复旦大学计算机科学技术学院

2013-2014 学年第一学期《操作系统》期末考试试卷

A 卷　共 9 页　　2014 年 1 月

课程代码：COMP130008.01-04　　考试形式：□开卷　☑闭卷

（本试卷答卷时间为 120 分钟，答案必须写在试卷上，做在草稿纸上无效）

NO aids (books, notes, discussion etc.) are allowed. Ask the instructor if you have any questions.

Student ID_____Name_____Score_____

Question	1	2	3	4	Sum
Score					

（装订线内不要答题）

1. Choice (30%) fill in the blank with one most appropriate answer

(1) Which level of I/O system is in charge of issuing commands to device controllers? C

A. Hardware
B. Interrupt handlers
C. device drivers
D. device-independent I/O software

(2) Which one of the following state transitions are impossible for classical processes? A

A. Ready --> Blocked
B. Blocked --> Ready
C. Running --> Ready
D. Ready --> Running

(3) Consider a file with absolute path name "/usr/ast/mbox". If its relative path name can be represented as "../mbox", which one of the following may be the current working directory? C

A. /usr　　B. /usr/ast
C. /usr/ast/temp　　D. /

(4) The copy-on-write mechanism provides D

A. an efficient way to create new processes
B. a clever way to share virtual memory pages (at least temporarily)
C. a way to avoid unnecessary page copying
D. all of the above

E. none of the above

(5) Which one of following scheduling algorithms may lead to starvation? C

A. First come first served
B. Round robin
C. Shortest Process Next
D. Highest Response Ratio Next

(6) The Banker's Algorithm is an example of a technique for B

A. deadlock prevention
B. deadlock avoidance
C. deadlock detection
D. deadlock recovery
E. none of the above

(7) In a file system, if the block size gets larger, what would NOT be observed on average? C .

A. Fewer disk seeks for the same amount of data
B. More internal fragmentation
C. More complex free space management
D. Smaller block allocation structure

(8) Which disk scheduling algorithm can always achieve the best performance in all situations? D

A. SCAN algorithm
B. C-SCAN algorithm
C. Shortest Seek Time First algorithm
D. None of the above

(9) Given the following page reference steam: 3 5 7 5 0 5 3 0 8 6 0 8 6 0 1 2 5 3 6 5 2 1. What is the working set by the end of the reference string, given the working-set window size being 10. A

A. {0 1 2 3 5 6}
B. {6 0 1 2 5 3 6 5 2 1}
C. {3 5 7 5 0 5 3 0 8 6 }
D. {0 3 5 7 8}

(10) Which of the following statements about comparison of paging and segmentation is NOT true? D

A. Segmentation facilitates dynamic linking of segments.

B. Address calculation is slightly more complicated with segmentation
C. Segment protection and sharing is more logical than page protection
D. Simple segmentation suffers from internal fragmentation

2. Short Answer (36%)

(1) For each of the following pairs of terms, identify the context(s) in which they occur. Then define each term and clarify the key difference (s) between the two terms.

(a) "message passing" and "shared memory"
(b) "reference bit" and "dirty bit"

Referential Answer:

(a) "message passing" and "shared memory"
context: IPC
message passing: Exchange messages between cooperating processes.
shared memory: Read and write data in shared region
key differences: message passing Communicate & Sync actions without sharing the same address space, but slower than shared memroy

(b) "reference bit" and "dirty bit"
context: paging-based memory management
reference bit: indicates if a page has been accessed (recently)
dirty bit: indicates if a page has been modified (relative to disk)
These bits influence decision-making in page replacement policies

(2) Identify three contexts in which caching was used as a solution technique. Briefly discuss the technical issues involved, and the benefits of caching as a solution.

Referential Answer:

CPU: on-chip instruction/data caches to avoid memory latency

buffer cache: in-memory caching to avoid disk latency

disk cache: remember recent data blocks to avoid disk access latency

write cache: buffer outgoing writes to improve system performance

storage: sequential read-ahead and prefetch for I/O controller

file system: in-memory caching of superblock, inodes, directory info

NFS: client-side caching of info to avoid network latency

Caching optimizes for the common case, and improves performance.

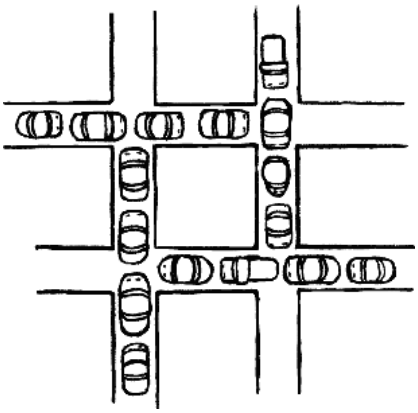
TLB

(3) When a process creates a new process using the fork() operation, which of the following state is shared between the parent process and the child process?

a. Stack
b. Heap
c. Shared memory segments

Referential Answer
Among the three choices, only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

(4) Gridlock is a term describing a traffic situation in which there are so many cars in the streets and the intersections that essentially no car can move any direction because other cars are in the way. Explain how this is the same as deadlock in an operating system by showing how each of the four conditions for deadlock hold in this situation.



Answer:

The four conditions that must hold for deadlock to occur are:

Mutual exclusion - holds because two cars cannot occupy the same spot.

Hold-and-wait - holds because a car in a particular spot holds that spot while waiting for a place to move into.

No preemption - holds because cars cannot magically steal another car's spot while the other car is in it.

Circular wait - holds because in gridlock we have lots of cars waiting for each other to move.

(5) Although DMA does not use the CPU, the maximum transfer rate is still limited. Consider reading a block from the disk. Name three factors that might ultimately limit the rate transfer.

Referential Answer
The first factor could be the limiting speed of the I/O device; Second factor could be the speed of bus, the third factor could be no internal buffering on the disk controller or too small internal buffering space. Fourth factor could be erroneous disk or transfer of block.

(6) What resources are used when a thread is created? How do they differ from those used when a process is created?

Referential Answer
Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

3. Analysis （22%）

(1) Suppose two processes enter the ready queue with the following properties:

P1:Process 1 has a total of 8 units of work to perform, but after every 2 units of work, it must perform 1 unit of I/O (so the minimum completion time of this process is 12 units). Assume that there is no work to be done following the last I/O operation.

P2:Process 2 has a total of 20 units of work to perform. This process arrives just behind P1.

Show the resulting schedule for the shortest-job-first (preemptive) and the round-robin algorithms. Assume a time slice of 4 units for RR. What is the completion time of each process under each algorithm? [10 points]

Answer:

SJF:

StartTime	0	2	3	5	6	8	9	11	28
Process	P1	P2	P1	P2	P1	P2	P1	P2	

P1 completes (with I/O) at time unit 12. P2 completes at 28.

RR:

StartTime	0	2	6	8	12	14	18	20	28
Process	P1	P2	P1	P2	P1	P2	P1	P2	

P1 completes (with I/O) at time unit 21. P2 completes at 28.

(2) Given the following stream of page references by an application, calculate the number of page faults the application would incur with the following page replacement algorithms. Assume that all page frames are initially free.

Reference Stream: A B C D A B E A B C D E B A B

a. FIFO page replacement with 3 frames available [3 points]
b. LRU page replacement with 3 frames available [3 points]
c. OPT page replacement with 3 frames available. [3 points]
d. Someone claims that if we increase the number of frames from 3 to 4, the number of page faults always decreases using FIFO page replacement. Is this statement correct? Briefly explain. [3 points]

Answer:

a. 11 page faults

Refrenence stream:	*A*	*B*	*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*B*	*A*	*B*
oldest page	*A*	*A*	*A*	*B*	*C*	*D*	*A*	*A*	*A*	*B*	*E*	*E*	*C*	*D*	*D*
		B	*B*	*C*	*D*	*A*	*B*	*B*	*B*	*E*	*C*	*C*	*D*	*B*	*B*
Newest page			*C*	*D*	*A*	*B*	*E*	*E*	*E*	*C*	*D*	*D*	*B*	*A*	*A*
page fault	√	√	√	√	√	√	√			√	√		√	√	

b. 12 page faults

Refrenence stream:	*A*	*B*	*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*B*	*A*	*B*
least recently used	*A*	*A*	*A*	*B*	*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*E*
		B	*B*	*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*B*	*A*
most recently page			*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*B*	*A*	*B*
page fault	√	√	√	√	√	√	√			√	√	√	√	√	

c. 8 page faults

Refrenence stream:	*A*	*B*	*C*	*D*	*A*	*B*	*E*	*A*	*B*	*C*	*D*	*E*	*B*	*A*	*B*
	A	*B*	*C*	*D*	*D*	*D*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*
		A	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*
			A	*A*	*A*	*A*	*A*	*A*	*A*	*C*	*D*	*D*	*D*	*A*	*A*
page fault	√	√	√	√			√			√	√			√	

d. False due to Belady's anomaly. For instance, for this problem if we increase the number of physical pages available from 3 to 4, then the number of page faults increases from 11 to 12 using a FIFO page replacement algorithm.

4. Advanced Problem（12%）

(1) Given the following correct serial implementation of a stack:

```
1  typedef uint elem_t;
2  elem_t *head; // the top of the stack
3  void push(int key, int value)
4  {
5    elem_t *e = malloc(sizeof(*e));
6    e->next = head;
7    e->key = key;
8    e->value = value;
9    head = e;              // Put it on the stack
10 }
11
12 elem_t *pop(void)
13 {
14   elem_t *e = head;
15   if (e) head = e->next;
16   return e;
17 }
```

```
18
19 elem_t *search(int key)
20 {
21    for (elem_t *e = head; e; e = e->next) {
22       if (e->key == key) {
23          return e;
24       }
25    }
26    return NULL;
27 }
```

We want investigate whether or not it can be run correctly on a multicore computer in which all cores share a cache-coherent memory. We place head and other variables in shared memory so that different cores can push and pop from the shared stack.

a. Is there any race-conditions if two threads on different cores concurrently invoke search, push, and pop? Give an example or a counterexample to support your answer. [4 points]

b. To improve the solution (for correctness and/or performance), we rewrite above implementation by using a read-write lock as below:

```
struct rwlock {
spinlock_t l;
volatile unsigned nreader;
};

void acquire_read(struct rwlock *rl)
{
spin_lock(&rl->l);
atomic_increment(&rl->nreader);
spin_unlock(&rl->l);
}

void acquire_write(struct rwlock *rl)
{
spin_lock(&rl->l);
while (rl->nreader) /* spin*/ ;
}
```

It guarantees that either one writer can acquire the lock in write mode, or several readers can acquire the lock in read mode. The resulting locked versions of push and search are:

```
struct rwlock rwlock;
void locked_push(int key, int value)
{
  acquire_write(&rwlock);
  push(key, value);
  release_write(&rwlock);
}

elem_t *locked_search(int key)
{
  acquire_read(&rwlock);
```

```
  elem_t *e = search(key);
  release_read(&rwlock);
  return e;
}
```

If we invoke many locked_search's concurrently, why does the version with a read/write lock perform better than if we had used a regular spin lock? [4 points]

c. After extensive experiment, one talent observes that even with a read/write lock and no concurrent push's or pop's, 48 concurrent locked_search's run slower than 48 concurrent search's. Explain briefly why. [4 points]

Referential Answers:

a: Consider the following interleaving of two threads, performing a push and a pop respectively. After this, *the popped element will still appear in the stack.*

```
6  e->next = head;
7  e->key = key;
8  e->value = value;
                                  14 elem_t *e = head;
                                  15 if (e) head = e->next;
9  head = e;
```

!!! NOTE!!! Other reasonable interleaving scenarios are possible!

b: The read/write lock allows searches to proceed in parallel, whereas a spinlock serializes all search operations.

c: Answer: acquire_read is itself serialized because it briefly holds a spinlock, which incurs overhead in locked_search.

（装订线内不要答题）