THE OPEN UNIVERSITY OF KENYA

DESIGN PLAN

Programme title	Bachelor of Science in Cybersecurity and Digital Forensics
Course title	Operating Systems
Learning Module number	07
Learning module title	Threads
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 threads in Operating System. 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 fundamental concepts of threads such as creation, synchronization and termination. We shall also learn how to develop multithreaded applications using C or C++ programming languages.
Module objectives:	This module aims at facilitating learners to acquire knowledge about: 1. The meaning and context of threads in computer processes 2. Interaction between processes and threads 3. Approaches for implementing threads in Operating Systems 4. Concurrent programming with threads
Module learning outcomes:	By the end of the module, you should be able to: 1. Define threads in computer processes 2. Describe the interaction between processes and threads 3. Illustrate different approaches for implementing threads in Operating Systems

<table>
<tr><td></td><td>4. Develop multithreaded programs in a high level programming language (C or C++)</td></tr>
<tr><td>Planned Learning Resources</td><td></td></tr>
<tr><td>ACTIVITY 1: INTRODUCTION
VIDEO 1: Pre-recorded lecture on topic emphasizing LEARNING OUTCOME 1: Factual knowledge.
  </td><td>
Video 1: Introduction to Threads (5 minutes)

Welcome to the seventh module of this course in Operating Systems. In the previous two modules, you have been learning about processes. In this module, you will learn about a concept that is very closely related to processes and is referred to as a thread.

A thread can be defined as a flow of execution through the process code. Each thread has its own program counter, system registers and a stack.

Since threads exist within processes, a thread will therefore have other items that they share with their peers. These include: code segment, data segment and open files. This implies that when one thread alters a code segment in memory, all the other threads see that.

A thread is also called a lightweight process.

Some of the operations applicable on threads include: thread creation, termination, synchronization (joins,blocking), scheduling, data management and process interaction.

A thread does not maintain a list of created threads, nor does it know the thread that created it. All threads within a process share the same address space.

Threads in the same process share:

● Process instructions
● Most data
● open files (descriptors)
● signals and signal handlers
● current working directory
● User and group id
● Each thread has a unique:
● Thread ID
● set of registers, stack pointer
● stack for local variables, return addresses
● signal mask
● priority

The table below shows what is unique to a thread and what is shared within the process:
<table>
<tr><td>Per process Items</td><td>Per Thread Items</td></tr>
<tr><td>Address Space
Global variables</td><td>Program counter
Registers</td></tr>
</table>
</td></tr>
</table>

Open files	Stack
Child processes	State
Pending alarms	
Signal and signal handlers	
Accounting information	

Threats have the same states as those of a process. These include:

1. Running
2. Ready
3. Blocked

Video 2: POSIX Threads (Pthreads): 5 minutes

POSIX refers to the "Portable Operating System Interface". It refers to a family of standards specified by the IEEE computer society for maintaining compatibility between operating systems.

The POSIX (IEEE 1003.1c) standard defines both the system and user-level application programming interfaces (APIs), along with command line shells and utility interfaces, for compatibility (portability) with variants of Unix and other operating systems.

The standard also provides specification for implementation of threads, usually known as Pthreads. Pthreads may be provided either as user-level or kernel-level

A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization. It provides a specification and not an implementation guide. An API specifies behavior of the thread library but leaves implementation to developers of the library.

The Pthreads provides more than 60 functions that manipulate threads from C programs. These include:

1. Creating and reaping threads

   pthread_create, pthread_join

2. Determining your thread ID

   pthread_self

3. Terminating threads

   pthread_cancel, pthread_exit

   exit [terminates all threads] , return [terminates current thread]

4. Synchronizing access to shared variables

   pthread_mutex_init, pthread_mutex_[un]lock

pthread_cond_init, pthread_cond_[timed]wait

Video 3: Implementing Threads (8 minutes)

There are three approaches to thread implementation, based on the address space in which they are running. These are: user Space threads, kernel space threads and hybrid threads.

Implementing Threads in User Space

illustrate the user space threads here using a diagram:

(a) A user-level threads package. (b) A threads package managed by the kernel.

Advantages of user space threads

These are some of the advantages of implementing user space threads:

1. Thread table contains information about threads (program counter, stack pointer...) so that the runtime system can manage them.
2. If thread blocks, the runtime system stores thread information in the table and finds a new thread to run.
3. Saving of state and scheduling are invoked faster than kernel call since there is no need for issuing of traps and cache flushing

Disadvantages of user space threads

Some of the disadvantages of implementing threads in the user address space include:

1. Cannot execute blocking system calls because of the risk of blocking all other threads
2. There is a need to tweak the system library to avoid blocking calls
3. This level of threads do not voluntarily give up CPU
4. Could interrupt periodically to give control to run time system

Implementing threads in the Kernel Space

Insert a diagram on kernel space threads here

	Advantages of Threads in kernel space The following are some of the advantages of implementing threads in the kernel space. 1. Kernel keeps the same thread table as the user table. 2. If a thread blocks, the kernel just picks another one, not necessarily from the same process. **Disadvantages of kernel threads approach** 1. Expensive to manage the threads in the kernel and takes valuable kernel space In order to get benefits of both approaches, a hybrid thread implementation approach has been utilized. **Hybrid approach** In this approach, user-level threads are multiplexed onto kernel level threads. Kernel is aware of kernel threads only and user level threads are scheduled, created and destroyed independently of kernel thread. It is at the discretion of the programmer to determine how many user-level and how many kernel level threads to use.
ACTIVITY 2: READING READING MATERIAL 1 	In this section, you have been provided with additional reading materials that should help you in gaining better understanding of the concepts that have been covered in this module. **a. Threads** 1) Andrew S Tanenbaum. (2016). Modern Operating Systems Paperback. Pearson. pp 97 - 119; https://www.amazon.com/Modern-Operating-Systems-Andrew-Tanenbaum/dp/9332575770#detailBullets_feature_div 2) Remzi H.,Andrea C. (2014). Operating Systems - Three Operating Systems. pp 25 - 33

ACTIVITY 3: Comprehension questions: 	1) What is the biggest advantage of implementing threads in user space? What is the biggest disadvantage? 2) Consider a system in which threads are implemented entirely in user space, with the run-time system getting a clock interrupt once a second. Suppose that a clock interrupt occurs while some thread is executing in the run-time system. What problem might occur? Can you suggest a way to solve it? 3) If a multithreaded process forks, a problem occurs if the child gets copies of all the parent's threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem ever occur in single-threaded processes? 4) Why would a thread ever voluntarily give up the CPU by calling *thread yield*? After all, since there is no periodic clock interrupt, it may never get the CPU back. 5) Describe two techniques for creating Thread objects in Java. 6) Distinguish between coarse-grained and fine-grained multithreading.
LEARNING OUTCOME 2: Conceptual knowledge ACTIVITY 4: Video to be used.	
CASE 1: 	An eatery has only one bathroom space for its staff. It decrees that when a woman is in a bathroom, other women may enter, but no men, and vice versa. A sign with a sliding marker on the door of each bathroom indicates which of three possible states it is currently in: • Empty • Women present • Men present Use concepts in C or Java programming languages to write the following procedures: i. *woman wants to enter*, ii. *man wants to enter*, iii. *woman leaves*, iv. *man leaves*. You may use whatever counters and synchronization techniques you like.
ACTIVITY 5: READING MATERIAL 	In this section, you are required to read more about threads from the following online sources. You will then be required to write a blog about what you have read.

	a) https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4_Threads.html b) https://www.geeksforgeeks.org/thread-in-operating-system/ c) https://www.javatpoint.com/threads-in-operating-system#:~:text=A%20thread%20is%20a%20single,process%20of%20any%20operating%20system. a) Having read the above articles, you are required to write a blog in the LMS with focus on comparing and contrasting Thread libraries: Pthreads, Windows Thread and Java Threads.
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 section, you are required to watch the following videos which cover some of the aspects we have looked at in this course: 1) Introduction to Threads (14 minutes) 2) Multithreading models (17:56 minutes) 3) Threading Issues (10:32 minutes)
ACTIVITY 7: Learner practice sessions	In this session, you are required to do a "lightning talk" focusing on "Thread Implementation Approaches". In the "lightning talk", use your smartphone or any other video camera to record yourself in not more than "60 seconds" while explaining three different approaches in "***Thread Implementation***" and the pros and cons of each. Note: 1. Upload your video with the captions <*fname_lname_talk7*>. 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_talk7***>. 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	The following materials and online resources have been carefully selected for you for further reading. They will help you achiever further grasp of the fourth objective in this module. 1) POSIX thread (pthread) libraries: • https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html 2) How to create a simple thread in C: • https://www.educative.io/answers/how-to-create-a-simple-thread-in-c 3) Writing Multithreaded applications in C • https://beej.us/guide/bgc/html/split/multithreading.html 4) Thread programming examples: • https://users.cs.cf.ac.uk/dave/C/node32.html 5) Multithreading in C++ • http://www.tutorialspoint.com/cplusplus/cpp_multithreading.htm
ACTIVITY 8	In this module, you have learnt about several concepts in Threads. You are required to write a two page essay on approaches used by Operating Systems in management of Threads. 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 suitable for 'thread'? a. Heavyweight process b. Process c. Lightweight process d. A task 2. What are the components of a thread? a. Thread Id, Program counter, Register set and Stack b. Thread Id, Program counter, Register set, and Queue

	c. Thread Id, Register set, Stack and Queue d. Thread Id, Register set, Program counter, Stack and Queue 3. Which of the following statements is false about thread? a. Thread is a lightweight process. b. Thread takes more time to terminate a running thread than a process. c. Thread takes less time for switching in between threads. d. Thread consumes less system resources. 4. Which of the following is/are the type(s) of threads? a. User level threads b. Kernel level threads c. Many-to-Many d. Both (1) and (2) 5. At which level is the management of User level threads done? a. Kernel b. Application c. Operating system d. All of the above 6. Which thread cancellation approach allows to terminate the thread immediately? a. Deferred cancellation b. Asynchronous cancellation c. Immediate termination is not possible. d. Both (1) and (2) 7. IEEE 1003.1c-1995 standard defines: a. Pthreads specification b. Pthreads implementation c. Pthreads specification and implementation d. Green threads specification 8. A thread life cycle consists of _____ states. a. 2 b. 3 c. 4 d. 5 9. Which of the following methods is/are defined in Object class? A. wait() B. sleep() C. stop() D. Both options (A) and (B) 10. What is a target thread?

	a. A thread to be executed next. b. A currently running thread. c. A newly created thread. d. A thread to be cancelled. 11. Which of the following level of a thread is the fastest to create and manage thread? a. Many-to-Many b. Kernel c. Many-to-One d. User 12. Many-to-One model defines: a. many user level threads to one kernel level threads b. many kernel level threads to one user level threads c. many user level threads to one user level threads d. many user kernel threads to one kernel level threads 13. Pthreads refers to: a. the POSIX defining an API for thread creation only b. the POSIX defining an API for thread implementation only c. the POSIX defining an API for thread synchronization only d. the POSIX defining an API for thread creation and synchronization 14. In which of the following operating systems is signal handling used to notify a process, that a particular event has occurred? a. Unix b. Linux c. Mac d. Windows 15. How do threads communicate with each other? a. Threads cannot communicate with each other. b. Threads communicate with each other by wait(), notify() and notifyAll() methods. c. Threads communicate with each other through stack. d. Threads communicate with each other through queue. Answers: 1. c. Lightweight process 2. a. Thread Id, Program counter, Register set and Stack 3. b. Thread takes more time to terminate a running thread than a process. 4. d. D. Both (1) and (2) 5. b. Application 6. b. Asynchronous cancellation 7. a. Pthreads specification

	8. c. 4: New, Runnable, Running, Non-Runnable (Blocked) and Terminated. 9. a. wait() 10. d. A thread to be cancelled. 11. d. User 12. a. many user level threads to one kernel level threads 13. d. the POSIX defining an API for thread creation and synchronization 14. a. Unix 15. b. Threads communicate with each other by wait(), notify() and notifyAll() methods.
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