

Lara Terzi SSMUAE sem.1 2018

by Lara Terzi

FILE	3643_LARA_TERZI_LARA_TERZI_SSMUAE_SEM.1_2018_95675_2051735020.DOCX (27.97K)		
TIME SUBMITTED	25-JAN-2018 04:18PM (UTC+0000)	WORD COUNT	2549
SUBMISSION ID	82376868	CHARACTER COUNT	16462

Lara Terzi

P148014

System Software & Multi-User
Applications Engineering

M5X01055

Research Report

Dr. Tim Bashford

Contents

1. Introduction	3
1.1. Multiprogramming systems	3
1.2. Basic computer operations and devices.	3
1.3. An introduction to processes and threads.....	4
2. Processes.....	4
2.1. What is a process and how to multi-process systems work?	4
2.1.1. Constituents of a process.....	4
2.2. How to programs relate to processes?	5
2.2.1. Scheduling occurrences and algorithms	5
2.2.2. Inter-Process Communication.....	5
2.2.3. Synchronization.....	6
3. Threads	6
3.1. Threads and why they are required.....	6
3.1.1. Constituents of a thread and the lightweight process.....	6
3.2. Scheduling	7
3.2.1. Thread scheduling and algorithms	7
3.3. Thread Communication	7
3.3.1. Methods of thread communication (functions)	7
3.4. Thread Synchronization	7
3.4.1. Methods of thread synchronization	7
4. Summary	8
4.1. Summarize points made.	8
5. References	8

1. Introduction

1.1. Multiprogramming systems

A multi-programming system uses parallel processing to allow a user to use multiple programs at the same time. As a computer's central processing unit can only execute one task at a time, so the computer's operating system uses context switching. This is a process which executes part of a program, then part of another, and so on making it look to the user as if the computer were processing all programs simultaneously, however, each program stops running before the next one starts and resumes only when each program's instructions have been executed. Protection must be applied to each program so that one program's tasks does not change another's. This is done by loading programs onto separate areas of the main memory. Fragmentation, wasteful memory allocation, can occur in the computer's main memory when multiprogramming, which can reduce storage capacity as well as diminishing computer performance. Another disadvantage of multiprogramming is the size capacity as large programs may not fit in the computer's main memory, however, virtual memory can be a remedy for this.

1.2. Basic computer operations and devices.

The fetch-decode-execute cycle is the basic computer operation cycle construct. The fetch operation 'fetches' a program's instructions and temporarily stores them in portions of the memory called registers. The 'address bus' is then used to 'fetch' the next instruction's address, a computer bus can be defined as a series of connecting lines that bridges devices together. Data is then moved from the RAM into the CPU via the 'data bus'. During the decoding process, the CPU decodes and interprets the information given. The execute operation processes and carries out the instructions. That data is then stored in a different register and the CPU arranges to re-start the cycle. This cycle continues repeating whilst the computer is switched on.

Input devices can be used to enter information and data into a computer. These includes other computers, disks, keyboards, image scanners, microphones, and more. Data and information can also be transferred via internet and network connections. Output devices are used to visualize and use data entered onto the computer. These include monitors, speakers, printers, projectors, any computer errors displayed, and more. A computer's "central processing unit is the heart of the computer, where the information processing takes place" [1]. This includes any process, task, program, device, etc. executed on the computer.

1.3. An introduction to processes and threads.

A computer's central processing unit processes lines of code that act as instructions for the operation of a program and, of course, its system. In order for programs to run, these instructions must be sent to all the appropriate hardware. They must also communicate with one another and with all the different components of a computer. As some instructions must come before others, scheduling is a very important part of process and thread execution. Synchronization also plays an important role in processing as if there is no communication and backup of the processes and threads that are executed, then the restarting point of execution would be unknown. In this sense, synchronization ensures that the executed tasks' progress has been saved and that it does not have to redo these processes at the next computer start up, saving time and money for all who use a computer.

2. Processes

2.1. What is a process and how do multi-process systems work?

2.1.1. Constituents of a process.

A process is "a series of actions" [2], or a set of instructions that are completed by the central processing unit for a program to run. A process has five states; the first is the 'start' or 'new' state, where a process is created. Once the creation is complete, the process is sent to the 'ready' state where it awaits to be assigned to a CPU core. Once that state is complete, the process is ready to run and is sent to the 'running' state by the scheduler where that process is then given the specific resources needed to run its instructions. Upon completion of those instructions, the process is sent to the 'terminated' state. In the case of an interrupt, a term used to define the need of the CPU core for another process, the process in the 'running' state is sent back to the 'ready' state, where it awaits to advance to the 'running' state again. If the process needs permissions from external drives or input and output devices, the process is sent to the 'waiting' state as it takes time for these data to be accessed, time that the processor can spend on another process. When the process has reached the 'terminated' state, the operating system performs clean-up tasks before eliminating the process.

A multi-process system can support two or more processors that share the same central memory, RAM. Having a multiprocessing system makes for a faster computer as two or more of the program's sets of instructions, depending on how many microprocessors the computer has, are being executed in the time that a single-processing computer would process one. The processors can also designate tasks to other processors whilst they start working on new ones which again, increases speed and efficiency as more data can be accessed contemporarily. However, as the speed increases so does the heat produced and cooling mechanisms need to be put in place. A scheduler is needed to ensure that processes that are reliant upon each other are executed in right order. It is also needed to ensure storage space optimization.

2.2. How to programs relate to processes?

2.2.1. Scheduling occurrences and algorithms

Scheduling occurs when processes are moved from a ready state to a running state. A scheduler decides which process or thread to execute based on size, novelty and priority, depending on what scheduling algorithm is used by the CPU. There are three scheduling levels; long-term scheduling, midterm scheduling and short-term scheduling. Long-term scheduling focuses on a program's processing admission and multiprogramming. Midterm scheduling focuses on context switching and the management of multiprogramming, and lastly, short-term scheduling focuses on interrupts and signal communication, and is the most frequently executed scheduling level.

There are two types of scheduling algorithms, preemptive and non-preemptive. Preemptive scheduling allows for the scheduler to move the process from the 'running' state to the 'ready' state so that another process, one with higher priority, can take its place. In a non-preemptive algorithm, the process cannot be stopped until it completes its allocated time cycle.

First Come First Serve (FCFS) is an algorithm that can be used in modern computers. Processes are carried out in a 'first come first serve' fashion and First In First Out (FIFO) model where the first and oldest process is executed first. This however wastes time as the processes that should have higher priority are not all executed first. Shortest Job Next (SJN) is the least time-wasting algorithm as the faster jobs are done immediately, leaving plenty time for the larger ones, but the CPU time must be known beforehand.

Priority Based Scheduling is commonly used in computers with batch processing. Processes are scheduled in order of priority as it is a non-preemptive algorithm. The First Come First Serve algorithm is imbued in this priority-based algorithm as the processes that have the same priorities are executed in that manner.

Shortest Remaining Time (SRT) is similar to the Shortest Job Next algorithm as the shortest task is given priority, however, unlike the SJN algorithm, the SRT algorithm is preemptive. CPU time must be known in order to use this algorithm. Round Robin Scheduling is a preemptive algorithm where each process has a quantum, a time limit for each process execution. The use of a quantum and context switching is used to save time. Lastly, Multiple-Level Queues Scheduling is embedded into other algorithms and uses those algorithms to group process categories together.

2.2.2. Inter-Process Communication

2.2.2.1. Methods

Inter-process communication (IPC) defines how the different processes communicate and share resources and data, usually via a client-server platform, where the server adheres to the client's requests depending on permissions. In today's era there are many inter-process communication methods, seven of which will be discussed in the following paragraph.

A signal is a common concurrent method of inter-process communication for software and hardware event data exchange. Semaphores are objects used to connect resources to those who have the permission. In software programming, an object is a specific class which can use a mixture of variables, functions and data structures. Similarly, condition variables are objects that ensure the sole resource's permission. Barriers ensure the readiness of the process before it is sent to be processed. Message queues are objects that exchange queues of messages to other message queues. When access to all or most of a file is obtained it is called a file lock. Lastly, sockets connect different systems virtually.

2.2.3. Synchronization

2.2.3.1. Methods

Data synchronization involves an algorithm to synchronize two or more processes together. The first of the three synchronization methods lets the first process alter the second. The second method lets the second process alter the first. The last method lets the two processes alter each other and exchange data. Data synchronization works so that the smaller devices are synchronized on the larger devices, almost like a data backup, so there are many methods of synchronization available. Database synchronization is the backing up of a database as well as the exchange of data throughout database entities. Email synchronization is a method of downloading the data from the email databases onto the computer, the exchange of information from that database to the user's computer.

File synchronization is the exchange of files from external drives such as USBs or external hard disk drives to a computer. This is an advantageous method of synchronization as it is faster, errors are less likely, and file duplication is prevented. Distributed file systems method ensure that multiple copies of the file are synchronized. The mirroring method creates a duplicate of any piece of data and is thus more efficient as data is very unlikely to be corrupted. Mirroring also offers an 'automatic failover', which is a process that moves data to secure servers in the case of failure or diminished computer performance.

3. Threads

3.1. Threads and why they are required.

3.1.1. Constituents of a thread and the lightweight process

A thread is a set of a program's instructions. A well written thread has as few lines of code as possible to ensure as small a size, as therefore execution time, as possible. Single threading executes one thread whilst multithreading executes multiple threads inside a process. The differences between a thread and a process are as follows; whilst a habitual process is independent, a thread is usually a fragment of a process and thus multiple threads can share the same memory address and resources. Thread context switching also uses less CPU time than process context switching. Multithreading can also be used on a process to carry out multiple processes at the same time.

The lightweight process (LWP) is another term for multitasking. Multitasking is the execution of multiple processes at within a quantum, by stopping and saving the state of the thread when a higher priority thread needs to be executed. The more processes are executed at a time, the faster a computer will be therefore the lightweight process is an important constituent of threading. Threading is very useful as it gives the illusion of multi-process processing, as multiple threads can be processed in one single process.

3.2. Scheduling

3.2.1. Thread scheduling and algorithms.

The thread scheduler is the part of the operating system that oversees any thread scheduling. As it is part of the operating system, it is responsible in understanding the thread's data and for thread context switching. Lastly, the most scheduler's most important responsibility is creating a schedule that executes threads and processes in the right quantum and in the right order to maximize the computer's efficiency and the central processing unit's time.

Fixed priority scheduling is a thread scheduling algorithm used in Java. This is similar to the priority-based process scheduling algorithm as it is based on a thread's priority, however with fixed priority scheduling, a thread's priority can be changed with the `setPriority` function.

3.3. Thread Communication

3.3.1. Methods of thread communication (functions)

Cooperation is the name for the two methods used for inter-thread communication. `Wait()` is a command that tells the thread to wait until another thread uses the `notify()` or `notifyall()` command. The `notify` command ensures the use of that object's and the `notifyall` command ensures the use of all threads needing to use that object. `Interrupt()` interrupts the thread so that the processor can be used by another. `Join()` ensures one thread waits for the other thread's completion.

3.4. Thread Synchronization

3.4.1. Methods of thread synchronization

Synchronization is an advantage and disadvantage of thread creation as multiple threads can have the same memory address, because of this, changes to one thread saved in the same memory address of other threads can modify all the threads in that memory address. However, to avoid this, threads with the same memory addresses or with the need for the same critical resources can be executed contemporarily.

4. Summary

4.1. Summarize points made.

A computer reads different instructions to make each program run. Those different instructions must communicate and work together to ensure the computer functions correctly. As time is one of the only resources that the world does not have enough of, processing speed is one of the most important factors in the computer market. As a processor can only process one task at a time, measures are taken so that processes have many sub-processes inside, these are referred to as threads and are very important. These allow a computer to reading multiple sets of instructions simultaneously in the time an older computer would process a single task.

5. References

- [1] E. E. Brent Jr., R. E. Anderson, "Computer Hardware, Configurations, and Peripherals," Computer Applications in the Social Sciences. Philadelphia: McGraw-Hill, 1990, ch. 2., pp.18.
- [2] English Dictionary and Thesaurus. Glasgow, Scotland: Collins, 2013, pp. 215.
- [3] M. Herlihy, N. Shavit, The Art of Multiprocessor Programming. Waltham: Elsevier, 2012.
- [4] J. Friesen, Java Threads and the Concurrency Utilities. New York: Apress, 2015
- [5] G. Tomolei, Multiprogramming, Multiprocessing, Multitasking and Multithreading[online]. Available: <https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/multiprogramming-multiprocessing-multitasking-multithreading/> 22/01/2018
- [6] Multiprocessing[online]. Available: <https://www.britannica.com/technology/multiprocessing> 22/01/2018
- [7] Operating System – Processes[online]. Available: https://www.tutorialspoint.com/operating_system/os_processes.htm 22/01/2018
- [8] Chapter 2. Interprocess communications[online]. Available: http://menehune.opt.wfu.edu/Kokua/More_SGI/007-2478-008/sgi_html/ch02.html 24/01/2018
- [9] Types of scheduling[online]. Available: <https://www.go4expert.com/articles/types-of-scheduling-t22307/> 24/01/2018

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

58/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

Be careful - this is straight from source, but isn't quoted. Be sure to paraphrase.



Comment 2

Also a bit too close to the original source for comfort.

PAGE 6

PAGE 7

PAGE 8

GRADING FORM: SMU GENERIC_GJ

LARA TERZI

58

RESEARCH-INFORMED LITERATURE

 A good number of references used, but the latter half especially of somewhat questionable academic credibility. You should ideally try to stick to journal articles and similar peer-reviewed sources, distantly followed by non-peer reviewed publications (such as textbooks), with web sources as a last resort. Sources 5, 7 and 9 especially are better avoided. I suspect you've misunderstood how citations should be applied inline in text. You've cited sources you've quoted, but nothing else. You need to quote sources you've paraphrased too. We'll have a chat about this to clarify.

23

KNOWLEDGE AND UNDERSTANDING

 Very good knowledge and understanding displayed - good depth throughout the report.

22

ANALYSIS

 Lack of synthesis again limits your mark in this section. A reasonable conclusion, however. As a comment, your written style is very good, but you could improve how you thread sections together.

5

PRACTICAL APPLICATION AND DEP.

 N/A - no marks assigned to this section.



You write extremely well - good spelling, punctuation and grammar, and a nicely-presented document. I enjoyed reading this work. You might consider including a few figures to help explain key points.

8