Noroff
School of technology
and digital media

Assessment 3 Cover Sheet

Course Code:	**Course Title:**	**Lecturer:**
UC2OPS2021_10	Operating Systems	Fabricio Bortoluzzi
Assignment No:	**Total number of pages in this document:**	**Maximum Word Count:**
3 of 3	7	N/A
Assignment Title:	Assessment 3: Online Test	
Date Set:	**Submission Date:**	**Feedback Date:**
27/09/2021	Online Test 01.10.2021 08h00 to 17h00	3 working weeks after submission

Assessment Details

This cover sheet outlines the final Cyber Security and Digital Forensics online test, which must be taken on Friday, October 1, 2021. The test will be available from 08h00 to 17h00.

Assessment 3 grade composition

- This test accounts for 80% (eighty percent) of the final grade.
- Part A accounts for 40% of the test weight, therefore it is worth 32% of the final grade.
- Part B accounts for 60% of the test weight, therefore it is worth 48% of the final grade.

Assessment 3: Exam Layout

Assessment 3 must be taken by clicking on the two links that will be available on Moodle on the date and time of the exam.

The test is comprised of two parts: **Part A** ”MCQs” and **Part B** ”Essays”.

- **Part A**: Set of 30 multiple-choice questions randomly extracted from a database containing MCQ created by the teaching team and exclusively designed for this exam. The questions in this group are balanced to a difficulty level of 4 marks per question, accounting for 120 marks, therefore you will have 120 minutes (two hours) to complete the exam.
- **Part B**: A set of 4 essay type questions randomly extracted from a database containing essay questions created by the teaching team and designed exclusively for this exam. The questions in this group are balanced to a difficulty level of 30 marks per question, accounting for additional 120 marks, therefore you will have additional 120 minutes (two hours) to complete the exam.

We have designed the test on a basis of 1-minute per mark, meaning you are expected to spend no more than 4 minutes to answer each MCQ and up to 30 minutes on each essay type question. This includes extra-time you could need to revisit any MCQ you find relevant and have a final check on the essays you will have written.

You should start taking Part A by 13h00 to have access to the full two hours for it plus two hours for Part B.

Both Parts will close at 17h00 automatically submitting your answers. You must finish Part A to be able to start Part B.

Ideally, you should start taking Part A as early as possible, for example, at 08:00. Then you should have a break of at least 1 hour. Only then you should take Part B.

Our suggestion of timing for this Exam:

- 08:00 Start taking Part A.
- 09:59 Finish and submit Part A, unlocking Part B.
- 10:00 1-hour break starts.
- 10:59 1-hour break ends.
- 11:00 Start taking Part B.
- 12:59 Finish and submit Part B.

Academic posture and plagiarism checks

On the date of exam, we will be monitoring the platform for potential student misconduct. During the exam you are required to:

- Find answers alone.
- Avoid by all means to paste text from the internet to compose your answers. Pasting will very likely increase similarity score in TurnItIn towards plagiarism detection.

- Refuse to exchange possible answers to similar or identical questions being presented to more than one student during the exam.
- Refuse to provide another student with answer to essays.
- Write original answers to essays. You are **not** being asked to reference and cite original textbook authors. Answers should be your own and reflect your current understanding of topics being asked.

Moodle's live logging and stored logging features will be put to use to their fullest extent to establish a potential correlation of coordinated academic misconduct. We strongly advice against sharing answers to anyone by any means.

Essays will go through TurnItIn plagiarism checks. TurnItIn similarity score is final to the detection of plagiarism, either by comparing student's answer against public internet resources or between another student's submission.

MCQs

Here is an example of an MCQ question that could be presented on the first section of the test:

Question:
Operating systems must perform many tasks: They must allocate CPU time for apps, ensure several processes can be loaded on RAM at the same time as long as they do not overlap and arrange processes in queues when they request access to I/O devices such as network cards and hard drives. Other than managing the hardware beneath, operating systems must provide programmers with a programmable layer, often described as the virtual machine, extended machine or abstract machine.

Which of the following alternative contains a valid example of the virtual/extended/abstract machine perspective?

a) It is the set of programming frameworks, libraries and developer resources, including an API – Application Programming Interface designed to hide low-level details about how the underlying hardware really works

b) It is the set of hardware-provided mechanisms aimed to provide isolation, protection or abstraction, including a memory management unit and the timer that generates interrupts that triggers context-switch actions

c) It is the set of circuit-level mechanisms providing hardware-assisted virtualization which speeds up virtualization minimizing overhead related to instruction translation to acceptable levels

d) It is the set of resources allocated to the creation of I/O queues in which processes are placed when they request any instruction that cannot be executed by the Arithmetic Logic Unit – ALU

As you can see, MCQ's will require a fair understanding of key topics presented in Lectures and explained in the literature.

Alternative D cannot be considered correct because the creation of queues is a topic related to the ”Resource Management” part of an Operating System, not the ”Abstract Machine” view.

Alternative C cannot be considered correct because the design of an abstract machine has absolutely no relation with the circuit-level mechanisms for hardware-assisted virtualization.

Alternative B cannot be considered correct because protection is the key concept of ”Resource Management” part of an Operating System, not the ”Abstract Machine” view.

Alternative A is correct because all statements fit the extended/abstract definition as presented on Lecture 1, and in accordance to slides and text-book.

Another MCQ example

Let's have a look at another possible question:

Question:
The kernel is the most important, relevant and critical part of an operating system. The operating system, on the other hand, is comprised of a kernel combined with system programs and a collection of application programs in most cases. When it comes to the kernel, which statement is true?

a) The kernel is the code that manages the underlying hardware, including CPU scheduling, memory allocation and I/O device queuing. It also serves processes by providing them with collection of system calls.

b) The kernel must provide applications with access to the CPU kernel-mode. When applications run in kernel mode, they can perform logic and arithmetic functions as well as request for I/O devices such as reading from the keyboard or writing to a hard drive.

c) The kernel is the only piece of code that is running at all times. It performs logic and arithmetic operations on behalf of processes, as well as the kernel grants processes with access to any portion of memory only accessible in kernel mode.

d) The kernel is partially written as CPU circuit and partially written as software. An operating system must give equal privileges to code running in kernel mode and code running in user mode, as long as a process is granted with access to the CPU in kernel mode whenever the process needs.

Alternative D is filled with wrong statements, being ”a process is granted with access to the CPU in kernel mode whenever the process needs” the most relevant false statement.

Alternative C is wrong because processes should be the entities running in the CPU most of the time. The kernel never performs ALU operations on behalf of processes, etc.

Alternative B is critically wrong, stating processes should acquire the CPU while it is in kernel-mode.

Alternative A has only correct statements.

Essays

Now let’s see how an essay will look like and how you can prepare yourself towards providing good quality answers:

Question
Critical sections are difficult to protect. Virtually all computer programs can benefit from better response time, higher performance, and higher throughput when they are programmed under a concurrency and parallelism approach such as the multi-threaded programming techniques discussed in lectures. The challenge associated with multi-threading is in the care that must be taken at crucial moments in the execution of the program when a shared data structure, such as a variable, may get corrupted if ideal safety conditions are not met. Demonstrate that you understand how risky concurrent programming is, and how can programmers mitigate, considering the following aspects:

- Adequate definition of the critical section according to concurrent programming principles
- Definition of the technical reasons that cause damage to shared data structures in the critical section
- The impact of the preemptive execution principle on the creation of race conditions
- Existing practices for perfect protection of the critical section

There is no universal template that could be considered the right answer to short-essays. However, when we assess a question like this we are looking for specific patterns that reveals how mature an answer is.

- Does the provided answer cover the reasons why critical sections are difficult to protect?
- Do they explain the problem relates to simultaneous writes over a same shared data structure, such as a variable?
- Do they cover a brief explanation of race conditions? Is there any reference to one line of high-level code becoming more than one machine-level instruction?
- Do they mention the problem relates to context switches happening at unpredictable times?
- Is the word ”preemptive” present in the answer, denoting they understand preemption is desired, but it is one key factor contributing to variable corruption?
- Do they argue saying the critical section can be effectively protected by using mutex, spin-locks and eventually semaphores? Is an explanation given to when should we use semaphores to keep processes sleeping and avoid busy wait? Is busy wait mentioned across the answer?

Express thoughts using a language other than your native tongue can be difficult, yet necessary. Good quality writing is expected, as much as possible, aiming at the following:

- You must provide comprehensive answers to the question presented. Your arguments must be well-founded, objective, clear, and the thoughts you express must remain tied to the core of what is being asked.
- Your answer should preferably use auxiliary verbs that better reflect your understanding of the matter: Avoid conjugations that denote uncertainty as could, would, should. Rather prefer the simple present: is, does, must, can, etc.
- Answering with examples is discouraged. Examples do not guarantee that you understand the subject. Examples are acceptable only if they stand as a *bridge* to clear and consistent arguments effectively addressing the proposed question.
- Be original. Answers will be checked for plagiarism against your colleagues' submissions and against publicly information made available on the Internet.

As a general guidance of expectations:

- Short essay awarded with more than 25 out of 30 marks will show an in-depth understanding of key topics asked, with a clear ability to evidence learning and development. The answer clearly demonstrates critical engagement with materials and concepts related to the course and stays upon topic. No grammatical or typographical errors will be present and formatting is consistent, clear and appropriate.
- Short essay awarded 20 to 24 out of 30 marks will follow the above guidance, containing few (if any) errors or inaccuracies, but does evidence an in-depth understanding or critical engagement with materials.
- Short essay awarded 15 to 19 out of 30 marks will follow the above guidance, some errors or inaccuracies, but does not evidence an in-depth understanding or critical engagement with materials.
- Short essay awarded 10 to 14 out of 30 marks will mostly follow the above guidance, containing significant errors, omissions or inaccuracies, although the work may be lacking supporting evidence or descriptions, with a focus more upon the descriptive.
- Short essay awarded 1 to 9 out of 30 marks will more or less follow the above guidance, although it will contain errors or inaccuracies or lack detailed discussion, and will be lacking any sort of relevant discussion.
- Short essay awarded 0 out of 30 marks will generally not follow the above guidance, be of a poor standard, will not demonstrate an understanding of the technical concepts, and/or will not provide adequate evidence of engagement.

How to get prepared to the test?

The Online Test is designed assuming you:

- Attended all lecture sessions;
- Attended online support sessions;
- Completed all the tutorials and online quizzes associated to them;
- Developed an understanding of key topics presented on slides;
- Has a fair understanding of key topics highlighted on the annotated version of the Dinosaur's book Table of Contents available as a PDF on Moodle.

Therefore, you should spend time studying by going through the following resources:

1. Check out the annotated version of the Dinosaur's book Table of Contents available as a PDF on Moodle.
2. Give priority to study and understand the basic elements highlighted in red.
3. Watch recorded lecture sessions, perhaps at 2x, to ensure you've got a brief understanding of all contents discussed there.

4. Browse over the handouts to ensure you have a grasp at all contents presented there.
5. Read the text-book or equivalent online resources going through the explanation of key topics such as ”processor-modes”, ”interrupts”, ”MMU”, ”Round Robin Algorithm” etc.

List of most relevant topics

This is a **non-exhaustive list of topics** you must be ready to explain.

- Operating system as an ”abstract machine” in opposition to ”resource manager”;
- The role of kernel as the most critical component of an operating system;
- How interrupts work;
- The role of system timer, (or hardware clock) as a generator of timed interrupts;
- What RAM is and how the OS uses RAM;
- Process context-switch;
- Dual mode operation of CPUs;
- How OS reacts to errors, such as invalid access to RAM and division by zero;
- System calls as programmatic access to OS facilities and services;
- Minimal understanding of computer programming languages like C and Java;
- The difference between executing ALU operations versus I/O requests;
- When and why processes issue system calls;
- What portability means for operating systems;
- Design versus Implementation;
- Policy versus Mechanism;
- Key differences between processes and threads;
- Process life-cycle: loading, ready, running, waiting, terminating;
- Process Control Block;
- Memory Management Unit;
- Process execution versus interrupt handling;
- fork(), exec() and wait();
- Interprocess Communication;
- Producer Consumer;
- Critical section: Risks and protection. Including race conditions, spin-locks, semaphores;
- General understanding of dead locks;
- Practical operating system aspects such as pipes and sockets;
- Single versus multi-threaded execution;
- OpenMP, Barriers and SIMD;
- CPU Scheduling algorithms;
- Memory allocation. Fragmentation, internal and external.
- Virtualization, hypervisors, hardware-assistance, overall benefits.
- Mass storage structure. Volatile and non-volatile memory, hard-drive queue scheduling, first-come, first served, SCAN, elevator algorithms.
- Root file-system. Partitioning. File-System Mounting. The master block record, MBR.
- Storage technologies, storage arrays, storage-area networks, network-attached devices.

- RAID. RAID levels. Mirroring. Stripping. Parity. Spare disks.
- ZFS file-system as a special use-case. ZFS advanced features including snapshots, deduplication and integrity checks.
- I/O management. I/O variety. Bus. Overview of storage bus.
- I/O techniques: Programmed I/O (Polling), Interrupt-based, Direct-Memory Access.
- Block-devices. Character-oriented devices. Clock and timer. Overall I/O protection and performance.
- File attributes. Metadata. File and file-system permission.
- File operations: open, seek, read, write, close. Locking.
- File handling: Sequential access. Direct access.
- File-system directories. File indexing. Tree-structured directories. Mount points. Unix access-control lists. Unix permission.
- File-system implementation: Contiguous allocation. Linked lists. File Allocation Table. Extent-based file-systems.
- Security violations in operating systems. Security measures for operating systems. Program threats. Buffer overflows. Code injection. Boot-sector computer virus. User authentication. Passwords. Security defenses. SELinux. AppArmor.
- Principle of least-privilege on operating systems. Operating system permissions. Protection rings. Active Directory and UNIX Kerberos domain.
- Distributed systems: Definition. Resource sharing. Computation speedup. Load balancing. Reliability. Network support for loosely coupled systems. Distributed services: DNS, Active Directory. Distributed file-systems. Design goals: robustness, transparency, scalability and fault-tolerance.

”*With hard work in your strides, good luck will always be by your side.*”
Good luck!