Operativsystem för civilingenjörer: Hemtentamen 2021-06-01

Det här är hemtentan som går tisdag 1 juni 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: 08:15 - 13:15

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 30. 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.

Information från institutionen angående fusk

Instruktioner inför digital hemtentamen/ examination

Jag vill undvika fusk - hur gör jag? / All form av fusk anmäls
Du ska följa instruktionerna för uppgiften. Om du är osäker, fråga ansvarig lärare om något i instruktionerna är oklart.
Du får inte samarbeta. Det här är en individuell examinationsuppgift. Du ska inte prata med någon, ställa frågor till eller ta hjälp av andra studenter eller kurskamrater. Att hjälpa andra under en individuell examination är också fusk.

Lägg undan mobilen. Stäng av sociala medier.

Du får inte använda hjälpmedel. Det vill säga att du får inte använda dig av något annat än det som står angivet i instruktionerna.
Du får inte använda andra formuleringar än dina egna. Dina svar ska vara **självständigt formulerade** och **redovisa dina kunskaper.** Det betyder att inga citat eller referat ska förekomma i dina svar om det inte står i instruktionerna att du får använda citat eller referat.
Dina svar kontrolleras via Urkund.
All misstanke om fusk anmäls till universitetets rektor och kan leda till en prövning i universitets disciplinnämnd.

Konsekvenser av fusk

Om du fuskar kan detta leda till en avstängning som kan få följder både för dina studier och privat:

- Uteblivet studiemedel som t.ex. som kan påverka din möjlighet att behålla din bostad.
- Ingen åtkomst till digitala plattformar.
- Du kan behöva meddela dina kursare om att du är fälld för fusk, om du t.ex. ingår i ett grupparbete på en pågående kurs men blir avstängd.
- Tillfällen för examination går förlorade vilket kan innebära att du inte kommer vidare i dina studier nästa läsperiod/termin och din studiegång blir därmed påverkad.
- Beslutet är en offentlig handling som begärs regelbundet ut av en nyhetsbyrå. Och kan även begäras ut av framtida arbetsgivare eller andra.
- Om du fuskar dig igenom din utbildning har du inte den kunskap som arbetsmarknaden förväntar av dig.

OBS: Tänk efter en gång till innan du påbörjar och genomför din tentamen!

Uppgift 1 (5 p)

a) Varför kan ett systemanrop innebära ett avbrott ("interrupt")?

b) Långt efter att systemanropet gjorts kan det förekomma fler interrupt, som en del av arbetet som utförs av systemanropet. Beskriv hur det kan gå till!

Uppgift 2 (5 p)

Task-structen, eller PCB som den också kallas, är en av de viktigaste datastrukturerna som finns i operativsystemkärnan.

a) Vad används den till?

b) Vad innehåller den?

c) Task-structen förekommer i flera olika köer i operativsystemkärnan. Vad är det för köer, och vad används de till?

Uppgift 3 (6 p)

Vi konstruerar en processor där fysiska adresser är 37 bitar, medan virtuella adresser är 33 bitar, varav 22 bitar är sidnummer ("page number" på engelska) och 11 bitar är offset.

a) Hur stor är en minnessida ("page")? Visa hur du räknat.

b) Fysiska ramar ("frames") kommer att vara lika stora som minnessidorna. Förklara varför!

c) På den här processorn har fysiska adresser fler bitar än virtuella adresser. Är det något konstigt med det? Vad innebär det?

d) Hur mycket plats (i byte) tar page-tabellen för en process? Gör rimliga antaganden, redovisa dem, och förklara hur du räknat!

e) Den virtuella minnessidan nummer **2** (binärt: **10**) lagras i fysisk frame nummer **0** (binärt: **0**). På vilken fysisk minnesplats finns den virtuella adressen **4097** (binärt **1000000000001**)?

Uppgift 4 (7 p)

Här är programmet **threads-1**, som startar två trådar, och där varje tråd ökar variabeln **data** med ett en miljon gånger. Slutresultatet borde förstås bli att variabeln innehåller värdet två miljoner.

Programmet threads-1.c:

```
// threads-1.c

#include <stdio.h>
#include <pthread.h>

volatile long long data = 0;

void *thread_body(void *arg) {
    for (int i = 0; i < 1000*1000; ++i) {
        ++data;
    }
    return NULL;
}

int main(void) {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, thread_body, NULL);
    pthread_create(&thread2, NULL, thread_body, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    printf("Result: data = %lld\n", data);
}
```

a) Vi kör programmet, men variabelns värde blir inte alls två miljoner, utan 1028606. Vad beror det på att det blir fel värde?

b) Vi flyttar om raderna i **main**-funktionen så den ser ut enligt nedan. Vad blir variabelns värde nu, och vad beror det på?

Ur programmet threads-2.c:

```
int main(void) {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, thread_body, NULL);
    pthread_join(thread1, NULL);
    pthread_create(&thread2, NULL, thread_body, NULL);
    pthread_join(thread2, NULL);
    printf("Result: data = %lld\n", data);
}
```

c) Kan man gissa något om hur körtiden för **threads-2** kommer att bli, jämfört med **threads-1**? Vad beror det på? (Vi talar om den så kallade "väggklocketiden", som är den verkliga tid som det tog att köra programmen, och som man till exempel skulle kunna mäta genom att titta på en väggklocka).

d) Vi har glömt hur man skrev för att använda pthread-paketets lås, så vi försöker göra ett lås själva, så programmet ser ut enligt nedan. Fungerar låset? Kommer variabelns värde att bli två miljoner nu? Förklara!

Programmet threads-3.c:

```
// threads-3.c

#include <stdio.h>
#include <pthread.h>

volatile long long data = 0;
volatile int locked = 0;

void *thread_body(void *arg) {
    for (int i = 0; i < 1000*1000; ++i) {
        // First, wait for the lock in a loop:
        while (locked) { }
        // Lock is free, lock it:
        locked = 1;
        ++data;
        // Release the lock:
        locked = 0;
    }
    return NULL;
}

int main(void) {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, thread_body, NULL);
    pthread_create(&thread2, NULL, thread_body, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    printf("Result: data = %lld\n", data);
}
```

e) Visa hur man gör låsningen på rätt sätt, med pthread-paketets lås.

Uppgift 5 (7 p)

Här är fem program som alla skriver en tio megabyte (10485760 byte) stor fil som heter **garbage**, och som bara innehåller bokstaven 'x'.

- Programmet **create-garbage-1** skriver ett tecken 10485760 gånger.
- Programmet **create-garbage-2** skriver en kilobyte (1024 byte) 10240 gånger.
- Programmet **create-garbage-3** skriver 10 kilobyte (10240 byte) 1024 gånger.
- Programmet **create-garbage-4** skriver hela filen (10485760 byte) på en gång.
- Programmet **create-garbage-5** startar tio trådar, som var och en skriver en tiondel av filen (en megabyte, 1048576 byte) på en gång.

Programmen:

create-garbage-1.c

```
// create-garbage-1.c

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

#define KILO 1024
#define MEGA KILO*KILO

int main(void) {
    char buffer[1];
    buffer[0] = 'x';
    int outfd = open("garbage", O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU);
    for (int i = 0; i < 10*MEGA; ++i)
        write(outfd, buffer, sizeof buffer);
    close(outfd);
}
```

create-garbage-2.c

```
// create-garbage-2.c

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

#define KILO 1024
#define MEGA KILO*KILO

char buffer[KILO];

int main(void) {
    for (int i = 0; i < sizeof buffer; ++i)
        buffer[i] = 'x';
    int outfd = open("garbage", O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU);
    for (int i = 0; i < 10*KILO; ++i)
        write(outfd, buffer, sizeof buffer);
    close(outfd);
}
```

create-garbage-3.c

```
// create-garbage-3.c

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

#define KILO 1024
#define MEGA KILO*KILO

char buffer[10*KILO];

int main(void) {
    for (int i = 0; i < sizeof buffer; ++i)
        buffer[i] = 'x';
    int outfd = open("garbage", O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU);
    for (int i = 0; i < KILO; ++i)
        write(outfd, buffer, sizeof buffer);
    close(outfd);
}
```

create-garbage-4.c

```
// create-garbage-4.c

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

#define KILO 1024
#define MEGA KILO*KILO

char buffer[10*MEGA];

int main(void) {
    for (int i = 0; i < sizeof buffer; ++i)
        buffer[i] = 'x';
    int outfd = open("garbage", O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU);
    write(outfd, buffer, sizeof buffer);
    close(outfd);
}
```

create-garbage-5.c

```
// create-garbage-5.c

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <pthread.h>

#define KILO 1024
#define MEGA KILO*KILO

char buffer[MEGA];

void *thread_body(void *arg) {
    int thread_number = (int)arg;
    printf("starting thread %d...\n", thread_number);
    int outfd = open("garbage", O_WRONLY);
    // lseek moves to the place in the file where this thread will write
```

```
    lseek(outfd, thread_number * MEGA, SEEK_SET);
    write(outfd, buffer, sizeof buffer);
    close(outfd);
    printf("done thread %d\n", thread_number);
    return NULL;
}

int main(void) {
    for (int i = 0; i < sizeof buffer; ++i)
        buffer[i] = 'x';
    // To ensure the file exists and is empty before the threads start
    int outfd = open("garbage", O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU);
    close(outfd);
    pthread_t threads[10];
    for (int thread = 0; thread < 10; ++thread)
        pthread_create(&threads[thread], NULL, thread_body, (void*)thread);
    for (int thread = 0; thread < 10; ++thread)
        pthread_join(threads[thread], NULL);
}
```

När vi provkör programmen mäter vi upp dessa körtider:

Program	Körtid
create-garbage-1	6.253 s
create-garbage-2	0.019 s
create-garbage-3	0.013 s
create-garbage-4	0.018 s
create-garbage-5	0.005 s

a) Varför är **create-garbage-1** så väldigt långsamt? Förklara!

b) Kan man förklara körtiderna för programmen **create-garbage-2**, **create-garbage-3** och **create-garbage-4**?

c) Kan man säga något om hur många processorkärnor som datorn vi provkörde på har? Vad, och varför?

d) Ett program kan vara flertrådat, men man talar också om flertrådade operativsystemkärnor. Kan man säga något om ifall operativsystemkärnan på datorn som vi provkörde på är flertrådad? Vad, och varför?

Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), 31 maj 2021