PDS/SDP OS internals 25/06/2024

PDS: L'esame è comune ai corsi PDS e SDP (in inglese). Si è predisposta un'unica proposta di soluzione in inglese. Qualora ci fossero errori nelle soluzioni (possibili, specie tenendo conto di modifiche e aggiustamenti fatti nel preparare l'esame), si pubblicheranno aggiornamenti.
SDP ON/OFF: As the ON/OFF exam is a subset of the standard one, the ON/OFF solution can be easily extracted from the standard one. If there are errors in the solutions (possible, especially taking into account last minute changes and adjustments made in preparing the exam), updates will be published.

1 In the context of virtual memory management, consider a system with demand paging. Answer the following questions:

A) Is the working set page replacement policy a fixed size policy? (YES/NO)
Briefly explain/motivate:
No, the size of the resident set changes, depending on the set of pages included in the Delta window.

B) Why is it hard to implement it (the working set policy)? (choose the right answer(s), multiple answers are possible)

- Because it's difficult to guess a good delta time NO. The value of delta only affects performance, not feasibility of the policy
- Because the resident set should be updated even in case of no PF YES. Whenever the page accessed at time t-Delta was not accessed later.
- Because the technique needs to keep track of last access time for each page YES, because only pages with last access > t-Delta should be kept in the resident set
- Because the technique require keeping a list of access times for each page NO, the full list is redundant, just the last access time is needed.

Briefly explain/motivate the answers

C) Consider an LRU strategy based on a stack implementation (with 5 frames). Given the following reference string: 4, 6, 4, 1, 7, 8, 2, 2, 3 (T1), 4, 2 (T2). Show the stack (the list) at time T1 (after accessing page 3) and T2 (after accessing page 2). (represent the stack by comma-separated numbers, with stack top first (e.g. 1,2,3,4,5 means that 1 is top of the stack)

T1) 3,2,8,7,1
T2) 2,4,3,8,7

D) Consider two reference string w1 and w2 with equal length. We know that p2 = 2*p1 (probabilities of strings). Two page replacement algorithms A1 and A2 are evaluated using the same value of (fixed) m. A1 produces the same number of page faults (F1 = F2 = 5000) with w1 and w2. In the case of A2, F1 and F2 change, but their sum is unchanged. Since the PF frequency with A1 is 20% worse than A2, A2 is finally selected. Compute the values of F1 and F2 measured with algorithm A2.

(m is omitted in formulas as it is constant/unchanged in all experiments)

F(Ai) = p1*F1(Ai,w1)+p2*F2(Ai,w2)
F1 and F2 are given for algorithm 1 (A1). Let's use unknowns x and y for F1 and F2 in A2
We know that
F(A1)/F(A2) = 1.2
x + y = 10000

F(A1)/F(A2) = p1*(5000+2*5000) / p1*(x+2y) = 1.2
1.2*(x+2y) = 15000
x+2y = 12500

using second equation (x+y = 10000) leads to x=7500, y=2500

2 Consider a file system based on indexed allocation in which, to manage large files, index blocks are organized in a linked list. The pointers/indexes are 32 bits in size and the disk blocks are 4KB in size. The file system resides on a 1TB disk partition, which includes both data blocks and index blocks.

A) Given a binary file of size 23033 KB, calculate exactly how many index blocks and data blocks the file occupies. Also calculate the internal fragmentation, for both data and index blocks.

#DB = ceil(23033KB/4KB) = 5759
IF = 4KB*(1-23033KB%4KB) = 4KB*0.75 = 3KB

*for index blocks we need to consider that one index block can hold 1023 pointers to data bloicks, plus a pointer for the list of index blocks)
#IB = ceil(5759/1023) (double ind.) = 6
IF (for index blocks) = 6*1023 – 5759 = 379 unused indexes = 379*4B = 1516B

B) A text file B, of size 15300 B, contains a sequence of variable length lines, each terminated by '\n'. We know that the average line length is 50 characters ('\n' excluded) and the maximum line length is 100. Compute the number of lines in the file (in case the number ranges between a minimum and a maximum value, compute the range).

x: number of lines
The average line length (\n included) is computed as size/x

So x = 15300/51 = 300

The result is exact (no min/max needed)

C) Consider the file B (of the previous question).
Does the variable length record format impact on allocation? No. La allocation strategy depends on the file system, not the type of file.
Do all allocated blocks store the same amount of file lines, given by the size of a block, divided by the maximum line length? No. File allocation is handled in the file allocation module, whereas lines in the text file are a problem at application level.

(Explain/motivate both answers)

3 Consider memory management with paging, and a MMU with a TLB. For each of the following questions, answer YES or NO, and give a motivation/explanation.

A) Does the TLB reach decrease when the page size increases?
NO. it increases, as the number of entries in the TLB is a constant

B) Does fragmentation increase, when the page size increases, because larger contiguous partitions are needed?
NO. Internal fragmentation increases with the page size, but the reason proposed is wrong: there is no contiguous allocation.

C) Is prepaging beneficial only if the probability of a prepaged page to be effectively used is > 80%?
NO. It is beneficial with high probability of usage, but no 80% threshold is given.

D) Do all kernel data structures require contiguous allocation?
NO. Just part of the kernel data structures need contiguous allocation (for efficiency): e.g. the page table.

E) Does the slab allocator use only power-of-2 sizes?
NO. The Buddy allocator uses power-of-2 sizes.

F) Does a free list of pages have an average internal fragmentation of half a page?
NO. Free lists do not have a problem of internal fragmentation, as they do not contain data, they just represent available space.

4 An oS161 system is given, running on a sys161 MIPS simulator with 4MB of RAM memory.
For each of the following addresses, say if they can be a logical user address, a logical kernel address, a physical address (explain/motivate answers):

- 0x80803005: it cannot be a user logical address because it is > 2G. it cannot be a physical address because it is > 4M, it could be a logical kernel address, but it is not compatible with the RAM (it should be < 0x80400000)
- 0x312010: it can be both a logical user and a physical address because it is < 2G and <4M, respectively. It cannot be a kernel logical address
- 0x532100: similar to the previous one, but it cannot be a physical address as it is > 4M

Given a logical user address 0x4010, convert it to the related physical address. It is known that as->as_pbase1, as->as_pbase2, as->as_vbase1, as->as_vbase2, as->as_npages1, as->as_npages2 have the following values: 0x100000, 0x200000, 0x3000, 0x6000, 2, 4.
OS161 user addresses can be split with 12 bits (3 hex digits) for the page offset, so the address is in page 4, offset/displacement 0x10. As segment 1 spans within pages 3 and 4, the address is in segment 1. The physical address is 0x100000 + (0x4010 – 0x3000) = 0x101010.

Physical memory in dumbvm is allocated by multiple of a page, despite being a contiguous allocation scheme, because

- allocating by multiple of a page reduces internal fragmentation
  NO. It could increase it, as standard contiguous allocation (without page alignment/padding, has no internal fragmentation.

- the MMU in MIPS has a TLB, so logical-to-physical translation needs pages
  YES. As the MMU uses the TLB, translation needs a correspondence page-frame.

- dumbvm implements a page table
  NO. There is no page table in dombvm

- kmalloc can just allocate by multiple of pages.
  NO. Any size can be asked to kmalloc.

5 Consider the implementation of locks and condition variables in OS161. For each of the following sentences, say YES/NO and provide a motivation:

Function cv_wait receives a lock as a parameter because

- it is needed as parameter in the inner call to wchan_sleep NO. Function wchan_sleep needs a slinlock.
- the calling thread has to be the owner of the lock YES. This is one requirement of cv_wait
- the lock has to be released and acquired again by cv_wait YES: this is exactly what cv_wait should do with the lock
- the lock has to be released and acquired again by wchan_sleep NO: wchan_sleep works on a spinlock.

The lock can be implemented

- by a binary semaphore, without any other element/requirement NO. Because the semaphore does not handle ownership.
- by a binary semaphore, plus an additional element/requirement YES. The additional requirement is on ownership.
- by a condition variable NO. The condition variable NEEDS an existing lock and it is not thought for mutual exclusion
- by a wait channel YES, combined with a spinlock and with a proper management of ownership.

The implementation of function lock_acquire can include a call to

- functions spinlock_data_get and spinlock_data_testandset NO. The two functions are at a much lower layer: they are used to implement spinlocks.
- function P on a semaphore YES, if the lock is implemented by a semaphore.
- function cv_wait NO. The condition variable NEEDS an existing lock and it is not thought for mutual exclusion
- function wchan_sleep YES, in the implementation based on wait channel and spinlock