Linköpings universitet
Institutionen för datavetenskap 2017-06-03

Tentamen i
TDIU16 Process- och operativsystemprogrammering

Datum 2017-06-03

Tid 14-18

Institution IDA

Kurskod TDIU16

Provkod TEN1

Examinator

Klas Arvidsson (klas.arvidsson@liu.se)

Administratör

Madeleine Häger Dahlqvist

Jourhavande lärare

Klas Arvidsson (013-28 21 46)

Tillåtna hjälpmedel

Inga hjälpmedel.

Instruktioner

- Fyll i tentamensomslaget och läs dess instruktioner innan du börjar. Läs instruktionerna och alla uppgifter innan du börjar.
- Ange din tolkning av frågan och alla antaganden du gör.
- Skriv tydligt. Oläsliga svar bedöms som noll poäng.
- Var precis i dina påståenden. Bevisa ditt resonemang när möjligt. Svävande eller tvetydiga formuleringar leder till poängavdrag.
- Motivera tydligt och djupgående alla påståenden och resonemang. Förklara uträkningar och lösningsmetoder.
- Tentamen har 7 uppgifter på 7 sidor (inklusive denna sida).
- Tentamen är på 30 poäng och betygsätts U, 3, 4, 5 (prel. gränser: 16.0p, 22.0p, 26.0p). Poäng ges för motiveringar, förklaringar och resonemang. Enbart rätt svar ger inte (full) poäng.
- Lycka till!

1. (a) Tabell 1 visar den totala mängden av fyra olika resurser som finns i ett system. I detta system körs för närvarande fyra olika processer. Tabell 2 visar den maximala resursanvändningen för dessa fyra processer och tabell 3 visar hur många resurser av varje typ processerna använder just nu. Process $P2$ begär ytterligare en resurs av typ A. Beräkna med hjälp av Banker's algorithm om begäran ska tillåtas. Systemet är i ett säkert läge, du behöver inte börja med att visa det. [2p]

A	B	C	D
7	4	7	7

Tabell 1: Totalt antal resurser

	A	B	C	D
P1	3	1	1	1
P2	7	3	3	4
P3	2	2	2	3
P4	0	1	1	5

Tabell 2: Maximalt resursbehov

	A	B	C	D
P1	0	0	1	0
P2	5	2	2	1
P3	0	1	2	1
P4	0	0	1	0

Tabell 3: Nuvarande resursanvändning

(b) Vad innebär det att ett system är i ett **osäkert** läge när Bankers Algorithm tillämpats? Välj ett av följande alternativ och motivera varför det stämmer: [1p]

- Systemet hamnar i deadlock efter allokeringen.
- Systemet kommer hamna i deadlock efter kommande allokeringar.
- Systemet kanske hamnar i deadlock efter kommande allokeringar.

2. Kim har implementerat `start`, `exit` och `wait` enligt koden i kodlistning 1. Dessa ska fungera liknande hur `exit` respektive `wait` fungerar i Pintos: vi antar att `wait` endast anropas en gång för varje process, och endast för processer man får köra wait på.

```
1  struct proc_exit {
2    int exit_status;
3    struct semaphore sema;
4  };
5  struct proc_exit proc_list[100];
6  struct lock exit_lock;
7
8  // Anropas precis innan en process startar. Processen har id 'proc_id'.
9  void start(int proc_id) {
10   struct proc_exit *child;
11
12   lock_acquire(&exit_lock);
13     child = &proc_list[proc_id];
14     sema_init(&child->sema, 0);
15     child->exit_status = -1;
16   lock_release(&exit_lock);
17 }
18 // Anropas när processen med id 'proc_id' avslutas. 'status' är
19 // statuskoden.
20 void exit(int proc_id, int status) {
21   struct proc_exit *child;
22
23   lock_acquire(&exit_lock);
24     child = &proc_list[proc_id];
25     sema_up(&child->sema);
26     child->exit_status = status;
27   lock_release(&exit_lock);
28 }
29 // Anropas när en process vill vänta på att processen med id 'child_id'
30 // ska avslutas. 'wait' ska alltså vänta på att processen avslutas och
31 // returnera den statuskod som skickades till 'exit' för den processen.
32 int wait(int child_id) {
33   struct proc_exit *child = NULL;
34   int status = -1;
35
36   lock_acquire(&exit_lock);
37     child = &proc_list[child_id];
38   lock_release(&exit_lock);
39   sema_down(&child->sema);
40   status = child->exit_status;
41
42   return status;
43 }
```

Listing 1: Implementation av start, exit och wait

(a) Finns det risk för deadlock? I så fall, var och varför? Tänk igenom de fyra kraven för deadlock för att göra din motivering. [2p]

(b) Vilken resurs hanterar semaforen `sema` som finns i `proc_exit` för varje process? [1p]

(c) Finns det några synkroniseringfel? I så fall, var och varför? [1p]

Tips: Ägna särskild uppmärksamhet åt ordningen som operationer sker i.

3. Kirsten sitter och implementerar låsfria datastrukturer med hjälp av atomiska operationer. Tyvärr har Kirsten bara tillgång till `compare_and_swap` och `atomic_swap` men behöver `test_and_set`. Kodlistning 2 visar hur funktionerna ska fungera. Implementationerna ska fortfarande vara atomiska, det går alltså att använda funktionerna för att modifiera samma data ifrån olika trådar.

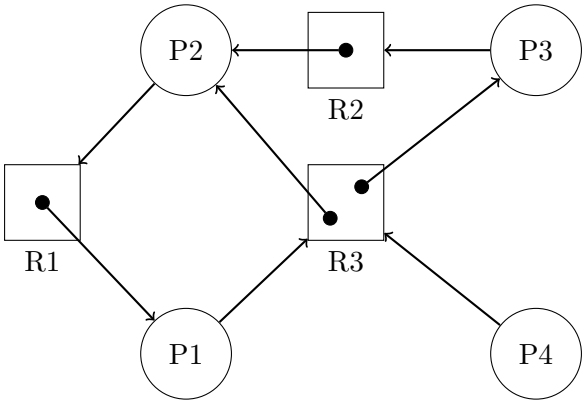
```
1  int compare_and_swap(int *val, int compare, int swap) {
2    atomic {
3      int tmp = *val;
4      if (*val == compare)
5        *val = swap;
6      return tmp;
7    }
8  }
9
10 void atomic_swap(int *a, int *b) {
11   atomic {
12     int tmp = *a;
13     *a = *b;
14     *b = tmp;
15   }
16 }
17
18 int test_and_set(int *val) {
19   atomic {
20     int old = *val;
21     *val = 1;
22     return old;
23   }
24 }
```

Listing 2: Implementation av atomiska operationer

(a) Implementera `test_and_set` med hjälp av `compare_and_swap`. [2p]

(b) Implementera `test_and_set` med hjälp av `atomic_swap`. [2p]

4. (a) Figur 1 visar en resursallokeringsgraf för ett system med tre resurstyper, R1, R2 och R3, samt fyra processer, P1, P2, P3 och P4. Det finns inga andra processer eller resurser i detta system. Är systemet i deadlock? Motivera ditt svar. Använd gärna villkoren för deadlock i din motivering. [2p]



Figur 1: Resursallokeringsgraf

(b) Antag att ytterligare en resurs av typ R3 finns i systemet i fig. 1. Är systemet fortfarande i deadlock? Motivera ditt svar. [2p]

Tips: Tänk på att det inte finns några ytterligare processer eller resurser i systemet.

Bakgrund till uppgift 5-7

På sidan 6-7 finns given kod för att implementera ett system för användarstudier. Det kan som mest vara 20 användare inloggade samtidigt. Systemet klarar bara att lagra svar från 100 användare utan att startas om. Varje användare representeras av en tråd som kör funktionen `do_survey`. Ett godtyckligt antal sådana trådar startas under systemets körning. Vid inloggning kontrollerar systemet om fler än 100 svar redan har lämnats, och nekar i så fall tillgång till systemet.

5. Kan ett olägligt trådbyte göra att fler än 20 användare loggas in i systemet? [3p]

6. (a) Förklara begreppet busy-wait. [1p]

(b) Var i den givna koden förekommer busy-wait? Hur kan man eliminera dessa? Det är självklart tillåtet att lägga till godtyckliga funktioner och variabler var som helst i koden. Du får använda de primitiver som är givna nedan om du vill. [3p]

7. Återanvänd gärna din kod från uppgift 6 när du svarar på följande frågor.

(a) Förklara vilka kritiska sektioner som finns i koden och använd lås för att synkronisera dessa. För mycket eller för lite omotiverad låsning kan ge poängavdrag. [4p]

(b) Använd atomiska operationer för att leta upp en ledig plats, se hur dessa fungerar i Kodlistning 2. Du har tillgång till `atomic_swap`, `test_and_set`, och `compare_and_swap`. [2p]

(c) Många användare mot slutet av varje körning hör av sig om att deras svar inte kunde sparas (se utskriften på rad 38 i Kodlistning 4). Vad beror detta på? Hur kan man lösa det? [2p]

Tillgängliga synkroniseringsprimitiver

```
struct semaphore;
void sema_init(struct semaphore *, int);
void sema_down(struct semaphore *);
void sema_up(struct semaphore *);

struct lock;
void lock_init(struct lock *);
void lock_acquire(struct lock *);
void lock_release(struct lock *);

struct condition;
void cond_init(struct condition *);
void cond_wait(struct condition *, struct lock *);
void cond_signal(struct condition *, struct lock *);
```

Listing 3: Tillgängliga synkroniseringsprimitiver

Kod till uppgift 5-7

```
struct account {
  bool in_use;
};

struct answer {
  // Dessa variabler innehåller svaren vi vill spara.
  bool a1, a2, a3;
};

struct account accounts[20];
struct answer answers[100];
int num_answers;

// Anropas en gång innan några trådar har startat.
void init() {
  num_answers = 0;
  for (int i = 0; i < 20; i++) {
    accounts[i].in_use = false;
  }
}

// Be användaren svara på några frågor. Returnerar svaren.
// Denna funktion är trådsäker.
struct answer get_answer();

```

```
26 // Godtyckligt antal trådar kör denna funktionen vid olika tidpunkter.
27 int do_survey() {
28   int account = login();
29   if (account == -1) {
30     printf("Sorry, you were too late :'(\n");
31     return -1;
32   }
33
34   struct answer answer = get_answer();
35   bool saved = save_answer(&answer);
36
37   if (!saved)
38     printf("Your answers could not be saved. Contact an admin.\n");
39
40   logout(account);
41 }
42
43 bool answers_full() {
44   return num_answers >= 100;
45 }
46
47 // Hitta ett ledigt användarkonto och logga in på det.
48 int login() {
49   if (answers_full())
50     return -1;
51
52   int i = 0;
53
54   while (accounts[i].in_use) {
55     i = (i + 1) % 20;
56   }
57
58   accounts[i].in_use = true;
59   return i;
60 }
61
62 void logout(int account) {
63   accounts[account].in_use = false;
64 }
65
66 // Spara svaren om det finns plats.
67 bool save_answer(struct answer *answer) {
68   if (answers_full())
69     return false;
70
71   answers[num_answers] = *answer;
72   num_answers++;
73   return true;
74 }
```

Listing 4: Kod till uppgift 5-7