

SSMUAE Robert McIntyre

by Robert McIntyre

FILE	2783_ROBERT_MCINTYRE_SSMUAE_ROBERT_MCINTYRE_95675_7139 20619.DOCX (34.76K)		
TIME SUBMITTED	24-JAN-2018 09:00PM (UTC+0000)	WORD COUNT	2658
SUBMISSION ID	82335429	CHARACTER COUNT	16750

**Robert McIntyre
P149064
System Software Multi-User Application Engineering
Research Report
Tim Bashford.**

Table of Contents

Introduction	3
Processes.....	3
Constituents of a process.....	3
Asymmetric vs Symmetric Multiprocessor system	4
Scheduling	4
Inter-Process Communication	4
Synchronisation.....	5
Threads	5
Scheduling.....	6
Thread communication.....	6
Thread Synchronisation	6
Summary	6
References	7

Introduction

Multiprogramming systems are used on both single and multiple CPU computers. They work in similar ways but, there are some differences. Multiprocessing is where there is more than one program is running at a time. [1] This is used in a CPU, which processes instructions while it is running. A multi-programming system is a type of parallel programming. A multi-programming system is where the system needs to be managed to treat all jobs fairly. In the IBM Knowledge Center it states that it is “the simultaneous operation of two or more processors that share the various hardware resources”[2]. A CPU will have many programs loaded into the memory ready to be executed. If the user runs two programs the priority transfers from the first to the second. The first program will be interrupted to free the execution for the second program.

The programs are executed using the fetch, decode and execute cycle which is part of the Von Neumann Architecture. The Von Neumann Architecture is known as a stored-program concept [3], where data and instructions are stored in the memory. There are many registers that you can find in the CPU. Each register will have its own function. Von Neumann is one type of architecture that a CPU might use. Other registers that a CPU uses are Program Counter, Memory Data Register, Memory Address Register and Current Instruction Register. These are all used in the fetch, decode and execute cycle. The program counter contains the address of the next instruction. When data is needed it is copied into the memory address register and the current instruction register and the program counter increments by one ready for the next instruction. The current instruction is then decoded in the control unit and translated into opcode where it is then executed.

There are two different types of multi-processing systems, symmetric and asymmetric. Both use scheduling to prioritise their processes. There are advantages and disadvantages to both. This report contains how processes and threads are used within the computer system. Explaining in detail the methods of scheduling, communication and synchronisation.

Processes

A program is a set of instructions needed to run an application. A process is the execution of a set of instructions that are carried out in a logical order[4]. A process will contain all the necessary information for a program to run[5]. There can several processes associated with a program.

Constituents of a process

A process will always have a changing state[6]. The operating system will store information about the process. The constituents of the process are that:

- Each task will have an identifier linked with the process (Address);
- With the identifier there will be a name of the process entity;
- The state of the process will be identified (Start/Ready/Running/ Waiting/ Ended)
- Resources such as a program counter and stack pointer will be allocated that are required for the process to run.
- The permission authority for each task – whether the user is an admin or not

A multiprocessing system is a computer with two or more CPUs in a system sharing the same resources. Multi-processing is more complex than single processing because the operating system needs to manage the resources needed for all the processes in the system in a timely manner to avoid conflicts [7]. The need for scheduling is important for the computer to be able to multi task. Processes need to be scheduled as some processes have higher priority than others. Multiprocessing systems will produce a higher performance than a single processor.

Asymmetric vs Symmetric Multiprocessor system

There are two different types of multi-processing systems, symmetric and asymmetric multi-processor systems.

Symmetric multi-processing is where multiple CPUs can share the processing workload evenly [8]. This is done automatically where this is up to the operating system to share the resources between the CPU cores. In a symmetric multi-processing system there is two or more CPU with the same architecture. In an asymmetric multi-processor system, you can have a system with two or more CPU with different architectures in the system. In asymmetric multi-processor systems processes are often processed on one processor and applications are processed on the other processor. Symmetric processors are more common in a computer system.

Scheduling

Scheduling is a term given for managing the CPU time so that the CPU is working as efficiently as possible. This will determine which process should be executed at a given time. Scheduling is important for multi-processing as the operating system will have to execute more than one process at a time. There are different states for the processors (ready, running and terminated). When a process starts it's in the ready state. When the process is scheduled the process is in the running state. Then once it is finished with the process is terminated.

Single CPU computer systems had only one processor which made scheduling much easier. However, most modern computers have multiple CPU and are multiprocessor systems.

There are different scheduling algorithms that are suitable for single and multi-processors. These can be described as Non-pre-emptive and Pre-emptive. Non-Pre-emptive is when a process is executed, and no other process can interrupt the execution until the first process is terminated. Examples of non-pre-emptive algorithms are time slice is where the processor gives all the processes equal time. First come first serve is where a process enters a queue, the first one to enter is dealt with first. Shortest Job First is when the CPU looks in a list to see which job can be completed the quickest and completes that process first. Pre-emptive is where processes have priorities. Processes can be interrupted if a new process has a higher priority. An example of a pre-emptive algorithm is Shortest Remaining Time Next. This is when the execution time is compared to the running process and if the new process is shorter the current process is interrupted. [9] There are also multi-level queue scheduling and multi-level feedback scheduling is where there is more than one queue with multiple algorithms. The difference between them are process can move between queues in the multi-level feedback, whereas in the multi-level queue the process must stay within the queue it has entered even when the other queues are empty.

There are issues to be addressed in scheduling for multiprocessor systems. Firstly, the operating system will allocate the processors tasks.[10] The operating system needs to decide which processor should execute a particular set of instructions. It also needs to prioritise the set of tasks allocated to a processor [9].

Inter-Process Communication

Inter-Process Communication is where processes pass information to each other. There are different techniques for the ways for processes to talk to one another. There are many methods for inter-process communication such as Shared memory, pipes, semaphores, message queuing and sockets[11]. Shared memory is where there is a shared space of memory that is used in multiprocessors that all processors can read and write to .[5] Processes share memory so that they share information. [5]. When a process needs to share data, it would often have a shared variable.

Pipes allow information to be sent immediately from one process to another. They are a one-way communication. Therefore, for two way communication this requires two pipes [12][5]. Semaphores are a way for flexibly synchronising processes and applications in the multiprocessor system. Semaphores have a value to indicate the status of a resource that can be read by a process and changed. The process checks the value to see if the resource can be used. If the resource can be used, the process then changes the semaphore value to show other processes that the resource is now busy. If the resource is busy the process waits and checks later. Message queuing is where data is created by processes and placed in a queue, once created the data can be used by any other process.[12] The message is given a specific type when the message is created. Other processes can select messages by type or the order they are taken from the queue [9]. Sockets allow communication between network processes and local processes. Processes can be passed through from the client's computer to the servers. A socket is an endpoint therefore, sockets are needed on each side to allow connection to take place.[9]

Synchronisation

For processes to communicate with each other they need to synchronise. Messages can then be sent and received between processes. Once a process has sent or received a message it needs to state what job is required to do next [9]. Process synchronisation manages all of the processes to stop them interfering with each other while being executed [13]. Process synchronisation is in place to prevent problems such as critical section problems. Critical section is the code uses shared variables that can be shared over many processes. When a process is using the critical section, it cannot be accessed by other processes. The other processes must wait for the process to finish with the critical section. There are different methods for processes to send or receive. The most common versions are blocking and non-blocking messages.[9] Blocking send and receive is when both processes are interrupted until the message is received. Non – blocking send, and Blocking receive is where the sending process does not require acknowledgement for the process to carry on, but the receiving end will wait until the message has been received.

There are four policy conditions for synchronisation. Mutual exclusion is where other processes cannot access the shared resources if a process is using them in its critical section. Hold and wait is where a process is on hold and waiting for resources held by other processes. No pre-emption is where a resource cannot be taken away from a process. Circular wait is where each process is waiting for a resource from the next process in a circular queue. [9] These policies can prevent a process deadlock situation. A deadlock situation is when two or more active processes are waiting for each other to complete an instruction before they can carry on. There are two version of a deadlock situation, deadlock prevention and deadlock avoidance. Deadlock avoidance is a much more efficient way of using the resources because decisions are made dynamically whereas with deadlock prevention there are a set of rules in place.

Threads

Threads are sub versions of processes. A process will contain many threads. They are a point of execution and handle a small set of instructions that cannot be run independently [14] Threads share their own memory space and consist of a program counter and a stack [9]. All threads can communicate with other threads. A thread is a series of instructions that are executed in order[15].

The lightweight processes are a layer between the kernel and user level threads. There is always one kernel level to every lightweight process[9]. The user level threads however seek the next freely available lightweight processes rather than using the same one every time. The kernel level threads

tell the user level threads how many lightweight processes should be running [16]. The kernel does not know the existence of user level threads because these are managed by the user thread library.

Scheduling

The responsibility for thread scheduling is partly up to the programmer to program the application on the threads and it is also up to the operating system to schedule threads accordingly. Threads need to be scheduled to make the process run efficiently. Thread scheduling is very similar to process scheduling. They both have a priority based scheduling. This is when there are two threads the first one is higher ranked than the second one. There are different types of algorithms to order when threads execute their instructions. Time slice is used to give time equally to all threads. There are other algorithms that are more efficient as some priority for threads is needed.

Many of the scheduling algorithms are used in both process and thread scheduling. Threads only use a priority system such as time slice whereas processes can also use non-priority scheduling such as first come first serve.

Thread communication

The programmer can decide how the threads communicate with each other but cannot tell when the threads are scheduled. This is done by the operating system. There are many ways for threads to communicate with each other. One way of doing this is that the threads are halted to allow another thread to update data in the shared spaces.[17]

Thread Synchronisation

Thread synchronisation is where there are two or more threads that are controlled while accessing the shared memory. This is needed because only one thread can access the shared memory at a time [18]. There are methods to prevent threads interfering when accessing the shared memory space. Synchronisation manages which threads can access the shared memory space. Threads are ranked by priority, the highest priority thread is executed first.[19] The thread is only interrupted if there is another thread with higher priority or the thread runs out of its allocated processing time. If there are threads with the same priority the scheduler chooses which thread to execute.

Summary

Multiprocessor systems use threads and processes in order to execute multiple programs in a system. These threads and processes need algorithms and methods to make efficient use of the CPU. From the information gathered there are many similarities between threads and processes. They communicate with each other and use many of the same algorithms to schedule within the CPU. One difference is that threads use priority algorithms whereas processes can use non-priority algorithms as well. Synchronisation is an important part of how threads and processes are executed because they both use shared memory that is managed within the CPU. In summary, the report covers important information regarding how processes and threads work.

References

- [1] D. Thakur, "What are Multi Tasking, Multi Programming and Multi Threading." [Online]. Available: <http://ecomputernotes.com/fundamental/disk-operating-system/what-are-multi-tasking-multi-programming-and-multi-threading>. [Accessed: 20-Dec-2017].
- [2] "IBM Knowledge Center - Multiprogramming and multiprocessing," *IBM Knowledge Center*. [Online]. Available: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconcept_s_75.htm. [Accessed: 20-Dec-2017].
- [3] S. Srikanth, T. M. Conte, E. P. DeBenedictis, and J. Cook, "The Superstrider Architecture: Integrating Logic and Memory Towards Non-Von Neumann Computing," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, 2017, pp. 1–8.
- [4] "Unix / Linux Processes Management." [Online]. Available: <https://www.tutorialspoint.com/unix/unix-processes.htm>. [Accessed: 31-Dec-2017].
- [5] A. S. Tanenbaum, *Modern operating systems*, 3rd ed. Pearson-Prentice International, 2009.
- [6] "Process definition by The Linux Information Project (LINFO)." [Online]. Available: <http://www.linfo.org/process.html>. [Accessed: 20-Dec-2017].
- [7] Marcraft International, *Server+ Certification Exam Cram (Exam SKO-002)*. Pearson IT Certification, 2005.
- [8] L. Kent, "Symmetric multiprocessing," *Infoworld Media Gr.*, vol. 15, no. 47, p. 82, 1993.
- [9] W. Stallings, *Operating systems : internals and design principles*, 5th ed. Pearson Prentice Hall, 2005.
- [10] R. I. Davis and A. Burns, "35 A Survey of Hard Real-Time Scheduling for Multiprocessor Systems," *Article*, vol. 43, no. 35, 2011.
- [11] "What is interprocess communication (IPC)? - Definition from WhatIs.com." [Online]. Available: <http://whatis.techtarget.com/definition/interprocess-communication-IPC>. [Accessed: 01-Jan-2018].
- [12] "What is pipe? - Definition from WhatIs.com." [Online]. Available: <http://whatis.techtarget.com/definition/pipe>. [Accessed: 12-Jan-2018].
- [13] "Process Synchronization in OS | Operating System Tutorial | Studytonight." [Online]. Available: <http://www.studytonight.com/operating-system/process-synchronization>. [Accessed: 11-Jan-2018].
- [14] "16.6. multiprocessing — Process-based 'threading' interface — Python 2.7.14 documentation." [Online]. Available: <https://docs.python.org/2/library/multiprocessing.html>. [Accessed: 31-Dec-2017].
- [15] "Ask Ars: what is a CPU thread? | Ars Technica," *4/12/2011*. [Online]. Available: <https://arstechnica.com/information-technology/2011/04/ask-ars-what-is-a-cpu-thread/>. [Accessed: 13-Jan-2018].
- [16] "Lightweight Processes (Multithreaded Programming Guide)," *Oracle Corporation*, 2010. [Online]. Available: <https://docs.oracle.com/cd/E19455-01/806-5257/guide-60747/index.html>. [Accessed: 14-Jan-2018].
- [17] J. English and John, "Multithreading in C++," *ACM SIGPLAN Not.*, vol. 30, no. 4, pp. 21–28,

Apr. 1995.

- [18] "What is synchronization and why is it important in Java?" [Online]. Available: <http://www.careerride.com/Java-what-is-synchronization.aspx>. [Accessed: 13-Jan-2018].
- [19] "Thread Priority," *Southampton university*, 1999. [Online]. Available: <http://journals.ecs.soton.ac.uk/java/tutorial/java/threads/priority.html>. [Accessed: 17-Jan-2018].

FINAL GRADE

GENERAL COMMENTS

Instructor

66/100

PAGE 1

PAGE 2

PAGE 3



Comment 1

Comma to the other side of the "but".



Comment 2

A little close to the source material. More paraphrasing and an inline citation needed.

PAGE 4



Comment 3

No need for this citation - because you use it again in the next sentence.

PAGE 5

PAGE 6

PAGE 7

PAGE 8

GRADING FORM: SMU GENERIC_GJ

ROBERT MCINTYRE

66

RESEARCH-INFORMED LITERATURE

 An excellent number of sources used, and cited mostly correctly (refer to in-line comments for more on this). There are, however, quite a number of web sources with poor academic credibility. Try to target more journals in future, even if it's at a reduction in the number of sources overall. I'd prefer to see 10 academically credible sources than 20, some of which are far less so. Some excellent use of citations, though a tad sparse in one or two places, e.g. the first half of the "scheduling" section.

30

KNOWLEDGE AND UNDERSTANDING

 Some very good knowledge and understanding displayed - good detail included.

23

ANALYSIS

 Good analysis, including synthesis. You could consider backing up information with more than one source where appropriate - e.g. [1][2][3] This is something which'll become increasingly important in the 3rd year of the BSc.

7

PRACTICAL APPLICATION AND DEP.

 N/A - no marks assigned to this section.

SKILLS FOR PROFESSIONAL PRAC.

 Generally good spelling, punctuation and grammar -

6

a few small errors detailed in the in-line comments.
A generally well-presented document. A few
diagrams would actually help to illustrate key points
in places.