Front side

Emnekode: DAT320

Emnenavn:

Operativsystemer og systemprogrammering

År og semester: 2022 Høst

Eksamensdato: 16.12.2022

Klokkeslett: 09:00-13:00

Tillate hjelpemidler: Kalkulator

Faglærer: Robert Ewald - 51875111

1 Process

What is a process?
Select one alternative:

- A hardware driver
- The operating system
- A program in memory
- A program on disk

Maximum marks: 1

2 I/O Changes

Which of the following is responsible to notify the CPU about I/O changes?
Select one alternative:

- Exception
- Signal
- Notification
- Interrupt

Maximum marks: 1

3 Address translation

Which device is responsible for address translation?
Select one alternative:

- Translation Look-aside Buffer
- Virtual Frame Translator
- Memory Management Unit
- Page Table Converter
- Address Bus

Maximum marks: 1

4 State transition

Which state transitions are possible between RUNNING, READY and BLOCKED?
Select one or more alternatives:

- BLOCKED - READY
- RUNNING - READY
- READY - RUNNING
- BLOCKED - RUNNING
- RUNNING - BLOCKED
- READY - BLOCKED

Maximum marks: 2

5 create process

Which system call is used to create a process?
Select one alternative:

- create_proc()
- fork()
- spawn()
- new()

Maximum marks: 1

6 Terminate process

Which shell command is used to terminate a process?
Select one alternative:

- terminate
- end
- conclude
- kill

Maximum marks: 1

7 Restrict process

Which feature allows to restrict a process?
Select one alternative:

- pipelining
- user accounts
- kernel/user mode
- access control lists

Maximum marks: 1

8 return

The fork() system call returns twice
Select one alternative:

- True
- False

Maximum marks: 1

9 Execution mechanism

What is the mechanism called which isolates processes efficiently and securely?

Maximum marks: 1

10 Context switch

Which resources don't need to be saved in a context switch?
Select one alternative:

- Program counter
- General Purpose CPU registers.
- Stack pointer
- Translation look-aside buffer

Maximum marks: 1

11 Parent child 1

Select one or more alternatives. All ticks need to be correct get points:

- Alternative 3
- Alternative 4
- Alternative 2
- Alternative 1

Maximum marks: 2

12 Parent Child 2

Select one or more alternatives:

- [ ] Alternative 2
- [ ] Alternative 4
- [ ] Alternative 3
- [ ] Alternative 1

Maximum marks: 2

13 preemptive context switch

What hardware feature is needed to enable preemptive scheduling?
Select one alternative:

- ○ preemption register
- ○ disk interrupt
- ○ timer interrupt
- ○ system call

Maximum marks: 1

14 System call

Which statements are true about system calls?
Please match the values:

	False	True
Executed using a trap (software interrupt)	○	○
Are defined at boot time	○	○
Executed in user mode	○	○

Maximum marks: 3

15 Memory virtualization

Virtualization of the whole memory is described by which term?
Select one alternative:

- ○ Virtual address
- ○ Address space
- ○ Base and bounds
- ○ Paging

Maximum marks: 1

16 Memory types

This is a memory type allocated by the operating system. True or false?

	False	True
Read only memory	○	○
Random access memory	○	○
Stack	○	○
Heap	○	○

Maximum marks: 4

17 Memory accesses

Consider paging without caching.

How many memory accesses are necessary to load a value from an address: [ ] .

Maximum marks: 1

18 PFN size

Given a page size of 4KB and a 64 bit architecture. How many bits are used for the PFN?

[ ] bits

Maximum marks: 2

19 TLB

What is the translation look-aside buffer?
Select one alternative:

- A cache for main memory
- a buffer for translated instructions
- A cache for page tables
- Memory for page tables

Maximum marks: 1

20 Allocation

Replace with question text
Mark the statements as true or false

	True	False
The function returns a pointer		
Memory space for r is allocated in ROM		
Another thread can safely use the returned value		
Memory space for r is allocated on the stack		
Memory space for r is allocated on the heap		

Maximum marks: 5

21 Address Resolution 1

Given a 16 bit architecture and a page size of 256 bytes.

Page Table

PFN	P	U/S	R/W	V
a1	1	0	1	1
b2	1	1	0	1
c4	1	0	1	1
--- 250 items omitted ---				
01	1	0	1	1
02	1	0	1	0
03	1	0	0	0

What is the physical address for virtual address **02fe**? Enter the address in hex:

What is the physical address for virtual address **0201**? Enter the address in hex:

What is the physical address for virtual address **0101**? Enter the address in hex:

What is the physical address for virtual address **feab**? Enter the address in hex:

Maximum marks: 8

22 Address Resolution 2

Given a 16 bit architecture and a page size of 512 bytes.

Page Table

PFN	P	U/S	R/W	V
a2	1	0	1	1
b2	1	1	0	1
c4	1	0	1	1
--- 122 items omitted ---				
01	1	0	1	1
02	1	0	1	0
03	1	0	0	0

What is the physical address for virtual address **02fe**? Enter the address in hex:

What is the physical address for virtual address **0201**? Enter the address in hex:

What is the physical address for virtual address **0101**? Enter the address in hex:

What is the physical address for virtual address **feab**? Enter the address in hex:

Maximum marks: 8

23 Critical section

After which line must the mutex be locked? Enter the line number here:

After which line must the mutex be unlocked? Enter the line number here:

Maximum marks: 2

24 Trace

Fill out the trace table for the segmentation fault case.

Thread 1	Thread 2
Select alternative (c, not running, b, a)	Select alternative (c, b, a, not running)
Select alternative (c, b, not running, a)	Select alternative (not running, a, b, c)
Select alternative (b, a, not running, c)	Select alternative (a, c, b, not running)

Maximum marks: 3

25 deadlock

Which conditions need to be present to cause a deadlock?
Please match the values:

	required	not required
hold-and-wait	○	○
no preemption	○	○
circular wait	○	○
I/O operations	○	○
mutual exclusion	○	○
multiple CPUs	○	○
limited memory	○	○

Maximum marks: 7

26 Create thread

Which function is used to create a thread?
Select one alternative:

- ○ pthread_fork()
- ○ pthread_create()
- ○ pthread_start()
- ○ pthread_new()

Maximum marks: 1

27 Wait for another thread

Which function is used wait for completion of another thread?
Select one alternative:

- sleep()
- pthread_wait()
- pthread_join()
- pthread_suspend()

Maximum marks: 1

28 Turnaround time

Assume the First Come First Served / First In First Out scheduling algorithm is used. The processes arrived in the order noted in the table. The jobs are not preempted. Calculate the average turnaround time.

Process	Arrival Time	Job runtime
P_1	0	10
P_2	0	100
P_2	0	10

Enter the average turnarount time here:

Maximum marks: 2

29 Round Robin

Consider these statements about round-robin scheduling. Which are true/false?

Please match the values:

	False	True
RR is fair	○	○
RR enables low response time	○	○
RR takes into account I/O	○	○
RR enables low turnaround time	○	○

Maximum marks: 4

30 Multi-level feedback queue

Which rules apply for multi-level feedback queues to avoid causing starvation?

Please match the values:

	applies	does not apply
If a job gives up CPU before the time slice is up, it stays at the same priority level	○	○
If priority(A) > priority(B): A runs	○	○
When a job enters the system it is placed at the highest priority	○	○
After some time period S, move all the jobs in the system to the topmost queue	○	○
If priority(A) = Priority(B): A & B run in RR	○	○
Once a job uses up its time allotment at a given level its priority is reduced	○	○

Maximum marks: 6

31 Process Priority

Which shell command is used to change the priority of a process
Select one alternative:

- chgpri
- nice
- yield
- priority

Maximum marks: 1

32 Page access cache

Consider a newly-created process that has been allocated a cache size of 5 pages, and then generates the following page accesses:
A C E F D B A B F F D C C G A G C E F D G E B D G G

How many cache misses are observed for this access stream when using of the FIFO page replacement algorithm?

What is its hit rate in percent? Hit rate %

How many cache misses are observed for this access stream when using of the FIFO page replacement algorithm and a cache size of 6?

The cache algorithm is replaced with the least frequently used algorithm. The cache size remains at 6. How many cache misses can be observed?

Maximum marks: 12

33 Locality

Which properties of the access pattern enable a cache perform well?
Please match the values:

	Does enable	Does not enable
Process locality	○	○
Address locality	○	○
Temporal locality	○	○
Remote locality	○	○
Spatial locality	○	○

Maximum marks: 5

34 Harddisk

Loud noise around a harddisk increases its latency
Select one alternative:

- ○ True
- ○ False

Maximum marks: 1

35 RAID

Which raid configuration should be used to maximize capacity and ensure operation with one failed drive?

Select one alternative

- RAID5
- RAID0
- RAID1
- RAID4

Which raid configuration should be used to minimize latency and ensure operation with one failed drive?

Select one alternative:

- RAID4
- RAID5
- RAID0
- RAID1

Maximum marks: 2

Question 11
Attached

Which outputs of this program are possible?

```
 1  int main(int argc, char *argv[]) {
 2    printf("hello world (pid:%d)\n", (int) getpid());
 3    int rc = fork();
 4    if (rc < 0) {         // fork failed; exit
 5      fprintf(stderr, "fork failed\n");
 6      exit(1);
 7    } else if (rc == 0) {
 8      printf("hello, I am child (pid:%d)\n", (int) getpid());
 9    } else {
10      int rc_wait = wait(NULL);
11      printf("hello, I am parent of %d (rc_wait:%d) (pid:%d)\n",
12      rc, rc_wait, (int) getpid());
13    }
14    return 0;
15  }
```

- Alternative 1

```
hello world (pid:29146)
hello, I am child (pid:29147)
hello, I am parent of 29147 (pid:29146)
```

- Alternative 2

```
hello world (pid:29146)
hello, I am parent of 29147 (pid:29146)
hello, I am child (pid:29147)
```

- Alternative 3

```
hello world (pid:29146)
hello world (pid:29147)
hello, I am parent of 29147 (pid:29146)
hello, I am child (pid:29147)
```

- Alternative 4

```
hello world (pid:29147)
hello, I am child (pid:29147)
hello world (pid:29146)
hello, I am parent of 29147 (pid:29146)
```

Question 12
Attached

Consider the program carefully. Which outputs of this program are possible?

```
 1  int main(int argc, char *argv[]) {
 2    printf("hello world (pid:%d)\n", (int) getpid());
 3    int rc = fork();
 4    if (rc < 0) {         // fork failed; exit
 5      fprintf(stderr, "fork failed\n");
 6      exit(1);
 7    } else if (rc == 0) { // child (new process)
 8      printf("hello, I am child (pid:%d)\n", (int) getpid());
 9    } else {              // parent goes down this path (main)
10      printf("hello, I am parent of %d (pid:%d)\n",
11      rc, (int) getpid());
12    }
13    return 0;
14  }
```

- Alternative 1

```
hello world (pid:29146)
hello, I am child (pid:29147)
hello, I am parent of 29147 (pid:29146)
```

- Alternative 2

```
hello world (pid:29146)
hello, I am parent of 29147 (pid:29146)
hello, I am child (pid:29147)
```

- Alternative 3

```
hello world (pid:29146)
hello world (pid:29147)
hello, I am parent of 29147 (pid:29146)
hello, I am child (pid:29147)
```

- Alternative 4

```
hello world (pid:29147)
hello, I am child (pid:29147)
hello world (pid:29146)
hello, I am parent of 29147 (pid:29146)
```

Question 20
Attached

Consider the following code.

```
1  int *some_function() {
2    int r;
3    r = 1;
4    return &r;
5  }
```

Question 23
Attached

Consider the following code. The function `queue_enqueue` can be called by multiple threads. Identify the critical section in this function.

```
 1  typedef struct __node_t {
 2    int value;
 3    struct __node_t *next;
 4  } node_t;
 5
 6  typedef struct __queue_t {
 7    node_t *head;
 8    node_t *tail;
 9    pthread_mutex_t lock;
10  } queue_t;
11
12  void queue_init(queue_t *q) {
13    node_t *tmp = malloc(sizeof(node_t));
14    tmp->next = NULL;
15    q->head = q->tail = tmp;
16    pthread_mutex_init(&q->lock0, NULL);
17  }
18
19  void queue_enqueue(queue_t *q, int value) {
20    node_t *tmp = malloc(sizeof(node_t));
21    assert(tmp != NULL);
22    tmp->value = value;
23    tmp->next  = NULL;
24    q->tail->next = tmp;
25    q->tail = tmp;
26  }
```

Question 24
Attached

1 Trace

Consider the following code. This code sometimes fails with a Segmentation fault. Fill out the trace table for this case. Assume a single CPU.

```
 1  #include <pthread.h>
 2
 3  typedef struct __info {
 4    int counter = 0;
 5  } __info_t;
 6
 7  __info_t *info;
 8
 9  void *count() {
10    if (info) {  // a
11      info->counter = info->counter + 11;  // b
12    }
13  }
14
15  void *ended() {
16    info = NULL; // c
17  }
18
19  int main(int argc, *char[] argv) {
20    info = malloc(sizeof(__info_t));
21    assert(info != NULL);
22    pthread_t thread_1, thread_2;
23    phtread_create(&t1, NULL, count);
24    pthread_create(&t2, NULL, ended);
25  }
```