Örebro universitet
Institutionen för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i

Operativsystem för civilingenjörer

lördag 4 januari 2020

Gäller som tentamen för:
DT513G Operativsystem för civilingenjörer, provkod 0100

Hjälpmedel:	Inga.
Poängkrav:	Maximal poäng är 38. För godkänt betyg krävs 20 poäng.
Resultat:	Meddelas på kursens hemsida eller via e-post senast lördag 25 januari 2020.
Återlämning av tentor:	Elektroniskt via webbportalen Studenttjänster.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070 - 73 47 013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara och tvetydiga formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges. Gjorda antaganden får inte förändra den givna uppgiften.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

Uppgift 1 (4 p)

a) Vad är ett operativsystem?
b) Vad menas med operativsystemkärnan? Hur är den relaterad till processorkärnorna i en flerkärnig CPU?
c) Ingår användargränssnittet i operativsystemet? Diskutera!

Uppgift 2 (3 p)

Man brukar säga att operativsystemet ska erbjuda en virtuell maskin som applikationerna kan använda. Vad menar man med det?

Uppgift 3 (4 p)

Processortillverkaren AMD har nyligen lanserat processorn *Ryzen Threadripper 3970X*, som har 32 kärnor. Med hjälp av hyperthreading kan den köra 64 trådar parallellt. Den kostar ungefär 25000 kronor.

Om man tar ett enkeltrådat program och gör om det till ett flertrådat program med 64 trådar, skulle man kunna tänka sig att programmet går 64 gånger så fort på den här processorn. I praktiken blir det sällan eller aldrig så snabbt. Till en del beror det på rent hårdvarumässiga orsaker, till exempel att när alla kärnorna kör för fullt så kan processorn bli överhettad, och därför kan en del kärnor köras långsammare, automatiskt av processorn.

a) Det finns också mjukvarumässiga orsaker som har med operativsystemet och programmet självt att göra, till att programmet inte går så fort. Vilka?

b) Hur bör man skriva det flertrådade programmet för att få så hög uppsnabbning som möjligt?

Uppgift 4 (7 p)

Förklara kort följande termer från operativsystemområdet:

a) systemanrop
b) process
c) tråd ("thread")
d) användartråd ("user thread")
e) kärntråd ("kernel thread")
f) busy-wait
g) filsystem

Uppgift 5 (7 p)

Vi har en byteadresserad dator som arbetar med minnesadresser på 16 bitar. Det finns alltså 2^{16}, dvs 65536, olika minnesadresser. Datorn har bara hälften så mycket fysiskt minne (32768 byte), som dessutom ska delas mellan de olika processerna. Med hjälp av virtuellt minne och så kallad "paging" kan varje process ändå ha tillgång till 65536 byte virtuellt minne.

a) Vi antar att en sida ("page") är 128 minnesadresser stor. Hur många bitar går det åt för att ange en minnesplats inom en sida ("offset")?

b) Hur många bitar går det åt för att ange sidnumret ("page number")?

c) Hur många olika sidor kan varje process ha?

d) Hur många olika ramar ("frames") finns det i det fysiska minnet?

e) Hur stor (i byte räknat) blir varje process sidtabell ("page table")? Redovisa hur du räknat!

f) Förklara, med den beskrivna datorn som exempel, hur paging fungerar. Hur är det möjligt att en process kan ha mer minne än vad som finns? Och hur är det möjligt att flera processer var och en kan ha mer minne än vad som finns?

Uppgift 6 (4 p)

För schemaläggningen ("schedulering") av processer används flera olika köer där processerna placeras. Förklara vilka det är, och vad man har varje kö till!

Uppgift 7 (2 p)

Förklara skillnaden mellan symmetrisk och asymmetrisk kryptering!

(Det är två uppgifter till på nästa sida.)

Uppgift 8 (5 p)

Skriv ett C-program för Linux som använder **fork** för att skapa en ny process, och sedan, i den nya processen, först använder **chdir** för att byta aktuell katalog ("directory") till **/home/thomas** och sedan använder **execl** för att starta programmet **/bin/ls**.

Här är deklarationer för de olika systemanropen:

```
#include <sys/types.h>
#include <unistd.h>

pid_t fork(void);
int chdir(const char *path);
int execl(const char *path, const char *arg, ... /* (char *) NULL */ );
```

Uppgift 9 (2 p)

a) Vad gör systemanropet **wait**?
b) Vilket eller vilka problem är det man vill lösa med det systemanropet?
