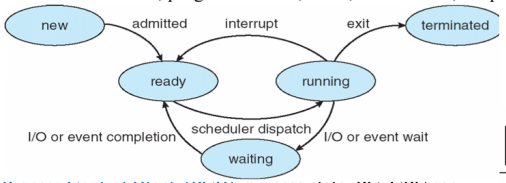
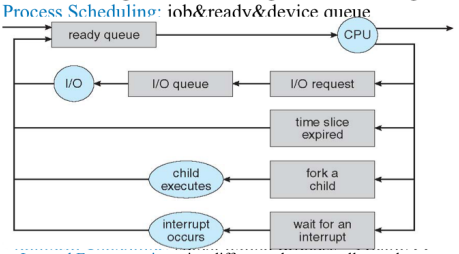
Process:text section, program counter, stack, data section, heap



Process Control Block (PCB): process state, PC, CPU reg, CPU scheduling info, mem-manage info, accounting info, I/O

Process Scheduling: job&ready&device queue



Internal Fragmentation: size difference between allocated mem and used is memory internal to a partition, but not being used.

Segmentation: logical addr <segment num, offset>; vary length

Segmentation table: base contains the starting physical address where the segments reside in memory; limit specifies the length of the segment. Each entry is associated to validation bit

Segmentation base/length reg (STBR, STLR)

Paging: PM -> frames; LM -> pages; Page table to translate; a N pages process need N free frames to be loaded. Backing store likewise split into pages. Still has fragmentation.

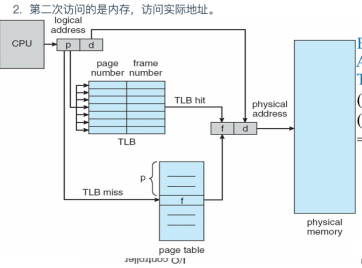
Page Num & Page Offset

Trade-off: small page size, less fragmentation but larger page table

Page Table Implementation: Page table is kept in the mem.

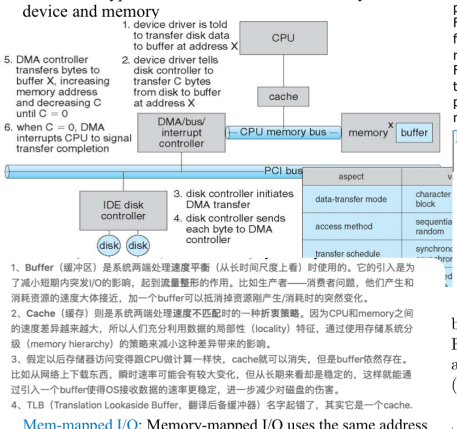
Page table base / length reg: this scheme every data/instruction access requires two memory accesses. Translation look-aside buffers / Associative Reg (TLBs) are used for the two men access procedure. Address-space identifiers (ASIDs) may be stored in each entry.

1. 第一次访问内存是访问页表，取出虚拟页对应的物理页。
2. 第二次访问内存是访问实际内存地址。
1. 第一次访问TLB，得到虚拟页对应的物理页
2. 第二次访问的是内存，访问实际地址。



Effective Access Time:
$(1+\varepsilon)*\alpha+(2+\varepsilon)*(1-\alpha) = 2+\varepsilon-\alpha$

Direct Mem Access (DMA): used to avoid programmed IO (*one byte at a time*) for large data movement. Requests DMA controller. Bypasses CPU to transfer data directly between I/O device and memory



1、Buffer（缓冲区）是系统两端处理速度平衡（从长时间尺度上看）时使用的。它的引入是为了减小短期内突发I/O的影响，起到流量整形的作用。比如生产者——消费者问题，他们产生和消耗资源的速度大体接近，加一个buffer可以抵消掉资源刚产生/消耗时的突然变化。

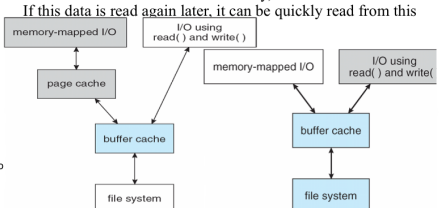
2、Cache（缓存）则是系统两端处理速度不匹配时的一种折衷策略。因为CPU和memory之间的速度差异越来越大，所以人们充分利用数据的局部性（locality）特征，通过使用存储系统分级（memory hierarchy）的策略来减小这种差异带来的影响。

3、假定以后存储器访问变得跟CPU做计算一样快，cache就可以消失，但是buffer依然存在。比如从网络上下载东西，瞬时速率可能会有较大变化，但从长期来看却是稳定的，这样就能通过引入一个buffer使得OS接收数据的速率更稳定，进一步减少对磁盘的伤害。

4、TLB（Translation Lookaside Buffer，翻译后备缓冲器）名字起错了，其实它是一个cache.

Mem-mapped I/O: Memory-mapped I/O uses the same address space to address both memory and I/O devices. The memory and registers of the I/O devices are mapped to (associated with) address values. So when an address is accessed by the CPU, it may refer to a portion of physical RAM, or it can instead refer to memory of the I/O device. Especially for large address spaces (graphics)

Page Cache/Buffer Cache: accelerates many accesses to files on non volatile storage. This happens because, when it first reads from or writes to data media like hard drives, Linux also stores data in unused areas of memory, which acts as a cache. If this data is read again later, it can be quickly read from this



Page Cache: caches pages rather than disk blocks using virtual memory techniques and addresses.

Unified buffer cache: uses the same page cache to cache both memory-mapped pages and ordinary file system I/O to avoid double caching

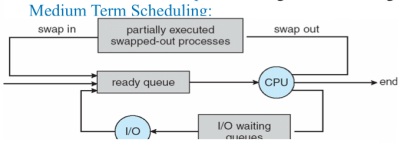
Recovery: consistent checking, backup (to other storage), restoring.

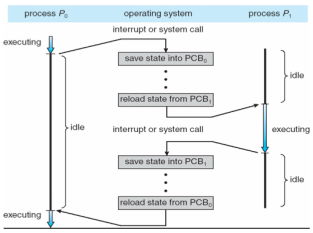
Longterm Scheduler: select which process -> ready Q (frequently invoked - milliseconds) control the degree of multi-programming

Shortterm Scheduler: select which process -> exec & CPU (infrequently invoked - seconds, minutes)

CPU-bound / IO-bound process: longterm sch finds good mix

Medium Term Scheduling:



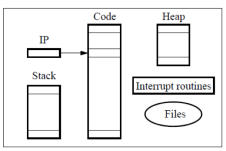


Context Switch

As a process executes, it changes state

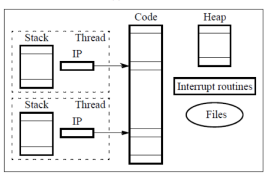
- **new**: The process is being created
- **running**: Instructions are being executed
- **waiting**: The process is waiting for some event to occur
- **ready**: The process is waiting to be assigned to a processor
- **terminated**: The process has finished execution

"heavyweight" process - completely separate program with its own variables, stack, and memory allocation.



(a) Process

Threads - *shares* the same memory space and global variables between routines.



(b) Threads

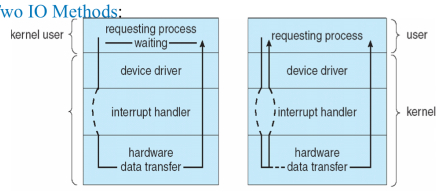
Blocking IO: process suspended until I/O completed

Non-Blocking IO: I/O call returns (when available) as much as available. Implemented by multithreading.

Asynchronous IO: in most cases, same as non-blocking; return immediately, not waiting for the IO; 不一定对signal作出反应

Well blocking IO means that a given thread cannot do anything more until the IO is fully received (in the case of sockets this wait could be a long time).
Non-blocking IO means an IO request is queued straight away and the function returns. The actual IO is then processed at some later point by the kernel.
For blocking IO you either need to accept that you are going to wait for every IO request or you are going to need to fire off a thread per request (Which will get very complicated very quickly).
For non-blocking IO you can send off multiple requests but you need to bear in mind that the data will not be available until some "later" point. This checking that the data has actually arrived is probably the most complicated part.

Two IO Methods:



before it can be accessed. Mount point: the root path that a FS will be mounted to. Mount timing: boot time, automatically at run-time, manually at run-time. Acess control list. Each user: (userID, groupID). Each file has 3 sets of attributes: owner, group,

Frame allocation. Fixed allocation, equal allocation, proportional allocation (process size). Priority allocation, proportional allocation based on priority. Local allocation: each process select from its own set of allocated frames. Global allocation: in mem, better performance.

Allocation methods. How disk blocks are alloca Contiguous Allocation, disk seeks are minimized, s random access. external fragmentation (compaction) grow. Extent-Based, an extent is a contiguous blo Linked, each block contains a pointer to the next block, sequ access (cluster of blocks), reliability. Indexed, directory trac addr of the file index block, index block stores block num fo *Free space mgt.* Free-space list: records all free disk block Vector (bitmap), HW support, bit operations. Linked List the first free block pointer in a special location on the dis caching in memory. Grouping, pointers to free blocks and a p

Interrupt mechanism also used for exceptions
Terminate process, crash system due to hardware error

Page fault executes when memory access error

System call executes via trap to trigger kernel to execute request

Multi-CPU systems can process interrupts concurrently
If operating system designed to handle it

Used for time-sensitive processing, frequent, must be fast

Process. A program in execution in memory. It includes: text section, data section, stack (temporary local vars and fcns), heap (dynamic allocated vars or classes), current activity (program coun register contents), a set of associated resources (e.g. open file handlers). Threads. All threads belonging to the same process share text section, data section, and OS resources. But each thread has its own thread ID, program counter, register set, and a stack. Process Control Block (PCB): Process state, Program counter, CPU registers, CPU scheduling information (e.g. priority), Memory-management information (e.g. base/limit register), I/O status information, Accounting information. Process states: New, Ready, Running, Waiting, Terminated. Context Switch: Kernel saves the state of the old process and loads the saved state for the new process. Context-switch time is purely overhead. Switch time depends on memory speed, number of registers, existence of special instructions, hardware support.

If a process does not have "enough" pages, the page-fault rate is very high
- Page fault to get page
- Replace existing frame
- But quickly need replaced frame back
- This leads to:
  - Low CPU utilization
  - Operating system thinking that it needs to increase the degree of multiprogramming
  - Another process added to the system

Thrashing □ a process is busy swapping pages in and out

Page table is kept in main memory

Page-table base register (PTBR) points to the page table

Page-table length register (PTLR) indicates size of the page table

In this scheme every data/instruction access requires two memory accesses
- One for the page table and one for the data / instruction

The two memory access problem can be solved by the use of a special fast-lookup hardware cache called associative memory or translation look-aside buffers (TLBs)

Some TLBs store address-space identifiers (ASIDs) in each TLB entry – uniquely identifies each process to provide address-space protection for that process
- Otherwise need to flush at every context switch

TLBs typically small (64 to 1,024 entries)

On a TLB miss, value is loaded into the TLB for faster access next time
- Replacement policies must be considered
- Some entries can be wired down for permanent fast access

Interrupt. Interrupts allow a device to change the flow of control in the CPU. Hardware may trigger an interrupt at any time by sending a signal to CPU. Software may trigger an interrupt either by an error or by a user request for an operating system service (trap). Interrupt transfers control to the interrupt service through interrupt vector, which contains the addresses (function pointer) of all the service routines. Architecture must save the address of the interrupted instruction. Incoming interrupts are disabled.

Polling can happen in 3 instruction cycles
- Read status, logical-and to extract status bit, branch if not zero
- How to be more efficient if non-zero infrequently?

CPU Interrupt-request line triggered by I/O device
- Checked by processor after each instruction

Interrupt handler receives interrupts
- Maskable to ignore or delay some interrupts

nterrupt vector to dispatch interrupt to correct handler
- Context switch at start and end
- Based on priority
- Some nonmaskable
- Interrupt chaining if more than one device at same interrupt number

Thrashing: a process is busy swapping pages in and out; cause low CPU utilization, OS think it should increase the degree of multiprogramming, add processes to the sys.

For each byte of I/O
- Read busy bit from status register until 0
- Host sets read or write bit and if write copies data into data-out register
- Host sets command-ready bit
- Controller sets busy bit, executes transfer
- Controller clears busy bit, error bit, command-ready bit when transfer done

Step 1 is busy-wait cycle to wait for I/O from device
- Reasonable if device is fast
- But inefficient if device slow
- CPU switches to other tasks?
  - But if miss a cycle data overwritten / lost

What is Operating System: 1.is system software that manages computer hardware, software resources, and provides common services for computer programs, which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. 2.also provides an interface for users to interact with the system(system call). 3.Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory and storage. The operating system coordinates all of this to make sure each program gets what it needs. 5.The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently.6. the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.(resoule allocator, Control program, interrupt driven)

Kenel and User mode: 1.The key difference between User Mode and Kernel Mode is that user mode is the mode in which the applications are running and kernel mode is the privileged mode to which the computer enters when accessing hardware resources.用户态 -->内核态： 由“中断”引发， 硬件自动完成转化的过程，触发中断信号意味着操作系统重新夺回控制权。内核态 --> 用户态： 操作系统执行一条特权指令，修改PSW的标志为“用户态”，意味着操作系统主动让出CPU控制权。（系统调用，内中断（异常），外中断（IO)). 2.用户态的应用程序不用关系这些复杂的计算资源的管理，只是需要使用内核提供的能力使用这些资源就好，将重心放在应用程序的逻辑上。3. 更加安全。如果让用户态的应用程序直接去操作计算资源，各种中各样的应用参差不齐，实现应用的人的水平也参差不齐，那么都去实现这么复杂的计算资源管理逻辑，很容易就将整个底层的资源给搞乱掉，导致全盘皆输。3.无法使用中断。中断在用户空间不可用

interrupt：1.interrupt transer control to the interrupt service routine generally through a interrupt vetor, which contains the address of all the service routine.interrupt architecture must save the address of the interrupted instruction. 2.signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the *Interrupt Service Routine (ISR)*.3.When a device raises an interrupt at let's say process i, the processor first completes the execution of instruction i. Then it loads the Program Counter (PC) with the address of the first instruction of the ISR. Before loading the Program Counter with the address, the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt the processor can continue with process i+1. 4.Two types.5.实现中断响应和中断返回，**实现优先权排队**，**中断**可以使CPU从**用户态切换为核心态，使操作系统获得计算机的控制权**。有了中断，才能实现多道程序并发执行。

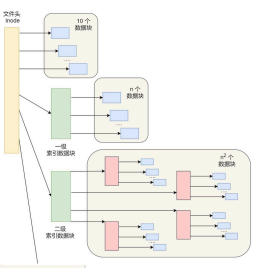
DMA channels are used to communicate data between the peripheral device and the system memory. All four system resources rely on certain lines on a bus. Some lines on the bus are used for IRQs, some for addresses (the I/O addresses and the memory address) and some for DMA channels.A DMA channel enables a device to transfer data without exposing the CPU to a work overload. Without the DMA channels, the CPU copies every piece of data using a peripheral bus from the I/O device. Using a peripheral bus occupies the CPU during the read/write process and does not allow other work to be performed until the operation is completed.

To overcome this problem a high-speed cache is set up for page table entries called a Translation Lookaside Buffer (TLB). Translation Lookaside Buffer (TLB) is nothing but a special cache used to keep track of recently used transactions. TLB contains page table entries that have been most recently used. Given a virtual address, the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as index while processing page table. TLB first checks if the page is already in main memory, if not in main memory a page fault is issued then the TLB is updated to include the new page entry. 1.hit:CPU generates virtual (logical) address. It is checked in TLB (present). Corresponding frame number is retrieved, which now tells where in the main memory page lies. 2.miss: CPU generates virtual (logical) address. It is checked in TLB (not present). Now the page number is matched to page table residing in main memory (assuming page table contains all PTE). Corresponding frame number is retrieved, which now tells where in the main memory page lies. The TLB is updated with new PTE (if space is not there, one of the replacement technique comes into picture i.e either FIFO, LRU or MFU etc).

MMIO: gives you a single address space and a common set of instructions for both data and I/O operations. You can define memory ordering rules and memory barriers that apply both to device accesses and normal memory. You don't need a whole separate set of opcodes for I/O instructions.

Process means a program is in execution, whereas thread means a segment of a process. A Process is not Lightweight, whereas Threads are Lightweight. A Process takes more time to terminate, and the thread takes less time to terminate. Process takes more time for creation, whereas Thread takes less time for creation.

连续空间存放方式顾名思义，文件存放在磁盘「连续的」物理空间中。这种模式下，文件的数据都是紧密相连，读写效率很高，但是有「磁盘空间碎片」和「文件长度不易扩展」的缺陷。非连续空间存放方式分为「链表方式」和「索引方式」。链表的方式存放是离散的，不用连续的，于是就可以消除磁盘碎片，可大大提高磁盘空间的利用率，同时文件的长度可以动态扩展。根据实现的方式的不同，链表可分为「隐式链表」和「显式链接」两种形式。因而不仅显著地提高了检索速度，而且大大减少了访问磁盘的次数。但也正是整个表都存放在内存中的关系，它的主要的缺点是不适用于大磁盘。索引的实现是为每个文件创建一个「索引数据块」，里面存放的是指向文件数据块的指针列表，说白了就像书的目录一样，要找哪个章节的内容，看目录查就可以。文件头需要包含指向「索引数据块」的指针，这样就可以通过文件头知道索引数据块的位置，再通过索引数据块里的索引信息找到对应的数据块索引的方式优点在于： - 文件的创建、增大、缩小很方便； - 不会有碎片的问题； - 支持顺序读写和随机读写.由于[索引数据]也是存放在磁盘块的，如果文件很小，明明只需一块就可以存放的下，但还是需要额外分配一块来存放索引数据，所以缺陷之一就是存储索引带来的开销。



方式	访问磁盘次数	优点	缺点
顺序分配	需访问磁盘 1 次	顺序**存取速度快**，**当文件是定长时**可以根据文件起始地址及记录长度**进行随机访问**	**要求连续的存储空间**，会**产生外部碎片**，**不利于文件的动态扩充**
链表分配	需访问磁盘 n 次	**无外部碎片**，**提高了外存空间的利用率**，**动态增长较方便**	只能按照文件的指针链顺序访问，**查找效率低**，指针信息存放消耗内存或磁盘空间
索引分配	m 级需访问磁盘 m+1 次	可以**随机访问**，**易于文件的增删**	索引表增加存储空间的开销，索引表的查找策略对文件系统效率影响较大

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.
8.	Process is called heavy weight process.	A Thread is lightweight as each thread in a process shares code, data and resources.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.
10.	If one process is blocked then it will not effect the execution of other process	Second thread in the same task could not run, while one server thread is blocked.
11.	Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.
12.	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
13.	Changes to the parent process does not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.