THE OPEN UNIVERSITY OF KENYA

DESIGN PLAN

Programme title	Bachelor of Science in Cybersecurity and Digital Forensics
Course title	Operating Systems
Learning Module number	04
Learning module title	Memory Management
Module Developer	Elisha Abade
Module duration in hours	
Instructional Hour Equivalent (Divide duration by 2)	
Reviewed by	
Vision	The innovative university for inclusive prosperity
Audience description	Learners of Cyber Security in first semester of second year
Instructions to learners ⓘ	In this course we shall be learning about memory management. We'll begin by watching videos on these important aspects of Operating systems. You are encouraged to ensure that you have access to a reliable Internet and that your devices (computer, tablet or phone) have properly working multimedia systems. This module also presents a number of interactive and non-interactive activities. You will be required to complete all the activities before you can proceed to the next module.
Learning module description	This module aims to enable learners to understand the concept of Memory Management in Operating Systems. The key aspects that you will learn include memory structure, memory addressing as well as design and implementation issues in computer memory management.
Module objectives:	This module aims at facilitating learners to acquire knowledge about: 1. Main memory of computer systems 2. Memory management techniques in operating systems 3. Virtual memory paging 4. Segmentation and protection.
Module learning outcomes:	By the end of the module, you should be able to: 1. Define the main memory of computer systems

	2. Describe Memory management techniques in operating systems 3. Compare and contrast Virtual memory and paging 4. Illustrate memory segmentation and protection mechanisms
Planned Learning Resources	
ACTIVITY 1: INTRODUCTION VIDEO 1: Pre-recorded lecture on topic emphasizing LEARNING OUTCOME 1: Factual knowledge.	**Video 1: Memory Management (8 minutes)** Welcome to the fourth module of this course in Operating Systems. In this session, we shall be learning about memory management techniques used in Operating Systems. This will help us to understand the various types of memory and how the Operating System makes them available to users and processes. We shall be looking at memory abstraction concepts of address spaces and swapping then delve into the virtual memory and page replacement algorithms that can be used by Operating Systems during memory management. **A memory address space** Physical memory cannot be exposed to processes because if this is allowed, a process can accidentally or intentionally throw out the Operating System. It could also be very difficult to have multiple processes running concurrently. Therefore an abstraction of memory through the notion of address spaces was introduced. This helps to solve the two problems mentioned earlier. An address space is the set of addresses that a process can use to address memory. Therefore each process has its own address space, independent of those belonging to other processes. **Swapping** The total amount of Random Access Memory (RAM) that is required by all processes is much more than can fit in memory. In order to deal with this, two approaches have been developed over the years. These are swapping and virtual memory. Of the two, swapping is the simpler one. It entails bringing in each process in its entirety into memory, running it for a while then putting it back onto disk. In this concept, idle processes are mostly stored in the disk so that they do not take up memory space when they are not running. Swapping also entailed splitting a program into little pieces called **overlays**. ***Insert a graphical illustration of how memory changes with time as processes run and get swapped back to hard disks. hint: We can borrow from figure 3-4 in the core reference text.*** Swapping can create multiple holes (unused blocks) in memory. When this happens, the Operating System needs to move all running processes as downward as possible in memory. This process is called **memory compaction**.

In the next video, we shall be learning about a concept of abstraction of memory so that it appears to be larger than the physical main memory, a technique referred to as virtual memory.

Video 2: Virtual memory and Paging (4 minutes)

Welcome to the second video of this module. In this session, we will be looking at two interesting concepts of memory management, namely: virtual memory and paging.

Each program has its own address space which is further broken down into smaller chunks known as **pages**. Each page comprises a range of contiguous addresses. The corresponding units of pages in the physical memory are referred to as **page frames**.

Memory addresses can be generated using indexing, base registers and other ways. These program generated addresses are referred to as **virtual addresses** and form the **virtual address space**. When virtual addresses are used, the virtual address space goes to a **Memory Management Unit (MMU)** that maps the virtual memory onto the physical memory. The MMU uses a special data structure called a **Page Table** to perform this mapping.

A simple mapping splits the virtual address into a virtual page number (high-order bits) and an offset (low-order bits). The virtual page number is used as an index into the page table to find the entry for that virtual page. From the page table entry, the page frame number (if any) is found. The page frame number is attached to the high-order end of the offset, replacing the virtual page number, to form a physical address that can be sent to the memory.

Video 3: Page Replacement Algorithms (8 minutes)

In this session we shall be looking at how pages get replaced by the Operating Systems when need arises.

There are times when programs look for pages that cannot be found in memory at that particular time. This is referred to as **a page fault**. When a page fault occurs, the operating system has to choose a page to remove from memory so as to create room for the incoming page.

In an event that the page to be removed has been modified while in memory, it must be rewritten to the disk to ensure that the copy in the disk is consistent with what was in memory. However, if the page had not been modified in memory, the page to be read just overwrites it. In order to manage the process of replacing pages in the event of page faults, there are a number of algorithms that have been proposed. These include:

1. **Optimal Page Replacement Algorithm**
   This algorithm works on the premise that at the time of a page fault, some set of pages are in memory and one of them will be referenced on the very next instruction (the page containing that instruction). Other pages may not be referenced until N instructions later. Each page can then be labeled with the number of instructions that will be executed before that page is first referenced. This algorithm provides that the page with the highest label should be removed.

2. **Not Recently Used (NRU) Page Replacement Algorithm**
   This algorithm is based on the fact that the Operating System will collect page usage statistics. At the time a process is started, all its page table entries are marked as not in memory. When any of its pages is referenced, a page fault occurs. The Operating System then uses two bits to track when a page is referenced or modified. Therefore for each page, the Operating system keeps track of when they are used. This results in the following classes:

   Class 0: not referenced, not modified.
   Class 1: not referenced, modified.
   Class 2: referenced, not modified.
   Class 3: referenced, modified.

   The NRU algorithm will remove a page at random from the lowest numbered non-empty class.

3. **First-In, First-Out (FIFO) Page Replacement Algorithm**
   In this approach, the operating system maintains a queue-like structure in which it keeps track of pages currently in memory. Since the structure behaves like a queue, additions happen at the tail. When a page fault occurs, the page at the head of the queue is removed, all pages are then pushed one step towards the head of the data structure and the newly loaded page is added at the tail of the queue.

4. **Second-Chance Page Replacement Algorithm**
   One potential weakness of the FIFO algorithm is that it creates a scenario where a frequently used page is thrown out of memory simply because it is at the head of the queue. A second chance algorithm was proposed with an objective of solving this problem.

   In the second chance algorithm, modification has been done to the simple FIFO such that the reference (R) bit is checked before a page is moved out of memory. If the R bit is reset (is zero), the page is both old and unused therefore it can just be replaced immediately. However if the R bit is set to 1, the bit is cleared then the page is put onto the end of the queue and its load time is updated as though it had just arrived in memory. The search for the next page to be evicted continues by repeating the same process of bit checking.

Insert a graphical illustration of how the second chance algorithm works. We can borrow from figure 3-15 in the core reference text.

5. **Clock Page Replacement Algorithm**

   The second chance algorithm discussed previously is unnecessarily inefficient because pages have to be moved around and shifted in the queue. The clock page algorithm further improves onto the second chance algorithm. It keeps the page frames in a circular list, hence the name, “clock”. It also maintains a special pointer to a page in the circle.

   On a page fault, the page being pointed to by the pointer is inspected. If its R bit is 0, the page is evicted and new page is inserted into the clock in its place. The hand is advanced one position. If R is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found with R = 0.

Insert a graphical illustration of how the clock page replacement algorithm works. We can borrow from figure 3-16 in the core reference text.

6. **Least Recently Used (LRU) Page Replacement Algorithm**

   The LRU algorithm works on the assumption that pages that have been heavily used in the last few instructions will probably be heavily used again soon. Therefore when a page fault occurs, the page that has been unused for the longest time is thrown out.

Insert a graphical illustration of how LRU algorithm works. We can borrow from figure 3-17 in the core reference text.

Video 4: Segmentation (8 minutes)

Welcome to this fourth session of the course in Operating Systems. We shall be looking at an alternative to paging. It is a technique called segmentation.

In this approach, the Operating System offers a view memory as a set of segments of no particular size, with no particular ordering. The length of a segment depends on the complexity of the function. In segmentation, the logical address space becomes a collection of segments that are typically disjointed.

A segment has a name (or a number) and a length. An address then specifies a segment and an offset within the segment. To access memory, a user program specifies "*segment + offset*". This is in contrast to paging where the user is unaware of the memory structure and everything is managed invisibly.

Implementing Segmentation

In this approach, logical addresses are given as (segment, offset) pairs. The loader then maps each segment onto a physical segment number. The mapping is supported by a special data structure called the **Segment Table (ST)**. Each process has its own segement table.

A segment table will typically have the following columns:
Segment, Access, Base, Size and Others.

If there are too many segments then the table is kept in memory, pointed to by **ST Base Register (STBR).** Since the number of segments used by different programs will vary, there is also a special register, the **ST Length Register (STLR)** that will also be maintained by the Operating System. The segment table (ST) is maintained as part of the process context therefore it is changed on each context switch. Since it is accessed on each memory reference, speed is critical.

The actual application of segmentation is as follows:
1. A user program presents an address (s,d).
2. If $s \geq STLR$ *then give up*
3. Obtain table entry at reference s+STBR, a tuple of form (b_s, l_s)
4. If $0 \leq d \ll ls$ *then this is a valid address at location* (b_s, d)*, else fault*

Protection

The greatest advantage of segmentation is to provide protection between components. The protection provided corresponds to the logical view hence provided per component. In this approach, protection bits are associated with each Segmentation Table entry. Each ST entry is checked such that they should not be self-modifying, hence they are protected against writes.

Sharing

Segmentation also facilitates sharing of code or data among processes. Each process has its own STBR/STLR but sharing is enabled when two processes have entries for the same physical locations. In this case, sharing occurs at segment level such that each segment has its own protection bits.

ACTIVITY 2: READING READING MATERIAL 1 (	**1. Memory Management** Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 182 -239 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div **2. Virtual memory and paging** Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 194 -208 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div **3. Segmentation** Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 240 -252 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div
ACTIVITY 3: Comprehension questions:	Questions are based on the lecture and reading material. Some of the questions in this activity have been taken from the key textbook in this course. 1) What is the difference between a physical address and a virtual address? 2) A swapping system eliminates holes by compaction. Assuming a random distribution of many holes and many data segments and a time to read or write a 64-bit memory word of 8nsec, about how long does it take to compact 8 GB? For simplicity, assume that word 0 is part of a hole and that the highest word in memory contains valid data. 3) Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of (a) 12 MB (b) 10 MB (c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

LEARNING OUTCOME 2:
Conceptual knowledge

ACTIVITY 4: Video to be used.

CASE 1:

We consider a program which has the two segments shown below consisting of instructions in segment 0, and read/write data in segment 1. Segment 0 has read/execute protection, and segment 1 has just read/write protection. The memory system is a demand paged virtual memory system with virtual addresses that have a 4-bit page number, and a 10-bit offset. The page tables and protection are as follows (all numbers in the table are in decimal):

Segment 0		Segment 1	
Read/Execute		Read/Write	
Virtual Page#	Page frame #	Virtual Page#	Page frame #
0	2	0	OnDisk
1	OnDisk	1	14
2	11	2	9
3	5	3	6
4	OnDisk	4	OnDisk
5	OnDisk	5	13
6	4	6	8
7	3	7	12

For each of the following cases, either give the real (actual) memory address which results from dynamic address translation or identify the type of fault which occurs (either page or protection fault).

(a) Fetch from segment 1, page 1, offset 3
(b) Store into segment 0, page 0, offset 16
(c) Fetch from segment 1, page 4, offset 28
(d) Jump to location in segment 1, page 3, offset 32

ACTIVITY 5: READING MATERIAL	In this section you have been provided with links to online resources where you are required to get more information with regards to memory management. Read all of them and use the information gathered to write your blog as required at the end of this activity. 1) Memory Management in Operating System https://www.researchgate.net/publication/340816041_Memory_Management_in_Operating_System 2) Memory Management, University of California, Davis: https://web.cs.ucdavis.edu/~pandey/Teaching/ECS150/Lects/06memory.pdf 3) Operating Systems. Memory Management https://www.dc.fi.udc.es/~so-grado/2022-23/Temas/SO-Memoria.pdf 4) Segmentation in Operating Systems https://www.geeksforgeeks.org/segmentation-in-operating-system/ a) Having read the above articles, you are required to write a blog in the LMS with focus on swapping, compaction and thrashing as aspects in main memory management.
ACTIVITY 6: ONLINE DISCUSSION	Activities based on reading Material 5. Your course instructor will create a discussion forum in the LMS to facilitate online group discussions. You are required to read the discussion topic and give comments. You are also encouraged to comment on contributions from at least three members of your group. You can use the LMS platform to send questions to your instructor on the discussion topics that he/she has posted on the LMS. The group discussion will be graded based on a weight that will be indicated on the LMS.
LEARNING OUTCOME 3: PRACTICAL SKILLS VIDEO 3:	In this section you will be watching videos that show how paging and segmentation work in Operating Systems in order to further understand the differences between the two techniques. Watch this 8-minute video, in which we take a look at the mathematical illustration of the Paging technique. In this other 10-minute video, we also find a practical illustration of memory management using segmentation.
ACTIVITY 7: Learner practice sessions	1) In this session, you are required to do two lightning talks and record them in a video format using your smartphone or any other video camera. The first lighting "lightning talk" focuses on "*Segmentation and Paging*". In this "lightning talk",

	record yourself in not more than “30 seconds” while explaining the differences between “***Segmentation and Paging***” and the pros and cons of each. Note: • Upload your video with the captions <*fname_lname_talk4a*>. where “*fname*” is your first name and “*lname*” is your last name (or surname). • The video must not be more than 30 seconds long. 2) The second session of “lightning talk” focuses on “*Translation Lookaside Buffer (TLB) and Page Tables*”. In a video of not more than “60 seconds” long, record yourself while explaining what “*TLB and Page Tables are*” are and their roles in memory management. Note: • Upload your video with the captions <*fname_lname_talk4b*>. where “*fname*” is your first name and “*lname*” is your last name (or surname). • The video must not be more than 60 seconds long.
ASSESSMENT OF PRACTICAL SKILL:	In the above activity, you uploaded your video, <***fname_lname_talk4a***> and <***fname_lname_talk4b***>. Each of them will be assessed out of 15 marks by the instructor by looking at among others: a) Accuracy of the assertions you have made (5 Marks) b) Degree of completeness of your response to the task (3 Marks) c) Adherence to the requirements with regards to topic and length of the video. (2 Marks)
LEARNING OUTCOME 4: KEY/TRANSFERABLE SKILLS	In this section you have further been given information on additional reading material that you are required to read. These materials serve to help you achieve the fourth learning objective in this course. 1) Remzi H., Andrea C. (2014). Operating Systems – Three easy pieces. PP 141 – 152. 2) https://www.javatpoint.com/os-segmentation 3) https://ecomputernotes.com/fundamental/disk-operating-system/memory-segmentation 4) https://www.pvpsiddhartha.ac.in/dep_it/lecture%20notes/os/unit4.pdf
ACTIVITY 8	In this module, you have learnt about several concepts in Memory Management. You are required to write a two page essay on page replacement algorithms clearly elucidating the strengths and weaknesses of each.

	Your instructor will create an activity in the LMS that will allow you to submit this essay for assessment. The essay will be marked out of 15 Marks. Some of the guidelines to success in this activity include: a) Originality (avoid copying from the Internet and other sources) (5 Marks) b) Level of accuracy of the essay content (5 Marks) c) Completeness of content (3 Marks) d) Sticking to length (number of pages) requirements (1 Mark) e) Keeping to the theme (1 Mark)
QUIZZ:	1. What is the name of the facility that allows programmers to address memory from a logical point of view, without regard to the main memory, physically available? a) Visual memory b) Real memory c) Virtual memory d) Secondary memory 2. Which of the following options represent the types of tables maintained by the operating system? a) memory, logical , I/O file b) memory, I/O, file, physical c) memory, I/O, file, process d) memory, logical, I/O, physical 3. Which of the following information is not included in memory table? a) The allocation of main memory to process. b) The allocation of secondary memory to process c) Any information needed to manage virtual memory d) Any information about the existence of file 4. With ………………… A page is written out to secondary memory only when it has been selected for replacement. a) pre-cleaning b) demand cleaning c) required cleaning d) fast cleaning 5) The mechanism that brings a page into memory only when it is needed is called..... a) segmentation b) fragmentation c) demand paging d) page replacement 6) FIFO scheduling is..........

	a) preemptive scheduling b) non preemptive scheduling c) deadline scheduling d) fair share scheduling Answers: 1. c) Virtual memory 2. c) memory, I/O, file, process 3. d) Any information about the existence of a file 4. b) demand cleaning 5. c) demand paging 6) b. non preemptive scheduling
TAKE HOME MESSAGE	Your course instructor will create a feedback section in the LMS to facilitate provision of your take home message. You are required to give a brief description of what you have learnt in this module in not more than half a page (typed) in the feedback section provided.
Reference list	1. Andrew S Tanenbaum. (2016). *Modern Operating Systems Paperback, 5th Edition*. Pearson. 2. Silberschatz A., Galvin P. B. and Gagne G. (2008). *Operating System Concepts*, 8th Edition. Wiley. ISBN: 9780470128725 3. Meyers, M. (2016). *CompTIA A+ Certification Guide*. McGraw-Hill Education