16

TENTAMEN (EXAMINATION)

Tentamensdatum/*Examination date:* 2018-06-05
(åå-mm-dd/*yy-mm-dd*)

AID-nummer
AID number

Ifylles av student

4	5	8	7	5	

Completed by student

Ifylles av vakt

4	5	8	7	5	

Completed by supervisor

Kurskod/*Course code:* TDDB68 **Provkod/*Exam code:*** TEN1

Kursnamn/*Course title:* Concurrent Programming and Operating Systems

Institution/*Department:* IDA

Jag intygar att varken mobil eller något annat otillåtet hjälpmedel finns tillgängligt under tentamen.
I confirm that no mobile or other non-permitted aids are available during the examination. ☒

Inlämnat: antal lösblad 10 **tentamensformulär** ☐
Enclosed: number of sheets ***exam booklet***

Markera behandlade uppgifter med X/*Mark tasks attempted with an X*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X här/*here*	X	X	X	X	X	X									
Erhållna poäng *Points obtained*	8	2	6	4.5	6	2,5									
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
X här/*here*															
Erhållna poäng *Points obtained*															

Anvisningar/*Instructions*

1. Skriv AID-nummer, datum, kurskod och provkod på varje blad som lämnas in/
   Write AID number, date, course code and exam code on every sheet that is handed in
2. På varje papper får högst en uppgift lösas om inget annat anges/
   Maximum one task per sheet unless otherwise instructed
3. Skriv endast på papprets ena sida om inget annat anges/
   Use only one side of each sheet unless otherwise instructed
4. Numrera de papper som lämnas in/*Number every sheet that is handed in*
5. Använd inte röd penna/*Do not use a red pen/pencil*

Sen inlämning ☐
Late hand in

Klockslag _____
Time

Orsak _____
Reason

_____

∑ Poäng/*Points:* 29 Betyg/*Grade:* 4

Examinator/*Examiner:* _____

LINKÖPINGS UNIVERSITET

45875 2018-06-05

TPDB68 TEN1

1

Multiple choice form for answering question 1. Please put X:s in the appropriate cells:

	A	B	C	D	
1 a)	X				0
1 b)			X	X	1
1 c)		X			1
1 d)		X	X		1
1 e)	X		X		1
1 f)		X		X	1
1 g)		X	X	X	1
1 h)		X			1
1 i)	X			X	0
1 j)	X	X	X	X	1

/8r

2a

```
item_type popFront(&D)
{
  wait(s); // s is a global semaphore
  if(!isempty(D) && D.front() != null)
  {
    elem temp = D.pop();
    signal(s);
    return temp.v;
  }
  else
  {
    signal(s);
    return null;
  }
}
```

What happens here?

1p

Define D's struct.

What happens if D is short?

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 45875	Datum: *Date:* 2018-06-05	Blad nummer: *Sheet number:* 3
Kurskod: *Course code:* TDDB68	Provkod: *Exam code:* TEN1	

2b

del 1

```
item_type popFront(&D)
{
  sema_up(front_S); // global semaphore
  sema_up(pop_S); // global semaphore
    if(!isempty(D) && D.front() != null)
    {
      elem temp = D.pop();
      sema_down(front_S);
      sema_down(pop_S);
      return temp.v;
    }
  else
  {
    sema_down(front_S);
    sema_down(pop_S);
    return null;
  }
}
```

This solution has has three semaphores. The idea is to prevent race conditions while still allowing elements to be added in the back.

pop_S : this semaphore prevents anyone else to remove elements at the same time. Goes up to 2.

front_S : Only one can access the front at a time.

back_S : — " — back at a time.

2e

fort_

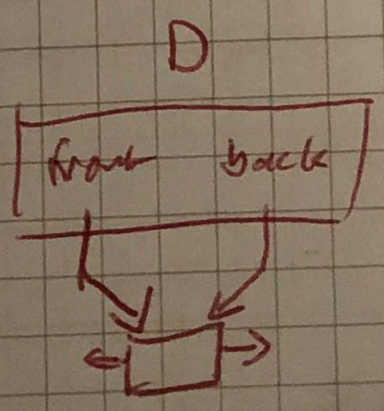
LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 45875	Datum: *Date:* 2018-06-05	Blad nummer: *Sheet number:* 4
Kurskod: *Course code:* TDDB68	Provkod: *Exam code:* TEN1	

2b
del 2.

The front and back semaphores make sure that only one process is executing there at a time.
The pop semaphore makes sure that only one process can remove elements at a time but still allows two processes to add elements at different ends at the same time.

still problematic with short queues



pip front. & push Back will interfere

1p

1p

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 45875	Datum: *Date:* 2018-06-05	Blad nummer: *Sheet number:* 5
Kurskod: *Course code:* TDDB68	Provkod: *Exam code:* TEN1	

3a It requires that there are a fixed amount of resources..

The processes does not know how many of each resource it needs (1)

3b If there are circles in the wait-for-graph of the resources. (1)

3c del 1

ALLOC

	R1	R2	R3
P1	0	1	0
P2	0	1	3
P3	1	0	0
P4	0	0	0

MAX

	R1	R2	R3
P1	3	3	2
P2	2	1	3
P3	3	0	3
P4	0	2	0

avail: [3, 5, 0]

Grant request [2,0,0] to P3. NEED = MAX - ALLOC'

ALLOC'

	R1	R2	R3
P1	0	1	0
P2	0	1	3
P3	3	0	0
P4	0	0	0

NEED

	R1	R2	R3
P1	3	2	2
P2	2	0	0
P3	0	0	3
P4	0	2	0

available: [1, 5, 0]

need ≤ available

Only P4 has enough available resources. P4 use and release its resources. Finnished: [F,F,F,T], available: [1, 5, 0]

fort..

3c del 2

Now, no other process can get enough resources that they request. (4)

P3s request should not be granted.

4a

A process is a program in execution. Each process has atleast one kernel thread and one user-level-thread. Each process usually has their own memory while all the threads belonging to a process share heap and memory.

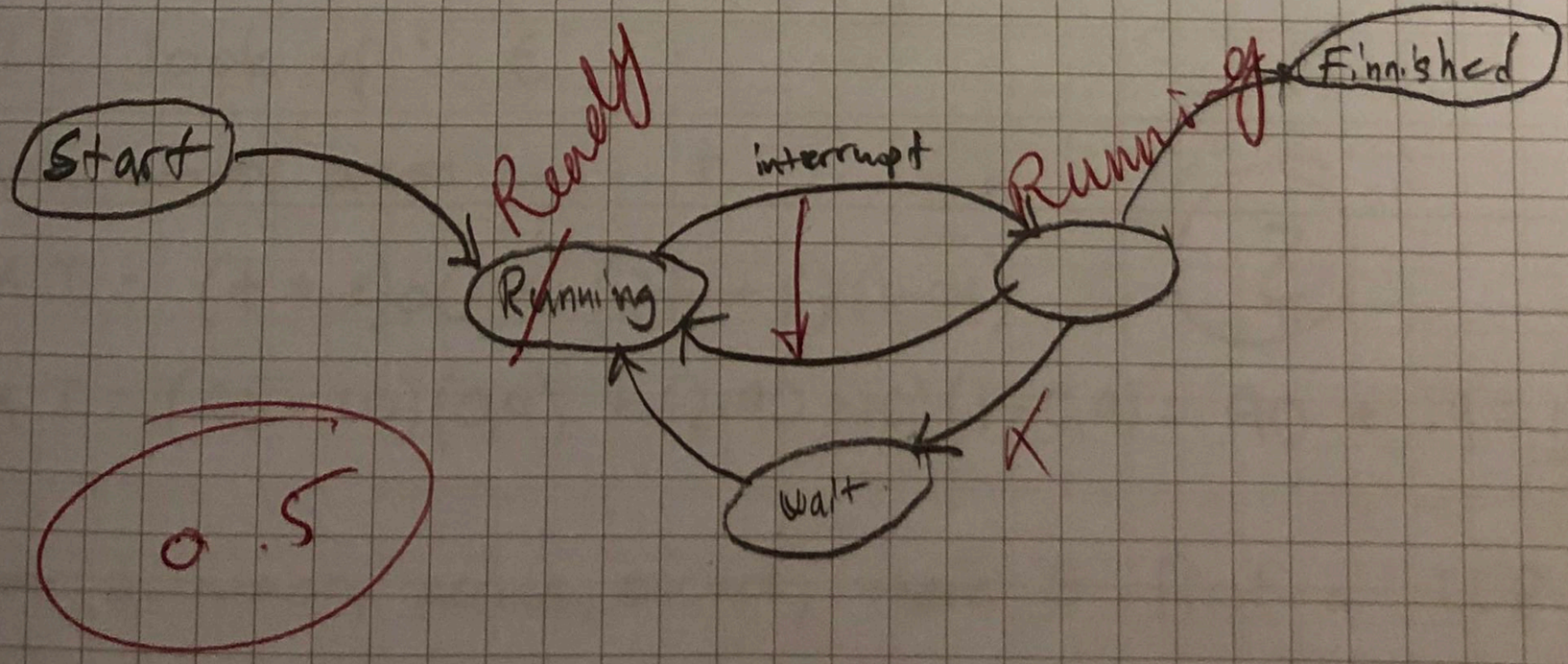
2

4b

One model is one-to-one when each kernel thread has only one user-level-thread. It is appropriate for multiprocessor systems because it is scaleable. There is a draw back with more overhead as each kernel-thread only has one user-level-thread.

2

4c

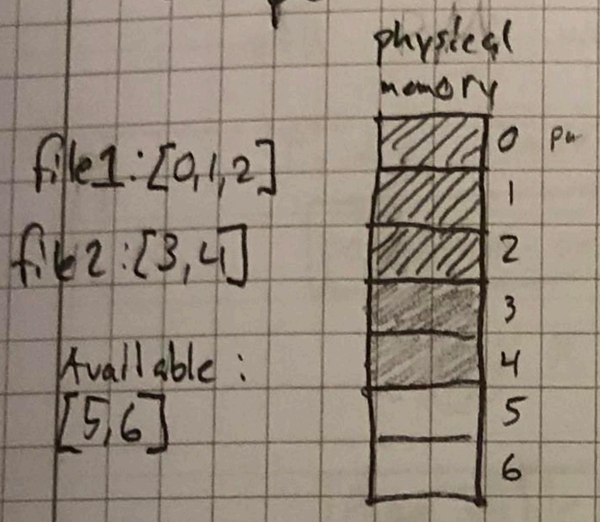


5a

External fragmentation.

Because all the allocated memory has to be contiguous empty spaces that are too small will not be used. (1)

Example



If file1 does not need it's memory anymore that space will be made available. If another file needs more memory than 3 pages then page 0-2 will be unused.

5b

TLB hit rate: α

TLB look up: ε

Physical memory access time: t

$EAT = (t + \varepsilon)\alpha + (2t + \varepsilon)(1-\alpha)$ (2)

$EAT = (90 + 10)(0,9) + (2 \cdot 90 + 10)(1 - 0,9) = 90 + 19 = \underline{109 ns}$

When a memory access occurs, there is first a TLB lookup if that address has recently been used. If the address exists in the TLB then it accesses the physical memory right away. However if it is not in the TLB then it accesses the page table to find where in the physical memory it is stored. Then it accesses the physical memory.

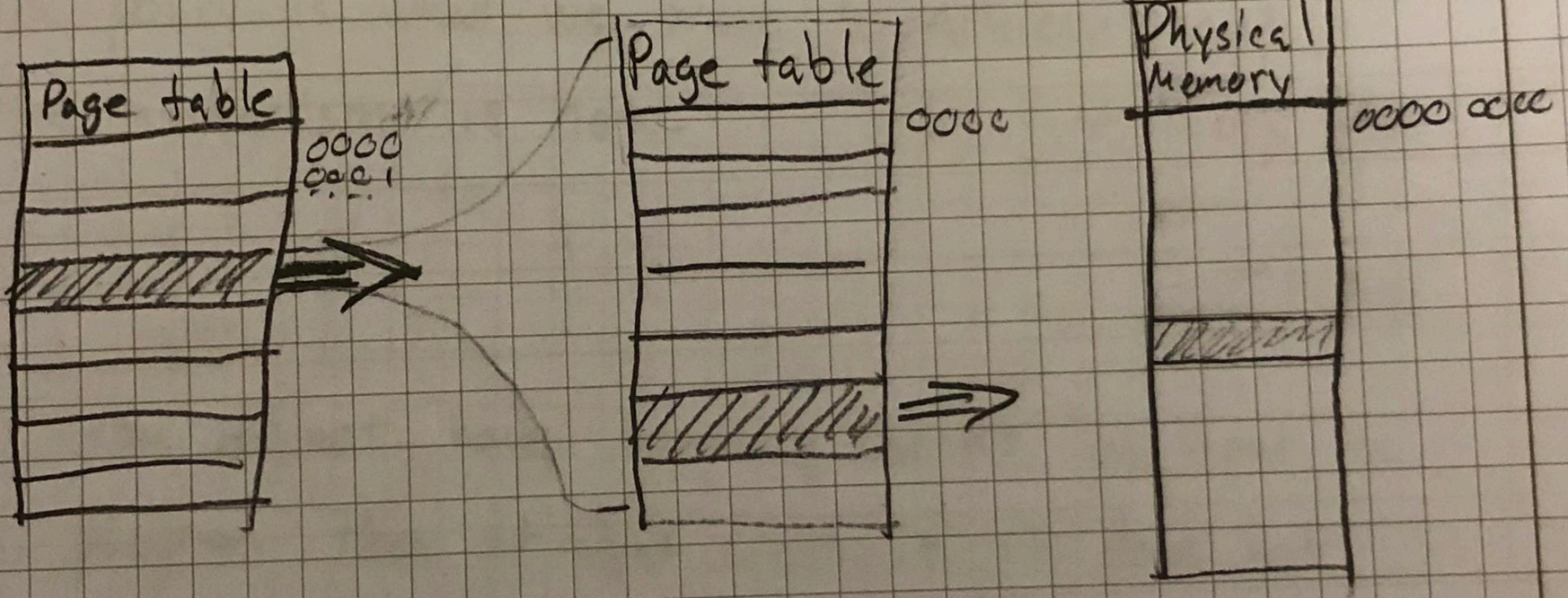
5C

3

One solution is multilevel-page-table.
Lets say that we have a two-level-page-table then
then the first level of the page-table will link
to multiple-page-tables below it. These page-tables
will reveal where the pages are stored in physical-memory.
If we have a 16bit address it could look likethis.

page-level-one: 2^4	page-level-two: 2^4	information 2^8

XXXXYYYYZZZZZZZZ



The X's represent in what page table the page
is stored. The Y's represent where in physical memory
the page is stored.

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 45875	Datum: *Date:* 2018-06-05	Blad nummer: *Sheet number:* 10
Kurskod: *Course code:* TDDB68	Provkod: *Exam code:* TEN1	

6a

If you want to restict access to a website you can have a username and password to ensure that no unwanted guests gets access.
The policy in this case is "no unwanted guests gets access" and the mechanism to achieve that is the authentication with username and password.
The policy is what we want to achieve and the mechanism is how.

1p

ok, but why?

6b

If you do not have boundary checks on input to a program then it might crash if it does not get what it expects. That is a vulnerability for the program. If someone writes input with the intent of damaging or extracting data with that flaw then thats an attack.

1,5p

ok example, but description not so clear