CZ2005 Operating Systems
Nanyang Technological University
Benjamin Dreyer
Exam Date: 11/28/2019

ALL info from review slides from prof and geek for geek.com

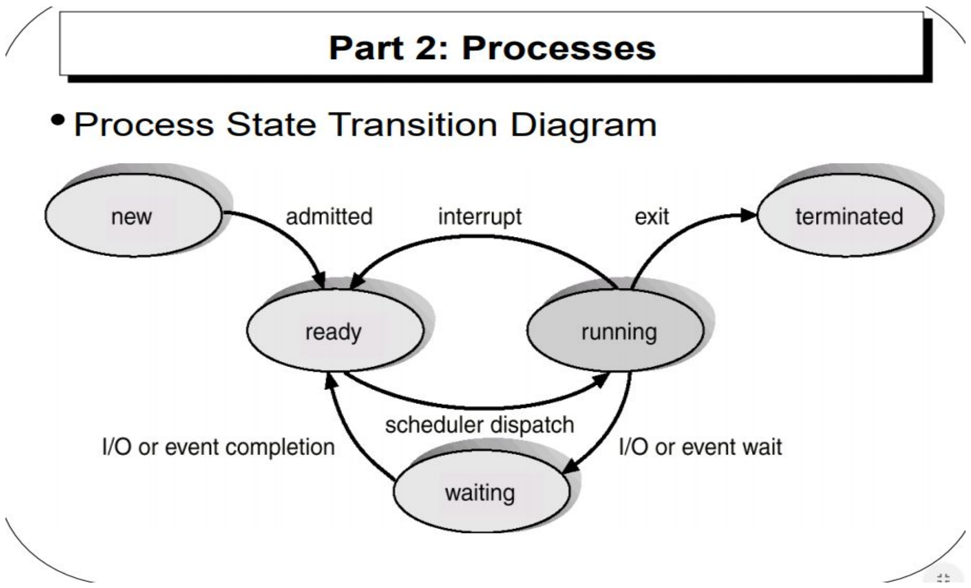
Overview (Part 1)

- *Understand the evolution of OS*
  - Batch, multi-programmed, time-sharing systems
    - **Batch Operating System**: In this, jobs which are of similar type of grouped together and treated as a batch.
      - Jobs are determined to be similar by an operator then placed into a batch where the jobs contain similar features.
    - **Multi-programmed**: A multiprogramming operating system is one which can run multiple programs concurrently.
    - **Time sharing systems**: A time sharing operating system is that in which each task is given some time to execute and all tasks are given time so that all process run seamlessly without any problem.
- *Understand the purpose and goals of OS*
  - Resource manager, control program, kernel
    - **Resource Manager**: The Operating System acts as a resource manager by managing resources internally lol
    - **Control Program:** A program that enhances an operating system by creating an environment in which you can run other programs. Control programs generally provide a graphical interface and enable you to run several programs at once in different windows.
    - **Kernel:** The kernel is a computer program that is the core of a computer's operating system.
  - Efficiency and convenience
- ***Basic Concepts:***
  - **Interrupts, traps, system calls**
    - **Interrupts**: In digital computers, an interrupt is an input signal to the processor indicating an event that needs immediate attention.
    - **Traps:** A trap also known as an exception or a fault, is typically a type of synchronous interrupt caused by an exceptional condition(a breakpoint, division by zero, invalid memory access, etcc….)
    - **System Calls**: a system call is the programatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- ***OS Components***

    - The parts of an operating system all exist as to make the various parts of a computer system work together.
    - All user software program has to undergo the operating system in order to utilize any of the hardware.
        - Kernel
        - Process Execution
        - Interrupt
        - Memory Management
        - Multitasking
        - Networking
        - Security
        - User Interface

Processes(Part 2)

- ***Process State Transition Diagram***



    - New
    - Ready
    - Waiting
    - Running
    - Terminated
- ***Process Context Switch***
    - **Context switching** involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier.

- ***Level of Scheduling***
  - **Long term scheduling:** Long term scheduling involves selecting the processes from the storage pool in the secondary memory and loading them into the ready queue in the main memory for execution.
  - **Medium term scheduling:** Medium term scheduling involves swapping out a process from main memory. That process can be swapped in later from the point it stopped executing. This can also be called as suspending and resuming the process and is does by the medium term scheduler.
  - **Short Term Scheduling:** Short term scheduling involves selecting one of the process from the ready queue and scheduling them for execution.

- ***Inter-Process Communication***
  - **Direct communication**: In direct communication, each process that wants to communicate must explicitly name the recipient or sender of the communication.
    - Send(P, message) send a message to P
    - Receive(Q, message): Receive a message from Q
  - **Indericeted communication:** In indirect communication the messages are sent to and received from mailboxes. A mailbox can be viewed abstractly as an object into which messages can be placed by process and from which messages can be removed. The send and receive primitives are defined as follows
    - Send(A, message) send a message to mailbox A
    - Received(A, message) Receive a message from mailbox A
  - **Message Buffering:** A message sent by a process needs to be kept in some memory area until the receiving machine has received it. It may be kept in the sender's address space or may be buffered in an address space managed by the operating system such as the process table.
- ***Basic Concepts***
  - Process address space and execution context **NO CLUE**
  - i/o bound, CPU bound processes
  - Independent, cooperative processes

CPU Scheduling(Part 3)

- ***Definition**: Cpu scheduling is the basis of multiprogramming operating systems. The objective of multiprogramming is to have some process running at all times, in order to maximize CPU utilization. Scheduling is a fundamental operating-system function. Almost all computer science resources are scheduled before use.*
- ***Basic Concepts***
    - **CPU-I/O burse cycle**: Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate between these two states. process execution begins with a CPU burst. That is followed by an I/O burst, then another CPU burst, then another I/O burst. Eventually, the last CPU burst will end with a system request to terminate execution, rather than with another I/o burst.
- ***Non-Preemptive, preemptive scheduling:***
    - **Preemptive**:Preemptive scheduling is used when a process switches from running state to a ready state or from waiting state to ready state. In this mode, the CPU may release the process depending on if there might be another process that can be finished faster or is more important.
    - **Non-Preemptive**: used when a process terminates or a process switches from running to waiting state. In this mode, once the resources are allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.
- ***Scheduling algorithms:***
    - **FCFS**: *First Come First Serve* a method of scheduling where the first process to enter the ready queue will be processed, then finished, then the next process that was behind that one in the ready queue will be processed.
    - **SJF:** *Shortest Job first,* whichever job requires the least amount of time to process will be processed. (Lowest Anticipated Time)
    - **SRTF:** *Shortest Remaining Time*, whichever process requires the least amount of time to finish will be processed. Practically the same as SJF but

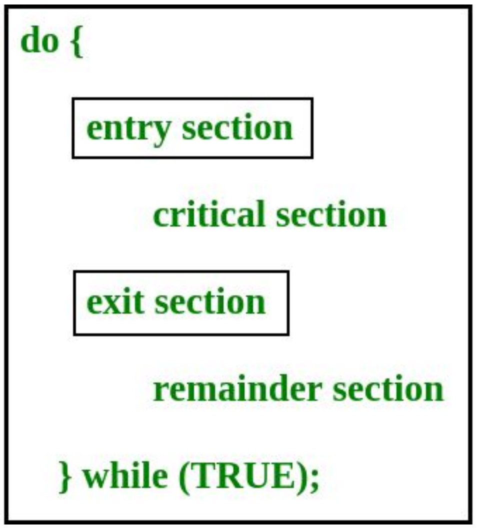
only is used in preemptive kernels, where if a shorter job enters the ready queue, the scheduler will switch to that job.

  - **Non-Preemptive Priority:** NPP, the processes are scheduled according to the priority number assigned to them. Once the processes get scheduled, it will run until completion. Generally, the lower the priority number, the higher is the priority of the process. So a processed assigned with the number 1 would have the highest priority. Literally the mcdonalds ordering system.
  - **Preemptive Priority:** in PP, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The process that has the highest priority of all the processes in the ready queue will be executed
  - **Round Robin: (**this shit is fucking complicated**)** RR scheduling is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.
    - Simple, easy to implement, starvation free as all processes get fair share of CPU
    - One of the most commonly used techniques in CPU scheduling as a core.
    - It is preemptive
    - Disadvantage: more overhead of context switching
    - Uses completion time, turn around time, and waiting time (turnaround time - waiting time) to calculate which program should be run when.

Process Synchronization(Part 4)

- ***Definition****: Process Synchronization is a way to coordinate processes that use shared data. It occurs in an operating system among cooperating processes. While executing many concurrent processes, process synchronization helps to maintain shared data consistency and cooperating process execution.*
- On the process of synchronization, processes are categorized as one of the following two types:
  - **Independent Process**: Execution of one process does not affect the execution of another process

    - **Cooperative Process:** Execution of one process directly affects the execution of others.
- Process sync problem arises in the case of coop process also because resources are shared in coop processes.
- ***Race Condition***: An undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.
- **Mutual Exclusion:** Mutual exclusion is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource.
- ***Critical Section Problem:*** Critical section is a code segment that can be accessed by only one process at a time. Critical section contains shared variables which need to be synchronized to maintain consistency of data variables.



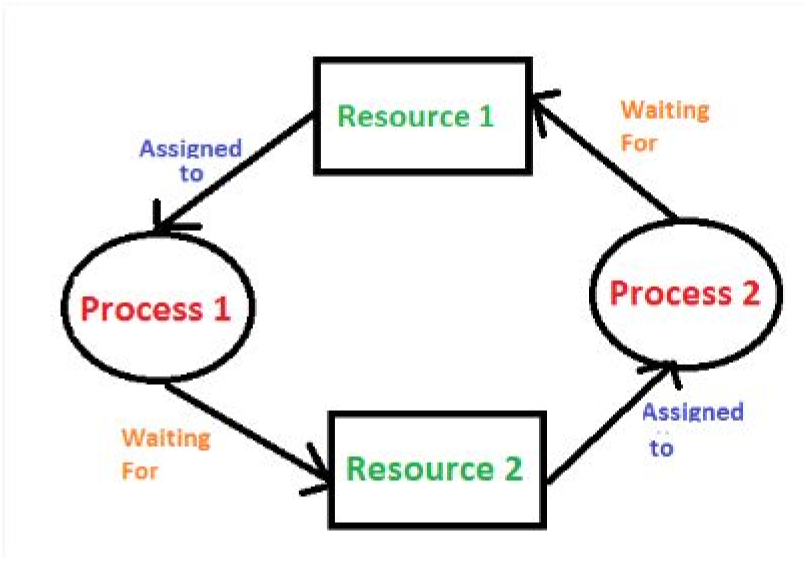
```
do {
    entry section
        critical section
    exit section
        remainder section
  } while (TRUE);
```

- **Test and Set:** Test and set is a hardware solution to the synchronization problem. In TestAndSet, we have a shared lock variable which can take either of the two values, 0 or 1.
    - 0: unlock
    - 1: lock
- **Test and Set cont:** Before entering into the critical section, a process inquires about the lock. If it is locked, it keeps on waiting till it becomes free and if it is not locked, it takes the lock and executes the critical section.

- **Test on Set cont:** In TAS, Mutual exclusion and progress are preserved but bounded waiting cannot be preserved.
- **Bounded waiting:** a bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

- **Semaphores:**
    - Semaphore was proposed by Djikstra in 1965, which is a very significant technique to manage concurrent processes by using a simple integer value, which is known as a semaphore.
        - **Binary Semaphore:** This is also known as a mutex lock. It can have only two values, 0 and 1. It's value is initialized to 1. It is used to implement the solution of critical section problem with multiple processes.
        - **Counting Semaphore:** Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.
        - **Wait():**
        - **Signal():**
        - **Using TestAndSet to implement semaphore:** dunno

Deadlocks (Part 5)



- ***Definition***: *Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.*
- ***F*our necessary conditions:**
    - **Mutual Exclusion:** One or more than one resource are non-sharable(Only one process can use at a time)
    - **Hold and Wait:** A process is holding at least one resource and waiting for resources.
    - **No Preemption:** A resource cannot be taken from a process unless the process releases the resource.
    - **Circular Wait:** A set of processes are waiting for each other in circular form.
- **Methods for handling deadlocks:**
    - **Deadlock Prevention:** The idea is to not let the system into deadlock state. One can zoom into each category individually, prevention is done by negating one of the above mentioned necessary conditions for deadlock.
    - **Avoidance:** kind of futuristic in nature. By using the strategy of avoidance we have to make an assumption. We need to ensure that all information about resources which process WILL need are known to us prior to execution of the process. Then you use ???Banker's algorithm??? In order to avoid a deadlock.
    - **Deadlock detection and recovery:** Let deadlock occur, then do preemption to handle it once occurred. (How do you just switch to preemption though?)
    - **Ignore the problem all together:** if deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

- **Deadlock avoidance**
    - **Resource allocation state:** the number of available and allocated resources, and the maximum requirements of all process in the system.
    - **Safe state:** a state is safe if the system can allocate all resources requested by all processes without entering a deadlock state (how would a system know that a deadlock wouldn't occur?)
    - **Algorithm to determine safe state:**
        - **Bankers algorithm:** a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources. Then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Memory Organization (Part 6) NOT FINISHED

- ***Definition:*** *A memory unit is the collection of storage units or devices together. The memory unit stores the binary information in the form of bits. Generally, memory/storage is classified into two categories*
    - ***Volatile Memory:*** *This loses its data when power is switched off.*
    - ***Non-volatile Memory:*** *This is permanent storage and does not lose any data when power is switched off.*
- ***Another Definition(GeeksforGeeks):*** *In Operating Systems, Memory Management is the function responsible for allocating and managing computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.*
- **Summary from the actual slides:**
    - Programs must be brought into memory for execution -> memory allocation
        i. **Contiguous allocation**: a classical memory allocation model that assigns a process consecutive memory blocks(that is, memory blocks having consecutive addresses.)
            - **Fix partition vs Dynamic Partition**

        - **Fixed Partitioning:** This is the oldest and simplest technique used to put more than one processes in the main memory. In this partitioning, numbers of partitions in RAM are **fixed but the size** of each partition may or **may not be the same**. No spanning is allowed.
        - **Dynamic Partition:**
    ii. **Non-contiguous allocation:** In non-contiguous allocation, Operating system needs to maintain the table which is called **Page Table** for each process which contains the base address of each block which is acquired by the process in memory space. In non-contiguous memory allocation, different parts of a process is allocated different places in Main Memory.

Virtual Memory (Part 7)

- ***Definition:*** *Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory(even though its not). The addresses a program may use to reference memory are distinguished from the addresses the memory system used to identify physical storage sites. (physical storage and virtual storage process are different???)*
- **The motivation of virtual memory:** There are a few advantages that virtual memory can bring
    - More processes may be maintained in the main memory: Because we are going to load only some of the pages of any particular process, there is room for more processes. This leads to more efficient utilization of the processor because it is more likely that at least one of the more numerous processes will be in the ready state at any particular time.
    - A process may be larger than all of main memory: One of the most fundamental restrictions in programming is lifted. A process larger than the main memory can be executed because of demand paging. The OS itself loads pages of a process in main memory as required.
    - It allows greater multiprogramming levels by using less of the available memory for each process.

- **Demand Paging:** The process of loading the page into memory on demand is known as demand paging.
    - **Page:** The operating system retrieves data from secondary storage in same-size blocks called pages. (Chunks of memory)
    - **The Process involves the following steps:**
        1. If CPU tries to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
        2. The OS puts the interrupted process in a blocking state. For the execution to process the OS must bring the required page into the memory.
        3. The OS will search for the required page in the logical address space.
        4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.
        5. The page table will update accordingly
        6. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.
- **Page Fault:** A page fault is a type of exception raised by computer hardware when a running program accesses a memory page that is not currently mapped by the memory management unit (MMU) into the virtual address space of a process.
    - A page fault can trigger demand paging
- **Performance of demand paging:** given memory access time, page fault time, and page fault rate, how to compute EAT?
    - EAT: Effective access time
- **Page replacement is essential to virtual memory**
    - **Page replacement:** a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in
    - **Page:** A page, memory page, or virtual page is a fixed-length contiguous block of virtual memory, described by a single entry in the page table. It is the smallest unit of data for memory management in a virtual memory operating system.
    - **Page replacement algorithms:**
        1. **FIFO:** first in first out
        2. **Optimal**
        3. **LRU**
        4. **Second-Chance(Clock)**
- **Allocation of frames**
    - **Fixed vs. variable allocation**
    - **Local vs. global Replacement**
- **Thrashing**
    - **What is it?**

    - **How to combat it: working-set model and page fault frequency**

File Systems (Part 8)

- **File structure**
    - **Structured files**
    - **Unstructured files**
- **File access methods**
    - **Sequential access**
    - **Random access**
- **Directory Organization**
    - **Required features: efficiency, naming and grouping**
    - **One level, two level, tree structured, acyclic graph**