1、什么是进程（Process）和线程（Thread）？有何区别？

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动，进程是系统进行资源分配和调度的一个独立单位。线程是进程的一个实体，是 CPU 调度和分派的基本单位，它是比进程更小的能独立运行的基本单位。线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源（如程序计数器，一组寄存器和栈），但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源。一个线程可以创建和撤销另一个线程，同一个进程中的多个线程之间可以并发执行。

进程与应用程序的区别在于应用程序作为一个静态文件存储在计算机系统的硬盘等存储空间中，而进程则是处于动态条件下由操作系统维护的系统资源管理实体。

2、Windows 下的内存是如何管理的？

Windows 提供了 3 种方法来进行内存管理：虚拟内存，最适合用来管理大型对象或者结构数组；内存映射文件，最适合用来管理大型数据流（通常来自文件）以及在单个计算机上运行多个进程之间共享数据；内存堆栈，最适合用来管理大量的小对象。

Windows 操纵内存可以分两个层面：物理内存和虚拟内存。

其中物理内存由系统管理，不允许应用程序直接访问，应用程序可见的只有一个 2G 地址空间，而内存分配是通过堆进行的。对于每个进程都有自己的默认堆，当一个堆创建后，就通过虚拟内存操作保留了相应大小的地址块（不占有实际的内存，系统消耗很小）。当在堆上分配一块内存时，系统在堆的地址表里找到一个空闲块（如果找不到，且堆创建属性是可扩充的，则扩充堆大小），为这个空闲块所包含的所有内存页提交物理对象（在物理内存上或硬盘的交换文件上），这时就可以访问这部分地址。提交时，系统将对所有进程的内存统一调配，如果物理内存不够，系统试图把一部分进程暂时不访问的页放入交换文件，以腾出部分物理内存。释放内存时，只在堆中将所在的页解除提交（相应的物理对象被解除），继续保留地址空间。

如果要知道某个地址是否被占用/可不可以访问，只要查询此地址的虚拟内存状态即可。如果是提交，则可以访问。如果仅仅保留，或没保留，则产生一个软件异常。此外，有些内存页可以设置各种属性。如果是只读，向内存写也会产生软件异常。

3、Windows 消息调度机制是？

A）指令队列；B）指令堆栈；C）消息队列；D）消息堆栈

答案：C

处理消息队列的顺序。首先 Windows 绝对不是按队列先进先出的次序来处理的，而是有一定优先级的。优先级通过消息队列的状态标志来实现的。首先，最高优先级的是别的线程发过来的消息（通过 sendmessage）；其次，处理登记消息队列消息；再次处理 QS_QUIT 标志，处理虚拟输入队列，处理 wm_paint；最后是 wm_timer。

4、描述实时系统的基本特性

在特定时间内完成特定的任务，实时性与可靠性。

所谓“实时操作系统”，实际上是指操作系统工作时，其各种资源可以根据需要随时进行动态分配。由于各种资源可以进行动态分配，因此，其处理事务的能力较强、速度较快。

5、中断和轮询的特点

对 I/O 设备的程序轮询的方式，是早期的计算机系统对 I/O 设备的一种管理方式。它定时对各种设备轮流询问一遍有无处理要求。轮流询问之后，有要求的，则加以处理。在处理 I/O 设备的要求之后，处理机返回继续工作。尽管轮询需要时间，但轮询要比 I/O 设备的速度要快得多，所以一般不会发生不能及时处理的问题。当然，再快的处理机，能处理的输入输出设备的数量也是有一定限度的。而且，程序轮询毕竟占据了 CPU 相当一部分处理时间，因此，程序轮询是一种效率较低的方式，在现代计算机系统中已很少应用。

程序中断通常简称中断，是指 CPU 在正常运行程序的过程中，由于预先安排或发生了各种随机的内部或外部事件，使 CPU 中断正在运行的程序，而转到为响应的服务程序去处理。

轮询——效率低，等待时间很长，CPU 利用率不高。

中断——容易遗漏一些问题，CPU 利用率高。

6、什么是临界区？如何解决冲突？

每个进程中访问临界资源的那段程序称为临界区，每次只准许一个进程进入临界区，进入后不允许其他进程进入。

（1）如果有若干进程要求进入空闲的临界区，一次仅允许一个进程进入；

（2）任何时候，处于临界区内的进程不可多于一个。如已有进程进入自己的临界区，则其它所有试图进入临界区的进程必须等待；

（3）进入临界区的进程要在有限时间内退出，以便其它进程能及时进入自己的临界区；

（4）如果进程不能进入自己的临界区，则应让出 CPU，避免进程出现“忙等”现象。

7、说说分段和分页

页是信息的物理单位，分页是为实现离散分配方式，以消减内存的外零头，提高内存的利用率；或者说，分页仅仅是由于系统管理的需要，而不是用户的需要。

段是信息的逻辑单位，它含有一组其意义相对完整的信息。分段的目的是为了能更好的满足用户的需要。

页的大小固定且由系统确定，把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而一个系统只能有一种大小的页面。段的长度却不固定，决定于用户所编写的程序，通常由编辑程序在对源程序进行编辑时，根据信息的性质来划分。

分页的作业地址空间是一维的，即单一的线性空间，程序员只须利用一个记忆符，即可表示一地址。分段的作业地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址。

8、说出你所知道的保持进程同步的方法？

进程间同步的主要方法有原子操作、信号量机制、自旋锁、管程、会合、分布式系统等。

9、Linux 中常用到的命令

显示文件目录命令 ls　　如 ls

改变当前目录命令 cd　　如 cd /home

建立子目录 mkdir　　如 mkdir xiong

删除子目录命令 rmdir　　如 rmdir /mnt/cdrom

删除文件命令 rm　　如 rm /ucdos.bat

文件复制命令 cp　　如 cp /ucdos /fox

获取帮助信息命令 man　　如 man ls

显示文件的内容 less　　如 less mwm.lx

重定向与管道 type　　如 type readme>>direct，将文件 readme 的内容追加到文 direct 中

10、Linux 文件属性有哪些？（共十位）

-rw-r--r--那个是权限符号，总共是- --- --- ---这几个位。

第一个短横处是文件类型识别符：-表示普通文件；c 表示字符设备（character）；b 表示块设备（block）；d 表示目录（directory）；l 表示链接文件（link）；后面第一个三个连续的短横是用户权限位（User），第二个三个连续短横是组权限位（Group），第三个三个连续短横是其他权限位（Other）。每个权限位有三个权限，r（读权限），w（写权限），x（执行权限）。如果每个权限位都有权限存在，那么满权限的情况就是：-rwxrwxrwx；权限为空的情况就是- --- --- ---。

权限的设定可以用 chmod 命令，其格式位：chmod ugoa+/-/=rwx filename/directory。例如：

一个文件 aaa 具有完全空的权限- --- --- ---。

chmod u+rw aaa（给用户权限位设置读写权限，其权限表示为：- rw- --- ---）

chmod g+r aaa（给组设置权限为可读，其权限表示为：- --- r-- ---）

chmod ugo+rw aaa（给用户，组，其它用户或组设置权限为读写，权限表示为：- rw- rw- rw-）

如果 aaa 具有满权限- rwx rwx rwx。

chmod u-x aaa（去掉用户可执行权限，权限表示为：- rw- rwx rwx）

如果要给 aaa 赋予制定权限- rwx r-x r-x，命令为：

chmod u=rwx，go=rx aaa

11、makefile 文件的作用是什么？

一个工程中的源文件不计其数，其按类型、功能、模块分别放在若干个目录中。make file 定义了一系列的规则来指定哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作。因为 makefile 就像一个 Shell 脚本一样，其中也可以执行操作系统的命令。makefile 带来的好处就是——“自动化编译”。一旦写好，只需要一个 make 命令，整个工程完全自动编译，极大地提高了软件开发的效率。make 是一个命令工具，是一个解释 makefile 中指令的命令工具。一般来说，大多数的 IDE 都有这个命令，比如：Delphi 的 make，Visual C++的 nmake，Linux 下 GNU 的 make。可见，makefile 都成为了一种在工程方面的编译方法。

12、简术 OSI 的物理层 Layer1，链路层 Layer2，网络层 Layer3 的任务。

网络层：通过路由选择算法，为报文或分组通过通信子网选择最适当的路径。

链路层：通过各种控制协议，将有差错的物理信道变为无差错的、能可靠传输数据帧的数据链路。

物理层：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。

13、什么是中断？中断时 CPU 做什么工作？

中断是指在计算机执行期间，系统内发生任何非寻常的或非预期的急需处理事件，使得 CPU 暂时中断当前正在执行的程序而转去执行相应的事件处理程序。待处理完毕后又返回原来被中断处继续执行或调度新的进程执行的过程。

14、你知道操作系统的内容分为几块吗？什么叫做虚拟内存？他和主存的关系如何？内存管理属于操作系统的内容吗？

操作系统的主要组成部分：进程和线程的管理，存储管理，设备管理，文件管理。虚拟内存是一些系统页文件，存放在磁盘上，每个系统页文件大小为 4K，物理内存也被分页，每个页大小也为 4K，这样虚拟页文件和物理内存页就可以对应，实际上虚拟内存就是用于物理内存的临时存放的磁盘空间。页文件就是内存页，物理内存中每页叫物理页，磁盘上的页文件叫虚拟页，物理页+虚拟页就是系统所有使用的页文件的总和。

15、线程是否具有相同的堆栈？dll 是否有独立的堆栈？

每个线程有自己的堆栈。

dll 是否有独立的堆栈？这个问题不好回答，或者说这个问题本身是否有问题。因为 dll 中的代码是被某些线程所执行，只有线程拥有堆栈。如果 dll 中的代码是 exe 中的线程所调用，那么这个时候是不是说这个 dll 没有独立的堆栈？如果 dll 中的代码是由 dll 自己创建的线程所执行，那么是不是说 dll 有独立的堆栈？

以上讲的是堆栈，如果对于堆来说，每个 dll 有自己的堆，所以如果是从 dll 中动态分配的内存，最好是从 dll 中删除；如果你从 dll 中分配内存，然后在 exe 中，或者另外一个 dll 中删除，很有可能导致程序崩溃。

16、什么是缓冲区溢出？有什么危害？其原因是什么？

缓冲区溢出是指当计算机向缓冲区内填充数据时超过了缓冲区本身的容量，溢出的数据覆盖在合法数据上。

危害：在当前网络与分布式系统安全中，被广泛利用的 50%以上都是缓冲区溢出，其中最著名的例子是 1988 年利用 fingerd 漏洞的蠕虫。而缓冲区溢出中，最为危险的是堆栈溢出，因为入侵者可以利用堆栈溢出，在函数返回时改变返回程序的地址，让其跳转到任意地址，带来的危害一种是程序崩溃导致拒绝服务，另外一种就是跳转并且执行一段恶意代码，比如得到 shell，然后为所欲为。通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，使程序转而执行其它指令，以达到攻击的目的。

造成缓冲区溢出的主原因是程序中没有仔细检查用户输入的参数。

17、什么是死锁？其条件是什么？怎样避免死锁？

死锁的概念：在两个或多个并发进程中，如果每个进程持有某种资源而又都等待别的进程释放它或它们现在保持着的资源，在未改变这种状态之前都不能向前推进，称这一组进程产生了死锁。通俗地讲，就是两个或多个进程被无限期地阻塞、相互等待的一种状态。

死锁产生的原因主要是：？ 系统资源不足；？ 进程推进顺序非法。

产生死锁的必要条件：

（1）互斥（mutualexclusion），一个资源每次只能被一个进程使用；

（2）不可抢占（nopreemption），进程已获得的资源，在未使用完之前，不能强行剥夺；

（3）占有并等待（hold andwait），一个进程因请求资源而阻塞时，对已获得的资源保持不放；

（4）环形等待（circularwait），若干进程之间形成一种首尾相接的循环等待资源关系。

这四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

死锁的解除与预防：理解了死锁的原因，尤其是产生死锁的四个必要条件，就可以最大可能地避免、预防和解除死锁。所以，在系统设计、进程调度等方面注意如何不让这四个必要条件成立，如何确定资源的合理分配算法，避免进程永久占据系统资源。此外，也要防止进程在处于等待状态的情况下占用资源。因此，对资源的分配要给予合理的规划。

死锁的处理策略：鸵鸟策略、预防策略、避免策略、检测与恢复策略。

1、程序和进程

进程由两个部分组成：1）操作系统用来管理进程的内核对象。内核对象也是系统用来存放关于进程的统计信息的地方。2）地址空间。它包含所有可执行模块或 DLL 模块的代码和数据。它还包含动态内存分配的空间。如线程堆栈和堆分配空间。

	定义	使用系统运行资源情况
程序	计算机指令的集合，它以文件的形式存储在磁盘上。程序是静态实体（passive Entity），在多道程	不使用【程序不能申请系统资源，不能被系统调度，也不能作为独立运行

	序系统中，它是不能独立运行的，更不能与其他程序并发执行。	的单位，因此，它不占用系统的运行资源】。
进程	通常被定义为一个正在运行的程序的实例，是一个程序在其自身的地址空间中的一次执行活动。 定义：进程是进程实体（包括：程序段、相关的数据段、进程控制块 PCB）的运行过程，是系统进行资源分配和调度的一个独立单位。	使用【进程是资源申请、调度和独立运行的单位，因此，它使用系统中的运行资源。】

2、进程与线程

如果说操作系统引入进程的目的是为了提高程序并发执行，以提高资源利用率和系统吞吐量。那么操作系统中引入线程的目的，则是为了减少进程并发执行过程中所付出的时空开销，使操作系统能很好的并发执行。

进程 process 定义了一个执行环境，包括它自己私有的地址空间、一个句柄表，以及一个安全环境；线程则是一个控制流，有他自己的调用栈 call stack，记录了它的执行历史。

线程由两个部分组成：1）线程的内核对象，操作系统用它来对线程实施管理。内核对象也是系统用来存放线程统计信息的地方。2）线程堆栈，它用于维护线程在执行代码时需要的所有参数和局部变量。当创建线程时，系统创建一个线程内核对象。该线程内核对象不是线程本身，而是操作系统用来管理线程的较小的数据结构。可以将线程内核对象视为由关于线程的统计信息组成的一个小型数据结构。

进程与线程的比较如下：

比较	进程	线程
活泼性	不活泼（只是线程的容器）	活泼
地址空间	系统赋予的独立的虚拟地址空间（对于 32 位进程来说，这个地址空间是 4GB）	在进程的地址空间执行代码。线程只有一个内核对象和一个堆栈，保留的记录很少，因此所需要的内存也很少。因为线程需要的开销比进程少
调度	仅是资源分配的基本单位	独立调度、分派的基本单位
并发性	仅进程间并发（传统 OS）	进程间、线程间并发
拥有资源	资源拥有的基本单位	基本上不拥有资源

系统开销	创建、撤销、切换开销大	仅保存少量寄存器内容，开销小。

3、进程同步

进程同步的主要任务：是对多个相关进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作，从而使程序的执行具有可再现性。

同步机制遵循的原则：

（1）空闲让进；

（2）忙则等待（保证对临界区的互斥访问）；

（3）有限等待（有限代表有限的时间，避免死等）；

（4）让权等待，（当进程不能进入自己的临界区时，应该释放处理机，以免陷入忙等状态）。

4、进程间的通信是如何实现的？

进程通信，是指进程之间的信息交换（信息量少则一个状态或数值，多者则是成千上万个字节）。因此，对于用信号量进行的进程间的互斥和同步，由于其所交换的信息量少而被归结为低级通信。

所谓高级进程通信指：用户可以利用操作系统所提供的一组通信命令传送大量数据的一种通信方式。操作系统隐藏了进程通信的实现细节。或者说，通信过程对用户是透明的。

高级通信机制可归结为三大类：

（1）共享存储器系统（存储器中划分的共享存储区）；实际操作中对应的是“剪贴板”（剪贴板实际上是系统维护管理的一块内存区域）的通信方式，比如举例如下：word 进程按下 ctrl+c，在 ppt 进程按下 ctrl+v，即完成了 word 进程和 ppt 进程之间的通信，复制时将数据放入到剪贴板，粘贴时从剪贴板中取出数据，然后显示在 ppt 窗口上。

（2）消息传递系统（进程间的数据交换以消息（message）为单位，当今最流行的微内核操作系统中，微内核与服务器之间的通信，无一例外地都采用了消息传递机制。应用举例：邮槽（MailSlot）是基于广播通信体系设计出来的，它采用无连接的不可靠的数据传输。邮槽是一种单向通信机制，创建邮槽的服务器进程读取数据，打开邮槽的客户机进程写入数据。

（3）管道通信系统（管道即：连接读写进程以实现他们之间通信的共享文件（pipe 文件，类似先进先出的队列，由一个进程写，另一进程读））。实际操作中，管道分为：匿名管道、命名管道。匿名管道是一个未命名的、单向管道，通过父进程和一个子进程之间传输数据。匿名管道只能实现本地机器上两个进程之间的通信，而不能实现跨网络的通信。命名管道不仅可以在本机上实现两个进程间的通信，还可以跨网络实现两个进程间的通信。

	同一机器两个进程间通信	跨网络通信
剪贴板 Clipboard	可以	不可以
匿名管道 Pipe	可以	不可以
命名管道（点对点单一通信，数据量可较大）Namedpipe	可以	可以
邮槽（一对多，数据量较小，424 字节以下）Mailslot	可以	可以

5、线程同步

根据用户模式及内核模式下的同步方式的不同，分类及对比如下：

	内核对象/ 非内核对象	含义	缺点	适用
关键代码段（临界区）CriticalSection	非内核对象，工作在用户方式下，为用户模式对象	从程序代码的角度来控制线程的并发性	1.因为在等待进入关键代码段时无法设定超时值，所以其很容易进入死锁状态。2.不能跨进程使用。	单个进程中线程间的同步（同步速度快）
事件对象 Event	内核对象	所有内核对象中最基本的。	速度较慢（相比用户模式实现线程同步）	多个进程间的各个线程间实现同步
互斥对象 Mutex	内核对象	代表对一个资源的独占式访问		
信号量 Semaphore	内核对象	使用计数器来控制程序对一个共享资源的访问		

由于进程同步产生了一系列经典的同步问题“生产者-消费者”问题，“哲学家进餐”问题，“读者-写者”问题。

常见的操作系统使用的文件系统整理

文件系统是操作系统用于明确磁盘或分区上的文件的方法和数据结构；即在磁盘上组织文件的方法。也指用于存储文件的磁盘或分区，或文件系统种类。操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：与文件管理有关软件、被管理文件以及实施文件管理所需数据结构。从系统角度来看，文件系统是对文件存储器空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

【FAT】：

常 PC 机使用的文件系统是 FAT16。像基于 MS-DOS，Win 95 等系统都采用了 FAT 16 文件系统。在 Win 9X 下，FAT16 支持的分区最大为 2GB。我们知道计算机将信息保存在硬盘上称为“簇”的区域内。使用的簇越小，保存信息的效率就越高。在 FAT16 的情况下，分区越大簇就相应的要大，存储效率就越低，势必造成存储空间的浪费。并且随着计算机硬件和应用的不断提高，FAT16 文件系统已不能很好地适应系统的要求。在这种情况下，推出了增强的文件系统 FAT32。同 FAT16 相比，FAT32 主要具有以下特点：

1、同 FAT16 相比 FAT32 最大的优点是可以支持的磁盘大小达到 32G，但是不能支持小于 512MB 的分区。

*基于 FAT32 的 Win 2000 可以支持分区最大为 32GB；而基于 FAT16 的 Win 2000 支持的分区最大为 4GB。

2、由于采用了更小的簇，FAT32 文件系统可以更有效率地保存信息。如两个分区大小都为 2GB，一个分区采用了 FAT16 文件系统，另一个分区采用了 FAT32 文件系统。采用 FAT16 的分区的簇大小为 32KB，而 FAT32 分区的簇只有 4KB 的大小。这样 FAT32 就比 FAT16 的存储效率要高很多，通常情况下可以提高 15%。

3、FAT32 文件系统可以重新定位根目录和使用 FAT 的备份副本。另外 FAT32 分区的启动记录被包含在一个含有关键数据的结构中，减少了计算机系统崩溃的可能性。

【NTFS】：

NTFS 文件系统是一个基于安全性的文件系统，是 Windows NT 所采用的独特的文件系统结构，它是建立在保护文件和目录数据基础上，同时照顾节省存储资源、减少磁盘占用量的一种先进的文件系统。使用非常广泛的 Windows NT 4.0 采用的就是 NTFS 4.0 文件系统，相信它所带来的强大的系统安全性一定给广大用户留下了深刻的印象。Win 2000 采用了更新版本的 NTFS 文件系统？？NTFS 5.0，它的推出使得用户不但可以像 Win 9X 那样方便快捷地操作和管理计算机，同时也可享受到 NTFS 所带来的系统安全性。

NTFS 5.0 的特点主要体现在以下几个方面：

1、NTFS 可以支持的分区（如果采用动态磁盘则称为卷）大小可以达到 2TB。而 Win 2000 中的 FAT32 支持分区的大小最大为 32GB。

2、NTFS 是一个可恢复的文件系统。在 NTFS 分区上用户很少需要运行磁盘修复程序。NTFS 通过使用标准的事物处理日志和恢复技术来保证分区的一致性。发生系统失败事件时，NTFS 使用日志文件和检查点信息自动恢复文件系统的一致性。

3、NTFS 支持对分区、文件夹和文件的压缩。任何基于 Windows 的应用程序对 NTFS 分区上的压缩文件进行读写时不需要事先由其他程序进行解压缩，当对文件进行读取时，文件将自动进行解压缩；文件关闭或保存时会自动对文件进行压缩。

4、NTFS 采用了更小的簇，可以更有效率地管理磁盘空间。在 Win 2000 的 FAT32 文件系统的情况下，分区大小在 2GB～8GB 时簇的大小为 4KB；分区大小在 8GB～16GB 时簇的大小为 8KB；分区大小在 16GB～32GB 时，簇的大小则达到了 16KB。而 Win 2000 的 NTFS 文件系统，当分区的大小在 2GB 以下时，簇的大小都比相应的 FAT32 簇小；当分区的大小在 2GB 以上时（2GB～2TB），簇的大小都为 4KB。相比之下，NTFS 可以比 FAT32 更有效地管理磁盘空间，最大限度地避免了磁盘空间的浪费。

5、在 NTFS 分区上，可以为共享资源、文件夹以及文件设置访问许可权限。许可的设置包括两方面的内容：一是允许哪些组或用户对文件夹、文件和共享资源进行访问；二是获得访问许可的组或用户可以进行什么级别的访问。访问许可权限的设置不但适用于本地计算机的用户，同样也应用于通过网络的共享文件夹对文件进行访问的网络用户。与 FAT32 文件系统下对文件夹或文件进行访问相比，安全性要高得多。另外，在采用 NTFS 格式的 Win 2000 中，应用审核策略可以对文件夹、文件以及活动目录对象进行审核，审核结果记录在安全日志中，通过安全日志就可以查看哪些组或用户对文件夹、文件或活动目录对象进行了什么级别的操作，从而发现系统可能面临的非法访问，通过采取相应的措施，将这种安全隐患减到最低。这些在 FAT32 文件系统下，是不能实现的。

6、在 Win 2000 的 NTFS 文件系统下可以进行磁盘配额管理。磁盘配额就是管理员可以为用户所能使用的磁盘空间进行配额限制，每一用户只能使用最大配额范围内的磁盘空间。设置磁盘配额后，可以对每一个用户的磁盘使用情况进行跟踪和控制，通过监测可以标识出超过配额报警阈值和配额限制的用户，从而采取相应的措施。磁盘配额管理功能的提供，使得管理员可以方便合理地为用户分配存储资源，避免由于磁盘空间使用的失控可能造成的系统崩溃，提高了系统的安全性。

7、NTFS 使用一个“变更”日志来跟踪记录文件所发生的变更。

【Ext2】：

Ext2 是 GNU/Linux 系统中标准的文件系统，其特点为存取文件的性能极好，对于中小型的文件更显示出优势，这主要得利于其簇快取层的优良设计。

其单一文件大小与文件系统本身的容量上限与文件系统本身的簇大小有关，在一般常见的 x86 电脑系统中，簇最大为 4KB，则单一文件大小上限为 2048GB，而文件系统的容量上限为 16384GB。

但由于目前核心 2.4 所能使用的单一分割区最大只有 2048GB，实际上能使用的文件系统容量最多也只有 2048GB。

至于 Ext3 文件系统，它属于一种日志文件系统，是对 ext2 系统的扩展。它兼容 ext2，并且从 ext2 转换成 ext3 并不复杂。

【Ext3】：

Ext3 是一种日志式文件系统，是对 ext2 系统的扩展，它兼容 ext2。日志式文件系统的优越性在于：由于文件系统都有快取层参与运作，如不使用时必须将文件系统卸下，以

便将快取层的资料写回磁盘中。因此每当系统要关机时，必须将其所有的文件系统全部 shutdown 后才能进行关机。

如果在文件系统尚未 shutdown 前就关机 （如停电） 时，下次重开机后会造成文件系统的资料不一致，故这时必须做文件系统的重整工作，将不一致与错误的地方修复。然而，此一重整的工作是相当耗时的，特别是容量大的文件系统，而且也不能百分之百保证所有的资料都不会流失。

为了克服此问题，使用所谓'日志式文件系统 （Journal File System） '。此类文件系统最大的特色是，它会将整个磁盘的写入动作完整记录在磁盘的某个区域上，以便有需要时可以回溯追踪。

由于资料的写入动作包含许多的细节，像是改变文件标头资料、搜寻磁盘可写入空间、一个个写入资料区段等等，每一个细节进行到一半若被中断，就会造成文件系统的不一致，因而需要重整。

然而，在日志式文件系统中，由于详细纪录了每个细节，故当在某个过程中被中断时，系统可以根据这些记录直接回溯并重整被中断的部分，而不必花时间去检查其他的部分，故重整的工作速度相当快，几乎不需要花时间。

【Ext4】：

Linux kernel 自 2.6.28 开始正式支持新的文件系统 Ext4。Ext4 是 Ext3 的改进版，修改了 Ext3 中部分重要的数据结构，而不仅仅像 Ext3 对 Ext2 那样，只是增加了一个日志功能而已。Ext4 可以提供更佳的性能和可靠性，还有更为丰富的功能：

1、与 Ext3 兼容。执行若干条命令，就能从 Ext3 在线迁移到 Ext4，而无须重新格式化磁盘或重新安装系统。原有 Ext3 数据结构照样保留，Ext4 作用于新数据，当然，整个文件系统因此也就获得了 Ext4 所支持的更大容量。

2、更大的文件系统和更大的文件。较之 Ext3 目前所支持的最大 16TB 文件系统和最大 2TB 文件，Ext4 分别支持 1EB（1，048，576TB， 1EB=1024PB， 1PB=1024TB）的文件系统，以及 16TB 的文件。

3、无限数量的子目录。Ext3 目前只支持 32，000 个子目录，而 Ext4 支持无限数量的子目录。

4、Extents。Ext3 采用间接块映射，当操作大文件时，效率极其低下。比如一个 100MB 大小的文件，在 Ext3 中要建立 25，600 个数据块（每个数据块大小为 4KB）的映射表。而 Ext4 引入了现代文件系统中流行的 extents 概念，每个 extent 为一组连续的数据块，上述文件则表示为"该文件数据保存在接下来的 25，600 个数据块中"，提高了不少效率。

5、多块分配。当写入数据到 Ext3 文件系统中时，Ext3 的数据块分配器每次只能分配一个 4KB 的块，写一个 100MB 文件就要调用 25，600 次数据块分配器，而 Ext4 的多块分配器"multiblock allocator"（mballoc） 支持一次调用分配多个数据块。

6、延迟分配。Ext3 的数据块分配策略是尽快分配，而 Ext4 和其它现代文件操作系统的策略是尽可能地延迟分配，直到文件在 cache 中写完才开始分配数据块并写入磁盘，这样就能优化整个文件的数据块分配，与前两种特性搭配起来可以显著提升性能。

7、快速 fsck。以前执行 fsck 第一步就会很慢，因为它要检查所有的 inode，现在 Ext4 给每个组的 inode 表中都添加了一份未使用 inode 的列表，今后 fsck Ext4 文件系统就可以跳过它们而只去检查那些在用的 inode 了。

8、日志校验。日志是最常用的部分，也极易导致磁盘硬件故障，而从损坏的日志中恢复数据会导致更多的数据损坏。Ext4 的日志校验功能可以很方便地判断日志数据是否损坏，而且它将 Ext3 的两阶段日志机制合并成一个阶段，在增加安全性的同时提高了性能。

9、“无日志”（No Journaling）模式。日志总归有一些开销，Ext4 允许关闭日志，以便某些有特殊需求的用户可以借此提升性能。

10、在线碎片整理。尽管延迟分配、多块分配和 extents 能有效减少文件系统碎片，但碎片还是不可避免会产生。Ext4 支持在线碎片整理，并将提供 e4defrag 工具进行个别文件或整个文件系统的碎片整理。

11、inode 相关特性。Ext4 支持更大的 inode，较之 Ext3 默认的 inode 大小 128 字节，Ext4 为了在 inode 中容纳更多的扩展属性（如纳秒时间戳或 inode 版本），默认 inode 大小为 256 字节。Ext4 还支持快速扩展属性（fast extended attributes）和 inode 保留（inodes reservation）。

12、持久预分配（Persistent preallocation）。P2P 软件为了保证下载文件有足够的空间存放，常常会预先创建一个与所下载文件大小相同的空文件，以免未来的数小时或数天之内磁盘空间不足导致下载失败。Ext4 在文件系统层面实现了持久预分配并提供相应的 API（libc 中的 posix_fallocate（）），比应用软件自己实现更有效率。

13、默认启用 barrier。磁盘上配有内部缓存，以便重新调整批量数据的写操作顺序，优化写入性能，因此文件系统必须在日志数据写入磁盘之后才能写 commit 记录，若 commit 记录写入在先，而日志有可能损坏，那么就会影响数据完整性。Ext4 默认启用 barrier，只有当 barrier 之前的数据全部写入磁盘，才能写 barrier 之后的数据。（可通过“mount -o barrier=0”命令禁用该特性。）

【ZFS】：

ZFS 源自于 Sun Microsystems 为 Solaris 操作系统开发的文件系统。ZFS 是一个具有高存储容量、文件系统与卷管理概念整合、崭新的磁盘逻辑结构的轻量级文件系统，同时也是一个便捷的存储池管理系统。ZFS 是一个使用 CDDL 协议条款授权的开源项目。

【HFS】：

1、HFS 文件系统概念

分层文件系统（Hierarchical File System，HFS）是一种由苹果电脑开发，并使用在 Mac OS 上的文件系统。最初被设计用于软盘和硬盘，同时也可以在在只读媒体如 CD-ROM 上见到。

2、HFS 文件系统开发过程

HFS 首次出现在 1985 年 9 月 17 日，作为 Macintosh 电脑上新的文件系统。它取代只用于早期 Mac 型号所使用的平面文件系统 Macintosh File System（MFS）。因为 Macintosh 电脑所产生的数据，比其它通常的文件系统，如 DOS 使用的 FAT 或原始 Unix 文件

系统所允许存储的数据更多。苹果电脑开发了一种新式更适用的文件系统，而不是采用现有的规格。例如，HFS 允许文件名最多有 31 个字符的长度，支持 metadata 和双分支（每个文件的数据和资源支分开存储）文件。

尽管 HFS 象其它大多数文件系统一样被视为专有的格式，因为只有它为大多数最新的操作系统提供了很好的通用解决方法以存取 HFS 格式磁盘。

在 1998 年，苹果电脑发布了 HFS Plus，其改善了 HFS 对磁盘空间的地址定位效率低下，并加入了其它的改进。当前版本的 Mac OS 仍旧支持 HFS，但从 Mac OS X 开始 HFS 卷不能作为启动用。

3、构成方式

分层文件系统把一个卷分为许多 512 字节的“逻辑块”。这些逻辑块被编组为“分配块”，这些分配块可以根据卷的尺寸包含一个或多个逻辑块。HFS 对地址分配块使用 16 位数值，分配块的最高限制数量是 65536。

组成一个 HFS 卷需要下面的五个结构：

1）卷的逻辑块 0 和 1 是启动块，它包含了系统启动信息。例如，启动时载入的系统名称和壳（通常是 Finder）文件。

2）逻辑块 2 包含主目录块（Master Directory Block，简称 MDB）。

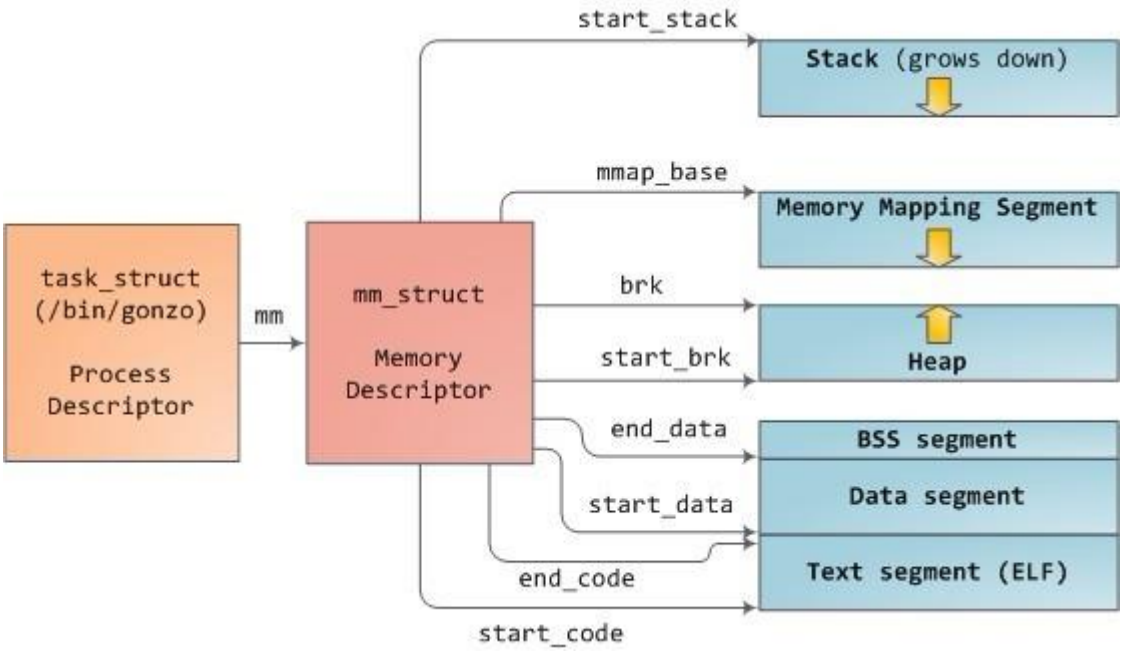
3）逻辑块 3 是卷位图（Volume Bitmap）的启动块，它追踪分配块使用状态。

4）总目录文件（Catalog File）是一个包含所有文件的记录和储存在卷中目录的 B*-tree。

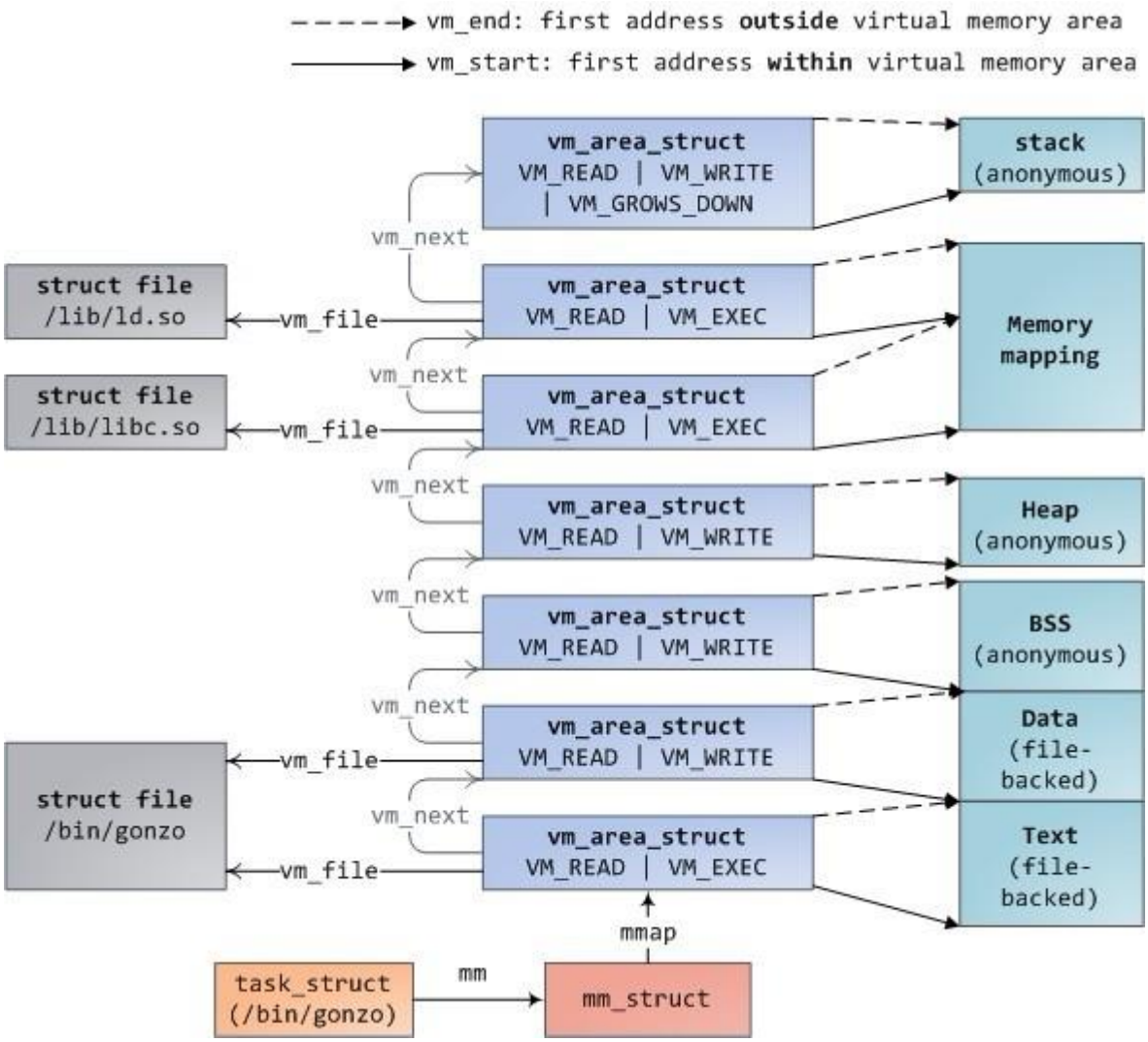
5）扩展溢出文件（Extent Overflow File）是当最初总目录文件中三个扩展占用后，另外一个包含额外扩展记录的分配块对应信息的 B*-tree。

内核怎样管理你的内存

在分析了进程的虚拟地址布局，我们转向内核以及他管理用户内存的机制。下图是 gonzo 的例子：



Linux 进程在内核中是由 task_struct 进程描述符实现的，task_struct 的 mm 字段指向内存描述符 mm_struct，他是进程的一个内存执行摘要。如上图所示，mm_struct 存储了内存各个段的开始和结束地址、进程所使用的内存页面数（rss 代表常驻集合大小）、使用的虚拟地址空间总数等等。在内存描述符中我们也可以找到两个用于管理进程内层的字段：虚拟内存集合和页表。Gonzo 的内存区域如下图：



每个虚拟内存区域（VMA）是一个虚拟地址空间上连续的区域；这些区域不会彼此覆盖。Vm_area_struct 结构描述了一个内存区域，包括他的开始和技术地址、flags 字段指定了他的行为和访问权限，vm_file 字段指定了该区域映射的实际文件。一个没有映射文件的 VMA 成为匿名的。除了内存映射段以外，上面的每个内存段（堆、栈等等）相当于一个单独的 VMA。这不是必须的，尽管在 x86 机器上通常是这样。VMA 不会关心他在哪个段里面。

一个进程的所有 VMA 以两种方式存储在他的内存描述符中，一种是以链表的方式存放在 mmap 字段，以开始虚拟地址进行了排序，另一种是以红黑树的方式存放，mm_rb 字段为这颗红黑树的根。红黑树可以让内核根据给定的虚拟地址快速地找到内存区域。当我们读取文件/proc/pid_of_process/maps，内核仅仅是通过进程 VMA 的链接同时打印出每一个。