Operativsystem för civilingenjörer: Hemtentamen 2021-01-07

Det här är hemtentan som går torsdag 7 januari 2021 i kursen DT513G Operativsystem för civilingenjörer vid Örebro universitet, provkod A001. Ansvarig lärare är Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), telefon **070-73 47 013**.

Tid: 09:15 - 14:15

Instruktioner

1. Den här hemtentan ersätter den planerade salstentan.
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 eller dela med dig av svar 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 på kursens Blackboard-sida, gå in i mappen **Hemtentamen**, och välj den rätta tentan. Där kan man sedan välja att skicka in sin lösning. Då blir bedömningen anonym. Glöm inte att klicka på knappen **Skicka** längst ner till höger.
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 finns 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 28. För godkänt betyg krävs minst 15 poäng.
15. Resultat meddelas senast 15 arbetsdagar efter tentamensdatum. Eftersom svaren skickas in elektroniskt scannas tentorna inte för retur.

Uppgift 1 (5 p)

Man brukar säga att operativsystemet erbjuder en virtuell maskin. Samtidigt finns det virtualisering, som också handlar om virtuella maskiner. Vad är skillnaden? Är det helt olika saker, eller hänger de ihop på något sätt? Jämför och förklara!

Uppgift 2 (3 p)

Man brukar säga att en viktig skillnad mellan trådar och processer i många vanliga operativsystem är att varje process har sin egen minnesrymd, medan flera trådar kan dela på samma minnesrymd. Men en annan skillnad är att operativsystemkärnan alltid måste känna till alla processer, men det kan hända att det finns trådar som operativsystemkärnan inte vet om. Förklara hur det kan vara möjligt!

Uppgift 3 (6 p)

a) Vad menas **spinlock** och **busy-wait** (även kallat **spinning**)?

b) Ge exempel på när det kan vara lämpligt för operativsystemkärnan att använda dessa.

c) Under vilka omständigheter bör operativsystemkärnan absolut *inte* använda dessa?

d) Varför är det normalt inte lämpligt att använda dem i vanliga program?

Uppgift 4 (5 p)

Den dyraste processor som just nu (6 januari) finns i lager enligt prisjämförelsesajten prisjakt.nu är **AMD Epyc 7H12**. Den har 64 kärnor, AMD:s motsvarighet till hyperthreading så den kan köra 128 parallella trådar, och en klockfrekvens på 2,6 GHz. Den kostar 85646 kronor.

Den billigaste processorn är **AMD Mobile Sempron 3200+**. Den har en kärna, ingen hyperthreading, och en klockfrekvens på 1,6 GHz. Den kostar 48 kronor.

När man ska ange hur snabb en dator är använder man ibland enheten **flops**, floating-point operations per second, som anger hur många flyttalsoperationer datorn kan utföra på en sekund. Här nedan finns ett program, flops.c, som utför en massa flyttalsdivisioner, i en eller flera trådar, och sen skriver ut hur många flops det blev. Man anger antalet trådar med ett kommandoradsargument när man startar programmet.

```
// Several threads, each performing many floating-point divisions
// Compile with:
// gcc -Wall -Wextra -std=c11 -O3 flops.c -lpthread -o flops
// Note 1: gcc with -O3 did not remove the actual calculations,
// but an aggressively optimizing compiler could.
// Note 2: Normal "int" is too small for some of the values,
// so we use "long long int".
// Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)
// Wed Jan  6 19:23:21 CET 2021
```

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <pthread.h>
#include <sys/time.h>

#define NR_DIVISIONS_PER_THREAD (1LL*1000*1000*1000)
#define STARTING_VALUE 1e50
#define FACTOR 1.0000001

int nr_finished_threads = 0;

void *thread_body(void *arg) {
    int thread_number = (int)arg;
    double value = STARTING_VALUE;
    printf("Thread %d starting with value %g, factor %.10f...\n",
           thread_number, value, FACTOR);
    for (long long int i = 0; i < NR_DIVISIONS_PER_THREAD; ++i) {
        value /= FACTOR;
    }
    printf("Thread %d finishing with value %g.\n", thread_number, value);
    ++nr_finished_threads;
    return NULL;
} // thread_body

int main(int argc, char *argv[]) {
    int nr_threads;
    if (argc != 2 || sscanf(argv[1], "%d", &nr_threads) != 1 || nr_threads < 1) {
        fprintf(stderr, "Usage: flops NUMBER-OF-THREADS\n");
        exit(EXIT_FAILURE);
    }

    printf("This program starts %d thread(s).\n", nr_threads);
    printf("Each thread performs %lld floating-point divisions\n",
           NR_DIVISIONS_PER_THREAD);

    pthread_t threads[nr_threads];
    printf("Starting the %d thread(s)...\n", nr_threads);

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

    for (int i = 0; i < nr_threads; ++i) {
        if (pthread_create(&threads[i], NULL, thread_body, (void*)(i + 1)) != 0) {
            fprintf(stderr, "Couldn't create thread %d.\n", i);
            exit(EXIT_FAILURE);
        }
    }

    printf("%d threads started. Waiting for them to finish...\n", nr_threads);

    for (int i = 0; i < nr_threads; ++i) {
        if (pthread_join(threads[i], NULL) != 0) {
            fprintf(stderr, "Couldn't join thread for thread %d.\n", i);
            exit(EXIT_FAILURE);
        }
    }

    struct timeval after;
    gettimeofday(&after, NULL);
    double elapsed_seconds =
        after.tv_sec - before.tv_sec + (after.tv_usec - before.tv_usec) / 1e6;

    printf("All threads finished (%d threads)!\n", nr_finished_threads);
    printf("Elapsed time: %.6f s\n", elapsed_seconds);
    long long int total_divisions = nr_threads * NR_DIVISIONS_PER_THREAD;
    printf("We have performed %lld floating-point divisions.\n", total_divisions);
    double flops = total_divisions / elapsed_seconds;
```

```
    printf("Flops (floating-point operations per second): %.0f = %.0f Mflops\n",
           flops, flops / (1000 * 1000));

    return EXIT_SUCCESS;
} // main
```

Vi provkör programmet ovan, med en tråd, på den dyra processorn. Det tar 2,0 sekunder, och ger 500 Mflops ("mega-flops", dvs miljoner flops). Vi provkör också programmet, med en tråd, på den billiga processorn. Det tar 4,0 sekunder, och ger 250 Mflops.

Hur lång tid kan vi räkna med att det tar, och hur många Mflops kan vi räkna med att vi får, om vi kör programmet:

a) med två trådar på den dyra processorn?
b) med två trådar på den billiga processorn?
c) med tio trådar på den dyra processorn?
d) med tio trådar på den billiga processorn?
e) med hundra trådar på den dyra processorn?
f) med hundra trådar på den billiga processorn?
g) med tusen trådar på den dyra processorn?
h) med tusen trådar på den billiga processorn?

Uppgift 5 (3 p)

När programmet i uppgiften ovan kört klart alla trådarna skriver det ut att de är klara, och hur många det var. Exempel:

```
All threads finished (2 threads)!
```

I en av våra provkörningar med 1000 trådar blir utskriften fel:

```
All threads finished (998 threads)!
```

Förklara varför det blev fel, och visa hur man kan ändra programmet så man slipper felet.

Uppgift 6 (3 p)

På labbarna i kursen har vi sett andra flertrådade program, bland annat i labb 2 där vi såg **thread-performance-test-2020** och (när ni gjort om det till en flertrådad version) **single-threaded-monster-world**. När man varierar antalet trådar i programmet **flops** varierar prestandan på ett liknande sätt som ett av dessa.

a) Vilket?

b) Förklara vad det är för likheter och skillnader mellan programmen som gör att det blir så!

Uppgift 7 (3 p)

Ju mindre minnessidor ("pages") man har, desto större blir page-tabellen. Förklara varför!

Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), 6 januari 2021