第五章作业

**1．说明Linux系统的体系结构分为哪几层。**

Linux系统大致可分为三层：1.靠近硬件的底层是内核，即Linux操作系统常驻内存部分。2.中间层是内核之外的shell层，即操作系统的系统程序部分。

3.最高层是应用层，即用户程序部分。

**2．说明Linux系统核心结构的组成情况。**

内存管理、进程管理、设备驱动程序、文件系统和网络管理等。

**3．什么是进程？什么是线程？Linux系统中的进程有哪些状态？如何获取系统中各进程的状态？**

进程：程序的一次执行过程。线程：进程中指令的不同执行路线。

进程状态：运行态、可中断等待态、不可中断等待态、停止态和僵死态。

获取进程状态：使用ps命令。可以使用man来查看ps的使用参数，以下是几个常使用到得

a 显示当前终端的所有进程信息

u 使用以用户为主的格式输出进程信息

x 显示当前用户在所有终端下的进程信息

-e 显示 系统内的所有进程

# ps 只显示当前用户打开的进程

]# ps aux 显示系统中所有进程信息

**5.4Linux系统中进程有哪两种模式？各有何特点？**

两种模式：用户模式和内核模式。特点：用户模式下运行的是内核之外的系统程序，内核模式下运行的是操作系统（即核心）程序。

**5．Linux系统中进程控制块的作用是什么？它与进程有何关系？**

进程控制块(PCB)的作用：进程控制块是进程组成中最关键的部分,它保存有进程的重要信息。关系：（1）每个进程有唯一的PCB。（2）操作系统根据PCB对进程实施控制和管理。(3）进程的动态、并发等特征是利用PCB.表现出来的。

（4）PCB是进程存在的唯一标志。

**6． Linux如何执行进程调度？**

查找所有在就绪队列中的进程,从中选出优先级最高且在内存的一个进程,如果队列中有实时进程,那么实时进程将优先运行。如果最需要运行的不是当前进程，则挂起当前进程，并保护它的现场，然后为选中的进程恢复运行现场。

**7．shell的基本工作过程是怎样的？**

（1）读取用户由键盘输入的命令行。（2）分析命令，以命令名作为文件名，其他参数改造为系统调用execve()内部处理所要求的形式。（3）终端进程调用fork()建立一个子进程。（4）终端进程本身用系统调用wait4()来等待子进程完成（如果是后台命令，则不等待）。当子进程运行时调用execve()，子进程根据文件名（即命令名）到目录中查找有关文件（这是命令解释程序构成的文件），调入内存，执行这个程序（即执行这条命令）。（5）如果命令末尾有&号（后台命令符），则终端进程不用执行系统调用wait4()，而是立即发提示符，让用户输入下一个命令，转步骤（1）。如果命令末尾没有&号，则终端进程要一直等待，当子进程（即运行命令的进程）完成工作后要终止时，向父进程（终端进程）报告，此时终端进程醒来，在做必要的判断等工作后，终端进程发提示符，让用户输入新的命令，重复上述处理过程。

**8． Linux系统一般采用哪种文件系统？其构造形式如何？**

一般采用ext2文件系统。ext2文件系统中的文件信息都保存在数据块中。在同一个ext2文件系统中，所有数据块的大小一样。ext2文件系统将逻辑分区划分成块组，每个块组重复保存一些有关整个文件系统的关键信息，以及实际的文件和目录的数据块。系统引导块总是介质上的第一数据块,只有根文件系统才放有引导程序。

**9．什么是块组？什么是超级块？超级块的功能是什么？**

Linux用的是虚拟文件系统，块组是相当于windows系统的硬盘分区，比如c盘d盘。超级块位于块组的最前面，描述文件系统整体信息的数据结构，主要描述文件系统的目录和文件的静态分布情况，以及描述文件系统的各种组成结构的尺寸、数量等。

**10．什么是索引节点？索引节点主要有哪些内容？它与文件有何关系？**

索引节点是一个结构，它包含了一个文件的长度、创建及修改时间、权限、所属关系、磁盘中的位置等信息。一个文件系统维护了一个索引节点的数组，每个文件或目录都与索引节点数组中的唯一一个元素对应。系统给每个索引节点分配了一个号码，也就是该节点在数组中的索引号，称为索引节点号。

索引节点与文件的关系：Linux为每个文件分配一个称为索引节点的号码inode，可以将inode简单理解成一个指针，它永远指向本文件的具体存储位置。系统是通过索引节点(而不是文件名)来定位每一个文件。文件系统处理文件所需要的所有信息都放在称为索引节点的数据结构中。文件名可以随时更改，但是索引节点对文件是唯一的，并且随文件的存在而存在。一个文件系统允许的inode节点数是有限的，如果文件数量太多，即使每个文件都是0字节的空文件，系统最终也会因为节点空间耗尽而不能再创建文件.

**11．为什么要设立虚拟文件系统（VFS）？它与实际文件系统的关系是怎样的？**

Linux系统支持多种文件系统，必须使用一种统一的接口，而VFS可以实现；VFS是建立在具体文件系统之上的，它为用户程序提供一个统一的、抽象的、虚拟的文件系统界面。

**12． Linux系统通常为什么要把硬盘划分为多个文件系统？简述文件系统安装的基本过程。**

Linux系统支持多种不同的文件系统，如ext，FAT，ext2，ext3，MINIX，MS DOS，SYSV等。

安装过程：在系统初启时，往往只有一个文件系统安装完成，即根文件系统。其他的文件系统（如由软盘构成的文件系统）可以根据需要（如从硬盘向软盘复制文件），作为子系统动态地安装到主系统中。

**13． Linux系统采用哪两种内存管理技术？各自的基本实现思想是什么？**

Linux系统采用交换和请求分页存储管理两种内存管理技术。（1）请求分页的基本思想是，当执行一个程序时，才把它换入内存，但并不把全部程序都调入内存，而是用到哪一页就调入哪一页；（2）内存交换的基本思想是，当系统中出现内存不足时，Linux内存管理子系统就需要释放一些内存页，从而增加系统中空闲内存页的数量。

**14．何谓虚拟存储器？Linux系统如何支持虚存？**

虚拟存储器：以透明的方式给用户提供一个比实际内存大的多的作业地址空间。它不是任何实际的物理存储器，而是一个非常大的存储器的逻辑模型。

用户程序（进程）通过有关文件系统操作的系统调用界面进入系统空间，然后经由VFS才可使用Linux系统中具体的文件系统。就是说，VFS是建立在具体文件系统之上的，它为用户程序提供一个统一的、抽象的、虚拟的文件系统界面。这个抽象的界面主要由一组标准的、抽象的有关文件操作构成，以系统调用的形式提供用户程序，如read()，write()，lseek()等。所以VFS必须管理所有的文件系统。它通过使用描述整个VFS的数据结构和描述实际安装的文件系统的数据结构来管理这些不同的文件系统。

**15． Linux系统中交换空间为何采用连续空间？**

作为交换空间的交换条件实际就是普通文件，但它们所占的磁盘空间必须是连续的，即文件中不能存在“空洞”。因为进程使用交换空间是临时性的，速度是关键性问题，系统一次进行多个盘块I/O传输比每次一块、多次传输的速度要快，所以核心在交换设备上是分配一片连续空间，而不管碎片的问题。

**16． Linux为什么要采用三级页表？该机制如何工作？**

由于Linux系统中页面的大小为4GB，进程虚存空间要划分为 个页面。如果直接用页表描述这种映射关系，那么每个进程的面表就要有个表项，而用大量的内存资源来存放页表办法是不可取，故Linux采用三级页表；

把一个线性地址映射成物理地址就分为以下四步：（1）以线性地址中最高位段作为下标在PGD中找到相应的表现，该表项指向相应的PMD。（2）以线性地址中第二个位段作为下标在PMD中找到相应的表项，该表项指向相应的PT。（3）以线性地址中第三个位段作为下标在PT中找到相应的表项，该表项指向相应的物理页面（即该物理页面的起始地址）（4）以线性地址中的最低位段是物理页面内的相对位移量，此位移量与该物理页面的起始地址相加就得到相应的物理地址。

**17． Linux信号机制是如何实现进程通信的？**

Linux信号机制基本上是从Unix系统中继承过来的。早期Unix系统中的信号机制比较简单和原始，后来在实践中暴露出一些问题，因此，把那些建立在早期机制上的信号叫做"不可靠信号"，信号值小于SIGRTMIN(Red hat 7.2中，SIGRTMIN=32，SIGRTMAX=63)的信号都是不可靠信号。这就是"不可靠信号"的来源。它的主要问题是：

1.进程每次处理信号后，就将对信号的响应设置为默认动作。在某些情况下，将导致对信号的错误处理；因此，用户如果不希望这样的操作，那么就要在信号处理函数结尾再一次调用signal()，重新安装该信号。

2.信号可能丢失，后面将对此详细阐述。因此，早期Unix下的不可靠信号主要指的是进程可能对信号做出错误的反应以及信号可能丢失。

Linux支持不可靠信号，但是对不可靠信号机制做了改进：在调用完信号处理函数后，不必重新调用该信号的安装函数（信号安装函数是在可靠机制上的实现）。因此，Linux下的不可靠信号问题主要指的是信号可能丢失。

信号值位于SIGRTMIN和SIGRTMAX之间的信号都是可靠信号，可靠信号克服了信号可能丢失的问题。Linux在支持新版本的信号安装函数sigation()以及信号发送函数sigqueue()的同时，仍然支持早期的signal()信号安装函数，支持信号发送函数kill()。 注：不要有这样的误解：由sigqueue()发送、sigaction安装的信号就是可靠的。事实上，可靠信号是指后来添加的新信号（信号值位于SIGRTMIN及SIGRTMAX之间）；不可靠信号是信号值小于SIGRTMIN的信号。信号的可靠与不可靠只与信号值有关，与信号的发送及安装函数无关。目前linux中的signal()是通过sigation()函数实现的，因此，即使通过signal()安装的信号，在信号处理函数的结尾也不必再调用一次信号安装函数。同时，由signal()安装的实时信号支持排队，同样不会丢失。

**18．管道文件如何实现两个进程间的通信？**

例如：$ls|more

在执行这个命令时，要创建一个管道文件和两个进程：“|”对应管道文件；命令ls对应一个进程，它向该文件中写入信息，称为写进程；命令more对应另一个进程，它从文件中读出信息，称为读进程。由系统自动处理二者间的同步、调度和缓冲。管道文件允许两个进程按“先入先出（FIFO）”方式传送数据，而它们可以彼此不知道对方的存在。管道文件不属于用户直接命名的普通文件，它是利用系统调用pipe()创建的、在同族进程间进行大量信息传送的打开文件（其实只存在于内存中）。

**19． Linux系统中设备驱动分层结构是怎样的？如何实现与设备的无关性？**

在面向对象的程序设计中，可以为某一类相似的事物定义一个基类，而具体的事物可以继承这个基类中的函数。如果对于继承的这个事物而言，其某函数的实现与基类一致，那它就可以直接继承基类的函数；相反，它可以重载之。这种面向对象的设计思想极大地提高了代码的可重用能力，是对现实世界事物间关系的一种良好呈现。

Linux内核完全由C语言和汇编语言写成，但是却频繁用到了面向对象的设计思想。在设备驱动方面，往往为同类的设备设计了一个框架，而框架中的核心层则实现了该设备通用的一些功能。同样的，如果具体的设备不想使用核心层的函数，它可以重载之。

Linux系统采用设备文件统一管理硬件设备，从而将硬件设备的特性及管理细节对用户隐藏起来，实现用户程序与设备无关性。

**20． Linux系统中可安装模块的思想是什么？**

可安装模块是可以在系统运行时动态地安装和拆卸的内核模块，即经过编译但尚未连接的目标文件（后缀为.o）。利用这一机制，根据需要在不必对内核重新编译连接的条件下，将可安装模块动态插入运行中的内核，成为其中一个有机组成部分或者从内核卸载已安装的模块。设备驱动程序或者与设备驱动紧密相关的部分（如文件系统）都是利用可安装模块实现的。

**21．什么是中断？中断的一般处理过程是什么？**

中断是CPU对系统发生的某个事件作出的一种反应，CPU暂停正在执行的程序，保留现场后，自动地转去执行相应的处理程序，处理完该事件后，再返回断点继续执行被“打断”的程序。

中断处理的一般过程：（1）保存正在运行进程的各寄存器的内容，把它们放入核心栈的新帧面中。（2）确定“中断源”或者检查中断发生，识别中断的类型（如时钟中断或盘中断）和中断的设备号（如哪个磁盘引起的中断）。中断向量因机器而异，但通常都包括相应中断处理程序入口地址和中断处理时处理机的状态字。（3）核心调用中断处理程序，对中断进行处理。（4）中断处理完成并返回。中断处理程序执行完以后，核心便执行与机器相关的特定指令序列，恢复中断时所保留的各寄存器内容，并执行核心栈的退栈操作，进程回到用户态。如果设置了重调度标志，则在本进程返回到用户态时进行进程调度。

**22． Linux系统怎样处理系统调用？**

Linux的系统调用是通过中断指令“INT 0x80”实现的，当CPU执行到中断指令“INT 0x80”时，硬件就做出一系列响应。CPU穿过陷阱门，从用户空间进入系统空间。相应地，进程的上下文从用户堆栈切换到系统堆栈。接着运行内核函数system\_call()。首先，进一步保存各寄存器内容；接着调用system\_trace()，以系统调用号为下标检索系统调用入口表sys\_call\_table，从中找到相应的函数；然后，转去执行该函数，完成具体的服务。