ACHIEVER'S EDUCATION CONSULTANT
08141710097
SUMMARY OF CIT211 WITH PAST QUESTIONS

ACHIEVER'S EDUCATION CONSULTANT 08141710097

CIT 211 – Introduction to Operating Systems is a three credit unit course of twenty-two units. This course is divided into six modules. The first module deals with the basic introduction to the concept of Operating Systems such as definition and functions of operating system, history and evolution of operating system. The second module treats, extensively, the various types of operating system.

The third module deals with concept of process management and discusses the concepts of Cooperating Processes, Threads, and CPU Scheduling. The fourth module discusses process synchronization issues such as Race Condition, Mutual Exclusion, Critical Section Problem, and other Classic Problems of Synchronization. The fifth module treats deadlock issues such as deadlock Characterization and methods for dealing with deadlocks. The last i.e. the sixth module discusses memory management functions of the operating system and issues such as memory management algorithms like paging, segmentation, contiguous memory allocation with their peculiar features were discussed.

It makes available the steps and tools that will enable you to make proper and accurate decision on designs issues and the necessary algorithms for a particular computing environment. Certain objectives have been set out to ensure that the course achieves its aims. In the course of the study, you will need to confirm, at the end of each unit, if you have met the objectives set at the beginning of each unit.

Describe the various CPU scheduling evaluation algorithms.

Describe some of the methods for handling deadlocks.

This course is designed to cover approximately seventeen weeks, and it will require your devoted attention. You should do the exercises in the Tutor-Marked Assignments and submit to your tutors.

Textbooks and References

«Operating System Structures to Support Security and Reliable Software». The logical design of Operating systems.

Assignments File

The selfassessment exercises will enable you monitor your performance by yourself, while the TutorMarked Assignment is a supervised assignment. The assignments take a certain percentage

of your total score in this course. The Tutor-Marked Assignments will be assessed by your tutor within a specified period. The examination at the end of this course will aim at determining the level of mastery of the subject matter.

This course includes twelve Tutor-Marked Assignments and each must be done and submitted accordingly. Your best scores however, will be recorded for you. Be sure to send these assignments to your tutor before the deadline to avoid loss of marks.

Presentation Schedule

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

The Presentation Schedule included in your course materials gives you the important dates for the completion of tutor marked assignments and attending tutorials. Remember, you are required to submit all your assignments by the due date. You should guard against lagging behind in your work. In tackling the assignments, you are expected to apply information and knowledge acquired during this course.

The assignments must be submitted to your tutor for formal assessment in accordance with the deadlines stated in the Assignment File. The work you submit to your tutor for assessment will count for 30% of your total course mark. At the end of the course, you will need to sit for a final three-hour examination.

Tutor Marked Assignments

You need to submit all the assignments. The total marks for the best three assignments will be 30% of your total course mark. Assignment questions for the units in this course are contained in the Assignment File. When you have completed each assignment, send it together with form to your tutor.

Make sure that each assignment reaches your tutor on or before the deadline given. If, however, you cannot complete your work on time, contact your tutor before the assignment is done to discuss the possibility of an extension. Examination and Grading The final examination for the course will carry 70% percentage of the total marks available for this course. The examination will cover every aspect of the course, so you are advised to revise all your corrected assignments before the examination.

This means that you teach yourself and that you learn, as your learning capabilities would allow. It also means that you are in a

ACHIEVER'S EDUCATIONAL

better position to determine and to ascertain the what, the how, and the when of your language learning. The course units are similarly designed with the introduction following the table of contents, then a set of objectives and then the dialogue and so on. The objectives guide you as you go through the units to ascertain your knowledge of the required terms and expressions.

Course Marking Scheme This table shows how the actual course marking is broken down.

[...]

In distance learning the study units replace the university lecturer. Think of it as reading the lecture instead of listening to a lecturer. In the same way that a lecturer might set you some reading to do, the study units tell you when to read your set books or other material. Just as a lecturer might give you an in-class exercise, your study units provide exercises for you to do at appropriate points.

Next is a set of learning objectives. These objectives enable you know what you should be able to do by the time you have completed the unit. You should use these objectives to guide your study. When you have finished the units you must go back and check whether you have achieved the objectives.

If you make a habit of doing this you will significantly improve your chances of passing the course. Remember that your tutor's job is to assist you. When you need help, don't hesitate to call and ask your tutor to provide it. Note the time you are expected to spend on each unit and how the assignments relate to the units.

Whatever method you chose to use, you should decide on it and write in your own dates for working on each unit. Once you have created your own study schedule, do everything you can to stick to it. You will almost always need both the study unit you are working on and one of your set of books on your desk at the same time. The content of the unit itself has been arranged to provide a sequence for you to follow.

As you work through the unit you will be instructed to read sections from your set books or other articles. Use the unit to guide your reading. Review the objectives for each study unit to confirm that you have achieved them. When you are confident that you have achieved a unit's objectives, you can then start on the next unit.

Proceed unit by unit through the course and try to pace your study so that you keep yourself on schedule. When you have submitted an assignment to your tutor for marking, do not wait for its return before starting on the next unit. When the assignment is returned, pay particular attention to your tutor's comments, both on the tutor-marked assignment form and also written on the assignment. Check that you have achieved the unit objectives and the course objectives .

Tutors and Tutorials

There are 12 hours of tutorials provided in support of this course. You will be notified of the dates, times and location of these tutorials, together with the name and phone number of your tutor, as soon as you are allocated a tutorial group. Your tutor will mark and comment on your assignments, keep a close watch on your progress and on any difficulties you might encounter and provide assistance to you during the course. You must mail or submit your tutor-marked assignments to your tutor well before the due date .

They will be marked by your tutor and returned to you as soon as possible. The following might be circumstances in which you would find help necessary. You should try your best to attend the tutorials. You can raise any problem encountered in the course of your study.

You will learn a lot from participating in discussions actively. The content of the course material was planned and written to ensure that you acquire the proper knowledge and skills for the appropriate situations. Real-life situations have been created to enable you identify with and create some of your own. The essence is to help you in acquiring the necessary knowledge and competence by equipping you with the necessary tools to accomplish this.

We wish you success with the course and hope that you will find it both interesting and useful.

ACHIEVER'S EDUCATIONAL CONSULT 08[illegible]710097

CIT 211

Operating system is basically an application program that serves as an interface to coordinate different resources of computer. An operating system processes raw system and user input and responds by allocating and managing tasks and internal system resources as a service to users and programs of the system.

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

But simply put, an can be defined as a suite of programs implemented either in software or firmware or both that makes the hardware usable. These applications have had command line interpreters as a basic user interface, but more recently have been implemented as a graphical user interface for ease of operation.

OS can further be described by what they do i.e. by their functions, goals and objectives.

Convenience for the User

When there is no OS, users of computer system will need to write machine-level program in order to manipulate the hardware. With OS, users can now easily and conveniently use the computer with no stress of directly programming the hardware. OS provide a convenient interface for using the computer system.

Efficiency

An OS allows computer system resources to be used in an efficient manner. This particularly important for large shared multi-user systems which are usually expensive.

Evolutionary Capabilities

An OS should be constructed in such a way that it permits the effective development, testing and introduction of new system functions without interfering with its service. OS can be viewed from the perspective of what they are. These views are diverse depending on the particular view point of a user. But some of these views are discussed below.

OS as a User/Computer Interface

ACHIEVER'S EDUCATIONAL CONSULTANT 08[illegible]710097

A computer system can be viewed as a layered or hierarchical structure consisting of the hardware, operating system, utilities, application programs and users. The users of application programs are called the end-users and are generally not concerned with the computer's architecture. The application is developed by the application programmer who uses a programming language and a language translator. A set of programs called the utilities is provided to assist the programmer in program creation, file management and the control of Input/Output devices.

The most important system program, operating system masks the details of the hardware from the programmer and provides a convenient interface for using the system. it acts as a mediator, making it easier for the programmer and for application programs to access and use the available services and facilities.

Text editor

ACHIEVER'S EDUCATIONAL CONSULTANT 08[illegible]10097

A computer system has a set of resources for the movement, storage and processing of data. The OS is responsible for managing these resources. The key difference is in the purpose of the program. The OS directs the processor in the use of the other system resources and in the timing of its execution of other programs.

The processor, in order to do any of these things, must cease execution of the OS program to execute other programs. Thus, the OS relinquishes control long enough to prepare the processor to do the next piece of work. The portion of the OS that is always in main memory is called the kernel or nucleus and it contains the most frequently used functions in the OS. The allocation of this resource is controlled jointly by the OS and the memory management hardware in the processor.

The services provided by the OS can be categorised into two

The program must be able to end its execution either normally or abnormally. In case of abnormal end to a program, it must indicate error. The OS takes care of the details so that the programmer can think in terms of reads and writes. In the case of the multi-user system, the OS must provide protection mechanisms to control access to the files.

It can take place between processes executing on the same computer. It can take place between processes executing on different computer systems that are linked by a computer network.

ACHIEVER'S EDUCATIONAL CONSULT 0814 3097

Communications may be implemented via a shared memory or by a technique of message passing in which packets of information are moved between processes by the OS. In each case, the OS must make a response that makes the less impact on running applications.

The response may range from ending the program that caused the error, retrying the operation or simply reporting the error to the application.

Efficiency of System: Single and Multi-User

Some may have general request and release codes. This statistics may be available tool for researchers who want to reconfigure the system to improve computing services. This is because such a terminal may allow the user to view several applications at the same time through 'windows' on the screen.

[...]

As you have learnt in this unit the OS is very important software in the computer system that provides a variety of services to the applications running on the system and the user. It evolved over time as you are going to learn in the next unit.

Enumerate the goals and functions of the OS

Introduction to Operating Systems. Zeroth Generation First Generation Second Generation Third Generation Fourth Generation Operating system was absent in the first commercial form of electronic computer launched in 1940's. Rows of mechanical switches were used to enter programs. At that time, programming languages were not in use.

Naturally, there was hardly any idea about operating system. The user had sole use of the machine and would arrive armed with program and data, often on punched paper tape. Programs could generally be debugged via a front panel using switches and lights.

Later machines came with libraries of support code, which would be linked to the users'

Program to assist in operations such as input and output. This was the genesis of the modern-day operating system. To see what operating systems are and what operating systems do, let us consider how they have developed over the last 30 years. By tracing that evolution we can identify the common elements of operating systems, and see how, and why they developed as they are now.

Operating systems have been evolving over the years. The Zeroth Generation The term zeroth generation is used to refer to the period when there was no OS. Perhaps the most significant of these early computers was the EDVAC, developed in 1944-46 by John von

the idea of the stored program and serial execution of instructions. The development of EDVAC set the stage for the evolution of commercial computing and operating system software.

The actual operation of these early computers took place without the benefit of an operating system. Early programs were written in machine language and each contained code for initiating operation of the computer itself. The individual user was responsible for all machine set up and operation, and subsequent clean-up and preparation for the next user. This system was clearly inefficient and depended on the varying competencies of the individual programmer as operators.

The First Generation, 1951-1956

The first generation marked the beginning of commercial computing. Operation continued without the benefit of an operating system for a time. There was no provision for moving a program to different location in storage for any reason. Similarly, a program bound to specific devices could not be run at all if any of these devices were busy or broken.

The OS consisted of a permanently resident kernel in main storage, and a job scheduler and a number of utility programs kept in secondary storage. User application programs were preceded by control or specification cards which informed the OS of what system resources were needed to run a particular application. The systems were designed to be operated as batch processing system. The first generation saw the evolution from hands-on operation to closed shop operation to the development of mono-programmed operating systems.

The most obvious was the inefficient use of system resources, which was most evident when the CPU waited while the relatively slower, mechanical I/O devices were reading or writing program data. In addition, system protection was a problem because the operating system kernel was not protected from being overwritten by an erroneous application program. Moreover, other user programs in the queue were not protected from destruction by executing programs.

The Second Generation, 1956-1964

The second generation of computer hardware was most not ably characterized by transistors replacing vacuum tubes as the hardware component technology. In addition, some very important changes in hardware and software architectures occurred during this period. For the most part, computer systems remained card and tape-oriented systems. Significant use of random access devices, that is, disks, did not appear until towards the end of the second generation.

The most significant innovations addressed the problem of excessive central processor delay due to waiting for input/output operations. Recall that programs were executed by processing the machine instructions in a strictly sequential order. As a result, the CPU, with its high speed electronic component, was often forced to wait for completion of I/O operations which involved mechanical devices that were order of magnitude slower. This problem led to the introduction of the data channel, an integral, special-purpose computer with its own instruction set, registers, and control unit designed to process input/output operations separate and asynchronously from the operation of the computer's main CPU near the end of the first generation, and its widespread adoption in the second generation.

The data channel allowed some I/O to be buffered. That is, a program's input data could be read «ahead» from data cards or tape into a special block of memory called a buffer. Then, when the user's program came to an input statement, the data could be transferred from the buffer locations at the faster main memory access speed rather than the slower I/O device speed. Similarly, a program's output could be written another buffer and later moved from the buffer to the printer, tape, or card punch.

What made this all work was the data channel's ability to work asynchronously and concurrently with the main processor. Thus, the slower mechanical I/O could be happening concurrently with main program processing. This process was called I/O overlap. This provided communication from the CPU to the data channel to initiate an I/O operation.

It remained for the channel to communicate to the CPU such events as data errors and the completion of a transmission. At first, this communication was handled by polling. The CPU stopped its work periodically and polled the channel to determine if there were any message. Polling was obviously inefficient and led to another significant innovation of the second generation - the interrupt.

These hardware developments led to enhancements of the operating system. Initially, this was accomplished by means of one or more physically separate small satellite computers.

ACHIEVER'S EDUCATIONAL CONSULTANCY

Toward the end of this period, as random access devices became available, tape-oriented operating system began to be replaced by disk-oriented systems. With the more sophisticated disk hardware and the operating system supporting a greater portion of the programmer's work, the computer system that users saw was more and more removed from the actual hardware - users saw a virtual machine.

The second generation was a period of intense operating system development. Also it was the period for sequential batch processing. But the sequential processing of one job at a time remained a significant limitation. Thus, there continued to be low CPU utilization for I/O bound jobs and low I/O device utilization for CPU bound jobs.

This was a major concern, since computers were still very large and expensive machines. Researchers began to experiment with multiprogramming and multiprocessing in their computing services called the time-sharing system. A noteworthy example is the compatible Time Sharing System , developed at MIT during the early 1960s.

The third generation officially began in April 1964 with IBM's announcement of its

Operating system development continued with the introduction and widespread adoption of multiprogramming. Operation of spooling system was, as before, transparent to the computer user who perceived input as coming directly from the cards and output going directly to the printer. The idea of taking fuller advantage of the computer's data channel I/O capabilities continued to develop. This concurrent operation of three programs required that additional features and complexity be added to the operating system.

Second, since the CPU was to be shared by the user

ACHIEVER'S EDUCATIONAL

program, the system reader, and the system writer, some processor allocation rule or policy was needed. This rule, called the static priority rule with preemption, was implemented in the operating system as a system dispatcher program. The spooling operating system, in fact, had multiprogramming since more than one program was resident in main storage at the same time. Later this basic idea of multiprogramming was extended to include more than one active user program in memory at time.

The advent of large scale multiprogramming was made possible by several important hardware innovations. The first was the widespread availability of large capacity, high speed disk units to accommodate the spooled input streams and the memory overflow together with the maintenance of several concurrently active program in execution. The second was relocation hardware which facilitated the moving of blocks of code within memory without an undue overhead penalty. Some of these hardware innovations involved extensions to the interrupt system in order to handle a variety of external conditions such as program malfunctions, storage protection violations, and machine checks in addition to I/O interrupts.

In addition, the interrupt system became the technique for the user program to request services from the operating system kernel. In this environment, several terminals, sometimes up to 200 of them, were attached to a central computer. Users at their terminals, «logged in» to the central system, and worked interactively with the system. The system's apparent concurrency was enabled by the multiprogramming operating system.

Users shared not only the system' hardware but also its software resources and file system disk space.

ACHIEVER'S EDUCATIONAL CONSULTANT 0097

The third generation was an exciting time, indeed, for the development of both computer hardware and the accompanying operating system. The Fourth Generation, 1979 - Present The fourth generation is characterized by the appearance of the personal computer and the workstation. Miniaturization of electronic circuits and components continued and large scale integration , the component technology of the third generation, was replaced by very large scale integration , which characterizes the fourth generation.

The operating system that control these desktop machines have brought us back in a full circle, to the open shop type of environment where each user occupies an entire computer for the duration of a job's execution. In this unit, you have been introduced to the brief history and evolution of operating system.

ACHIEVER'S EDUCATIONAL CONSULTANT 08142710097

Since you are going to be interacting with the computer machine in your day-to-day activities as a computer user or professional, it is necessary to have the basic knowledge of OS. In the next unit, you are going to be introduced to the various types of OS in the market today based on several criteria.

Tutor Marked Assignment.

[...]

In the previous unit we discussed the history and evolution of the operating system. In this unit you will be taken through the core component of the operating system which is the kernel. As a basic component of an operating system, a kernel provides the lowest-level abstraction layer for the resources that applications must control to perform their function. These tasks are done differently by different kernels, depending on their design and implementation.

The existence of a kernel is a natural consequence of designing a computer system as a series of abstraction layers, each relying on the functions of layers beneath itself. The kernel, from this viewpoint, is simply the name given to the lowest level of abstraction that is implemented in software. In most cases, the boot loader starts executing the kernel in supervisor mode, The kernel then initializes itself and starts the first process. Kernel development is considered one of the most complex and difficult tasks in programming.

Its central position in an operating system implies the necessity for good performance, which defines the kernel as a critical piece of software and makes its correct design and implementation difficult. For various reasons, a kernel might not even be able to use the abstraction mechanisms it provides to other software. Such reasons include memory management concerns and lack of reentrancy, thus making its development even more difficult for software engineers. A kernel will usually provide features for low-level scheduling of processes, Inter-process communication, process synchronization, context switch, manipulation of process control blocks, interrupt handling, process creation and destruction, process suspension and resumption .

Kernel Basic Responsibilities The kernel's primary purpose is to manage the computer's resources and allow other programs to run and use these resources. This is the most central part of a computer system, responsible for running or executing programs on it. The kernel takes responsibility for deciding at any time which of the many running programs should be allocated to the processor or processors The computer's memory. Memory is used to store both program instructions and data.

Typically, both need to be present in memory in order for a program to execute. Often multiple programs will want access to memory, frequently demanding more memory than the computer has available. This is discussed in module 3. Finally, a kernel must provide running programs with a method to make requests to access these facilities.

Process Management The main task of a kernel is to allow the execution of applications and support them with features such as hardware abstractions. Typically, the number of processes a system may run simultaneously is equal to the number of CPUs installed. Other systems may

provide cooperative multitasking, where each process is allowed to run uninterrupted until it makes a special request that tells the kernel it may switch to another process. Such requests are known as «yielding», and typically occur in response to requests for interprocess communication, or for waiting for an event to occur.

Older versions of Windows and Mac OS both used co-operative multitasking but switched to pre-emptive schemes as the power of the computers to which they were targeted grew. This typically means providing synchronization mechanisms to ensure that no two processors attempt to modify the same data at the same time. Memory Management The kernel has full access to the system's memory and must allow processes to access this memory safely as they require it. Virtual addressing allows the kernel to make a given physical address appear to be another address, the virtual address.

This allows every program to behave as if it is the only one running and thus prevents applications from crashing each other. On many systems, a program's virtual address may refer to data which is not currently in memory. This scheme is generally known as demand paging.

ACHIEVER'S EDUCATIONAL CONSULTA 0817 0097

Virtual addressing also allows creation of virtual partitions of memory in two disjointed areas, one being reserved for the kernel and the other for the applications.

The applications are not permitted by the processor to address kernel memory, thus preventing an application from damaging the running kernel. This fundamental partition of memory space has contributed much to current designs of actual general-purpose kernels and is almost universal in such systems, although some research kernels take other approaches.

Device Management

A kernel must maintain a list of available devices. As device management is a very OS-specific topic, these drivers are handled differently by each kind of kernel design, but in every case, the kernel has to provide the I/O to allow drivers to physically access their devices through some port or memory location. Very important decisions have to be made when designing the device management system, as in some designs accesses may involve context switches, making the operation very CPU-intensive and easily causing a significant performance overhead.

System Calls

This is implemented differently by each kernel, but most provide a C library or an API, which in turn invoke the related kernel functions. A call gate is a special address which the kernel has added to a list stored in kernel memory and which the processor knows the location of.

ACHIEVER'S EDUCATIONAL CONSULTA 11710097

When the processor detects a call to that location, it instead redirects to the target location without causing an access violation. System call instructions have been added to recent models of x86 processors, however, and some operating systems for PCs make use of them when available.

An application that makes large numbers of requests but does not need to wait for the result of each may add details of requests to an area of memory that the kernel periodically scans to find requests.

Fault Tolerance

Both approaches typically require some hardware or firmware support to be operable and efficient. When an application

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

needs to access an object protected by a capability, it performs a system call and the kernel performs the access for it. Approaches where protection mechanism are not firmware supported but are instead simulated at higher levels , are possible, but there are performance implications. Lack of hardware support may not be an issue, however, for systems that choose to use language-based protection.

An important kernel design decision is the choice of the abstraction levels where the security mechanisms and policies should be implemented. One approach is to use firmware and kernel support for fault tolerance , and build the security policy for malicious behaviour on top of that , delegating some responsibility to the compiler. Approaches that delegate enforcement of security policy to the compiler and/or the application level are often called language-based security.

Hardware-Based Protection or Language-Based Protection

Typical computer systems today use hardware-enforced rules about what programs are allowed to access what data. In systems that lack support for capabilities, processes are isolated from each other by using separate address spaces. Calls from user processes into the kernel are regulated by requiring them to use one of the above-described system call methods. An alternative approach is to use language-based protection.

In a language-based protection system, the kernel will only allow code to execute that has been produced by a trusted language compiler.

Advantages of this approach include

Changes to the protection scheme do not require new hardware.

ACHIEVER'S EDUCATIONAL CONSULTANT 081 10097

Disadvantages include

Longer application start up time. On traditional systems, applications frequently perform operations that are not type safe. Such operations cannot be permitted in a language-based protection system, which means that applications may need to be rewritten and may, in some cases, lose performance.

Examples of systems with language-based protection include JX and Microsoft's

Edsger Dijkstra proved that from a logical point of view, atomic lock and unlock operations operating on binary semaphores are sufficient primitives to express any functionality of process cooperation. However this approach is generally held to be lacking in terms of safety and efficiency, whereas a message passing approach is more flexible. The idea of a kernel where I/O devices are handled uniformly with other processes, as parallel co-operating processes, was first proposed and implemented by Brinch Hansen. In Hansen's description of this, the «common» processes are called internal processes, while the I/O devices are called external processes.

Kernel-Wide Design Approaches

Naturally, the above listed tasks and features can be provided in many ways that differ from each other in design and implementation. Most kernels do not fit exactly into one of these categories, but are rather found in between these two designs. These are called hybrid kernels. More exotic designs such as nanokernels and exokernels are available, but are seldom used for production systems.

A monolithic kernel instead tends to include many policies, therefore restricting the rest of the system to rely on them. In a monolithic kernel, all OS services run along with the main kernel thread, thus also residing in the same memory area. Some developers maintain that monolithic systems are easier to design and implement than other solutions, and are extremely efficient if well-written. The main disadvantages of monolithic kernels are the dependencies between system components - a bug in a device driver might crash the entire system - and the fact that large kernels can become very difficult to maintain.

IPC

The microkernel approach consists of defining a simple abstraction over the hardware, with a set of primitives or system calls to implement minimal OS services such as memory management, multitasking, and inter-process communication. Other services, including those normally provided by the kernel such as networking, are implemented in user-space programs, referred to as servers. Microkernels are easier to maintain than monolithic kernels, but the large number of system calls and context switches might slow down the system because they typically generate more overhead than plain function calls. This is due in large part to the overhead of moving in and out of the kernel, a context switch, to move data between the various applications and servers.

By the mid-1990s, most researchers had abandoned the belief that careful tuning could reduce

this overhead dramatically, but recently, newer microkernels, optimized for performance, such as L4 and K42 have addressed these problems. This is mitigated to some degree by perfecting the virtual memory system, but not all computer architectures have virtual memory support. Due to the problems that monolithic kernels pose, they were considered obsolete by the early 1990s. As a result, the design of Linux using a monolithic kernel rather than a microkernel was the topic of a famous flame war between Linus Torvalds and Andrew Tanenbaum.

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

microkernels offer little advantage over correct code. Microkernels are often used in embedded robotic or medical computers where crash tolerance is important and most of the OS components reside in their own private, protected memory space. This is impossible with monolithic kernels, even with modern module-loading ones. As in a monolithic kernel, there is no performance overhead associated with microkernel message passing and context switching between kernel and user mode.

Also, as with monolithic kernels, there are none of the benefits of having services in user space.

Nanokernels The term is sometimes used informally to refer to a very light-weight microkernel, such as L

Monolithic kernels generally suffer from a considerably bad security architecture because an inaccurate and insecure part directly affects the whole operating system.

Nanokernels Versus microkernels

A nanokernel in contrast has essentially none of those, therefore nanokernels are not independently executable operating systems, which is why they are not an operating system kernel in the traditional sense.

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

Nevertheless this significant difference potentiates extremely powerful techniques like multi scheduling or operating system emulation. An exokernel is a type of kernel that does not abstract hardware into theoretical models. In this unit you have been taken through the concept of kernel and its importance in operating systems design.

Also, its various responsibilities in the computer environment had been deeply discussed not leaving behind the kernel design issues and trade-offs. Most early computers operated this way during the 1950s and early 1960s, which were reset and reloaded between the execution of different programs. Eventually, small ancillary programs such as program loaders and

ACHIEVER'S EDUCATIONAL CONSULTANT 08141711097

debuggers were left in memory between runs, or loaded from ROM. As these were developed, they formed the basis of what became early operating system kernels.

The «bare metal» approach is still used today on some video game consoles and embedded systems, but in general, newer computers use modern operating systems and kernels.

[...]

In the last unit you have been introduced to the concept and history of operating system and how it evolved with each discovery and improvement in the technology of computer architecture. The types of computer they control and the sort of applications they support The nature of interaction that takes place between the computer user and his/her program during its processing.

o Categorise operating systems based on various criteria

They are used to control machinery, scientific instruments and industrial systems. An RTOS typically has very little user-interface capability, and no end-user utilities, since the system will be a sealed box when delivered for use. A very important part of an RTOS is managing the resources of the computer so that a particular operation executes in precisely the same amount of time every time it occurs. RTOS can be hard or soft.

A hard RTOS guarantees that critical tasks are performed on time. This is the type of operating system most people use on their desktop and laptop computers today. Windows 98 and the Mac O.S. are both examples of an operating system that will let a single user have several programs in operation at the same time. A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously.

ACHIEVER'S EDUCATION

requirements of the various users are balanced, and that each of the programs they are using has sufficient and separate resources so that a problem with one user does not affect the entire community of users.

Multi-User Operating Systems

MVS, are examples of multi-user operating systems. In a batch processing operating system environment, users submit jobs to a central place where these jobs are collected into a batch, and subsequently placed on an input queue at the computer where they will be run. In this case, the user has no interaction with the job during its processing, and the computer's response time is the turnaround time . The user, in this environment, has nearly full interaction with the program during its execution, and the computer's response time may be expected to be no more than a few second.

The systems, in this case, are designed to be interrupted by external signal that require the immediate attention of the computer system. In fact, many computer operating systems are hybrids, providing for more than one of these types of computing service simultaneously. A multiprogramming operating system is a system that allows more than one active user program to be stored in main memory simultaneously. A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit.

The term multiprocessing is generally used to refer to large computer hardware complexes found in major scientific or commercial applications. A networked computing system is a collection of physical interconnected computers. Each machine runs its own local operating system and has its own user . Network operating systems are designed with more complex functional capabilities.

Network operating systems are not fundamentally different from single processor operating systems. This unit has taken you through some of the various classifications of OS we have based on different criteria. In the next unit we will be discussing the disk operating system. Distinguish between Network OS and Distributed OS.

How is a soft RTOS different from hard RTOS

The previous unit introduced the various types of OS based on different criteria. In this unit, you will be taken through the disk operating system and its various characteristics and examples. This component was only loaded if needed. Otherwise, disk-access would be limited to low-level operations such as reading and writing disks at the sector-level.

In some cases, the disk operating system component was known as DOS. On the PC compatible platform, an entire family of operating systems was called DOS. 8080 based machines from IMSAI, MITS , Cromemco, North Star, etc used the CP/M-80 disk operating system.

The history of Microsoft disk operating system is closely linked to the IBM

Towards the end of the 1970's, a number of PCs appeared on the market, based on 8-bit microprocessor chips such as Intel 8080. IBM decided to enter this market and wisely opted for a 16-bit microprocessor, the Intel 8088. At that time, CP/M dominated the market. In 1979, a small company Seattle the Computer Products developed its own OS, 86-DOS to test some of its Intel based products .

DOS and in collaboration with Microsoft developed a commercial product. Both of which improved disk file performance.

4 MB floppies

Multiple 32MB disk partitions Support for Expanded Memory system The DOS operating system for the Apple Computer's Apple II family of computers. Usually it was called Apple DOS to distinguish it from MS-DOS. Commodore DOS, which was used by 8-bit Commodore computers. Unlike most other DOS systems, it was integrated into the disk drives, not loaded into the computer's own memory.

The Atari OS only offered low-level disk-access, so an extra layer called DOS was booted off a floppy that offered higher level functions such as filesystems. Disc Filing System This was an optional component for the BBC Micro, offered as a kit with a disk controller chip, a ROM chip, and a handful of logic chips, to be installed inside the computer.

GDOS and G+DOS, for the +D and DISCiPLE disk interfaces for the ZX

PCs type hardware using the Intel CPUs or their compatible cousins from other makers. Any DOS in this family is usually just referred to as DOS. The original was licensed to IBM by Microsoft, and marketed by them as «PC-DOS». When Microsoft licenced it to other hardware manufacturers, it was called MS-DOS.

Digital Research produced a compatible variant known as «DR-DOS», which was eventually taken over by Novell. «OpenDOS» for a while after the relevant division of Novell was sold to Caldera International, now called SCO. There is also a free version named «FreeDOS». This unit has taken you through some of the various sample of early OS.

The DOS is no longer popular because of the advanced GUI packages in the market now. But you still find it in some computing environment. Disk operating system can be the operating system itself or not . Distinguish between DOS that is the OS itself and the one that is not .

The previous unit discussed the disk operating system and its various characteristics and examples. Such applications include embedded systems , industrial robots, spacecraft, industrial control , and scientific research equipment. An RTOS will typically use specialized scheduling algorithms in order to provide the real-time developer with the tools necessary to produce deterministic behavior in the final system.

An early example of a large-scale real-time operating system was the so-called «control program» developed by American Airlines and IBM for the Sabre Airline Reservations

Debate exists about what actually constitutes real-time computing.

Two basic designs exist

Event-driven Blocked. Most tasks are blocked, most of the time. Only one task per CPU is running. In simpler systems, the ready list is usually short, two or three tasks at most.

The real key is designing the scheduler. If there are never more than a few tasks on the ready list, then a simple unsorted bidirectional linked list of ready tasks is likely optimal. The critical response time, sometimes called the flyback time, is the time it takes to queue a new ready task and restore the state of the highest priority task. In a welldesigned RTOS, readying a new task will take 3-20 instructions per ready queue entry, and restoration of the highest-priority ready task will take 5-30 instructions.

On a 20MHz 68000 processor, task switch times run about 20 microseconds with two tasks ready. In such systems, a scheduler ready list implemented as a linked list would be inadequate. A significant problem that multitasking systems must address is sharing data and hardware resources among multiple tasks. It is usually «unsafe» for two tasks to access the same specific data or hardware resource simultaneously.

Binary semaphores

Typically this method of protection is used only when the critical section is just a few source code lines long and contains no loops. This method is ideal for protecting hardware bitmapped registers when the bits are controlled by different tasks. A typical solution is to have the task that has a semaphore run at the priority of the highest waiting task. But this simplistic approach fails when there are multiple levels of waiting .

Handling multiple levels of inheritance without introducing instability in cycles is not straightforward. In a deadlock, two or more tasks lock a number of binary semaphores and then wait forever for other binary semaphores, creating a cyclic dependency graph. Deadlock is usually prevented by careful design, or by having floored semaphores . The other approach to resource sharing is for tasks to send messages.

Protocol deadlocks occur when two or more tasks wait for each other to send response messages. Although their real-time behavior is less crisp than semaphore systems, simple messagebased systems usually do not have protocol deadlock hazards, and are generally betterbehaved than semaphore systems. The scheduler often provides the ability to unblock a task from interrupt handler Memory allocation is even more critical in an RTOS than in other operating systems. This can cause a program to stall, unable to get memory, even though there is theoretically enough available.

The simple fixed-size-blocks algorithm works astonishingly well for simple embedded

systems

The RTOS is not found in most computer environment because of its complexity. But it is most suitable for use in life and time critical environment. This unit has extensively describe and discussed the RTOS and the way it carries out various OS functions such as memory allocation, scheduling, interrupt handling and interprocess communication . You are to compare all these with what you have learnt about DOS in the previous unit.

Memory allocation is even more critical in an RTOS than in other operating systems.

List and explain the two basic design philosophies for the RTOS

Describe how interprocess communication and resource sharing are implemented in the RTOS.

Unit 3: Object-oriented and Time sharing Operating Systems

In the previous unit, you been exposed to the real-time operating system, its history and application areas. It can be argued, however, that there are already object-oriented concepts involved in the design of a more typical operating system such as Unix. During the late 1980s, Steve Jobs formed the computer company NeXT. One of NeXT's first tasks was to design an object-oriented operating system, NEXTSTEP.

They did this by adding an object-oriented framework on top of Mach and BSD using the Objective-C language as a basis. NeXT was later acquired by Apple Computer and its operating system became the basis for Mac OS X most visibly in the form of the «Cocoa» frameworks.

Choices is an object-oriented operating system that was developed at the University of

Inheritance is used to separate the kernel into portable machine independent classes and small non-portable dependent classes. Choices has been ported to and runs on SPARC, x86 and ARM. Athene is an object based operating system first released in 2000 by Rocklyte Systems.

The user environment is constructed entirely from objects that are linked together at

Objects can be shared between processes by creating them in shared memory and locking them as required for access.

One attempt at creating a truly object-oriented operating system was the BeOS of the mid

1990s, which used objects and the C++ language for the application programming interface . But the kernel itself was written in C with C++ wrappers in user space. The system did not become mainstream though even today it has its fans and benefits from ongoing development. Syllable makes heavy use of C++ and for that reason is often compared to BeOS.

TAJ is India's first object oriented operating system. It is made in C++ with some part in assembly. The source code of TAJ OS is highly modularized and is divided into different

modules, each module is implemented as class. i.e. all the device drivers and other important OS modules are embedded into kernel itself.

This increases the speed of execution by reducing context switching time . TAJ OS is developed by Viral Patel.

Multitasking

In this area, ideally, the kernel would consist of the bare minimum required to support a JVM. This is the only component of such an operating system that would have to be written in a language other than Java.

Time-sharing

Similarly, small slices of time spent waiting for disk, tape, or network input could be granted to other users. Throughout the late 1960s and the 1970s computer terminals were multiplexed onto large institutional mainframe computers , which in many implementations sequentially polled the terminals to see if there was any additional data or action requested by the computer user. Later technology in interconnections were interrupt driven, and some of these used parallel data transfer technologies like, for example, the IEEE 488 standard. Generally, computer terminals were utilized on College properties in much the same places as desktop computers or personal computers are found today.

In the earliest days of personal computers, many were in fact used as particularly smart terminals for time-sharing systems. The Internet has brought the general concept of time-sharing back into popularity. Expensive corporate server farms costing millions can host thousands of customers all sharing the same common resources. As with the early serial terminals, websites operate primarily in bursts of activity followed by periods of idle time.

The bursty nature permits

ACHIEVER'S EDUCATION .COM 0814 710097

the service to be used by many website customers at once, and none of them not ice any delays in communications until the servers start to get very busy.

The Time-Sharing Business

In the 1960s, several companies started providing time-sharing services as service bureaus. Early systems used Teletype K/ASR-33s or K/ASR-35s in ASCII environments, and an IBM teleprinter in EBCDIC environments. They would connect to the central computer by dial-up acoustically coupled modems operating at 10-15 characters per second. Later terminals and modems supported 30-120 characters per second.

Users were charged rent for the terminal, a charge for hours of connect time, a charge for seconds of CPU time, and a charge for kilobyte-months of disk storage. Companies providing this service included Tymshare , Dial Data, and Bolt, Beranek, and Newman. By 1968, there were 32 such service bureaus serving the NIH alone. History The concept was first described publicly in early 1957 by Bob Bemer as part of an article in Automatic Control Magazine.

CTSS has a good claim to be the first time-sharing system and remained in use until 1973. DTSS influenced the design of other early timesharing systems developed by Hewlett Packard, Control Data Corporation, UNIVAC and others .

RSX-11

As you have learnt in this unit, several attempts had been made at developing a truly object-oriented operating system. The unit has also discussed the basic features of some the object-oriented OS in existence today. In the next unit you will be learning about some of the basic functions of the OS and how they are achieved.

What do you under by object-oriented OS

Discuss at least three of the attempts that have been made to develop object-oriented OS stating the characteristic features of each of the examples of these attempts.

Context-Switch

Early computer systems allowed one program to be executed at a time. This program has complete control of the system, and had access to all the system's resources. Current-day computer systems allow multiple programs to be loaded into memory and to be executed concurrently. These needs resulted in the not ion of a process, which is a program in execution.

A process is the unit of work in a modern time-sharing system. Although, the main concern of the OS is the execution of user programs, it also needs to take care of various system tasks that are better left outside the kernel itself.

Define a process List the possible states of a process Describe a process control block Describe process creation and process termination

It includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. In addition, a process generally includes the process stack, which contains temporary data , and a data section, which contains global variables. Several processes may be associated with the same program - each would execute independently , either synchronously or asynchronously . Although, two processes may be associated with the same program, they nevertheless considered two separate execution sequences.

For instance, several users may be running different copies of the mail program, or the same user may invoke many copies of the editor program. Modern computer systems allow multiple programs and processes to be loaded into memory at the same time and, through time-sharing , give an appearance that they are being executed at the same time even if there is just one processor.

Process states

The state of a process is defined in part by the current activity of that process.

[...]

These state names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems more finely delineate process states. The various process states, are displayed in the figure, with arrows indicating possible transitions between states.

Each process is represented in the operating system by a process control block –

A PCB contains many pieces of information associated with a specific process as shown in Figure 3.2 below.

Program counter

Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward. The objective of time-sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. A uniprocessor system can have only one running process. If more processes exist, the rest must wait until the CPU is free and can be rescheduled.

ACHIEV EDUCATIONAL NSULTANT 1710097

As processes enter the system, they are put into a job queue. This queue is generally stored as a linked list. We extend each PCB to include a pointer field that points to the next PCB in the ready queue. When a process is allocated the CPU, it executes for a while and eventually quits, is interrupted, or waits for the occurrence of a particular event, such as the completion of an I/O request.

Since the system has many processes, the disk may be busy with the I/O request of some other process. The process therefore may have to wait for the disk. The list of processes waiting for a particular I/O device is called a device queue. It waits in the ready queue until it is selected for execution .

The process could issue an I/O request, and then be placed in an I/O queue. The process could create a new sub process and wait for its termination. The process could be removed forcibly from the CPU, as result of an interrupt, and be put back in the ready queue. In the first two cases, the process eventually switches from the waiting state to the ready state, and is then put back in the ready queue.

A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated. A process migrates between the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes for these queues in some fashion. The selection process is carried out by the appropriate scheduler.

In a batch system, often more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device, where they a re kept for later execution. A process may execute for only a few milliseconds before waiting for an I/O request. There may be minutes between the creation of new processes in the system.

Context Switch

This task is known as context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context-switch time is pure overhead, because the system does no useful work while switching. Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions.

The processes in the system can execute concurrently, and they must be created and deleted dynamically. Therefore, the operating system must provide a mechanism for process creation and termination. A process may create several new processes, via a create-process system call, during the course of execution. The creating process is called a parent process, whereas the new processes are called the children of that process.

Each of these new processes may in turn create other processes, forming a tree of processes . In general, a process will need certain resources to accomplish its task. The parent may have to partition its resources among its children, or it may be able to some resources among several of its children. Restricting a child process to a subset of the parent's resources prevents any process

from overloading the system by creating too many subprocesses.

When a process is created it obtains initialization data that may be passed along from the parent process to the child process in addition to the various physical and logical resources. On such a system, the new process may get two open files, F1 and the terminal device, and may just need to transfer the datum between the two. The parent continues to execute concurrently with its children.

There are also two possibilities in terms of the address space of the new process

The child process has a program loaded into it. In UNIX, every process except process 0 is created when another process executes the fork system call. Every process has one parent process, but can have many child processes. In UNIX, a child process is in fact created as a copy of the parent.

The child process can then overlay itself with a different program as required. Each process may create many child processes but will have only one parent process, except for the very first process which has no parent. The first process, called init in UNIX, is started by the kernel at booting time and never terminates. Process 1, known as init, is the ancestor of every other process in the system.

A parent will typically retrieve its child's exit status by calling the wait system call. However, if a parent does not do so, the child process becomes a zombie process. Here is some sample C programming language code to illustrate the idea of forking. The code that is in the «Child process» and «Parent process» sections are executed simultaneously.

*** When fork returns 0, we are in**

* Here we count up to ten, one each second. * Again we count up to ten.

This code will print out the following

At that point, the process may return data to its parent process . All the resources of the process – including physical and virtual memory, open files, and I/O buffers – are deallocated by the operating system. A process can cause the termination of another process via an appropriate system call e.g. abort. Usually, only the parent of the process that is to be terminated can invoke such a system call.

The child has exceeded its usage of some of the resources that it has been allocated. This requires the parent to have a mechanism to inspect the state of its children. The task assigned to the child is no longer required. The parent is exiting, and the operating system does allow a child to continue if its parent terminate.

On such systems, if a process terminates , then all its children must also be terminated. This phenomenon, referred to as cascading termination, is normally initiated by the operating system. The wait system call returns the process identifier of a terminated child, so that the

parent can tell which of its possibly many children has terminated. If the parent terminates, however, all its children have assigned as their new parent the init process.

Thus, the children still have a parent to collect their This unit has introduced you to the concept of processes. It has extensively discussed process control block, process scheduling and the various operations that can be carried out on processes. In the next unit, you will be taken through co-operating processes and the means through which these co-operating processes communicate with one another. As you have learnt in this unit, a process is a program in execution.

The state of a process is defined by that process current activity. Each process is represented in the operating system by its own process-control block . A process, when it is not executing is placed in some waiting queue. The ready queue contains all the processes that are ready to execute and are waiting for the CPU.

Each process is represented by a PCB, and the PCB can be linked together to form a ready queue. Long-term scheduling is selection of processes to be allowed to contend for the CPU. Long-term scheduling is normally influenced by resource –

ACHIEVER'S EDUCATIONAL CONSULTANT 08111710097

allocation considerations, especially memory management.

What are possible information that a PCB will contain?

Describe the actions taken by the kernel to switch context between processes.

What do you understand by process scheduling?

Explain the various types of scheduling we have.

Interprocess Communication

Message Passing System The previous unit has introduced you to the concept of processes and the various operations that can be carried out on processes. Sometimes, when you have more than one process running on the computer system, there may be need for them to interact with one another. This unit takes you through the different ways that these various processes that may be running on the computer system at the same time interacts with one another.

Explain interprocess communication Describe message passing Describe some methods for logically implementing a link and the send/receive operations

Describe the concept of buffering and the various ways it can be implemented. Co-operating processes The concurrent processes executing in the operating system may be either independent processes or co-operating processes. A process is independent if it cannot be affected by the other processes executing in the system. Clearly, any process that does not share any data with any other process is independent.

Whereas, a process is co-operating if it can be affected by the other processes executing in the system. Clearly, any process that shares data with other processes is co-operating process.

We may want to provide an environment for process co-operation for several reasons

Such a speedup can be achieved only if the computer has multiple processing elements. Concurrent execution of co-operating processes requires mechanisms that allow processes to communicate with one another and to synchronize their actions. A producer process produces information that is consumed by a consumer process. For example, a print program produces characters that are consumed by the printer driver.

A producer can produce one item while the consumer consuming another item. The producer and consumer must be synchronised, so that it does not try to consume an item that has not yet been produced. In this situation, the consumer must wait until an item is produced. The consumer may have to wait for new items, but the producer can always produce new items.

The buffer may either be provided by the operating system through the use of an interprocess-communication facility , or explicitly coded by the application programmer with the use of shared memory. Interprocess Communication Processes can communicate with each other via

Inter-process communication . This is possible for both processes running on the same machine and on different machines. IPC is best provided by message-passing system and message-passing system can be defined in many ways.

We are now going to look at different issues when designing message-passing systems. Message-Passing system The function of a message system is to allow processes to communicate with one another without the need to resort to shared data. Messages sent by a process can be of either fixed or variable size. If only fixed-sized messages can be sent, the system-level implementation is straightforward.

This restriction, however, makes the task of programming more difficult. On the other hand, variable-sized messages require a more complex system-level implementation, but the programming task becomes simpler. If processes P and Q want to communicate, they must send messages to and receive message from each other, a communication link must exist between them. This link can be implemented in a variety of ways.

We are concerned here not with the link's physical implementation , but rather with its logical implementation.

[...]

We are going to look at each of these types of messages in the following section.

ACHIEVER'S EDUCATIONAL CONSULTANT 417 9097

Processes that want to communicate must have a way to refer to each other. They can use either direct or indirect communication. With direct communication, each process that wants to communicate must explicitly name the recipient or sender of the communication.

A communication link in this scheme has the following properties

a link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate. A link is associated with exactly two processes. A variant of this scheme employs asymmetry in addressing.

Only the sender names the recipient, the recipient is not required to name the sender. receive – Receive a message from any process, the variable id is set to the name of the process with which communication has taken place. The disadvantage in both symmetric and asymmetric schemes is the limited modularity of the resulting process definitions. Changing the name of a process may necessitate examining all other process definitions.

All refrences to the old name must be found, so that they can be modified to the new name. this situations is not desirable from the viewpoint of separate compilation. With indirect communication, the messages are sent to and received from mailboxes or ports. A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed.

In this scheme, a process can communicate with some other process via a number of different mailboxes. A link is established between a pair of processes only if both members of the pair have a shared mailbox. A link may be associated with more than two processes. A number different links may exist between each pair of communicating processes, with each link corresponding to one mailbox.

Ow suppose that processes P1, P2 and P3 all share mailbox A. Process P1 sends a message to A, while P2 and P3 each a receive from A. Allow at most one process at a time to execute a receive operation. Allow the system to select arbitrarily which process will receive the message . The system may identify the receiver to the sender. A mailbox may be owned by either a process or by the operating system.

if the mailbox is owned by a process , then we distinguish between the owner and the user . Since each mailbox has a unique owner, there can be no confusion about who should receive a message sent to this mailbox. Any process that subsequently sends a message to this mailbox must be not ified that the mailbox no longer exists. On the other hand, a mailbox owned by the operating system is independent and is not attached to any particular process.

Send and receive messages through the mailbox

Initially, the owner is the only process that can receive messages through the mailbox. However, the ownership and receive privilege mat be passed to other processes through

appropriate system calls. Of course, the provision would result in multiple receiver for each mailbox.

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

Communication between processes takes place by calls to send and receive primitives.

There are different design options for implementing each primitive. Message passing may be either blocking or nonblocking – also known as synchronous and asynchronous. When both the send and receive are blocking, we have a rendezvous between the sender and the receiver. Whether the communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue.

In this case, te sender must block until the recipient receives the message. If the link is full, the sender must block until space is available in the queue. The other cases are referred to as automatic buffering. In this unit, you have been taken through the concept of co-operating processes, the means through which they communicate, and the various means to ensure synchronization between communicate processes.

As you have learnt in this unit, the processes in the system can execute concurrently. Concurrent execution requires mechanism for process creation and deletion.

ACHIEVER'S EDUCATIONAL CONSULTANT 08[illegible]10097

The processes executing in the operating system may be either independent processes or co-operating processes. Co-operating processes must have the means to communicate with each other.

The processes are expected to exchange information through the use of these shared variables. The message-system method allows the processes to exchange messages. The responsibility for providing communication may resy with the operating system itself. These two schemes are not mutually exclusive, and can be used simultaneously within a single operating system.

What do you understand by interprocess communication

Suppose a process P wants to wait for two messages, one from mailbox A and one from mailbox B. What sequence of send and receive should it execute? b. What sequence of send and receive should P execute if P wants to wait for one message from mailbox A or from mailbox B
c. Devise a scheme that allows a process to wait until a mailbox is empty, or explain why such a scheme cannot exist. Briefly explain buffering and the various ways it can be implemented.

It shares with other threads belonging to the same process its code section, data section, and other operating system resources, such as open files and signals.

Distinguish between user and kernel threads

ACHIEVER'S EDUCATIONAL CONSULTANCY 9814171097

Describe various multithreading models and their advantages and disadvantages. State the advantages of thread pools and the motivation for thread pools. In modern operating systems, each process can have several threads of execution . Multiple threads share the same program code, operating system resources and operating system permissions .

A process that has only one thread is referred to as a single-threaded process, while a process with multiple threads is referred to as a multithreaded process. Multi-threaded processes have the advantage that they can perform several tasks concurrently without the extra overhead needed to create a new process and handle synchronised communication between these processes. An application typically is implemented as a separate process with several threads of control. A web browser might have one thread display images or text while another thread retrieves data from the network.

In certain situations a single application may be required to perform several similar tasks. A bus web server may have several clients concurrently accessing it. If the web server ran as a traditional single-threaded process, it would be able to service only one client at a time. The amount of time that a client might have to wait for its request to be serviced could be enormous.

One solution is to have the server run as a single process that accepts requests. In fact, this process-creation method was in common use before threads became popular. Process creation, as you have seen in the previous unit is very heavyweight. It is generally more efficient for one process that contains multiple threads to serve the same purpose.

ACHIEVER'S EDUCATIONAL

The benefits of multithreaded programming can be broken down into four major categories

Alternatively, because threads share resources of the process to which they belong, it is more economical to create and context switch threads. The threaded programming model provides developers with a useful abstraction of concurrent execution. Since the kernel is not aware of user-level threads, fiber creation and scheduling are done in user space without the need for kernel intervention. Therefore, fibers are generally fast to create and manage.

However, the use of blocking system calls in fibers can be problematic. If a fiber performs a system call that blocks, the other fibers in the process are unable to run until the system call returns. When an I/O operation is initiated, a system call is made, and does not return until the I/O operation has been completed. In the intervening period, the entire process is «blocked» by the kernel and cannot run, which starves other fibers in the same process from executing.

As mentioned earlier, fibers are implemented entirely in userspace. Since scheduling occurs in userspace, the scheduling policy can be more easily tailored to the requirements of the program's workload. Due to the fact that thread management is done by the operating system, kernel threads are generally slower to create and manage than are user threads. However, since the kernel is managing the threads, if a thread performs a blocking system call, the

ACHIEVER'S EDUCATIONAL CONS 710097

kernel can schedule another thread in the application for execution.

Also in multiprocessor environment, the kernel can schedule threads on different processors. The use of kernel threads simplifies user code by moving some of the most complex aspects of threading into the kernel. The program does not need to schedule threads or explicitly yield the processor. Each time a thread starts, blocks, or exits, the process must switch into kernel mode, and then back into user mode.

This context switch is fairly quick, but programs that create many short-lived threads can suffer a performance hit. Cooperative multithreading, on the other hand, relies on the threads themselves to relinquish control once they are at a stopping point. This can create problems if a thread is waiting for a resource to become available. In the late 1990s, the idea of executing instructions from multiple threads simultaneously has become known as simultaneous multithreading.

In this section we will look at the three common types of multithreading implementation. Thread management is done in user space, so it is efficient, but the entire process will block if a thread makes a blocking system call. Also, because only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multiprocessors. The only drawback to this model is that creating a fiber requires creating the corresponding kernel thread.

The number of kernel threads may be specified to either a particular application or a particular machine an application may be allocated more kernel threads on a multiprocessor than on a miniprocessor. whereas the many-to-one model allows the developer to create as many fibers as she wishes, true concurrency is not gained because kernel can schedule only one thread at a time. Synchronous signals are delivered to the same process that performed the operation causing the signal, hence the name synchronous. When a signal is generated by an event external to a running process, that process receives the signal asynchronously.

Examples of such signals include terminating a process with specific keystrokes or having a timer expire. Typically an asynchronous signal is sent to another process.

A default signal handler

ACHIEVER'S EDUCATION

Every signal has a default signal handler that is run by the kernel when handling the signal. This default action may be overridden by a user-defined signal handler function. In this instance, the user-defined function is called to handle the signal rather than the default action. Both synchronous and asynchronous signals may be handled in different ways.

Assign a specific thread to receive all signals for the process. The method of delivering a signal depends on the type of signal generated. Unlimited threads could exhaust system resources, such as CPU time or memory. One solution to this issue is to thread pools.

The general idea behind a thread pool is to create a number of threads at process startup and place them into a pool, where they sit and wait for work. When a server receives a request, it awakens a thread from this pool passing it the request to service. Once the thread completes its service, it returns to the pool awaiting more work.

Advantages of Thread Pools

The number of threads in the pool can be set heuristically based upon factors such as the number of CPUs in the system, the amount of physical memory and the expected number of concurrent client requests. More sophisticated thread-pool architectures can dynamically adjust the number of threads in the pool according to usage patterns. In this unit you have been introduced to the main concept of light-weight processes popularly known as threads. A multithreaded process contains several different flows of control within the same address space.

The benefits of multithreading include increased responsiveness to the user, resource sharing within the process, economy, and the ability to take advantage of multiprocessor architectures. In general, fibers are faster to create and manage than are kernel threads. The one-toone model maps each user thread to a corresponding kernel thread. The many-to-many model multiplexes many user threads to a smaller or equal number of kernel threads.

Other issues include thread cancellation, signal handling and thread-specific data.

Under what circumstances is one type better than the other?

Describe the action taken by a kernel to context switch between kernel threads. Describe the action taken by a thread library to context switch between fibers.

First-Come-First-Serve Scheduling

The problem of selecting an algorithm for a particular system will also be considered.

o Multilevel Queue Scheduling o Multilevel Feedback Queue Scheduling

Select a CPU scheduling algorithm for a particular system. Any other processes must wait until the CPU is free and can be rescheduled. A process is executed until it must wait, typically for the completion of some I/O request. In a simple computer system, the CPU would then sit idle.

All this waiting time is wasted. With multiprogramming, we try to use this time productively. Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process.

Scheduling is fundamental to operating system function. Almost all computer resources are scheduled before use. The success of CPU scheduling depends on the following observed property of processes. Although they vary greatly by process and by computer, they tend to have a frequency curve similar to that shown in Figure 3.2.

The curve is generally characterized as exponential or hyperexponential, with many short CPU bursts and a few long CPU bursts. An I/O-bound program would typically have many very short CPU bursts while a CPU-bound program might have a few very long CPU bursts. This distribution can help us select an appropriate CPU-scheduling algorithm.

Process B

Figure 3.1 Alternating Sequence of CPU and I/O Bursts of Two Processes.

CPU Scheduler

Whenever the CPU becomes idle, the operating system must select on the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler . The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them. The ready queue is not necessarily a first-in, first-out queue.

As you shall see, when we consider the various scheduling algorithms, a ready queue may be implemented as a FIFO queue, a priority queue, a tree, or simply an ordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks of the processes.

[...]

A new process must be selected for execution. When scheduling takes place only under circumstances 1 and 4, we say the scheduling scheme is nonpreemptive. This scheduling method is used by Microsoft Windows 3.1 operating system.

Dispatcher

Another component involved in the CPU scheduling function is the dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. Jumping to the proper location in the user program to restart that program. The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

Different CPU-scheduling algorithms have different properties and may favour one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU-scheduling algorithms. The characteristics used for comparison can make a substantial difference in the determination of the best algorithm.

CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40 percent to 90 percent . It affects only the amount of time that a process spends waiting in the ready queue. Waiting time is, therefore, the sum of the periods spent waiting in the ready queue.

i.e. the time from the submission of a request until the first response is produced.

ACHIEVER'S EDUCATIONAL CONSULTANCY 9814710097

We usually want to maximize CPU utilization and throughput, and to minimize turnaround time, waiting time, and response time. CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. In this section, we describe many CPU-scheduling algorithms that exist.

First-Come, First Served Scheduling This is the simplest CPU-scheduling algorithm. In this scheme, the process that requests the CPU first is allocated the CPU first. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue.

The running process is the removed from the queue. The code for FCFS scheduling is simple to write. The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3. When the CPU is available, it is assigned to the process that has the smallest next CPU burst.

If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie. The choice arises when a new process arrives at the ready queue while a previous process is executing. The new process may have a shorter next CPU burst than what is left of the current executing process. Preemptive SJF scheduling is sometime called shortest-remaining-time-first Example 3.

Process P1 is started at time 0, since it is the only process in the queue. A nonpreemptive SJF scheduling would result in an average waiting time of 7.75 milliseconds. A priority is associated with each process and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS.

Some systems use low numbers to represent low priority while others use low numbers for high priority. But in this course, we will use low numbers to represent high priority. When a process arrives at the ready queue, its priority is compared with that of the currently running process. A solution to indefinite blocking of low-priority processes is aging.

Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.

Round-Robin Scheduling

Round-robin is one of the simplest scheduling algorithms for processes in an operating system. It assigns time slices to each process in equal portions and in circular order, handling all processes without priority. Round-robin scheduling is both simple and

easy to implement, and starvation-free. The name of the algorithm comes from the round-robin principle known from other fields, where each person takes an equal share of something in turn.

It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time, called a time quantum , is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

New processes are added to the tail of the queue. The CPU scheduler picks from the head of the queue, sets a timer to interrupt after 1 time quantum, and dispatches the process. The process may have a CPU burst of less than one time quantum. A context switch will be executed, and the process will be put at the tail of the ready queue.

Multilevel Queue Scheduling

The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, etc. Each q.

CIT211 List of Questions
Latex formatted questions may not properly render

Q1 ..... is the strongest non-blocking guarantee of progress data

- Lock-freedom
- Wait-freedom
- Run-freedom
- Obstruction-freedom

Q2 In general, a lock-free algorithm can run in ..... phases

- two
- three
- four
- five

Q3 Correct concurrent assistance is typically the most complex part of a ..... algorithm

- Lock-free
- Wait-free
- Run-free
- obstruction-free

Q4 The services provided by the Operating System can be categorised into .....

- two

ACHIEVER'S EDUCATIONAL CONSULTANT 08[illegible]10097

- three
- five
- six

Q5 The ??closed-shop?? mode of operation was used during the ..... generation of the Operating System

- first
- second
- third
- fourth

Q6 In the ..... approach, the kernel itself only provides basic functionality that allows the execution of servers, separate programs that assume former kernel functions, such as device drivers, GUI servers, etc.

- picokernel
- microkernel
- nanokernel
- exokernel

Q7 The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ..... queue

- ready
- system
- job
- process

Q8 A process goes from the ??Ready?? state to ..... state

- waiting
- running
- finished
- ready

Q9 ..... section contains global variables

- data
- storage
- code

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- memory

Q10 A ..... operating system is one that appears to its users as a traditional uniprocessor system, even though it is actually composed of multiple processors

- network
- time-sharing
- distributed
- real-time

Q11 In ..... operating system the computer's response time is the turnaround time

- real-time
- time-sharing
- online
- batch-processing

Q12 ..... is about the weakest natural non-blocking progress guarantee

- Lock-freedom
- Wait-freedom
- Run-freedom
- Obstruction-freedom

Q13 When a process creates a new process, ... possibilities exist in terms of execution

- two
- three
- four
- six

Q14 ..... scheduler is another name for CPU scheduler

- short-term
- long-term
- medium-term
- instant

Q15 Preventing the system from continually live-locking is the task of a .....

- context switcher
- contention manager

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- scheduler
- context manager

Q16 A ..... queue consists of all processes in the system

- ready
- system
- job
- process

Q17 Which of the following is a type of synchronization?

- barrier
- lock
- thread join
- all of the above

Q18 > In which of the following situations can race condition occur?

- File system
- Networking
- Life-critical system
- All of the above

Q19 ..... scheduling produces the shortest waiting time

- RR
- FCFS
- SJF
- Priority

Q20 ..... scheduling is the simplest scheduling algorithm

- RR
- FCFS
- SJF
- Priority

Q21 A thread comprises of the following except:

- thread ID
- program counter

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- register set
- none of the above

Q22 Fibers are supported ..... the kernel

- below
- above
- beside
- any of the above

Q23 ..... are supported directly by the operating system

- fibers
- user threads
- kernel threads
- all of the above

Q24 Threads can be classified into ..... different types

- two
- three
- four
- six

Q25 Blocking message passing is known as .....

- asynchronous
- isochronous
- synchronous
- none of the above

Q26 Which of the following properties is possessed by a communication link in direct communication scheme?

- a link is established automatically between every pair of processes that want to communicate
- a link is associated with exactly two processes
- Exactly one link exists between each pair of processes
- all of the above

Q27 The class of OS that has very little user-interface capability, and no end-user utilities is ..... OS

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- real-time
- time-sharing
- network
- batch-processing

Q28 ..... is a very light-weight microkernel

- L3
- L4
- L6
- L8

Q29 The .....kernel represents the closest hardware abstraction layer of the operating system by interfacing the CPU, managing interrupts and interacting with the MMU

- exo
- monolithic
- micro
- nano

Q30 The main disadvantages of ..... kernels are the dependencies between system components

- exo
- monolithic
- micro
- nano

Q31 The loop that the kernel provides for execution whenever no processes are available to run is called the ..... process

- passive
- active
- dormant
- idle

Q32 In the ??.....?? mode of operation, the individual user (programmer) was responsible for all machine set up and operation, and subsequent clean-up and preparation for the next user

- open-shop
- closed-shop

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- all of the above
- none of the above

Q33 In the area of system efficiency, which of the following services is offered by the OS?

- Accounting
- Ease of Evolution of OS
- New Services
- All of the above

Q34 A ready queue may be implemented as the following except:

- FIFO queue
- priority queue
- a tree
- none of the above

Q35 Process execution begins with ..... burst

- I/O
- CPU
- Wait
- all of the above

Q36 The number of threads in the pool can be set heuristically based upon the following factors except:

- the expected number of concurrent client requests
- the number of CPUs in the system
- the amount of physical memory
- none of the above

Q37 Which of the following operating systems does not support kernel threads?

- Windows NT
- Solaris 2
- BeOS
- none of the above

Q38 The user of an Operating system is a(n) .....

- person

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- expert
- supervisor
- application

Q39 In ..... capacity buffer, the sender must block until the recipient receives the message

- Bounded
- Unbounded
- Zero
- All of the above

Q40 OS can be defined by the following except:

- functions
- goals
- objectives
- none of the above

Q41 How many common types of multithreading implementation are there?

- two
- three
- four
- six

Q42 Which of the following is not a disadvantage of Deterministic Modelling

- It is hard to express a system of complex algorithms and distributions,
- It requires too much knowledge to be useful
- It requires exact numbers for input and its answers apply to only those cases
- It is too specific

Q43 Which of the following scheduling algorithms is a type of priority scheduling algorithm?

- shortest-job-first
- first-come-first-served
- round-robin
- none of the above

Q44 Threads can be classified into ..... different types

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- two
- three
- four
- six

Q45 Blocking message passing is known as .....

- asynchronous
- isochronous
- synchronous
- none of the above

Q46 Which of the following properties is possessed by a communication link in direct communication scheme?

- a link is established automatically between every pair of processes that want to communicate
- a link is associated with exactly two processes
- exactly one link exists between each pair of processes
- all of the above

Q47 The class of OS that has very little user-interface capability, and no end-user utilities is ..... OS

- real-time
- time-sharing
- network
- batch-processing

Q48 ..... is a very light-weight microkernel

- L3
- L4
- L6
- L7

Q49 The .....kernel represents the closest hardware abstraction layer of the operating system by interfacing the CPU, managing interrupts and interacting with the MMU

- exo
- monolithic

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- micro
- nano

Q50 The main disadvantages of ..... kernels are the dependencies between system components

- exo
- monolithic
- micro
- nano

Q51 The number of processes a system may run simultaneously is ..... to the number of CPUs installed

- less than
- greater than
- equal
- inversely proportional

Q52 ..... kernels are able to give the user the illusion that the number of processes being run simultaneously on the computer is higher than the maximum number of processes the computer is physically able to run simultaneously

- multi-programming
- multi-processing
- multi-tasking
- all of the above

Q53 The loop that the kernel provides for execution whenever no processes are available to run is called the ..... process

- passive
- active
- dormant
- idle

Q54 In the ??.....?? mode of operation, the individual user (programmer) was responsible for all machine set up and operation, and subsequent clean-up and preparation for the next user

- open-shop
- closed-shop

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- all of the above
- none of the above

Q55 In the area of system efficiency, which of the following services is offered by the OS?

- Accounting
- Ease of Evolution of OS
- New Services
- All of the above

Q56 ..... and debuggers are accessed through the OS to assist programmers in creating programs

- creators
- launchers
- editors
- compilers

Q57 The kernel contains the ..... frequently used functions in the OS

- most
- least
- seldom
- all of the above

Q58 All of these are resources managed by the operating system except .....

- CPU time
- file storage space
- data
- printing paper

Q59 A set of programs called the ..... is provided to assist the programmer in program creation and file management

- system software
- operating system
- housekeeper
- utilities

Q60 The user of an Operating system is a(n) .....

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- person
- application
- supervisor
- expert

Q61 When a process creates a new process, there are ..... possibilities in terms of the address space of the new process

- two
- three
- four
- five

Q62 ..... scheduler is another name for CPU scheduler

- Short-term
- long-term
- medium-term
- instant

Q63 A ready queue is generally stored as a .....

- b-tree
- graph
- stack
- linked list

Q64 A ..... queue consists of all processes in the system

- ready
- system
- job
- process

Q65 A ..... is just a passive collection of instructions

- process
- task
- program
- job

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

Q66 The objective of ..... is to maximize CPU utilization

- multiprogramming
- multiprocessing
- distributed
- multitasking

Q67 A ..... system is a computer hardware configuration that includes more than one independent processing unit

- multiprogramming
- multiprocessing
- distributed
- multitasking

Q68 Typically, a task has ..... states

- two
- three
- four
- six

Q69 The user of an Operating system is a(n) .....

- person
- expert
- application
- supervisor

Q70 The function of a message system is to allow processes to communicate with one another without the need to resort to ..... data

- same
- separate
- shared
- conflicting

Q71 The objective of ..... is to switch the CPU among processes so frequently that users can interact with each program while it is running

- real-time
- time-sharing

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

- network
- batch-processing

Q72 The ??closed-shop?? mode of operation was used during the ..... generation of the Operating System

- second
- first
- fourth
- third

Q73 The services provided by the Operating System can be categorised into .....

- two
- three
- four
- six

Q74 A process goes from the ??Ready?? state to ..... state

- waiting
- running
- finished
- ready

Q75 In the ..... approach, the kernel itself only provides basic functionality that allows the execution of servers, separate programs that assume former kernel functions, such as device drivers, GUI servers, etc.

- picokernel
- nanokernel
- microkernel
- exokernel

Q76 The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ..... queue

- ready
- system
- job
- process

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097

Q77 ..... section contains global variables

- code
- storage
- data
- memory

Q78 A ..... operating system is one that appears to its users as a traditional uniprocessor system, even though it is actually composed of multiple processors

- network
- time-sharing
- distributed
- real-time

Q79 In ..... operating system the computer's response time is the turnaround time

- real-time
- time-sharing
- online
- batch-processing

Q80 When a process creates a new process, ... possibilities exist in terms of execution

- two
- three
- four
- five

ACHIEVER'S EDUCATIONAL CONSULTANT 08141710097