör *free*. Man kan även använda sig av en gömd fotnot som ligger efter arean där man kan skriva att arean är använd eller inte och kanske en pekare till huvudet. Vad är det för poäng med att lägga den informationen efter arean, räcker det inte med huvudet?

9.8 lite bättre [2 poäng*]

Så kallade *pipes* är ett mycket enkelt sätt att skicka data från en process till en annan. Det har dock sina begränsningar och ett bättre sätt är att använda s.k. *sockets*. Om vi istället för en pipe öppnar en *stream socket* mellan två processer så har vi flera fördelar. Beskriv två saker som en *stream socket* ger oss som vi inte får om vi använder en *pipe*.

Namn:_____ Persnr:_____

9.9 character device [2 poäng*]

Vi kan skapa en s.k. *character device* och interagera med den med hjälp av `ioctl`. I koden nedan, beskriv vad `fd`, `JOSHUA_GET_QUOTE` och `buffer` är och hur vårt *device* kan tänkas fungera.

```
if (ioctl(fd, JOSHUA_GET_QUOTE, &buffer) == -1) {
  perror("Hmm, not so good");
} else {
  printf("Quote - %s\n", buffer);
}
```

9.10 delat minne [2 poäng*]

Processer i ett Unix-system kan kommunicera med varandra via flera olika kanaler. De kan även dela minnesareor på liknande sätt som två trådar i en process can dela *heap* och global data. Hur kan vi få två processer att dela minne?

Namn:_____ Persnr:_____

10 Bara för omtentamen i ID2206 reggade före HT16

10.1 publika-nyklar [2 poäng]

Beskriv vad man kan göra med s.k. assymetriska krypteringsmetoder får vi har en publik och en provat nyckel som vi inte kan göra med system där vi bara har en nyckel.

10.2 hårda och mjuka [2 poäng]

Vad är skillnaden mellan vad som informellt kallas "hårda" och "mjuka" realtidssystem?

10.3 multiprocessor [2 poäng*]

Om vi har en multiprocessor så måste en schemaläggare naturligtvis kunna låta processer köra på de olika processornerna. Vi kan dock göra bättre eller sämre schemalägging, beskriv en aspekt som vi måste ta hansyn till och hur vi anpassar schemaläggaren.

Namn:_____ Persnr:_____

10.4 distribuerat operativsystem [2 poäng*]

Man kan låta ett operativsystem spänna över flera datorer s.k. distribuerat operativsystem istället för att ha oberoende system på de enskilda datorerna. Vad gör det för skillnad när vi skall implementera ett system med kommunicerande processer.