Tutorial Letter 202/3/2021

Solutions to assignment #2

Operating Systems and Architecture

Semester 1 and 2

School of Computing

Computer Science Department

IMPORTANT INFORMATION
Please register on myUnisa, activate your myLife e-mail account and make sure that you have regular access to the myUnisa module website, COS3721-2021-S1/S2, as well as your group website

BARCODE

Due date	30 June 2021
Submission procedure	**Written/Typed. Submission online via myUnisa in a PDF format;**
Chapters in SGG	Chapters 1 – 7 and 18 (section 18.1 and 18.2)
Marking Scheme	
Question 1 [10 marks]	1.1(4) and 1.2(6)
Question 2 [4 marks]	2.1(2), [2.2 or 2.3](2)
Question 3 [8 marks]	3.1(2), 3.2(3), [3.3 or 3.4](3)
Question 4 [6 marks]	4.1(2) and 4.2(4)
Question 5 [4 marks]	5.1(2) and 5.2.a(2)
Question 6 [6 marks]	6.1(3) and 6.2(3)
Question 7 [7 marks]	7.1(2), 7.2(3) and 7.3(2)
Question 8 [5 marks]	8.1(2) and 8.2(3)

Question 1 – Deadlocks [10 marks]

1.1 Assume that a multithreaded application uses only reader–writer locks for synchronization. Applying the four necessary conditions for deadlock, is deadlock still possible if multiple reader–writer locks are used? (SGG, Exo 8.13, P. 373)

Answer:

Marking: give 1 mark for each correct option

Yes. (1) Mutual exclusion is maintained, as locks cannot be shared if there is a writer. (2) Hold and wait is possible, as a thread can hold one reader—writer lock while waiting to acquire another. (3) You cannot take a lock away, so no preemption is upheld. (4) A circular wait among all threads is possible.

1.2 Consider the following snapshot of a system:

	Allocation	Max
	A B C D	A B C D
To	1 2 0 2	4 3 1 6
T1	0 1 1 2	2 4 2 4
T2	1 2 4 0	3 6 5 1
T3	1 2 0 1	2 6 2 3
T4	1 0 0 1	3 1 1 2

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe. (SGG, Exo 8.18, P. 373)

a. Available = (2, 2, 2, 3)
b. Available = (4, 4, 1, 1)
c. Available = (3, 0, 1, 4)
d. Available = (1, 5, 2, 2)

Answer:

Marking: give 4 marks for each correct answer without full development; 6 marks for the correct answer with full development. Mark only one correct option: a., b., c. or d. Please follow the student's approach to ensure the correctness of the answer. The students are requested to give only one correct safe sequence.

We start by calculating the needs for each process:

	Allocation	Max	Needs	Available
	A B C D	A B C D	A B C D	A B C D
To	1 2 0 2	4 3 1 6	3 1 1 4	
T1	0 1 1 2	2 4 2 4	2 3 1 2	
T2	1 2 4 0	3 6 5 1	2 4 1 1	
T3	1 2 0 1	2 6 2 3	1 4 2 2	
T4	1 0 0 1	3 1 1 2	2 1 1 1	

a. Yes. Possible safe sequences are <T4, T0, …> and <T4, T1, …> the dots indicating where any ordering of the remaining threads is correct.

Full working:

Available = (2, 2, 2, 3)
- The only thread that can execute to the completion is T4. Assuming T4 has completed, the new resources available are: **Available = (2, 2, 2, 3) + (2, 1, 1, 1,) = (4, 3, 3, 4)**

- The next threads that can be completed are: T0 and T1. So, <T4, T0, ..> and <T4, T1, …> are two acceptable sequences. Assuming T0 or T1 has completed then, the resources available are:
Available = (4, 3, 3, 4) + (1, 2, 0, 2) = (5, 5, 3, 6) {for T0} <T4, T0, …> any other thread can complete
Available = (4, 3, 3, 4) + (0, 1, 1, 2) = (4, 4, 4, 6) {for T1} <T4, T1, …> any other thread can complete
All possible safe sequences: <T4, T0, …> and <T4, T1, …>

b. Yes. One possible ordering are:
{<T2 T4 T1 T0 T3> , <T2 T4 T1 T3 T0>, <T2, T4, T3, T1, T0>, {<T4 T1 T0 T2 T3> <T4 T1 T0 T3 T2> <T4 T1 T2 T0 T3> <T4 T1 T2 T3 T0> <T4 T1 T3 T0 T2> < T4 T1 T3 T2 T0>, <T4, T2, T1, T0, T3>, <T4, T2 T1 T3 T0>, <T4, T2, T3, T1, T0>

Full working:

Available = (4, 4, 1, 1)

- The threads that can execute to the completion are: T2 and T4. Assuming T2 or T4 complete then, the resources available become:
T2 completes: Available = (4, 4, 1, 1) + (1, 2, 4, 0) = (5, 6, 5, 1) only T4 can complete <T2, T4, …>
T4 completes: Available = (4, 4, 1, 1) + (1, 0, 0, 1) = (5, 4, 1, 2) only T1 and T2 can complete

1) After the execution of T2

- Assuming T4 is executed after T2 then, the resources available become:
Available = (5, 6, 5, 1) + (1, 0, 0, 1) = (6, 6, 5, 2) T1 and T3 can complete but, not T0. So,

After the execution of T2 followed by T4

-Assuming T1 is executed then, Available = (6, 6, 5, 2) + (0, 1, 1, 2) = (6, 7, 6, 4) To and T3 can complete
The safe sequences: <T2, T4, T1, ..> {<T2 T4 T1 T0 T3> and <T2 T4 T1 T3 T0>}
-Assuming T3 is executed after T4 then,
Available = (6, 6, 5, 2) + (1, 2, 0, 1) = (7, 8, 5, 3) , T1 can complete but not T0
-Assuming T1 is completed after T2, T4 and T3 then, Available = (7, 8, 5, 3) +(0, 1, 1, 2) = (7, 9, 6, 5) T0 can complete and so, <T2, T4, T3, T1, T0> is safe.

2) After the execution of T4

- Assuming T1 is completed after T4 then,
Available = (5, 4, 1, 2) + (0, 1, 1, 2) = (5, 5, 2, 4) T0, T2 and T3 can complète.
<T4, T1, …> {<T4 T1 T0 T2 T3> <T4 T1 T0 T3 T2> <T4 T1 T2 T0 T3> <T4 T1 T2 T3 T0> <T4 T1 T3 T0 T2> < T4 T1 T3 T2 T0>}
- Assuming T2 is completed after T4 then,
  Available = (5, 4, 1, 2) + (1, 2, 4, 0) = (6, 6, 5, 2) T1 and T3 can complete but, not T0
- Assuming T1 is completed after T2
  Available = (6, 6, 5, 2) + (0, 1, 1, 2) = (6, 7, 6, 4), T0 and T3 can complete so,
  <T4, T2, T1, ..> is safe {<T4, T2, T1, T0, T3>, <T4, T2 T1 T3 T0>}

- Assuming T3 is completed after T2 then

Available = (6, 6, 5, 2) + (1, 2, 0, 1) = (7, 8, 5, 3) T1 can complete but not T0
- Assuming T1 is completed then, Available = (7, 8, 5, 3) + (0, 1, 1, 2) = (7, 9, 6, 5) T0 can complete.

<T4, T2, T3, T1, T0>

c. No.

Full working:

Available = (3, 0, 1, 4)

- T0 cannot complete as it needs 1 copy of the second resource which is not available
- T1 cannot complete as it needs 3 copies of the second resource which is not available
- T2 cannot complete as it needs 4 copies of B which is not available
- T3 cannot complete as it needs 4 copies of B
- T4 cannot complete as it needs 1 copy of B which is not available

d. Yes. Possible orderings: <T3, T2, T1, T4, T0>, <T3, T2, T4, T1, T0>, <T3, T4, T0, T1, T2>, <T3, T4, T0, T2, T1>, <T3, T4, T1, T0, T2>, <T3, T4, T1, T2, T0>, <T3, T4, T2, T0, T1>, <T3, T4, T2, T1, T0>.

Full working:

Available = (1, 5, 2, 2) on T3 can complete
- Assuming T3 has completed its execution then,

Available = (1, 5, 2, 2) + (1, 2, 0, 1) = (2, 7, 2, 3) T1, T2 and T4 can execute to the completion.
1) We now consider the case <T3, T1, … >. That is, when T1 completes after T3

Available = (1, 5, 2, 2) + (0, 1, 1, 2) = (1, 6, 3, 4) not thread can complete

2) Consider the case <T3, T2> that is when T2 completes its execution after T3
Available = (1, 5, 2, 2) + (1, 2, 4, 0) = (2, 7, 6, 2), T1 and T4 can complete
- Assuming T1 has completed <T3, T2, T1>,
Available = (2, 7, 6, 2) + (0, 1, 1, 2) = (2, 8, 7, 4) T4 can complete
  - ➢ Assuming T4 has completed <T3, T2, T1, T4> then,
Available = (2, 8, 7, 4) + (1, 0, 0, 1) = (3, 8, 7, 5) T0 can complete <T3, T2, T1, T4, T0>
- Assuming T4 has completed <T3, T2, T4> then
Available = (2, 7, 6, 2) + (1, 0, 0, 1) = (3, 7, 6, 3) only T1 can complete
  - ➢ Assuming T1 has completed <T3, T2, T4, T1> then,
Available = (3, 7, 6, 3) + (0, 1, 1, 2) = (3, 8, 7, 5) T0 can complete <T3, T2, T4, T1, T0>

3) Consider the case <T3, T4>
Available = (2, 7, 2, 3) + (1, 0, 0, 1) = (3, 7, 2, 4) so, T0, T1 and T2 can complete.
<T3, T4, any permutation of T0, T1 and T2 here>

Question 2 – Main memory [4 marks]

2.1 Most systems allow a program to allocate more memory to its address space during execution. Allocation of data in the heap segments of programs is an example of such allocated memory. What is required to support dynamic memory allocation in the following schemes? (SGG, Exo 9.13, P. 419)

a. Contiguous memory allocation

b. Paging

Answer:

Marking: give 1 mark for each correct option

a. Contiguous memory allocation: might require relocation of the entire program, since there is not enough space for the program to grow its allocated memory space.

b. Paging: incremental allocation of new pages is possible in this scheme without relocation of the program's address space.

2.2 Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers)? (SGG, Exo 9.16, P. 419)

a. 21205

b. 164250

c. 121357

d. 16479315

e. 27253187

Answer:

Marking: give 1 mark for 2 correct options. Mark only the 4 correct answers. No need to mark 2.3 if 2 marks are allocated to 2.2. If a student obtains less than 2 marks on question 2.2, 2.3 must be marked and the highest mark between 2.2 and 2.3 should be considered.

a. page = 20; offset = 725

b. page = 160; offset = 410

c. page = 118; offset = 525

d. page = 16093; offset = 83

e. page = 26614; offset = 451

2.3 Consider a computer system with a 32-bit logical address and 8-KB page size. The system supports up to 1 GB of physical memory. How many entries are there in each of the following? (SGG, Exo 9.18, PP. 419-420)

a. A conventional, single-level page table

b. An inverted page table

Answer:

Marking: give 1 mark for each correct option.

a. 219 = 524, 288 entries.
b. 217 = 131, 072 entries.

Question 3- Virtual Memory [8 marks]

3.1 The following is a page table for a system with 12-bit virtual and physical addresses and 256-byte pages. Free page frames are to be allocated in the order 9, F, D. A dash for a page frame indicates that the page is not in memory.

Page	Page Frame
0	0 x 4
1	0 x B
2	0 x A
3	-
4	-
5	0 x 2
6	-
7	0 x 0
8	0 x C
9	0 x 1

Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. In the case of a page fault, you must use one of the free frames to update the page table and resolve the logical address to its corresponding physical address. (SGG, Exo 10.17, P. 479)

- 0x2A1

- 0x4E6
- 0x94A
- 0x316

Answer:

Marking: give ½ mark for each correct answer.

- 0x2A1 → 0xAA1
- 0x4E6 → 0x9E6
- 0x94A → 0x14A
- 0x316 → 0xF16

3.2 Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings:

- 2, 6, 9, 2, 4, 2, 1, 7, 3, 0, 5, 2, 1, 2, 9, 5, 7, 3, 8, 5
- 0, 6, 3, 0, 2, 6, 3, 5, 2, 4, 1, 3, 0, 6, 1, 4, 2, 3, 5, 7
- 3, 1, 4, 2, 5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1
- 4, 2, 1, 7, 9, 8, 3, 5, 2, 6, 8, 1, 0, 7, 2, 4, 1, 3, 5, 8
- 0, 1, 2, 3, 4, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 4, 3, 2, 1, 0

Indicate the number of page faults for each algorithm assuming demand paging with three frames. (SGG, Exo 10.19, PP. 480-481)

Answer:

Marking: give 3 marks for one correct option.

- 2, 6, 9, 2, 4, 2, 1, 7, 3, 0, 5, 2, 1, 2, 9, 5, 7, 3, 8, 5
  FIFO = 18, LRU = 17, OPT = 13
- 0, 6, 3, 0, 2, 6, 3, 5, 2, 4, 1, 3, 0, 6, 1, 4, 2, 3, 5, 7
  FIFO = 16, LRU = 19, OPT = 13
- 3, 1, 4, 2, 5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1
  FIFO = 15, LRU = 16, OPT = 11
- 4, 2, 1, 7, 9, 8, 3, 5, 2, 6, 8, 1, 0, 7, 2, 4, 1, 3, 5, 8
  FIFO = 20, LRU = 20, OPT = 16
- 0, 1, 2, 3, 4, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 4, 3, 2, 1, 0
  FIFO = 12, LRU = 11, OPT = 11

3.3 The KHIE (pronounced “k-hi”) operating system uses a FIFO replacement algorithm for resident pages and a free-frame pool of recently used pages. Assume that the free-frame pool is managed using the LRU replacement policy. Answer the following questions:

a. If a page fault occurs and the page does not exist in the free-frame pool, how is free space generated for the newly requested page?

b. If a page fault occurs and the page exists in the free-frame pool, how are the resident page set and the free-frame pool managed to make space for the requested page?

c. To what does the system degenerate if the number of resident pages is set to one?

d. To what does the system degenerate if the number of pages in the free-frame pool is zero? (SGG, Exo 10.21, P. 481)

Answer:

Marking: give 1 mark for each correct option. Mark only the first 3 correct answers. If the total marks for question 3.3 is less than 3, mark 3.4 and consider the highest marks of the two questions 3.3 and 3.4.

a. When a page fault occurs and the page does not exist in the free frame pool, then one of the pages in the free-frame pool is evicted to disk, creating space for one of the resident pages to be moved to the free-frame pool. The accessed page is then moved to the resident set.

b. When a page fault occurs and the page exists in the free-frame pool, then it is moved into the set of resident pages, while one of the resident pages is moved to the free-frame pool.

c. When the number of resident pages is set to one, then the system degenerates to the page-replacement algorithm used in the free frame pool, which is typically LRU.

d. When the number of pages in the free-frame pool is zero, then the system degenerates to a FIFO page-replacement algorithm.

3.4 Suppose that your replacement policy (in a paged system) is to examine each page regularly and to discard that page if it has not been used since the last examination. What would you gain and what would you lose by using this policy rather than LRU or second-chance replacement? (SGG, Exo 10.23, P. 481)

Answer:

Marking: give 3 marks a correct answer.

Such an algorithm could be implemented through the use of a reference bit. After every examination, the bit is set to zero; it is set back to one if the page is referenced. The algorithm then selects an arbitrary page for replacement from the set of unused pages. The advantage of this algorithm is its simplicity—nothing other than a reference bit need be maintained. The disadvantage is that it ignores locality by using only a short time frame for determining whether to evict a page. For example, a page may be part of the working set of a process but may be evicted because it has not been referenced since the last examination (that is, not all pages in the working set may be referenced between examinations).

Question 4- Mass-Storage Structure [6 marks]

4.1 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is:

2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a. FCFS

b. SCAN

c. C-SCAN (SGG, Exo 11.12, P. 526)

Answer:

Marking: give 2 marks for only 1 correct option.

a. The FCFS schedule is 2,150; 2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681 The total seek distance is 13,011.

b. The SCAN schedule is 2,150; 2,296; 2,800; 3,681; 4,965; 2,069; 1,618; 1,523; 1,212; 544, 356. The total seek distance is 7,492.

c. The C-SCAN schedule is 2,150; 2,296; 2,800; 3,681; 4,965; 356; 544; 1,212; 1,523; 1,618; 2,069. The total seek distance is 9,917.

4.2 Consider a RAID level 5 organization comprising five disks, with the parity for sets of four blocks on four disks stored on the fifth disk. How many blocks are accessed in order to perform the following?

a. A write of one block of data

b. A write of seven continuous blocks of data (SGG, Exo 11.15, P. 526)

Answer:

Marking: give 2 marks for each correct option.

a. A write of one block of data requires the following: read of the parity block, read of the old data stored in the target block, computation of the new parity based on the difference between the new and old contents of the target block, and write of the parity block and the target block.

b. Assume that the seven contiguous blocks begin at a four-block boundary. A write of seven contiguous blocks of data could be performed by writing the seven contiguous blocks, writing the parity block of the first four blocks, reading the eighth block, computing the parity for the next set of four blocks, and writing the corresponding parity block onto disk.

Question 5 – I/O Systems [4 marks]

5.1 In most multi-programmed systems, user programs access memory through virtual addresses, while the operating system uses raw physical addresses to access memory. What are the implications of this design for the initiation of I/O operations by the user program and their execution by the operating system?

(SGG, Exo 12.9, P. 567)

Answer:

Marking: 2 marks for the correct answer.

The user program typically specifies a buffer for data to be transmitted to or from a device. This buffer exists in user space and is specified by a virtual address. The kernel needs to issue the I/O operation and needs to copy data between the user buffer and its own kernel buffer before or after the I/O operation. In order to access the user buffer, the kernel needs to translate the virtual address provided by the user program to the corresponding physical address within the context of the user program's virtual address space. This translation is typically performed in software and therefore incurs overhead. Also, if the user buffer is not currently present in physical memory, the corresponding pages need to be obtained from the swap space. This operation might require careful handling and might delay the data copy operation.

5.2 Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such functionality?

(SGG, Exo 12.11, P. 567)

Answer:

Marking: give 2 marks for the correct answer:

Direct virtual memory access allows a device to perform a transfer from two memory-mapped devices without the intervention of the CPU or the use of main memory as a staging ground; the device simply issues memory operations to the memory-mapped addresses of a target device, and the ensuing virtual address translation guarantees that the data are transferred to the appropriate device. This functionality, however, comes at the cost of having to support virtual address translation on addresses accessed by a DMA controller and requires the addition of an address-translation unit to the DMA controller. The address translation results in both hardware and software costs and might also result in coherence problems between the data structures maintained by the CPU for address translation and corresponding structures used by the DMA controller. These coherence issues would also need to be dealt with and would result in a further increase in system complexity.

Question 6 – File-System Implementation [6 marks]

6.1 Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such systems is the degree of variability in the size of the extents. What are the advantages and disadvantages of the following schemes?

a. All extents are of the same size, and the size is predetermined.

b. Extents can be of any size and are allocated dynamically.

c. Extents can be of a few fixed sizes, and these sizes are predetermined.

(SGG, Exo 14.7, P. 639)

Answer:

10

Marking: give 1 mark for each correct answer.

If all extents are of the same size, and the size is predetermined, then it simplifies the block-allocation scheme. A simple bit map or free list for extents will suffice. If the extents can be of any size and are allocated dynamically, then more complex allocation schemes are required. It might be difficult to find an extent of the appropriate size, and there might be external fragmentation. We could use the Buddy system allocator discussed in the previous chapters to design an appropriate allocator. When the extents can be of a few fixed sizes, and these sizes are predetermined, we must maintain a separate bit map or free list for each possible size. This scheme is of intermediate complexity and of intermediate flexibility compared with the other two schemes.

6.2 Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

a. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

b. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

(SGG, Exo 14.11, P. 640)

Answer:

Marking: give 3 marks for one correct option. Mark only one option.

Let Z be the starting file address (block number).

• **Contiguous**. Divide the logical address by 512, with X and Y the resulting quotient and remainder, respectively.

a. Add X to Z to obtain the physical block number. Y is the displacement into that block.

b. 1

• **Linked**. Divide the logical physical address by 511, with X and Y the resulting quotient and remainder, respectively.

a. Chase down the linked list (getting X + 1 blocks). Y + 1 is the displacement into the last physical block.

b. 4

• **Indexed**. Divide the logical address by 512, with X and Y the resulting quotient and remainder, respectively.

a. Get the index block into memory. Physical block address is contained in the index block at location X. Y is the displacement into the desired physical block.

b. 2

Question 7 – Security [7 marks]

7.1 What is the purpose of using a "salt" along with a user-provided password? Where should the salt be stored, and how should it be used? (SGG, Exo 16.2, P. 713)

Answer:

Marking: give 2 marks for the correct answer.

When a user creates a password, the system generates a random number (which is the salt) and appends it to the user-provided password. It encrypts the resulting string and stores the encrypted result and the salt in the password file. When a password check is to be made, the password presented by the user is first concatenated with the salt and then encrypted before checking for equality with the stored password. Since the salt is different for different users, a password cracker cannot check a single candidate password, encrypt it, and check it against all of the encrypted passwords simultaneously.

7.2 Make a list of six security concerns for a bank's computer system. For each item on your list, state whether this concern relates to physical, human, or operating-system security. (SGG, Exo 16.4, P. 713)

Answer:

Marking: give ½ mark for each answer.

- In a protected location, well guarded: physical, human
- Network tamperproof: physical, human, operating system
- Modem access eliminated or limited: physical, human
- Unauthorized data transfers prevented or logged: human, operating system
- Backup media protected and guarded: physical, human
- Programmers, data entry personnel, trustworthy: human

7.3 Mobile operating systems such as iOS and Android place the user data and the system files into two separate partitions. Aside from security, what is an advantage of that separation? (SGG, Exo 16.7, P. 714)

Answer:

Marking: give 2 marks for the correct answer.

User data can be encrypted easily (encrypt the user file system but not the system file system), user data can be erased without harming system (reset to factory defaults), and the system can be upgraded without harming user data.

Question 8 – Protection [5 marks]

8.1 What hardware features does a computer system need for efficient capability manipulation? Can these features be used for memory protection? (SGG, Exo 17.12, P. 749)

Answer:

Marking: give 2 marks for the correct answer.

A hardware feature is needed allowing a capability object to be identified as either a capability or accessible object. Typically, several bits are necessary to distinguish between different types of capability objects. For example, 4 bits could be used to uniquely identify 24 or 16 different types of capability objects. These could not be used for routine memory protection, as they offer

little for protection apart from a binary value indicating whether they are a capability object or not. Memory protection requires full support from virtual memory features discussed in Chapter 10.

8.2 Describe how the Java protection model would be compromised if a Java program were allowed to directly alter the annotations of its stack frame. (SGG, Exo 17.15, P. 749)

Answer:

Marking: give 3 marks for the correct answer.

When a thread issues an access request in a doPrivileged() block, the stack frame of the calling thread is *annotated* according to the calling thread's protection domain. A thread with an annotated stack frame can make subsequent method calls that require certain privileges. Thus, the annotation serves to mark a calling thread as being privileged. If a Java Program were allowed to directly alter the annotations of a stack frame, it could potentially perform an operation for which it did not have the necessary permissions, thus violating the security model of Java.

1.3 Other assessment

No other assessment, apart from the exam, is planned for this module.

1.4 The examination

As communicated to you by the University, due to the reasons included in that formal communication, the two semesters 1 and 2 will run concurrently this academic year from the 1st of April 2021. The exams will start from September or October 2021 and run until November or December 2021. The examination dates for COS3721 will be communicated to you by the Institution in due time.

For all other information regarding module cos3721, you should still refer to Tutorial letter 101.

Regards

COS3721 Team

@Unisa 2021