Unix 考试复习大纲(2018.11.8)

1.UNIX 操作系统最根本的功能特征是什么？包括哪些最基本的概念？

分时多用户、开放性。

分时多用户：多个用户多个进程同时在一个系统中运行，系统资源高度共享、有效协调

并发开放性：标准化，结构上的一致性

可移植性：应用软件的编码及系统应用接口

可互操作性：可保持用户原来的使用习惯、异种机之间的互操作

(1).交互式分时多用户：人机间实时交互数据，多个用户可同时使用一台机器，每个用户可同时执行多个任务

(2).软件复用：每个程序模块完成单一的功能，程序模块可按需任意组合，较高的系统和应用开发效率

(3).可移植性强：数千行汇编码，数十万行 C 语言代码

(4).配置灵活，适应性强：小内核，参数灵活可调,核外应用系统，任意裁减，限制规则很少

(5). 界面方便高效：内部: 系统调用丰富高效,外部: shell 命令灵活方便可编程,应用: GUI 清晰直观功能强大

(6)安全机制完善：口令、权限、加密等措施完善，抗病毒结构,误操作的局限和自动恢复功能

(7).多国语言支持，支持全世界现有的几十种主要语言

(8).网络和资源共享：内部: 多进程结构易于资源共享，外部: 支持多种网络协议

2、操作系统核心是什么？核心通过什么方式和什么原则向上层应用程序提供了哪些服务？

系统调用的集合及实现系统调用的内部算法就形成操作系统核心。

内核必须要提供的功能：中断处理、短程调度、原语管理

以函数形式提供给核外的命令和上层应用系统使用的一组程序,涵盖操作系统的所有功能。是应用程序请求操作系统服务的唯一通道。

3、数据缓冲区高速缓冲是建立的基础和原则是什么？要解决的根本问题是什么？他有什么优缺点？

(1)基础与原则是:

对于文件系统的一切存取操作，内核都能通过每次直接从磁盘上读或往磁盘上写来实现。但是，慢的磁盘传输速率会使系统响应时间加长、吞吐率降低。内核通过保持一个称为数据缓冲区高速缓冲的内部数据缓冲池来试图减少对磁盘的存取频率。

(2)解决的根本问题是:

磁盘机械运行速度大大低于处理机的运行速度;多进程并发运行,少量的磁盘(通道),I/O 成为瓶颈;数据访问的随机性,磁盘忙闲不均。

(3)数据高速缓冲区优缺点：

优点:

提供了对磁盘块的统一的存取方法; 消除了用户对用户缓冲区中数据的特殊对齐需要;减少了磁盘访问的次数,提高了系统的整体 I/O 效率;有助于保持文件系统的完整性。

缺点：

数据未及时写盘而带来的风险;额外的数据拷贝过程,大量数据传输时影响性能。

4、中断与例外分别是哪里产生的？他们的根本区别是什么？他们的处理方式和流程有何区别？中断级别是根据什么来划分的？为什么要在系统中设置不同的处理机执行级别？

(1)中断：来自进程之外的事件(外设、时钟等)引起的。

例外：来自进程内部的非期望事件(地址越界,除数为 0 等)。

(2) 根本区别：中断要保存上下文,发生在两条指令执行之间；例外不保存上下文，发生在一条指令执行过程中。

(3).处理方式和流程的区别:

中断：当接收到中断的时候，内核保存它的当前上下文，判定中断原因，为中断服务。当中断服务完毕后从下一条指令继续执行(中断服务是由核心中特殊的函数,而不是特殊的进程来执行的)。

例外：来自进程内部的非期望事件(地址越界,执行特权指令除数为 0 等),发生在一条指令执行过程中,系统在例外事件处理完后重新执行该指令。

(4).中断级别的划分:

中断事件	中断级别
硬盘故障	高
时钟	
硬盘	
网路	
终端	
软件中断	低

(5).不同级别？

用一组特权指令给处理机设置一个执行级,以屏蔽同级和低级的中断,最大限度地减少其它事件的干扰,使当前任务顺利执行并尽快完成;但开放更高级的中断,以响应更紧迫的请求。

5、操作系统中包括了哪些构建原语？他们是如何构建更大的功能模块的？

一般地，把系统态下执行的某些具有特定功能的程序段成为原语。一类是机器指令级的，其特点是执行期间不允许中断，正如在物理学中的原子一样，在操作系统中是一个不可分割的基本单位;另一类是功能型的，其特点是作为原语的程序段不允许并发执行。

原语均在系统态下执行，且都是为了完成某个系统管理所需要的功能被高层软件所调用，具有不可分割性；即原语的执行必须是连续的，在执行过程中不允许被中断。

两种构建原语：1 输入/输出重定向。2 管道。

进程相关：创建进程原语 Create()、撤销原语 destroy( )、挂起原语 suspend()、激活原语 active( )、唤醒原语 wakeup()

进程管理、主存储器管理、文件管理、I/O 系统管理、辅助存储器管理、网络管理、系统保护

操作系统的基本服务：程序执行、I/O 操作、文件系统处理、通信、错误检测

6、UNIX 的文件系统包括了哪些大的功能模块？什么是本地文件系统？什么是虚拟文件系统？设置虚拟文件系统的优缺点是什么？

超级块：文件系统中第一块被称为超级块。这个块存放文件系统本身的结构信息，比如每个区域的大小，未被使用的磁盘块的信息。

i 节点：超级块的下一个部分就是 i 节点表，文件系统中的每个文件在该表中都对应一个 i 节点。i 节点是固定长度的记录项，它包含有关文件的大部分信息。Linux 文件系统使用 索引节点 i 来记录文件信息，索引节点是一个结构，用固定长度，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。

数据区：文件的内容保存在这个区域上，磁盘上所有块的大小都一样，如果文件包含了超过一个块的内容，则文件内容会存放在多个磁盘块中，并把磁盘块的分配情况记录在文件的 i 节点中的磁盘序列表中

参考：http://www.cnblogs.com/qianye/archive/2012/11/24/2786345.html

本地文件系统是 UNIX 系统中的基本文件系统,它通常固定存放在本地机器的存贮设备上,任何一种结构形式的文件系统都必然会直接或间接地与某个本地文件系统相联系。本地文件系统由一个根文件系统 和 若干子文件系统所组成。

虚拟文件系统是整个操作系统的用户界面,它给用户提供一个统一的文件系统使用接口,避免用户涉及各个子文件系统的特征部分。用户感觉使用的是一个整体的,比本地机器上实际硬盘空间大得多的文件系统。虚构文件系统接受来自用户的操作请求,根据该操作所访问的文件是存放在本地机器上,还是存放在远地机器上而分别把操作交给本地文件系统或网络文件系统;本地文件系统或网络文件系统(实际上再传给远地机器上的本地文件系统)进行相应的操作后,将结果返回到虚拟文件系统中再传回给用户。

优点：使得整个文件系统结构统一,模块性强,增加功能非常方便;对用户来说,整个文件系统的透明性好,使用简便,避免了用户在对不同类型的文件或不同地点的文件系统进行操作时,分别来设置参数和安排操作过程。

缺点:PVFS 中应用系统 socket 相互通信，应用 TCP/IP 通信协议，每次通信需要内核嵌入，进行内存拷贝，CPU 的负载比较大，影响系统的运行效率；PVFS 并行虚拟文件系统本身具备良好的可扩展性，但是其动态配置的能力不强，如果要扩展一个 I/O 节点，就需要停止服务，并且不能做到空间的合理利用等。

参考：https://blog.csdn.net/foreverdengwei/article/details/8476008

7、划分数据块和数据片的目的是什么？

8、资源保护系统以什么方式保护了哪些类型的资源？

共享资源: 中断屏蔽、原子操作、自旋锁、信号量、环形缓冲区、互斥体

内存防护：checksec 用来检查可执行文件属性、CANNARY(栈溢出保护)是一种缓冲区溢出攻击缓解手段、内存地址随机化机制（address space layout randomization)

参考：https://blog.csdn.net/hudaweikevin/article/details/1687982
https://www.cnblogs.com/Spider-spiders/p/8798628.html

9、逻辑地址和物理地址的区别是什么？他们是怎么转换的？

物理地址：CPU 地址总线传来的地址，由硬件电路控制（现在这些硬件是可编程的了）其具体含义。物理地址中很大一部分是留给内存条中的内存的，但也常被映射到其他存储器上（如显存、BIOS 等）。在没有使用虚拟存储器的机器上，虚拟地址被直接送到内存总线上，使具有相同地址的物理存储器被读写；而在使用了虚拟存储器的情况下，虚拟地址不是被直接送到内存地址总线上，而是送到存储器管理单元 MMU，把虚拟地址映射为物理地址。

线性地址（Linear Address）也叫虚拟地址(virtual address)是逻辑地址到物理地址变换之间的中间层。在分段部件中逻辑地址是段中的偏移地址，然后加上基地址就是线性地址。是一个 32 位无符号整数，可以用来表示高达 4GB 的地址，也就是，高达 4294967296 个内存单元。线性地址通常用十六进制数字表示，值得范围从 0x00000000 到 0xffffffff）程序代码会产生逻辑地址，通过逻辑地址变换就可以生成一个线性地址。如果启用了分页机制，那么线性地址可以再经过变换以产生一个

物理地址。如果没有启用分页机制，那么线性地址直接就是物理地址。

逻辑地址：在有地址变换功能的计算机中,访内指令给出的地址（操作数）叫逻辑地址,也叫相对地址，也就是是机器语言指令中，用来指定一个操作数或是一条指令的地址。要经过寻址方式的计算或变换才得到内存储器中的实际有效地址即物理地址。一个逻辑地址由两部份组成，段标识符:段内偏移量。段标识符是由一个 16 位长的字段组成，称为段选择符。其中前 13 位是个索引号，后面 3 位包含一些硬件细节 。

CPU 将一个逻辑地址转换为物理地址，需要进行两步：首先将给定一个逻辑地址（其实是段内偏移量，这个一定要理解！！！），CPU 要利用其段式内存管理单元，先将为个逻辑地址转换成一个线程地址，再利用其页式内存管理单元，转换为最终物理地址。

参考：https://blog.csdn.net/prike/article/details/52722934

10、文件的 i 节点有什么样的特点和功能？

当一个文件被创建时，系统会分配一个 inode 给它，这样就将文件名和 inode 关联起来了，我们可以将 inode 看做文件系统的内部名称。当用户或程序引用该文件时，系统就会通过该文件的文件名来查找相应的 inode，然后通过 inode 来获取所需要的文件信息。也就是说在 UNIX 系统中文件名仅仅只是与 inode 相连，而不是与文件相连，这是和 Windows 系统中不同的一点。

i 节点是对文件进行控制和管理的一种数据结构。每一个文件都有自己的 i 节点，每个 i 节点都有一个唯一的 i 节点号。i 节点结构如下(参考/usr/include/sys/ino.h)：

```
struct dinode
{
    ushort di_mode;   /*文件类型+用户权限*/
    short di_nlink;   /*文件链接数*/
    ushort di_uid;   /*属主用户 id*/
    ushort di_gid;   /*属主用户组 id*/
    off_t di_size;   /*文件大小*/
    char di_addr[40]; /*文件数据区起点地址*/
    time_t di_atime; /*最后访问时间*/
    time_t di_mtime; /*最后修改时间*/
    time_t di_ctime; /*创建时间*/
};
```

从上面这个结构可以看出以下一些信息：

1、i 节点保存了文件的属性和类型、存放文件内容的物理块地址、最近一次的存取时间、最近一次的修改时间、创建此文件的时间。

2、i 节点中没有记录文件名字，那文件名是怎么关联到 i 节点

3、di_mode 保存文件类型+用户权限。

参考：https://blog.csdn.net/u011403897/article/details/9526237

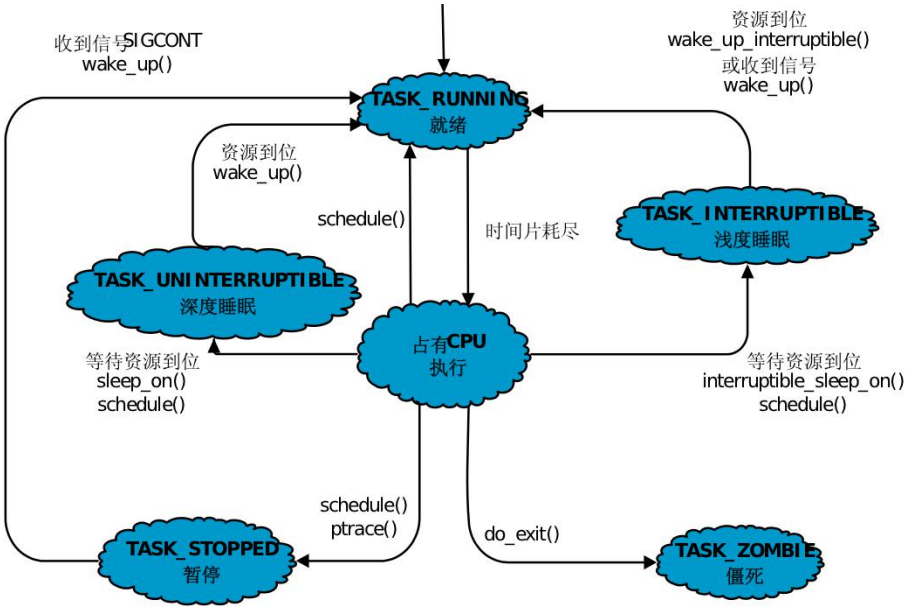
11、进程的三个最基本的进程状态？进程的生命周期中可能要经过哪些过程？

运行态：进程占用 CPU，并在 CPU 上运行；

就绪态：进程已经具备运行条件，但是 CPU 还没有分配过来；

阻塞态：进程因等待某件事发生而暂时不能运行；

LINUX 进程间状态转换和内核调用图解：



运行状态（TASK_RUNNING）：意味着进程处于可运行状态。并不意味着已经实际分配了 CPU。进程可能会一直等到调度器选中它。该状态确保进程可以立即运行，而无需等待外部事件。
可中断睡眠状态（浅度睡眠）（TASK_INTERRUPTIBLE）：针对等待某事件或其他资源的睡眠进程设置的。在内核发送信号给该进程表面事件已经发生时，进程状态变为 TASK_RUNNING，它只要调度器选中该进程即可恢复执行。
不可中断睡眠状态（深度睡眠状态）（TASK_UNINTERRUPTIBLE）：其和浅度睡眠基本类似，因内核指示而停用的睡眠进程。不能由外部信号唤醒，只能由内核亲自唤醒。
暂停状态（TASK_STOPPED）：进程暂停执行接受某种处理。如正在接受调试的进程处于这种状态
僵死状态（TASK_ZOMBIE）：进程已经结束但未释放 PCB

参考：https://blog.csdn.net/maliao1123/article/details/54973884
https://blog.csdn.net/qq_29168493/article/details/79386139

12、标准输入输出重定向的基本原理是什么？如何来实现标准输入输出重定向？

理解 I/O 重定向的原理需要从 Linux 内核为进程所维护的关键数据结构入手。对 Linux 进程来讲，每个打开的文件都是通过文件描述符(File Descriptor)来标识的，内核为每个进程维护了一个文件描述符表，这个表以 FD 为索引，再进一步指向文件的详细信息。在进程创建时，内核为进程默认创建了 0、1、2 三个特殊的 FD，这就是 STDIN、STDOUT 和 STDERR。所谓的 I/O 重定向也就是让已创建的 FD 指向其他文件。在 I/O 重定向的过程中，不变的是 FD 0/1/2 代表 STDIN/STDOUT/STDERR，变化的是文件描述符表中 FD 0/1/2 对应的具体文件，应用程序只关心前者。本质上这和接口的原理是相通的，通过一个间接层把功能的使用者和提供者解耦。

(1) 首先 fork 一个子进程，后续步骤都在子进程中完成，父进程通过 waitpid()系统调用等待子进程结束；

(2) 打开 open()系统调用打开 in.txt 和 out.txt，得到它们的描述符;

(3) 通过 dup2()系统调用把 STDIN 重定向到 fd_in，把 STDOUT 重定向到 fd_out（注意，重定向的影响范围是整个子进程）；

dup2(fd_in, 0); dup2(fd_out, 1);

参考：http://www.cnblogs.com/weidagang2046/p/io-redirection.html