Operativsystem för civilingenjörer: Hemtentamen 2020-06-03

Det här är hemtentan som går onsdag 3 juni 2020 i kursen Operativsystem för civilingenjörer. Ansvarig lärare är Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), telefon **070-73 47 013**.

Tid: 08:15 - 13:15

Instruktioner

1. Den här hemtentan ersätter den planerade salstentan, och är endast för de studenter som anmält sig till den tentan.
2. Uppgifterna ska lösas enskilt, dvs inga grupper av två eller flera studenter.
3. **Du får använda dator, böcker och vilka andra hjälpmedel som helst,** men du får inte samarbeta eller fråga någon (utom mig). Exempelvis är det tillåtet att söka och läsa på webbplatser som Stack Overflow, men inte att ställa egna frågor. Du kan alltså provköra kommandon och program, om du vill.
4. Diskutera inte uppgifterna med andra förrän tidigast dagen efter tentan.
5. Lös de angivna uppgifterna och samla svaren på lämpligt sätt, till exempel i en PDF. Skicka sen in lösningarna till mig i Blackboard. (Välj **Kursmaterial** i menyn till vänster, gå in i mappen **Hemtentamen**, och välj den rätta tentan. Där kan man sedan välja att skicka in sin lösning. Då blir bedömningen anonym.
6. Om det skulle bli problem med Blackboard, kan man i nödfall skicka in svaren antingen som ett kursmeddelande i Blackboard eller via vanlig e-post (thomas.padron-mccarthy@oru.se), senast vid tentatidens slut. Då blir bedömningen inte anonym.
7. I Blackboard ser du att du skickat in din lösning. Om du i stället skickade in med e-post, och inte senast en timme efter tentatidens slut fått ett svar från mig med en bekräftelse på att du skickat in svaren, bör du kontakta mig, enklast genom att ringa eller SMS:a mig (ifall det är e-posten som inte fungerar). Tänk på att en del mailtjänster (särskilt Microsoft-tjänster som Hotmail.com, Outlook.com och Live.com) ibland kastar bort brev med bilagor, utan att meddela det.
8. Skriv gärna svaren i ett ordbehandlingsprogram. Rita gärna eventuella diagram i ett ritprogram. Det är inte förbjudet att skriva och rita för hand, men då måste text och bilder scannas in eller fotograferas. Det finnas scanner-appar till Android och iPhone, till exempel Adobe Scan, som ger bättre resultat än att bara ta vanliga kort med kameran.
9. Tentatiden är utökad med en extra timme för att täcka in problem med e-post, inscanning eller fotografering av diagram, och liknande.

10. Om du behöver fråga något, så kontakta gärna mig. Ring eller skicka SMS, för jag kanske inte kommer att sitta vid datorn hela tiden.
11. Oklara och tvetydiga formuleringar kommer att misstolkas. Lösningar som inte går att läsa eller förstå kan naturligtvis inte ge några poäng.
12. Antaganden utöver de som står i uppgifterna måste anges. Gjorda antaganden får inte förändra den givna uppgiften.
13. 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.
14. Maximal poäng är 24. För godkänt betyg krävs minst 12 poäng.
15. Resultat meddelas på kursens hemsida eller via e-post senast 24 juni 2020. Eftersom svaren skickas in elektroniskt scannas tentorna inte för retur.

Uppgift 1 (6 p)

Med **time**-kommandot i Linux-skalet bash kan man se körtider för ett program. Det skriver ut flera olika tider, och det är "real" som är "väggklocketiden", den tid som man skulle kunna mäta upp med ett stoppur eller genom att titta på en väggklocka, och som anger den verkliga tid det tar från att programmet startar tills det att det avslutas.

Skriv ett C-program för Linux som mäter och skriver ut tiden för en programkörning, men bara väggklocketiden. I programmet **thread-performance-test-2020.c** från labb 2 i kursen kan man se hur man kan använda systemanropet **gettimeofday** för att mäta tider med bättre upplösning än hela sekunder:

```
struct timeval before;
gettimeofday(&before, NULL);

// ...

struct timeval after;
gettimeofday(&after, NULL);

double elapsed_milliseconds =
    (after.tv_sec - before.tv_sec) * 1e6 + (after.tv_usec - before.tv_usec);
double seconds = elapsed_milliseconds / 1e6;
printf("Elapsed time: %.6f s\n", seconds);
```

Programmet ska använda **fork** för att skapa en ny process, och sedan, i den nya processen, **execvp** för att starta programmet vars körtid ska mätas. Därefter ska föräldraprocessen vänta med **wait** eller **waitpid** på att programmet ska köra klart.

a) Först gör vi en enklare version. I den här versionen ska vi bara mäta tiden för ett sleep-anrop i barnprocessen, till exempel **sleep(5)**, som rimligen bör ta nästan exakt 5 sekunder. Vi behöver fortfarande **fork** och antingen **wait** eller **waitpid**, men här behövs inget anrop till **execvp**.

b) Ändra programmet så det tar argument på samma sätt som det vanliga **time**-kommandot, och kör det program och de argument som man gett som argument. Ett par körexempel:

```
> ./my-time echo A B C
A B C
Uppmätt körtid: 0.000439 s

> ./my-time sleep 5
Uppmätt körtid: 5.000820 s
```

Uppgift 2 (6 p)

I det sista körexemplet i uppgiften ovan finns två processer, dels den som kör programmet **my-time**, och dels den som skapas med **fork** och sedan kör programmet **sleep**. Vi har en flerkärnig processor, just nu inga andra processer igång på datorn, och ett modernt operativsystem som inte gör några onödiga kontextswitchar. Visa hur de två processerna flyttas mellan de olika köerna i operativsystemkärnan under körningen av det körexemplet.

Uppgift 3 (6 p)

En av de senaste processorerna från tillverkaren AMD heter Ryzen Threadripper 3990X. Den har 64 kärnor, och med hjälp av hyperthreading kan den köra 128 trådar parallellt. Den finns att köpa nu, och den kostar 46000 kronor. (Se Prisjakt.)

Vi har ett vanligt, enkeltrådat program, men när vi nu betalat 46000 kronor för den här processorn vill vi förstås utnyttja de många kärnorna, och göra om programmet till en flertrådad version.

a) Vad är den högsta uppsnabbning man teoretiskt kan få på den här processorn med en flertrådad version, jämfört med en enkeltrådad version?

b) I praktiken blir det sällan 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. Men det finns också mjukvarumässiga orsaker som har med operativsystemet och programmet självt att göra, till att programmet inte går så fort. Vilka?

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

Uppgift 4 (6 p)

Virtuellt minne brukar numera vara sidindelat, på engelska i "pages". Sidorna har en viss storlek. Vi ska utveckla en ny processorarkitektur för vanliga moderna datorer, och här är förslag på några sidstorlekar som man skulle kunna använda. Ange för var och en av dem om det är en rimlig storlek eller inte, och om det inte är en rimlig storlek så förklara (detaljerat) *varför* den inte är rimlig.

a) 4 byte
b) 1024 byte
c) 8192 byte
d) 8193 byte
e) 10000 byte
f) 1099511627776 byte (2^{40} byte)

Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), 2 juni 2020