THE OPEN UNIVERSITY OF KENYA

DESIGN PLAN

Programme title	Bachelor of Science in Cybersecurity and Digital Forensics
Course title	Operating Systems
Learning Module number	09
Learning module title	Input/Output
Module Developer	Elisha Abade
Module duration in hours	8
Instructional Hour Equivalent (Divide duration by 2)	4
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 the concepts of Input/Output management in Operating Systems. We'll begin by watching videos on 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.
Learning module description	This module aims to facilitate learners to have an understanding of the principles of hardware and software I/O management, then delve into internal workings of device drivers, programmed I/O, Interrupt driven I/O as well as other complex I/O mechanisms such as Direct Memory Access (DMA).
Module objectives:	This module aims at facilitating learners to acquire knowledge about: 1. Principles of I/O hardware 2. I.O devices and device controllers 3. Principles of I/O software; 4. Modes of I/O operations
Module learning outcomes:	By the end of the module, you should be able to: 1. Describe the principles of I/O hardware and I/O software

	2. Discuss how I/O device controllers interact with the Operating System 3. Interpret the internal working mechanism of device drivers 4. Illustrate modes of I/O operations such as programmed I/O, memory-mapped I/O and Direct Memory Access (DMA)
Planned Learning Resources	
ACTIVITY 1: INTRODUCTION VIDEO 1: Pre-recorded lecture on topic emphasizing LEARNING OUTCOME 1: Factual knowledge. 	**Video 1: Introduction to Input/Output management (8 minutes)** Welcome to the ninth module of the Operating Systems course. The module will introduce you to the role of Operating Systems in controlling all I/O activities of a Computer System. The Operating System plays a significant role in controlling I/O devices, namely: ● Issuing commands, ● handling of interrupts, ● error handling ● Providing easy to use interfaces to devices ● Allows for device independence. You will begin by being exposed to some of the principles of I/O hardware and then I/O software in general. I/O software can be structured in layers, with each having a well-defined task. You will be looking at these layers in details in order to see what they do and how they fit together. Next, you will be exposed to the internals and operations of several I/O devices in detail. **Principles of I/O hardware** Input/Output devices can be roughly divided into two categories: block devices and character devices. A block device is one that stores information in fixed-size blocks, each one with its own address. In block devices, all transfers are in units of one or more entire (consecutive) blocks. A key attribute for these devices is that you each block can be read or written independently of all other blocks. Common examples of block devices are Hard disks, Blu-ray discs, and USB. Character devices on the other hand, deliver or accept a stream of characters, without regard to any block structure. They are not addressable and do not have any seek operation. Common examples of character devices are printers, network interfaces and mice (for pointing). It is important to note that this broad classification scheme is not necessarily all inclusive. Therefore there are some devices that do not perfectly fall into either category. For instance, clocks are neither block addressable nor do they generate or accept character streams. All they do is to cause interrupts at well-defined intervals.

Additionally, memory-mapped screens and touch screens do not fit the model well either. Nonetheless, the model of block and character devices is general enough that it can be used as a basis for making some of the operating system software dealing with I/O device independent. This therefore leaves the file system to just deal with abstract block devices and let the device-dependent features to be handled by lower-level software.

Device Controllers
The conventional design of I/O units separates them into two major components, mechanical and electronic. The electronic component is called the device controller or an adapter while the mechanical component is the device itself. In personal computers, the electronic component often takes the form of a chip on the parentboard or a printed circuit card that can be inserted into an expansion slot.

The interface between the controller and the device is often a very low-level one. The controller's job is to convert the serial bit stream into a block of bytes and perform any error correction where necessary. Error detection and correction is done by calculating checksum values of the bytes received and once declared error-free, the blocks are copied to memory.

Video 2: Memory-Mapped I/O (5 minutes)
Each device controller has registers that are used for communicating with the CPU. The Operating System writes into these registers when it wants to communicate with the device. The Operating System can command the device to deliver data, accept data, switch itself off or on or even perform other functions by writing the commands into these registers.

On the other hand, by reading from these registers, the operating system can learn what the device's state is, whether it is prepared to accept a new command or not. In addition to the control registers, many devices have a data buffer that the operating system can read and write.

With this background, the challenge thus arises on how the CPU communicates with the control registers and also with the device data buffers. Two approaches have been proposed in order to deal with this situation. In the first approach, each control register is assigned an I/O port number. The set of all the I/O ports form the I/O port space, which is protected so that ordinary user programs cannot access it (only the operating system can).

The second approach is where all the control registers are mapped onto the memory space. This is achieved by assigning each control

register to a unique memory space. This approach is called memory-mapped I/O. In most systems, the assigned addresses are at or near the top of the address space.

The advantage of memory-mapped I/O is that with memory-mapped I/O, an I/O device driver can be written entirely in C. Without memory-mapped I/O, some assembly code is needed. Further, with memory-mapped I/O, no special protection mechanism is needed to keep user processes from performing I/O.

Memory-mapped I/O also has its disadvantages. First, most computers nowadays have some form of caching of memory words. Caching a device control register would be disastrous. Second, if there is only one address space, then all memory modules and all I/O devices must examine all memory references to see which ones to respond to.

Video 3: Direct Memory Access (5 minutes)

assume that the CPU accesses all devices and memory
via a single system bus that connects the CPU, the memory, and the I/O devices as shown in the figure below.

insert diagram on DMA here. Check figure 5-4 from reference book.

The operating system can use only DMA if the hardware has a DMA controller, which most systems do. Sometimes this controller is integrated into disk controllers and other controllers, but such a design requires a separate DMA controller for each device. More commonly, a single DMA controller is available (e.g., on the parentboard) for regulating transfers to multiple devices, often concurrently.

Video 4: Principles of I/O software
One of the key conepts of in the design of I/O software is **device independence.** This means that we should be able to write programs that can access any I/O device without having to specify the device in advance. Closely related to device independence is the goal of **uniform naming**. The name of a file or a device should simply be a string or an integer and not depend on the device in any way.

Error handling is another important issue for I/O software. Generally, errors should be handled as close to the hardware as possible. If the controller discovers a read error, it should try to correct the error itself if it can. If it cannot, then the device driver should handle it, perhaps by just trying to read the block again.

	Transfer mode is another very important concept in I/O software design. That is whether an I/O activity is synchronous (blocking) or asynchronous (interrupt-driven). In most cases, physical I/O is asynchronous. This means that the CPU starts the transfer and goes off to do something else until the interrupt arrives. It is important to note that user programs are much easier to write if the I/O operations are blocking. For example, after a read system call, the program is automatically suspended until the data are available in the buffer. It is up to the operating system to make operations that are actually interrupt-driven look blocking to the user programs. However, some very high-performance applications need to control all the details of the I/O therefore some operating systems make asynchronous I/O available to them. **Buffering** is yet another I/O software design issue that must be taken into consideration. In most cases, data that come off a device cannot be stored directly in their final destination. It has to be temporarily kept in a "store" from where it can then be accessed. The I/O software design must therefore be ready to deal with both situations of **buffer underruns** and **buffer overflow**. The final design issue is that of sharable versus dedicated devices. This helps in determining how to control some undesirable characteristics such as deadlocks and starvation. The Operating System must have mechanisms to handle all these challenges. **Programmed I/O** This is the simplest form of I/O design in which CPU is left to do all the work. The essential aspect of programmed is that after outputting a character, the CPU continuously polls the device to see if it is ready to accept another one. This behavior is often called **polling** or **busy waiting**. Programmed I/O is simple but has the disadvantage of tying up the CPU full time until all the I/O is done. **Interrupt-Driven I/O** The idea of interrupt-driven I/O is to block process which requests I/O and then schedule another process then return to calling process when I/O is done. For example, a printer generates interrupt to the CPU when a character is printed the CPU grants it the chance hence it keeps printing until the end of the string. The CPU then re-instantiate calling process.

A key disadvantage of the interrupt-driven I/O is that an interrupt occurs on every I/O request and for every character. Interrupts take time, so this scheme wastes a certain amount of CPU time. An alternative is to use DMA. In the next subsection we are going to discuss I/O using DMA.

I/O Using DMA

In this approach, the idea is to let the DMA controller feed the characters to the printer one at time, without the CPU being bothered. This way, I/O using DMA in essence behaves like programmed I/O only that the DMA controller does all the work, instead of the main CPU. This strategy requires special hardware (the DMA controller) but frees up the CPU during the I/O to do other work.

Video 5: Layers of I/O Software

I/O software are organized into layers and each layer has well defined functions and interfaces for interacting with its adjacent layers. The layers are (from hardware to application layer, i,e bottom-up): interrupt handlers, device drivers, device-independent operating system software and user-level I/O software.

Insert a diagram that shows the I/O software system in a layered fashion as described above with hardware at the bottom.

Interrupt handlers
Interrupt handlers are special routines that are invoked when an interrupt occurs and help in processing the interrupts. The basic idea is that a device driver starts an I/O process then blocks until an interrupt happens when the I/O activity finishes. The interrupt handler processes interrupt and wakes up the driver when processing is finished. Drivers are kernel processes with their very own stacks, Program Counters and process states.

The details of interrupt processing can be outlined as follows:

1. Save registers not already saved by the interrupt hardware.
2. Set up a context for the interrupt service procedure. Doing this may involve setting up the TLB, MMU and a page table.
3. Set up a stack for the interrupt service procedure.
4. Acknowledge the interrupt controller. If there is no centralized interrupt controller, re-enable interrupts.
5. Copy the registers from where they were saved to the process table.
6. Run the interrupt service procedure.
7. Choose which process to run next.
8. Set up the MMU context for the process to run next.

9. Load the new process' registers, including its PSW.
10. Start running the new process.

Device Drivers

Each hardware device has a device controller and each device controller has some registers that are used to give it commands or to read out its status or both. The number of device registers and the nature of the commands vary from device to device. Therefore, each I/O device attached to a computer needs some device-specific code for controlling it. This code is referred to as the **device driver**.

Device drivers are usually written by the manufactures of the device. Since each operating system needs its own drivers, device manufacturers commonly supply drivers for several popular operating systems.

Some of the facts about device drivers are as follows:

1. Device drivers contain code specific to the device
2. They are supplied by manufacturer
3. They are mostly installed in the kernel, even though it might be better to install them in the user space because a bad driver can mess up the kernel.
4. Device drivers need to interface to the Operating System.
5. Device drivers check input parameters for validity
6. They help move from abstract to concrete translation (block number to cylinder, head, track, sector)
7. They check device status. Might have to start it.
8. They put commands in device controller's registers
9. Driver blocks itself until interrupt arrives
10. Might return data to caller
11. Does return status information

Device-Independent I/O Software

In as much as some of the I/O software are device specific, other parts of it are device independent. The exact boundary between the drivers and the device-independent software is system (and device) dependent, because some functions that could be done in a device-independent way may actually be done in the drivers, for efficiency or other reasons.

Some of these tasks that are being done by the device-independent I/O software are:

1. Uniform interfacing for device drivers
2. Buffering
3. Error reporting
4. Allocating and releasing dedicated devices
5. Providing a device-independent block size

	User-Space I/O Software Although most of the I/O software is within the operating system, a small portion of it consists of libraries linked together with user programs, and even whole programs running outside the kernel. System calls, including the I/O system calls, are normally made by library procedures. Not all user-level I/O software consists of library procedures. Another important category is the spooling system. Spooling is a way of dealing with dedicated I/O devices in a multiprogramming system. Instead what is done is to create a special process, called a daemon, and a special directory, called a spooling directory.
ACTIVITY 2: READING READING MATERIAL 1 	Please read the following materials so as to be able to answer questions provided in the next activity. a. File Management Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 263 - 325 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div b. I/O Device Management Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 337 - 399 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div c. Secondary Storage Management Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 369 - 385 https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div

ACTIVITY 3: Comprehension questions: 	1) Explain how an OS can facilitate installation of a new device without any need for recompiling the OS. 2) Suppose that a computer can read or write a memory word in 5 nsec. Also suppose that when an interrupt occurs, all 32 CPU registers, plus the program counter and PSW are pushed onto the stack. What is the maximum number of interrupts per second this machine can process? 3) Explain the tradeoffs between precise and imprecise interrupts on a superscalar machine. 4) In which of the four I/O software layers is each of the following done. (a) Computing the track, sector, and head for a disk read. (b) Writing commands to the device registers. (c) Checking to see if the user is permitted to use the device. (d) Converting binary integers to ASCII for printing. 5) A typical printed page of text contains 50 lines of 80 characters each. Imagine that a certain printer can print 6 pages per minute and that the time to write a character to the printer's output register is so short it can be ignored. Does it make sense to run this printer using interrupt-driven I/O if each character printed requires an interrupt that takes 50 μ sec all-in to service?
LEARNING OUTCOME 2: Conceptual knowledge ACTIVITY 4: Video to be used.	
CASE 1: 	A local area network is used as follows. The user issues a system call to write data packets to the network. The operating system then copies the data to a kernel buffer. Then it copies the data to the network controller board. When all the bytes are safely inside the controller, they are sent over the network at a rate of 10 megabits/sec. The receiving network controller stores each bit a microsecond after it is sent. When the last bit arrives, the destination CPU is interrupted, and the kernel copies the newly arrived packet to a kernel buffer to inspect it. Once it has figured out which user the packet is for, the kernel copies the data to the user space. If we assume that each interrupt and its associated processing takes 1

	msec, that packets are 1024 bytes (ignore the headers), and that copying a byte takes 1 µ sec, what is the maximum rate at which one process can pump data to another? Assume that the sender is blocked until the work is finished at the receiving side and an acknowledgement comes back. For simplicity, assume that the time to get the acknowledgement back is so small it can be ignored.
ACTIVITY 5: READING MATERIAL 	In this section you have been provided with links to online resources that should help you improve your understanding of two key concepts in I/O role of the Operating System. You are required to read all of them carefully and use the information to write a brief blog in the LMS as you will be instructed. 1) Device Controllers a) https://www.geeksforgeeks.org/device-controllers-in-operating-system/ b) https://www.tutorialspoint.com/operating_system/os_io_hardware.htm#:~:text=The%20Device%20Controller%20works%20like,is%20called%20the%20device%20controller. 2) Memory-mapped I/O a) https://www.geeksforgeeks.org/memory-mapped-i-o-and-isolated-i-o/ b) https://www.baeldung.com/cs/memory-mapped-vs-isolated-io • Having read the above articles, you are required to write a blog in the LMS with focus on device controllers and memory mapped I/O. In your blog also discuss how device drivers interact with the device controllers and the Operating System. You can use examples where necessary.
ACTIVITY 6: ONLINE DISCUSSION 	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 session, you are provided with online videos that you are required to watch in order for you to re-inforce your understanding of the following data transfer modes: Programmed I/O, Interrupt driven I/O, Direct Memory Access (DMA).

	Programmed I/O (5:41 minutes): Data Transfer modes (11:47 minutes): Basics of OS (I/O Structure) - 12:44 minutes : Interrupt driven I/O (28 minutes):
ACTIVITY 7: Learner practice sessions	In this session, you are required to do a "lightning talk" focusing on "Input/Output management" function of the Operating System. In the "lightning talk", use your smartphone or any other video camera to record yourself in not more than "30 seconds" while explaining how "***Interrupt Processing***" is done by the Operating System. Note: 1. Upload your video with the captions <*fname_lname_talk9*>. where "*fname*" is your first name and "*lname*" is your last name (or surname). The video must not be more than 30 seconds long.
ASSESSMENT OF PRACTICAL SKILL:	In the above activity, you uploaded your video, <***fname_lname_talk9***>. It will be assessed 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 been provided with links to online resources where you are required to read more about Direct Memory Access (DMA) and Disk Space Management. Use these resources to help you undertake the activity in the next section. 1) Direct Memory Access a) https://www.minitool.com/lib/direct-memory-access.html b) http://inputoutput5822.weebly.com/direct-memory-access.html c) https://www.geeksforgeeks.org/direct-access-media-dma-controller-in-computer-architecture/ 2) Disk Space Management a) https://www.ibm.com/docs/en/imdm/11.6?topic=monitoring-disk-space-management b) https://www.tutorialandexample.com/free-space-management-in-operating-system c) https://www.geeksforgeeks.org/free-space-management-in-operating-system/

ACTIVITY 8	In this module, you have learnt about several concepts in Input/Output Devices Management. You are required to write a two page essay on Disk space management, disk arm scheduling and System clock (both hardware and software). 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. Which of the following is/are the technique(s) for performing I/O management function. A) Programmed I/O B) Interrupt driven I/O C) Direct Memory Access D) All of the above 2. In ……., the processor issues an I/O command, on behalf of a process, to an I/O module. A) Programmed I/O B) Interrupt driven I/O C) Direct Memory Access D) Virtual Memory Access 3. In ……., the processor issues an I/O command on behalf of a process, continues to execute subsequent instructions and interrupted by the I/O module when the latter has completed it's work. A) Programmed I/O B) Interrupt driven I/O C) Direct Memory Access D) Virtual Memory Access 4. A ……. module controls the exchange of data between main memory and an I/O module. A) Programmed I/O B) Interrupt driven I/O C) Direct Memory Access D) Virtual Memory Access 5. The …… unit is capable of mimicking the processor and indeed of taking over control of the system from the processor.

A) Programmed I/O
B) Interrupt driven I/O
C) Direct Memory Access
D) Virtual Memory Access

6. ……. layer deals with the logical structure of files and with the operations that can be specified by users such as open, close, read and write.
A) Physical organization
B) File system
C) Directory management
D) Scheduling and control

7. When a user process issues an I/O request, the operating system assigns a buffer in the system portion of main memory to the operation is called ……..
A) Double buffer
B) Single buffer
C) Linear buffer
D) Circular buffer

8. ……. may be inadequate if the process performs rapid bursts of I/O.
A) Double buffering
B) Single buffering
C) Linear buffering
D) Circular buffering

9. On a movable head system, the time it takes to position the head at the track is known as ……
A) seek time
B) rotational delay
C) access time
D) Transfer time

10. The time disk controller takes for the beginning of the sector to reach the head is known as ……..
A) seek time
B) rotational delay
C) access time
D) Transfer time

11. The …….. consists of two key components: the initial startup time, and the time taken to traverse the tracks that have to be crossed once the access arm is up to speed.
A) seek time
B) rotational delay
C) access time

	D) Transfer time 12. The …… policy is to select the disk I/O request that requires the least movement of the disk arm from its current position. A) Last in first out B) Shortest service time first C) Priority by process D) Random scheduling **Answers:** 1. D) All of the above 2. A) Programmed I/O 3. B) Interrupt driven I/O 4. C) Direct Memory Access 5. C) Direct Memory Access 6. B) File system 7. B) Single buffer 8. A) Double buffering 9. A) seek time 10. B) rotational delay 11. A) seek time 12. B) Shortest service time first
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