清华 OS 习题集

操作系统概述

单选题

（华中科技大学，2005）程序正在试图读取某个磁盘的第 100 个逻辑块，使用操作系统提供的（ ）接口

- [x] 系统调用
- [ ] 图形用户
- [ ] 原语
- [ ] 键盘命令

操作系统作为用户和计算机硬件系统之间的接口，用户可以通过 3 种方式使用计算机，命令方式、系统调用方式、图形方式。系统调用按照功能分为进程管理、文件操作、设备管理等，本题描述的是文件操作系统调用相关的执行。

(2009 计算机统考)单处理器系统中，可并行的是（ ）

1)进程与进程 2)处理器与设备 3)处理器与通道 4)设备与设备

- [ ] 1 2 3
- [ ] 1 2 4
- [ ] 1 3 4
- [x] 2 3 4

并行指同一时刻同时发生，同一时刻单个处理器只能运行一个进程。

(2010 统考）下列选项中，操作系统提供给应用程序的接口是（ ）

- [x] 系统调用

- [ ] 中断
- [ ] 库函数
- [ ] 原语

解释：略

(2011 统考)下列选项中，在用户态执行的是（ ）

- [x] 命令解释程序
- [ ] 缺页处理程序
- [ ] 进程调度程序
- [ ] 时钟中断处理程序

后 3 个选项都属于内核的功能，在内核态。命令解释程序则属于应用程序。

(2013 联考)计算机开机后，操作系统最终被加载到（ ）

- [ ] BIOS
- [ ] ROM
- [ ] EPROM
- [x] RAM

操作系统被加载到内存（RAM）中

操作系统属于__（单选）

- [ ] 硬件
- [x] 系统软件
- [ ] 通用库
- [ ] 应用软件

操作系统是管理计算机硬件与软件资源的计算机程序，例如 Windows，Linux，Android，iOS 等。应用软件一般是基于操作系统提供的接口，为针对使用者的某种应用目的所撰写的软件，例如 Office Word，浏览器，手机游戏等。而通用库，一般是指为了便于程序开发，对常用的程序功能封装后被调用的程序。

以 ucore OS 为例，它通过 I/O 子系统和各种驱动程序直接控制时钟，串口，显示器等计算机硬件外设，并通过系统调用接口给在其上运行的应用软件提供服务，并通过进程管理子系统、CPU 调度器、内存管理子系统、文件子系统、I/O 子系统来管理应用软件的运行和实现具体的服务。

以下哪个不能用于描述操作系统（单选）

- [ ] 使计算机方便使用
- [ ] 可以管理计算机硬件
- [x] 可以控制应用软件的执行
- [ ] 负责生成应用软件

解释：操作系统负责管理计算机的硬件资源，使得用户不需要关心硬件的工作过程，极大地方便了计算机的使用。我们日常使用计算机，往往已经在使用了特定的操作系统，例如 Windows，而在操作系统上，会同时运行多个应用软件，例如浏览器，音乐播放器等，为了让一个或者多个软件能够正常使用有限的硬件资源，操作系统需要管理应用程序的执行过程。一般来说，像浏览器，音乐播放器，和其他应用软件，都是由特定的个人和团队开发的，操作系统不负责生成应用软件。 以 ucore OS 开发为例，有了 ucore OS，应用软件访问硬件跟简单了，有统一的文件访问方式来访问磁盘或各种外设。ucore OS 可以通过 I/O 操作控制硬件和应用软件的运行等。但编写软件是程序员的工作，把基于 C 语言或汇编语言的程序转换并生成执行代码是编译器（如 gcc,gas）、连接器(如 link)的工作。操作系统可加载运行应用软件的执行代码。

以下不属于操作系统的功能是（）

- [ ] 进程调度
- [ ] 内存管理
- [x] 视频编辑
- [ ] 设备驱动

解释：视频编辑是一个特定的功能，不是系统范围内的共性需求，具体完成这个功能的是视频编辑应用软件。当然，视频编辑应用软件在涉及文件访问时，是需要操作系统中的文件子系统支持；在涉及视频显示方面，需要操作系统的显卡/GPU 等设备驱动支持。

操作系统中的多道程序设计方式用于提高__（单选）

- [ ] 稳定性
- [x] 效率
- [ ] 兼容性
- [ ] 可靠性

解释：是在计算机内存中同时存放几道相互独立的程序，使它们在管理程序（早期的操作系统）控制之下，相互穿插的运行。两个或两个以上程序在计算机系统中同处于开始到结束之间的状态（这里用进程来表示，在后续课程中会讲解“进程管理”）。这样可以使得几道独立的程序可以并发地共同使用各项硬件资源，提高了资源的利用率。 以 ucore OS 为例，在 lab5 中支持了用户进程，从而可以在内存中存放多个程序，并以进程的方式被操作系统管理和调度。

下面对于分时操作系统的说法，正确的是（）

- [ ] 应用程序执行的先后顺序是完全随机的
- [ ] 应用程序按照启动的时间依次执行
- [x] 应用程序可以交替执行
- [ ] 应用程序等待的时间越长，下一次调度被选中的概率一定越大

解释：选择 3 更合适。分时操作系统把多个程序放到内存中，将处理机（CPU）时间按一定的时间间隔（简称时间片）分配给程序运行，这样 CPU 就可以轮流地切换给各终端用户的交互式程序使用。由于时间片很短，远小于用户的交互响应延迟，用户感觉上好像独占了这个计算机系统。应用程序执行的先后顺序主要是由操作系统的调度算法和应用程序本身的行为特征来确定的。调度算法需要考虑系统的效率、公平性等因素。对于 1,2 而言，从系统的效率上看不会带来好处；对于 4 而言，可以照顾到公平性，但“一定”的表述太强了，比如如果调度算法是简单的时间片轮转算法（在后续章节“处理器调度”），则 4 的要求就不会满足了，且更实际的调度算法其实还需考虑等待的事件等诸多因素。 以 ucore OS 为例，在 lab6 中支持实现不同的调度算法。对于分时操作系统而言，体现其特征的一个关键点就是要实现时间片轮转调度算法或多级反馈队列调度算法(在后续章节“处理器调度”)。在 ucore OS 中，可以比较方便地实现这两种调度算法。

Unix 操作系统属于__()

- [x] 分时操作系统
- [ ] 批处理操作系统
- [ ] 实时操作系统
- [ ] 分布式操作系统

解释：选择 1 更合适。Unix 操作系统支持交互式应用程序，属于分时操作系统。比早期的批处理操作系统要强大。且它更多地面向桌面和服务器领域，并没有很强的实时调度和实时处理功能，所以一边不划归为实时系统。它虽然有网络支持（如 TCP/IP），但实际上它管理的主要还是单个计算机系统让的硬件和应用软件。 以 ucore OS 为例，它模仿的是 Unix 操作系统，实现了对应的分时调度算法（时间片轮转、多级反馈队列），所以也算是分时系统。如果 ucore 实现了实时进程管理、实时调度算法，并支持在内核中的抢占（preempt in kernel），则可以说它也是一个实时系统了。

批处理的主要缺点是__()

- [ ] 效率低
- [x] 失去了交互性
- [ ] 失去了并行性
- [ ] 以上都不是

解释：批处理操作系统没有考虑人机交互所需要的分时功能，所以开发人员或操作人员无法及时与计算机进行交互。 以 ucore OS 为例，如果它实现的调度算法是先来先服务调度算法（在后续章节“处理器调度”，相对其他调度算法，具体实现更简单），那它就是一种批处理操作系统了，没有很好的人机交互能力。

多选题

关于操作系统，说法正确的是（）

- [x] 操作系统属于软件
- [x] 操作系统负责资源管理
- [x] 操作系统使计算机的使用更加方便
- [ ] 操作系统必须要有用户程序才能正常启动

操作系统是一种软件，特定指是系统软件，其更功能是管理计算机资源，让用户和应用程序更方便高效地使用计算机。 以 ucore OS 为例，其实没有用户程序，操作系统也可以正常运行。所以选项 4 是部队的。

设备管理的功能包括__（）

- [x] 设备的分配和回收
- [ ] 进程调度
- [x] 虚拟设备的实现
- [x] 外围设备启动

进程调度是属于操作系统的进程管理和处理器调度子系统要完成的工作，与设备管理没有直接关系 以 ucore OS 为例（lab5 以后的实验），与进程调度相关的实现位于 kern/process 和 kern/schedule 目录下；与设备管理相关的实现主要位于 kern/driver 目录下

多道批处理系统主要考虑的是__()

- [ ] 交互性
- [ ] 及时性
- [x] 系统效率
- [x] 吞吐量

解释：交互性和及时性是分时系统的主要特征。多道批处理系统主要考虑的是系统效率和系统的吞吐量。 以 ucore OS 为例（lab6 实验），这主要看你如何设计调度策略了，所以如果实现 FCFS(先来想服务)调度算法，这可以更好地为多道批处理系统服务；如果实现时间片轮转（time-slice round robin）调度算法，则可以有比较好的交互性；如果采用多级反馈队列调度算法，则可以兼顾上述 4 个选项，但交互性用户程序获得 CPU 的优先级更高。

中断、异常和系统调用

单选题

(西北工业大学)CPU 执行操作系统代码的时候称为处理机处于（ ）

- [ ] 自由态
- [ ] 目态
- [x] 管态
- [ ] 就绪态

（知识点：3.4 系统调用）内核态又称为管态

(2012 统考)下列选项中，不可能在用户态发生的是（ ）

- [ ] 系统调用
- [ ] 外部中断
- [x] 进程切换
- [ ] 缺页

（知识点：3.3 中断、异常和系统调用比较）系统调用是提供给应用程序使用的，由用户态发出，进入内核态执行。外部中断随时可能发生；应用程序执行时可能发生缺页；进程切换完全由内核来控制。

（2012 统考）中断处理和子程序调用都需要压栈以保护现场。中断处理一定会保存而子程序调用不需要保存其内容的是（ ）

- [ ] 程序计数器
- [x] 程序状态字寄存器
- [ ] 通用数据寄存器

- [ ] 通用地址寄存器

（知识点：3.3 中断、异常和系统调用比较）程序状态字（PSW）寄存器用于记录当前处理器的状态和控制指令的执行顺序，并且保留与运行程序相关的各种信息，主要作用是实现程序状态的保护和恢复。所以中断处理程序要将 PSW 保存，子程序调用在进程内部执行，不会更改 PSW。

（2013 统考）下列选项中，会导致用户进程从用户态切换到内核态的操作是（ ） 1）整数除以 0 2）sin()函数调用 3）read 系统调用

- [ ] 1、2
- [x] 1、3
- [ ] 2、3
- [ ] 1、2、3

（知识点：3.4 系统调用）函数调用并不会切换到内核态，而除零操作引发中断，中断和系统调用都会切换到内核态进行相应处理。

（华中科技大学）中断向量地址是（ ）

- [ ] 子程序入口地址
- [ ] 中断服务例程入口地址
- [x] 中断服务例程入口地址的地址
- [ ] 例行程序入口地址

（知识点：3.3 中断、异常和系统调用比较） 中断向量地址，即存储中断向量的存储单元地址，中断服务例行程序入口地址的地址。

下列选项中， __可以执行特权指令？()

- [x] 中断处理例程
- [ ] 普通用户的程序
- [ ] 通用库函数
- [ ] 管理员用户的程序

（知识点：3.3 中断、异常和系统调用比较）中断处理例程（也可称为中断处理程序）需要执行打开中断，关闭中断等特权指令，而这些指令

只能在内核态下才能正确执行，所以中断处理例程位于操作系统内核中。而 1,3,4 都属于用户程序和用于用户程序的程序库。 以 ucore OS 为例，在 lab1 中就涉及了中断处理例程，可查看 intr_enable，sti，trap 等函数完成了啥事情?被谁调用了?

一般来讲，中断来源于__（）

- [x] 外部设备
- [ ] 应用程序主动行为
- [ ] 操作系统主动行为
- [ ] 软件故障

（知识点：3.3 中断、异常和系统调用比较）中断来源与外部设备，外部设备通过中断来通知 CPU 与外设相关的各种事件。第 2 选项如表示是应用程序向操作系统发出的主动行为，应该算是系统调用请求。第 4 选项说的软件故障也可称为软件异常，比如除零錯等。 以 ucore OS 为例，外设产生的中断典型的是时钟中断、键盘中断、串口中断。在 lab1 中，具体的中断处理例程在 trap.c 文件中的 trap_dispatch 函数中有对应的实现。对软件故障/异常的处理也在 trap_dispatch 函数中的相关 case default 的具体实现中完成。在 lab1 的 challenge 练习中和 lab5 中，有具体的系统调用的设计与实现。

系统调用的主要作用是（）

- [ ] 处理硬件问题
- [ ] 应对软件异常
- [x] 给应用程序提供服务接口
- [ ] 管理应用程序

（知识点：3.4 系统调用）应用程序一般无法直接访问硬件，也无法执行特权指令。所以，需要通过操作系统来间接完成相关的工作。而基于安全和可靠性的需求，应用程序运行在用户态，操作系统内核运行在内核态，导致应用程序无法通过函数调用来访问操作系统提供的各种服务，于是通过系统调用的方式就成了应用程序向 OS 发出请求并获得服务反馈的唯一通道和接口。 以 ucore OS 为例，在 lab1 的 challenge 练习中和 lab5 中，系统调用机制的初始化也是通过建立中断向量表来

完成的（可查看 lab1 的 challenge 的答案中在 trap.c 中 idt_init 函数的实现），中断向量表描述了但应用程序产生一个用于系统调用的中断号时，对应的中断服务例程的具体虚拟地址在哪里，即建立了系统调用的中断号和中断服务例程的对应关系。这样当应用程序发出类似“int 0x80”这样的指令时（可查看 lab1 的 challenge 的答案中在 init.c 中 lab1_switch_to_kernel 函数的实现），操作系统的中断服务例程会被调用，并完成相应的服务（可查看 lab1 的 challenge 的答案中在 trap.c 中 trap_dispatch 函数有关“case T_SWITCH_TOK:”的实现）。

用户程序通过__向操作系统提出访问外部设备的请求（）

- [ ] I/O 指令
- [x] 系统调用
- [ ] 中断
- [ ] 创建新的进程

（知识点：3.3 中断、异常和系统调用比较）具体内容可参见 10.的回答。 以 ucore OS 为例，在 lab5 中有详细的 syscall 机制的设计实现。比如用户执行显示输出一个字符的操作，由于涉及向屏幕和串口等外设输出字符，需要向操作系统发出请求，具体过程是应用程序运行在用户态，通过用户程序库函数 cputch，会调用 sys_putc 函数，并进一步调用 syscall 函数(在 usr/libs/syscall.c 文件中)，而这个函数会执行“int 0x80”来发出系统调用请求。在 ucore OS 内核中，会接收到这个系统调用号（0x80）的中断（参见 kernel/trap/trap.c 中的 trap_dispatch 函数有关 “case T_SYSCALL:”的实现）,并进一步调用内核 syscall 函数（参见 kernel/syscall/syscall.c 中的实现）来完成用户的请求。内核在内核态（也称特权态）完成后，通过执行“iret”指令（kernel/trap/trapentry.S 中的“__trapret:”下面的指令），返回到用户态应用程序发出系统调用的下一条指令继续执行应用程序。

应用程序引发异常的时候，操作系统可能的反应是（）

- [ ] 删除磁盘上的应用程序
- [ ] 重启应用程序
- [x] 杀死应用程序

- [ ] 修复应用程序中的错误

（知识点：3.3 中断、异常和系统调用比较）更合适的答案是 3。因为应用程序发生异常说明应用程序有错误或 bug，如果应用程序无法应对这样的错误，这时再进一步执行应用程序意义不大。如果应用程序可以应对这样的错误（比如基于当前 c++或 java 的提供的异常处理机制，或者基于操作系统的信号（signal）机制（后续章节“进程间通信”会涉及）），则操作系统会让应用程序转到应用程序的对应处理函数来完成后续的修补工作。 以 ucore OS 为例，目前的 ucore 实现在应对应用程序异常时做的更加剧烈一些。在 lab5 中有有对用户态应用程序访问内存产生错误异常的处理（参见 kernel/trap/trap.c 中的 trap_dispatch 函数有关 “case T_PGFLT: ”的实现），即 ucore 判断用户态程序在运行过程中发生了内存访问错误异常，这是 ucore 认为重点是查找错误，所以会调用 panic 函数，进入 kernel 的监控器子系统，便于开发者查找和发现问题。这样 ucore 也就不再做正常工作了。当然，我们可以简单修改 ucore 当前的实现，不进入内核监控器，而是直接杀死进程即可。你能完成这个修改吗？

下列关于系统调用的说法错误的是（）

- [ ] 系统调用一般有对应的库函数
- [x] 应用程序可以不通过系统调用来直接获得操作系统的服务
- [ ] 应用程序一般使用更高层的库函数而不是直接使用系统调用
- [ ] 系统调用可能执行失败

（知识点：3.4 系统调用）更合适的答案是 2。根据对当前操作系统设计与实现的理解，系统调用是应用程序向操作系统发出服务请求并获得操作系统服务的唯一通道和结果。如果操作系统在执行系统调用服务时，产生了错误，就会导致系统调用执行失败。 以 ucore OS 为例，在用户态的应用程序（lab5,6,7,8 中的应用程序）都是通过系统调用来获得操作系统的服务的。为了简化应用程序发出系统调用请求，ucore OS 提供了用户态的更高层次的库函数（user/libs/ulib.[ch]和 syscall.[ch]），简化了应用程序的编写。如果操作系统在执行系统调用服务时，产生了错误，就会导致系统调用执行失败。

以下关于系统调用和常规调用的说法中，错误的是（）

- [ ] 系统调用一般比常规函数调用的执行开销大
- [ ] 系统调用需要切换堆栈
- [ ] 系统调用可以引起特权级的变化
- [x] 常规函数调用和系统调用都在内核态执行

（知识点：3.4 系统调用）系统调用相对常规函数调用执行开销要大，因为这会涉及到用户态栈和内核态栈的切换开销，特权级变化带来的开销，以及操作系统对用户态程序传来的参数安全性检查等开销。如果发出请求的请求方和应答请求的应答方都在内核态执行，则不用考虑安全问题了，效率还是需要的，直接用常规函数调用就够了。 以 ucore OS 为例，我们可以看到系统调用的开销在执行“int 0x80”和“iret”带来的用户态栈和内核态栈的切换开销，两种特权级切换带来的执行状态（关注 kern/trap/trap.h 中的 trapframe 数据结构）的保存与恢复等（可参看 kern/trap/trapentry.S 的 alltraps 和 trapret 的实现）。而函数调用使用的是"call"和“ret”指令，只有一个栈，不涉及特权级转变带来的各种开销。如要了解 call, ret, int 和 iret 指令的具体功能和实现，可查看“英特尔 64 和 iA-32 架构软件开发人员手册卷 2a's,指令集参考（A-M）”和“英特尔 64 和 iA-32 架构软件开发人员手册卷 2B' s,指令集参考（N-Z）”一书中对这些指令的叙述。

操作系统与用户的接口包括__（）

- [x] 系统调用
- [ ] 进程调度
- [ ] 中断处理
- [ ] 程序编译

（知识点：3.3 中断、异常和系统调用比较）解释： 更合适的答案是 1。根据对当前操作系统设计与实现的理解，系统调用是应用程序向操作系统发出服务请求并获得操作系统服务的唯一通道和结果。

多选题

操作系统处理中断的流程包括__（）

- [x] 保护当前正在运行程序的现场

- [x] 分析是何种中断，以便转去执行相应的中断处理程序
- [x] 执行相应的中断处理程序
- [x] 恢复被中断程序的现场

（知识点：3.3 中断、异常和系统调用比较）解释：中断是异步产生的，会随时打断应用程序的执行，且在操作系统的管理之下，应用程序感知不到中断的产生。所以操作系统需要保存被打断的应用程序的执行现场，处理具体的中断，然后恢复被打断的应用程序的执行现场，使得应用程序可以继续执行。 以 ucore OS 为例（lab5 实验），产生一个中断 XX 后，操作系统的执行过程如下：

vectorXX(vectors.S)--> alltraps(trapentry.S)-->trap(trap.c) -->trap_dispatch(trap.c)-->-->……具体的中断处理 -->trapret(trapentry.S) 通过查看上述函数的源码，可以对应到答案 1-4。另外，需要注意，在 ucore 中，应用程序的执行现场其实保存在 trapframe 数据结构中。

下列程序工作在内核态的有__()

- [x] 系统调用的处理程序
- [x] 中断处理程序
- [x] 进程调度
- [x] 内存管理

（知识点：3.3 中断、异常和系统调用比较）这里说的“程序”是一种指称，其实就是一些功能的代码实现。而 1-4 都是操作系统的主要功能，需要执行相关的特权指令，所以工作在内核态。 以 ucore OS 为例（lab5 实验），系统调用的处理程序在 kern/syscall 目录下，中断处理程序在 kern/trap 目录下，进程调度在 kern/schedule 目录下，内存管理在 kern/mm 目录下

物理内存管理

单选题

(2009 联考)分区分配内存管理方式的主要保护措施是（ ）

- [x] 界地址保护
- [ ] 程序代码保护
- [ ] 数据保护
- [ ] 栈保护

为了防止程序的访问地址越界，所以需要进行界地址保护，由硬件负责检查程序访问地址是否合法。

以 ucore 为例，在 ucore lab1 中的，x86 保护模式中的分段机制在某种程度上可以看成是一种分区方式的物理内存分配。bootloader 在启动后，就要完成分段（分区）工作，即建立两个段，内核代码段和内核数据段，一个段就是一个分区。在表述段属性的段描述符（mmu.h 中的 segdesc 数据结构）中，有两个重要的域（字段，field），起始地址（sd_base_15_0,sd_base_23_16,sd_base_31_24）、段限长(sd_lim_15_0,sd_lim_19_16)。这个段大小就是分区的界地址。80386 CPU 在每一次内存寻址过程中，都会比较 EIP（即段内偏移）是否大于段限长，如果大于段限长，这会产生一个内存访问错误异常。

(2010 联考)某基于动态分区存储管理的计算机，其主存容量为 55MB（初始为空），采用最佳适配（Best Fit）算法，分配和释放的顺序为：分配 15MB，分配 30MB，释放 15MB，分配 8MB，分配 6MB，则此时主存中最大空闲分区的大小是（ ）

- [ ] 7MB
- [x] 9MB
- [ ] 10MB
- [ ] 15MB

空闲分区链变化：55（初始）；40（分配 15MB 后）；10（分配 30MB 后）；10->15（释放 15MB 后）；2->15（分配 8MB 后）；2->9（分配 6MB 后）。

以 ucore 为例，能否在 ucore 中做个试验完成上述算法？

(2009 联考）一个分段存储系统中，地址长度为 32 位，其中段号占 8 位，则最大段长为（ ）

- [ ] 2^8 字节

- [ ] 2^16 字节
- [x] 2^24 字节
- [ ] 2^32 字节

在段访问机制中，如果采用的是单地址方案，则段号的位数+段内偏移的位数=地址长度，所以段内偏移占了 32 - 8 = 24 比特。

以 ucore lab1 为例，在段访问机制上，80386 采用了不同，且更加灵活的段寄存器+地址寄存器方案，(可看 OS 原理部分的"物理内存管理：第 2 部分"ppt 的第 10 页)，即 CS 中的值（称为选择子，selector）是段号，作为索引，指向了一个段描述符表中的一个段描述符。 段描述符中的一个字段是段基址（32 位），这个 32 位段基址+段内偏移（即 32 位的 EIP）形成了最终的线性地址（如果使能页机制，则也就是最终的物理地址了）。 所以，如果是这道题说明了采用 80386 方式，结果就不一样了。

(2010 联考）某计算机采用二级页表的分页存储管理方式，按字节编址，页大小为 2^10 字节，页表项大小为 2 字节，逻辑地址结构为“|页目录号|页表|页内偏移量|”逻辑地址空间大小为 2^16 页，则表示整个逻辑地址空间的页目录表中包含表项的个数至少为（ ）

- [ ] 64
- [x] 128
- [ ] 256
- [ ] 512

页大小为 2^10B，页表项大小为 2B，一页可以存放 2^9 个页表项，逻辑地址空间大小为 2^16 页，需要 2^16 个页表项，需要 2^16/2^9 = 2^7 = 128 个页面保存页表项。所以页目录表中包含的表项至少为 128。

以 ucore lab2 为例，80386 保护模式在使能页机制后，采用的是二级页表，32 位地址空间，页大小为 2^12 字节，页表项为 4 字节，逻辑地址结构为“|页目录号|页表|页内偏移量|”逻辑地址空间大小为 2^20 页，则也目录表包含的表项个数为 1024. 你觉得对吗？

(武汉理工大学）段式存储管理系统中，一个程序如何分段是在（ ）决定的。

- [ ] 分配主存时
- [x] 程序员编程时
- [ ] 装作业时
- [ ] 程序执行时

> 程序员在编程时，其实已经确定了哪些是代码，哪些是数据。所以编译器可以把代码放到代码段，数据放到数据段，操作系统可以根据编译器的设置把程序加载到内存中的代码段和数据段执行。
>
> 以 ucore lab1 为例，学员编写文件有代码和数据两部分，gcc 把代码放到代码段，数据放到数据段，并形成 ELF 格式的 ucore kernel（这其实也是一个程序）。bootloader 把 ucore 的代码或数据放到它设置好的内核代码段和内核数据段中。ucore lab5 以后，ucore 也可以加载应用程序到用户代码段和用户数据段中。

一般情况下，___的速度最快

- [x] L1 cache
- [ ] L2 cache
- [ ] Main Memory
- [ ] Disk

> 解释：访问速度上 cache > Main Memory > Disk; cache 中 L1 > L2 > L3 ... 越靠近 cpu 速度越快，容量越小。
>
> 这个在 qemu 模拟器中无法体现，因为它没有模拟不同存储之间的访问速度。 :(

分页系统中，逻辑地址到物理地址的变换是由___决定的

- [ ] 段表
- [x] 页表
- [ ] 物理结构
- [ ] 重定位寄存器

> 解释：分页系统中，页表负责转换逻辑地址到物理地址。

分段系统中，逻辑地址到物理地址的变换是由____决定的

- [x] 段表
- [ ] 页表
- [ ] 物理结构
- [ ] 重定位寄存器

解释：看看 ucore lab1，了解 bootloader 和 ucore 是如何建立段表的。

连续内存分配算法中的 First Fit（首次适应）算法，其空闲分区链的顺序为____

- [x] 空闲区首地址递增
- [ ] 空闲区首地址递减
- [ ] 空闲区大小递增
- [ ] 空闲区大小递减

解释：First Fit 是指按地址来寻找第一个满足要求的空闲块，其空闲分区链的顺序也就是按空闲块首地址递增。

以 ucore 为例，能否在 ucore 中做个试验完成上述算法？

连续内存分配算法中的 Best Fit（最佳适应）算法，其空闲分区链的顺序为____

- [ ] 空闲区首地址递增
- [ ] 空闲区首地址递减
- [x] 空闲区大小递增
- [ ] 空闲区大小递减

解释：Best Fit 是指寻找一个大小最合适的空闲块，要求空闲块按照大小排列，其空闲分区链的顺序为按大小递增。

以 ucore 为例，能否在 ucore 中做个试验完成上述算法？

连续内存分配算法 First Fit 的缺点是____

- [ ] 算法复杂
- [ ] 大的空闲分区会被分割
- [x] 容易产生外部碎片

- [ ] 分配速度慢

  解释：*First Fit* 算法非常简单，分配速度也较快。但是 *First Fit* 不考虑实际的需求和找到的空闲分区的大小的匹配度，所以容易产生外部碎片。

  以 *ucore* 为例，能否在 *ucore* 中做个试验完成上述算法并看看其缺点能否重现？

连续内存分配算法 *Best Fit* 的缺点是____

- [ ] 算法复杂
- [ ] 大的空闲分区会被分割
- [ ] 分配速度慢
- [x] 回收速度慢

  解释：*Best Fit* 算法也非常简单，分配速度较快。由于选取的空闲分区大小都很合适，所以基本不会出现大的空闲分区总是被分割的情况。但是在此算法中，内存回收则涉及了很多操作：判断左右邻居是否是空闲分区，如果不是，则插入此空闲分区到合适的地方，如果是则合并空闲块，并把合并后的结果插入到合适地方；但是由于空闲分区链不是按地址排序的，所以上述操作需要遍历几次链表用于查找和插入，速度较慢。

连续内存分配算法 *Worst Fit* 的缺点是____

- [ ] 算法复杂
- [x] 大的空闲分区会被分割
- [ ] 分配速度慢
- [ ] 容易产生很小的空闲分区

  解释：*Worst Fit* 每次使用最大的空闲分区，按照需求分割相应的大小，所以会造成大的空闲分区总是被分割。其算法比较简单，分配速度也很快。

  以 *ucore* 为例，能否在 *ucore* 中做个试验完成上述算法并看看其缺点能否重现？

应用程序中的逻辑地址到物理内存中的物理地址的转换机制建立的过程发生____程序过程中

- [ ] 编译

- [ ] 链接
- [x] 加载
- [ ] 运行

解释：在编译器编译和链接程序的过程中都只涉及到逻辑地址，跟机器的配置无关，这也是编译链接所生成的可执行文件可以直接在相同系统的其它机器上使用的原因。而在操作系统加载应用程序时，操作系统负责建立应用程序的段表或页表。将逻辑地址和实际物理地址对应起来，之后应用程序在运行过程中 CPU 才能根据逻辑地址通过段表或页表正确访问到物理地址。

以 ucore lab5 为例，在加载应用程序时，将建立应用程序对应的进程，并在建立过程中，把应用程序对应进程的页表也建立好了。

某一作业完成后，系统收回其主存空间，并与相邻空闲区合并，为此需修改空闲区表，如果待回收的空闲区有相邻的低址空闲区,也有相邻的高址空闲区，那么空闲区表将____

- [ ] 项数不变，有一个空闲区的大小变大
- [ ] 项数不变，有一个空闲区的起始地址变小，大小变大
- [ ] 项数增加
- [x] 项数减少

解释：合并之后，原本的 2 个相邻空闲分区和被回收的分区合并成一个分区，所以分区项数变为 n - 2 + 1 = n - 1。

以 ucore 为例，能否在 ucore 中做个试验完成上述相邻空闲区合并功能，看看合并的效果如何？

对于分页系统与分段系统,下列说法正确的是( ).

- [ ] 页的大小跟具体执行程序有关
- [x] 都属于非连续分配
- [ ] 段的大小固定且由系统确定
- [ ] 分段技术和分页技术是不能共存在一个系统中的

解释：页的大小由 CPU 硬件规定的规范，并由操作系统进行初始化和管理，跟具体执行程序无关； 段的大小是指程序的数据段、代码段等

每段的大小，和具体程序相关；分段技术和分页技术是按照需求进行动态的分配和回收，是非连续分配，它们可以融合使用，也称段页式管理。

以 ucore 为例，在 lab2 之后，就采用基于 80386 的段页式管理了，但段机制的功能没有太用。

采用段页式管理时，程序按逻辑被划分成___

- [x] 段
- [ ] 页
- [ ] 区域
- [ ] 块

解释：程序按逻辑划分各段。而页、区域、块由操作系统负责分配、映射和管理，和程序逻辑没有关系。

采用段页式管理的多道程序环境下，一个应用程序都有对应的___

- [x] 一个段表和一个页表
- [ ] 一个段表和一组页表
- [ ] 一组段表和一个页表
- [ ] 一组段表和一组页表

解释：每道程序有一个段表，且还要有一个页表，才能完成段页式的内存管理。

以 ucore 为例，在 lab5 之后，就采用基于 80386 的段页式管理了，一个程序有一个段表，也有一个页表。

在分页式存储管理系统中时，每次 CPU 取指令或取操作数，至少要访问___次主存。

- [x] 0
- [ ] 1
- [ ] 2
- [ ] 3

解释：0 次。因为 CPU 会有 cache 和 mmu

以 ucore 为例，在 80386 中，如果取的指令和取的操作数都在 CPU 的 cache 中，且放在页表中的地址映射关系被缓存在 CPU 的 MMU 中，则不需要访问主存。

在分段式存储管理系统中时，每次 CPU 取指令或取操作数，至少要访问___次主存。

- [x] 0
- [ ] 1
- [ ] 2
- [ ] 3

解释：0 次。因为 CPU 会有 cache，mmu 和对段表项的缓存

以 ucore 为例，在 80386 中，如果取的指令和取的操作数都在 CPU 的 cache 中，且放在段表中的地址映射关系被缓存在 CPU 的段表项缓存中，则不需要访问主存。

在段页式存储管理系统中时，每次 CPU 取指令或取操作数，至少要访问___次主存。

- [x] 0
- [ ] 1
- [ ] 2
- [ ] 3

解释：0 次。因为 CPU 会有 cache 和对段表项的缓存 以 ucore 为例，在 80386 中，如果取的指令和取的操作数都在 CPU 的 cache 中，且放在段表中的地址映射关系被缓存在 CPU 的段表项缓存中，且放在页表中的地址映射关系被缓存在 CPU 的 MMU 中，则不需要访问主存。

每道程序能在不受干扰的环境下运行，主要是通过___功能实现的。

- [ ] 内存分配
- [x] 内存保护
- [ ] 内存回收
- [ ] 内存扩充

解释：内存访问需要将逻辑地址和重定位寄存器(基址寄存器)进行加运算之后才能访问物理地址，而内存保护主要是使用界地址寄存器来实现

对逻辑地址的限制，以免逻辑地址越界而造成物理地址访问越界，进而对别的程序进行干扰。

以 ucore 为例，在 lab5 中，ucore 对每个应用程序设置的页表，可以完成对不同程序的地址空间的隔离，从而避免了程序间可能的干扰。

可变分区存储管理方案中___作为存储保护使用。

- [ ] 逻辑地址寄存器
- [x] 长度寄存器
- [ ] 物理地址寄存器
- [ ] 基址寄存器

解释：长度寄存器或称界地址寄存器，用于存储保护。

以 ucore 为例，在 80386 中，分段机制可以看成是一种可变分区存储管理方案，其段描述符中的段限长字段类似这里提到的“长度寄存器“

分页系统中的页面对___透明，是___管理的。

- [ ] 程序员、编译器
- [x] 程序员、操作系统
- [ ] 操作系统、编译器
- [ ] 程序员、链接器

解释：分页由操作系统控制，用户并不能感知。

以 ucore 为例，在 lab5 以后，程序员写应用程序，确实不需要考虑有分页机制的存在。因为 ucore 已经为每个应用程序建立好了段表（也叫全局描述符表，简称 GDT）和页表。

多选题

关于分段系统和分页系统说法正确有___。

- [x] 页是系统层面的内存管理的单位，分页的目的主要是由于操作系统管理的需要；段是编写程序层面的内存管理的单位，分段的目的主要是为了能更好地满足程序员开发的需要
- [x] 页的大小是固定的，而且由系统确定。段的长度却是不固定的，决定于程序员所编写的程序

- [x] 分段系统会产生外碎片，分页系统会产生内碎片
- [x] 分段可灵活的控制存取访问，可根据各段的特点决定访问权

解释：1,2,4 解释略。3：分段系统中段的大小是跟程序相关的，分段系统中每次分配的大小就是相应段的真实大小所以没有内部碎片；但是却会产生不满足任何段大小的空闲分区，就是外部碎片。

Virtual Memory Management

单选题

(2012 联考)下列关于虚拟存储器的叙述中，正确的是（ ）

- [ ] 虚拟存储只能基于连续分配技术
- [x] 虚拟存储只能基于非连续分配技术
- [ ] 虚拟存储容量只受外存容量的限制
- [ ] 虚拟存储容量只受内容容量的限制

采用连续分配方式的时候，会使得相当一部分内存空间都处于空闲状态，造成内存资源的严重浪费，无法从逻辑上扩大内存容量。

（2011 年联考）在缺页处理过程中，操作系统执行的操作可能是（ ） 1)修改页表 2)磁盘 I/O 3)分配页框

- [ ] 仅 1、2
- [ ] 仅 2、3
- [ ] 仅 1、3
- [x] 1、2、3

如果还有可分配给程序的内存，那么会分配新的页框，修改页表，从磁盘读取内容放入到分配的页框中。

（2013 计算机联考）若系统发生抖动(Thrashing)时，可用采取的有效措施是（ ）1）撤销部分进程 2）增加磁盘交换区的容量 3）提高用户进程的优先级

- [x] 仅 1

- [ ] 仅 2
- [ ] 仅 3
- [ ] 仅 1、2

撤销部分进程可以增大可用内存，减少抖动。而磁盘交换区容量和进程优先级则跟抖动无关。

(南昌大学)一个虚拟存储器系统中，主存容量 16MB，辅存容量 1GB，地址寄存器位数 32 位。那么虚存最大容量为（ ）

- [ ] 1GB
- [ ] 16MB
- [ ] 1GB + 16MB
- [x] 4GB

虚拟存储器的最大容量跟虚拟地址空间有关，是 2^32。

（上海交通大学）分页式虚拟存储管理系统中，分页是（ ）实现的

- [ ] 程序员
- [ ] 编译器
- [ ] 系统调用
- [x] 系统

分页是系统内部实现的，对用户透明。

为了使得内存需求较大的程序能够正常运行，常需要通过外存和内存的交换技术，这被叫做__技术

- [ ] 虚拟机
- [ ] 内存分配
- [ ] 进程调度
- [x] 虚拟存储

解释：虚拟机用于模拟真实物理机器，单独的内存分配技术可以不考虑使用外存，进程调度则用于管理进程的执行时间和次序等。虚拟存储是指当真实内存不能满足需求的时候，可以将程序需要的代码和数据放到内存中，暂时不需要的放到外存上；通过内存和外存的不断交换，来满足程序的运行需求。

虚拟内存是为了应对__的问题（）

- [ ] 内存访问速度过慢
- [ ] 内存管理困难
- [x] 内存容量不满足程序需求
- [ ] 磁盘访问过慢

解释：虚拟内存是应对内存容量不能满足程序需求的情况，并不能解决内存内存和外存访问速度的问题。

一般来讲，虚拟内存使得程序的运行速度__

- [ ] 加快
- [ ] 不变
- [x] 变慢
- [ ] 变得极不稳定

解释：由于虚拟内存有可能造成外存和内存的不断交换，虽然能够满足大程序的运行需求，但是程序的运行速度相比没有虚拟内存的情况下会变慢。

虚拟内存常用的页面淘汰技术，主要利用了程序的__特征

- [ ] 健壮性
- [ ] 完整性
- [x] 局部性
- [ ] 正确性

解释：程序的局部性是指程序呈现在某段时间内只访问程序的某一部分代码和数据的特性，而页面置换算法可以利用这一特性使常被访问的页面不被淘汰也就减少了缺页率。

在一个系统中，页面大小设定为 4k，分配给每个进程的物理页面个数为 1，在某应用程序中需要访问一个 int[1024][1024]的数组（逐行访问），那么按行存储和按列存储的不同情况下，__

- [x] 按行存储时，执行效率高
- [ ] 按列存储时，执行效率高
- [ ] 执行效率相同

- [ ] 执行效率不确定

解释：按行存储的时候，每访问一行才会出现一次页面置换；而按列存储，则每访问一次就发生一次缺页。由于缺页的时候需要调用页面置换算法进行内外存交换，所以缺页率高的时候效率就低。

虚拟内存技术__

- [ ] 只能应用于分段系统
- [ ] 只能应用于分页系统
- [x] 可应用于分段系统、分页系统
- [ ] 只能应用于段页式系统

解释：虚拟内存技术是一种内外存交换的思想，可应用与分段系统、分页系统等。而目标系统是分段系统还是分页系统，影响的只是虚拟内存技术思想的具体实现。

在虚拟页式内存管理系统中，页表项中的‘访问位’给__提供参考价值。

- [ ] 分配页面
- [x] 页面置换算法
- [ ] 换出页面
- [ ] 程序访问

解释：页面置换算法可能需要根据不同页面是否被访问，访问时间和访问频率等进行淘汰页面的选择。

在虚拟页式内存管理系统中，页表项中的‘修改位’供__使用

- [ ] 分配页面
- [ ] 页面置换算法
- [x] 换出页面
- [ ] 程序访问

解释：页面换出的时候，需要判断外存上的相应页面是否需要重写。如果内存中该页面在使用期间发生了修改，则相应的修改位被设置，用于换出的时候通知操作系统进行外存相应页面的修改。

在虚拟页式内存管理系统中，页表项中的__供程序访问时使用

- [ ] 访问位

- [ ] 修改位
- [x] 状态位
- [ ] 保护位

解释：页表项的状态位用于指示该页是否已经调入内存，供程序访问时使用，如果发现该页未调入内存，则产生缺页中断，由操作系统进行相应处理。

在虚拟页式内存管理系统中，发生缺页的概率一般取决于__

- [ ] 内存分配算法
- [ ] 内存读取速度
- [ ] 内存写入速度
- [x] 页面置换算法

解释：缺页率的高低跟实际能分配的物理内存的大小，以及系统中的页面置换算法相关。差的页面置换算法可能造成需要访问的页面经常没有在内存中，而需要进行缺页中断处理。

页面置换算法的优劣，表现在__

- [ ] 程序在运行时能够分配到的页面数
- [ ] 单位时间内，程序在运行时得到的 CPU 执行时间
- [x] 程序在运行时产生的页面换入换出次数
- [ ] 程序本身的访存指令个数

解释：页面置换算法在满足程序运行需求的同时，应尽量降低页面的置换次数，从而降低运行开销。

选择在将来最久的时间内不会被访问的页面作为换出页面的算法叫做__

- [x] 最优页面置换算法
- [ ] LRU
- [ ] FIFO
- [ ] CLOCK

解释：LRU 是换出在过去的时间里最久未被访问的页面；FIFO 是换出最先被换入的页面；CLOCK 类似于 LRU，也是对 FIFO 的改进。但是以上三种算法都是根据过去一段时间内的页面访问规律进行换出页面

的选择。而最优页面置换算法是指换出将来在最久的时间内不会被访问的页面，是一种理想情况也是不可能实现的。

Belady 异常是指__

- [ ] 频繁的出页入页现象
- [x] 分配的物理页数变多，缺页中断的次数却增加
- [ ] 进程的内存需求过高，不能正常运行
- [ ] 进程访问内存的时间多于读取磁盘的时间

解释：一般情况下，分配的物理页数越多，缺页率会越低。但是某些页面置换算法如 FIFO 就可能造成相反的情况，也即分配的物理页数增多，缺页率却增高的情况。这种情况称为 Belady 异常。

在各种常见的页面置换算法中，__会出现 Belady 异常现象

- [x] FIFO
- [ ] LRU
- [ ] LFU
- [ ] CLOCK

解释：FIFO 可能出现 Belady 异常，如访问顺序 1,2,3,4,1,2,5,1,2,3,4,5，在最多分配 3 个物理块的情况下缺页 9 次，而在最多分配 4 个物理块的情况下缺页 10 次。

当进程访问的页面不存在，且系统不能继续给进程分配物理页面的时候，系统处理过程为__

- [ ] 确定换出页面->页面换出->页面换入->缺页中断
- [ ] 缺页中断->页面换入->确定换出页面->页面换出
- [ ] 缺页中断->确定换出页面->页面换入->页面换出
- [x] 缺页中断->确定换出页面->页面换出->页面换入

解释：首先在程序访问的时候发现页面不在内存中，从而发出缺页中断，进入页面置换的流程。需要确定换出页面才能执行页面交换，而页面换入之前要保证页面已经正确的换出，因为页面换出可能需要重写外存中相应的页面。

某进程的页面访问顺序为 1、3、2、4、2、3、1、2，系统最多分配 3 个物理页面，那么采用 LRU 算法时，进程运行过程中会发生__缺页。

- [ ] 三次
- [ ] 四次
- [x] 五次
- [ ] 六次

解释：1（缺页） - 3（缺页） - 2（缺页） - 4（缺页，换出 1） - 2 - 3 - 1（缺页，换出 4） - 2

在现代提供虚拟内存的系统中，用户的逻辑地址空间__

- [ ] 不受限制
- [ ] 受物理内存空间限制
- [ ] 受页面大小限制
- [x] 受指令地址结构

解释：逻辑地址空间受到逻辑地址的结构限制，也即为指令地址的结构限制。

多选题

以下哪些页面置换算法是可以实现的__

- [ ] 最优页面置换算法
- [x] LRU
- [x] FIFO
- [x] CLOCK

解释：最优页面置换算法是根据将来的页面访问次序来选择应该换出的页面，因为在程序执行之前不可能已知将来的页面访问次序，所以不可能实现。而其它的页面置换算法则是根据已经发生的页面访问次序来决定换出的页面，都是可以实现的。

影响缺页率的因素有__

- [x] 页面置换算法

- [x] 分配给进程的物理页面数
- [x] 页面本身的大小
- [x] 程序本身的编写方法

解释：总体来讲，缺页率的主要影响因素的页面置换算法和分配给进程的物理页面数。但是页面本身的大小和程序本身的编写方法则涉及到页面访问次序的变化，对缺页率也会造成影响。

判断题

发生缺页的时候，一定会使用页面置换算法__

- [ ] 对
- [x] 错

解释：发生缺页的时候，如果分配给程序的物理页面数还有空闲，则直接换入新的页面，不需要使用页面置换算法来挑选需要换出的页面。

进程、线程管理

单选题

(2010 年计算机联考真题)下列选项中，导致创建新进程的操作是（） 1)用户登陆成功 2)设备分配 3)启动程序执行

- [ ] 仅 1 和 2
- [x] 仅 2 和 3
- [ ] 仅 1 和 3

- [ ] 1、2、3

解释：设备分配是通过系统中设置相应的数据结构实现的，不需要创建进程

（2012 年计算机联考真题）下列关于进程和线程的叙述中，正确的是（）

- [x] 不管系统是否支持线程，进程都是资源分配的基本单位
- [ ] 线程是资源分配的基本单元，进程是调度的基本单位
- [ ] 系统级线程和用户级线程的切换都需要内核的支持
- [ ] 同一进程中的各个线程拥有各自不同的地址空间

解释：引入线程的操作系统中，通常都是把进程作为资源分配的基本单位，而把线程作为独立运行的基本单位。同一进程中的各个线程都可以共享进程所拥有的系统资源，这表现在所有线程都有相同的地址空间。对于用户级线程的切换，通常是发生在一个应用进程的诸多线程之间，这时，也同样无须内核的支持

(2010 年计算机联考真题)下列选项中，降低进程优先级的合理时机是（）

- [x] 进程时间片用完
- [ ] 进程刚完成 I/O 操作，进入就绪队列
- [ ] 进程长期处于就绪队列
- [ ] 进程从就绪状态转为运行状态

解释：进程时间片用完，从执行态进入就绪态应降低优先级以让别的进城那个调度进入执行状态，B 中进程刚完成 I/O，进入就绪队列后应该等待被处理器调度，故应提高优先级，C 中类似，D 中不应该降低，应该在时间片用完后再降低

（上海交通大学）OS 对（）分配内存资源

- [ ] 线程
- [ ] 高速缓冲存储器
- [x] 进程
- [ ] 快表

解释：进程是系统资源分配的基本单位，线程是调度的基本单位，高速缓冲存储器和快表都是硬件

(四川大学)一进程基本状态可以从其他两种基本状态转变过去，这个基本状态一定是（）

- [ ] 执行状态
- [ ] 阻塞状态
- [x] 就绪状态
- [ ] 完成状态

解释：处于就绪状态的进程，已具备了运行条件，但由于未能获得 CPU，故仍不能运行，就绪状态可以从运行状态和阻塞状态转换得到

（上海交通大学）下列说法（）不是创建进程必须的

- [ ] 建立一个进程的进程表项
- [ ] 为进程分配内存
- [x] 为进程分配 CPU
- [ ] 将进程表项放入就绪队列

解释：进程刚被创建时，实际上是处于就绪状态的，所以不需为进程分配 CPU

（2011 年全国统考）在支持多线程的系统中，进程 P 创建的若干个线程不能共享的是（）

- [ ] 进程 P 的代码段
- [ ] 进程 P 打开的文件
- [ ] 进程 P 的全局变量
- [x] 进程 P 中某线程的栈指针

解释：多线程系统中，一个进程的多个线程共享进程的代码段、文件和全局变量，进程中某线程的栈指针是归该线程所独有，对其他线程透明，但不恩能够与其他线程共享。

（2011 年全国统考）下列选项中，在用户态执行的是（）

- [x] 命令解释程序
- [ ] 缺页处理程序
- [ ] 进程调度层序
- [ ] 时钟中断处理程序

解释：缺页处理程序和时钟中断都属于中断，进程调度属于系统调用，均在核心态执行，命令解释程序属于命令借口，它在用户态执行

（南京理工大学）进程和程序之间有密切联系，但又有不同的概念，两者的一个本质区别是（）

- [x] 程序是静态概念，进程是动态概念
- [ ] 程序是动态概念，进程是静态概念
- [ ] 程序保存在文件中，进程存放在内存中
- [ ] 程序顺序执行，进程并发执行

解释：进程和程序的本质区别是程序是静态的，进程是动态的

（电子科技大学）若一进程拥有 100 个线程，这些线程属于用户级线程，则在系统调度执行时间上占用（）个时间片

- [x] 1
- [ ] 100
- [ ] 1/100
- [ ] 0

解释：在引入线程的系统中，资源仍然是按进程分配的，由于分配给该进程 1 个时间片，所以在执行时间上总共占 1 个时间片

(上海交通大学)一个进程被唤醒，意味着（）

- [x] 该进程可以重新占用 CPU
- [ ] 优先级变为最大
- [ ] PCB 移到就绪队列之首
- [ ] 进程变为运行态

解释：在一个进程被唤醒时，它将从阻塞状态变成就绪状态，从而可以重新获得 CPU 并投入运行

对进程的描述中，下列说法错误的是（）

- [x] 一个程序只对应一个进程
- [ ] 一个进程可以包含若干个程序
- [ ] 进程是有生命周期的
- [ ] 一个程序可以对应多个进程

解释：进程是执行中的程序，它是有生命周期的，程序本身不是进程，程序只是被动实体，一个程序可能会有多个进程相关

下列的进程状态变化中，()变化是不可能发生的

- [ ] 运行一等待
- [x] 等待一运行
- [ ] 等待一就绪
- [ ] 运行一就绪

解释： 进程状态是由当前活动所定义，运行状态表示指令正在被执行，等待状态表示进程等待某个事件的发生，就绪态表示进程等待分配处理器，由进程状态图我们可以看到等待状态无法直接转变成运行状态，需要从等待态先变成就绪态

一个运行的进程用完了分配给它的时间片后，它的状态变为（）

- [ ] 运行
- [ ] 等待
- [x] 就绪
- [ ] 终止

解释： 当一个进程用完了分配给它的时间片后，状态会变为就绪态，之后会继续等待分配处理器

将进程的（）连接在一起形成进程队列

- [ ] 堆栈段
- [ ] 数据段
- [ ] 堆
- [x] PCB

解释：进程调度选择一个可用的进程到 CPU 上执行，而进程进入洗头膏时，会被加到作业队列中，改队列包括系统中的所有进程，驻留在内存中就绪、等待运行的进程保存在就绪队列中，改队列通常用链表来实现，其头节点指向链表的第一个和最后一个 PCB 块的指针。

下列关于进程控制块的描述中，说法错误的是（）

- [ ] 进程控制块记录进程的状态及名称等
- [ ] 进程控制块位于主存储区内

- [x] 进程控制块对每个进程不止有一个
- [ ] 进程控制块的内容、格式及大小可能不同

解释：每个进程在操作系统内用一个进程控制块来表示，每个进程控制块都记录进程的状态及名称等，并且每个进程对应一个进程控制块，进程控制块的内容、格式及大小可能不同，并且进程控制快位于主存储区内

PCB 是进程存在的唯一标志，下列（）不属于 PCB

- [ ] 堆栈指针
- [ ] 全局变量
- [ ] 进程 ID
- [x] CPU 状态

解释：进程描述块包含许多与一个特定进程相关的信息，主要有：进程状态、程序计数器、CPU 调度信息、内存管理信息、记账信息以及 I/O 状态信息。从题目中我们可以看出 CPU 状态信息并不包含在内。

对于标准的线程，下列叙述中，错误的是（）

- [ ] 进程中可以包含多个线程
- [ ] 线程并不拥有资源，只是使用他们
- [ ] 线程可以创建其他线程
- [x] 线程没有生命期

解释：线程依然有生命周期

（）系统调用是用来被父进程等待子进程结束的

- [x] wait()
- [ ] fork()
- [ ] exit()
- [ ] exec()

解释：当进程完成执行最后的语句并使用系统调用的 exit()请求操作系统删除自身时，进程终止。这时，进程可以返回状态值到父进程，而这个父进程等待子进程结束的方法是通过父进程系统调用 wait()

多个进程的实体能存在于同一内存中，在一段时间内都得到运行。这种性质称为进程的（）

- [ ] 动态性
- [ ] 调度性
- [x] 并发性
- [ ] 独立性

解释：概念题,进程有四个特性，动态性：进程的实质是程序在多道程序系统中的一次执行过程，进程是动态产生，动态消亡的；并发性：任何进程都可以同其他进程一起并发执行；独立性：进程是一个能独立运行的基本单位，同时也是系统分配资源和调度的独立单位；异步性：由于进程间的相互制约，使进程具有执行的间断性，即进程按各自独立的、不可预知的速度向前推进

现在操作系统中，（）是资源分配的基本单位，（）是 CPU 调度的基本单位。

- [ ] 作业，程序
- [ ] 内存，进程
- [x] 进程，线程
- [ ] 代码，数据

解释：概念题，在现代操作系统中，进程使资源分配的基本单位，线程是 CPU 调度的基本单位。其中线程与属于同一进程的其他线程共享代码段、数据段和其他操作系统资源，如果进程有多个控制线程，那么它能同时做多个任务

下列各项工作步骤中，()不是创建进程所必需的步骤

- [ ] 为进程分配内存等资源
- [ ] 将 PCB 链入进程就绪队列
- [x] 作业调度程序为进程分配 CPU
- [ ] 建立一个 PCB

解释：创建进程时不需要用作业调度程序为进程分配 CPU

在多线程操作系统中，对线程具有属性阐述正确的是（）

- [x] 具有进程控制块，共享所属进程资源，处理机的独立调度单位，具有动态性
- [ ] 具有线程控制块，共享所属进程资源，处理机的独立调度单位，具有动态性

- [ ] 具有进程控制块，独享所属进程资源，处理机的独立调度单位，具有动态性
- [ ] 具有进程控制块，共享所属进程资源，处理机的独立调度单位，具有静态性

解释：概念题，线程具有进程控制块，共享所属进程资源，处理机的独立调度单位，具有动态

多选题

（西安电子科技大学）能正确描述进程和线程的概念是（）

- [x] 线程可以是进程中独立执行的实体，一个进程可以包含一个或多个线程
- [ ] 线程又称为轻型进程，因为线程都比进程小
- [x] 多线程计数具有明显的优越性，如速度快、通信简便、设备并行性高
- [ ] 由于线程不作为资源分配单位，线程之间可以无约束地并行执行
- [x] 一个线程可以属于一个或多个进程

解释：虽然线程被称为轻量级线程，这并不意味着线程比进程小，进程和线程之间无法进行大小比较

（电子科技大学）引起挂起状态的原因有（）

- [x] 终端用户的请求
- [x] 父进程请求
- [x] 负荷调节的需要
- [ ] 操作系统的需要
- [ ] 平衡各队列中的进程控制块

解释：考察引起挂起的原因

下列各项中属于进程特性的是（）

- [x] 动态性
- [x] 异步性

- [x] 独立性
- [x] 并发性

解释：概念题,进程有四个特性,动态性：进程的实质是程序在多道程序系统中的一次执行过程，进程是动态产生，动态消亡的;并发性：任何进程都可以同其他进程一起并发执行;独立性：进程是一个能独立运行的基本单位，同时也是系统分配资源和调度的独立单位；异步性：由于进程间的相互制约，使进程具有执行的间断性，即进程按各自独立的、不可预知的速度向前推进

采用多线程技术的操作系统具有()

- [x] 一个进程中可以有一个或多个线程
- [x] 把进程作为资源分配单位,把线程作为调度和执行单位
- [ ] 不同的线程一定执行不同的程序
- [x] 允许多个线程并发执行

解释：不同的线程可能执行相同的程序，一个线程中可以有一个或多个线程，把进程作为资源分配单位,把线程作为调度和执行单位，允许多个线程并发执行

判断题：

（北京工业大学）子进程可以继承它的父进程所拥有的所有资源（）

- [ ] 对
- [x] 错

解释：子进程继承了父进程的代码段和数据段资源，堆栈段则是自己的

-

（首都师范大学）属于同一进程的用户级线程阻塞了，那么同一个进程的其他用户级线程还可以占有 CPU 运行，直到时间片用完（）

- [x] 对
- [ ] 错

解释：在同一进程中，线程的切换不会引起进程的切换，在由一个进程中的线程切换到另一个进程中的线程时，将会引起进程的切换

在操作系统中，进程是一个静态的概念（）

- [ ] 对
- [x] 错

解释：动态概念

一般来说用户进程的 PCB 存放在用户区，系统进程的 PCB 存放在操作系统区（）

- [ ] 对
- [x] 错

解释： PCB 通常是系统内存占用区中的一个连续存区，它存放着操作系统用于描述进程情况及控制进程运行所需的全部信息。

在 linux 环境里使用 fork（）来创建新进程（）

- [x] 对
- [ ] 错

解释： 概念题，了解进程的创建是如何进行的

在多对一模型的线程中，如果一个线程执行了阻塞系统调用，并不影响整个进程（）

- [ ] 对
- [x] 错

解释： 多对一模型的线程中，如果一个线程执行了阻塞系统调用，会影响整个进程，整个进程会阻塞

启动一个线程使用的是 start()方法（）

- [x] 对
- [ ] 错

解释： 概念题

在父进程还存活的情况下，不会产生僵死状态（）

- [ ] 对
- [x] 错

解释：一个已经终止但是其父进程尚未对其进行善后处理（获取终止子进程的有关信息，释放它仍占用的资源）的进程称为僵尸进程(zombie)。这时进程在调用 exit 命令结束自己的生命的时候，其实它并没有真正的被销毁，而是留下一个称为僵尸进程（Zombie）的数据结构

CPU 调度

单选题

若当前进程因时间片用完而让出处理机时，该进程应转变为（）状态。

- [x] 就绪
- [ ] 等待
- [ ] 运行
- [ ] 完成

解释：只有处于就绪队列中的进程才能得到时间片，因此因为时间片用完而让出 CPU 的进程应该再次返回到就绪队列中。时间片是轮循调度算法中的概念，所有的进程都会按照顺序被分配一个时间片，当时间片用完时如果进程执没有结束，那么应该让出 CPU 进入就绪队列等待下一个属于自己的时间片。

最高响应比优先算法的特点是（）

- [ ] 有利于短作业但不利于长作业

- [x] 有利于短作业又兼顾到长作业
- [ ] 不利于短作业也不利于长作业
- [ ] 不利于短作业但有利于长作业

解释：最高响应比优先算法的响应值公式为R＝（w+s)/s，其中 w 为等待时间，s 为服务时间，因此在等待时间相同的情况下优先选择服务时间短的进程，而当服务时间长的进程等待到一定时间后，其响应值会增加到能够被首先选择，避免了一直被服务时间短的进程超过，所以该算法有利于短作业又兼顾到长作业。

在单处理器的多进程系统中，进程什么时候占用处理器和能占用多长时间，取决于（）

- [ ] 进程相应的程序段的长度
- [ ] 进程总共需要运行时间多少
- [x] 进程自身和进程调度策略
- [ ] 进程完成什么功能

解释：在单处理器的多进程系统中，系统是依靠所使用的调度策略来对进程进行调度的，而其所采用的调度策略可能不止一种，所以什么时候选择什么进程占用处理器和能占用多长时间并不仅仅取决于进程的某一项特性。

时间片轮转调度算法是为了（）

- [x] 多个终端都能得到系统的及时响应
- [ ] 先来先服务
- [ ] 优先级高的进程先使用 CPU
- [ ] 紧急事件优先处理

解释：时间片轮转调度算法在选择进程时是按照到达时间进行选择的，所以不存在优先级高的进程，而每个进程每次只能占用同等的 CPU 时间，所以优先执行的进程并不一定比后执行的进程先完成，对于新加入的进程，只要是队列中等待的进程不是很多，都可以很及时地得到时间片来使用 CPU，所以该算法能够使多个终端得到系统的及时响应。

在基于优先级的可抢占的调度机制中，当系统强制使高优先级任务等待低优先级任务时，会发生（）

- [x] 优先级反转
- [ ] 优先级重置

- [ ] 系统错误
- [ ] 死循环

解释：优先级反转的定义：（1）可以发生在任何基于优先级的可抢占的调度机制中；（2）当系统内的环境强制使高优先级等待低优先级任务时发生。

下面关于硬时限（hard deadlines）和软时限（soft deadlines）的描述错误的是（）。

- [ ] 如果错过了硬时限，将会发生严重的后果
- [x] 硬时限是通过硬件实现的，软时限是通过软件实现的
- [ ] 如果软时限没有被满足，系统也可以继续运行
- [ ] 硬时限可以保证系统的确定性

解释：硬时限是指必须满足的时间限制，如果没有满足可能会导致非常严重的后果；软时限是指在理想情况下应该被满足的时间限制，如果没有被满足，系统可以降低对该时限的要求，以保证不会产生太严重的后果。

下面的调度算法中那个是公平的（）

- [ ] FCFS 先来先服务
- [ ] SPN 短进程优先
- [x] RR 轮循
- [ ] SRT

解释：FCFS 算法可能导致某些进程长时间占用 CPU，所以并不公平；SPN 算法可能会使长进程在很长时间内得不到响应，所以也不公平；RR 算法由于每个进程都能及时得到响应，并且不会长时间占用 CPU，所以是公平的；SRT 也就是 SPN。

FCFS 调度算法的特点不包括（）

- [ ] 简单
- [ ] 平均等待时间变化大
- [x] 不会导致 I/O 和 CPU 之间的重叠处理
- [ ] 花费时间少的任务可能排在花费时间长的任务后面

解释：FCFS 算法的优点是简单，缺点有（1）平均等待时间变化较大；（2）花费时间较少的任务可能排在花费时间较长的任务后面；（3）可能导致 i/o 和 CPU 之间的重叠处理。

CPU 调度策略的目标不包括（）

- [ ] 减少响应时间
- [x] 提高系统处理单任务的速度
- [ ] 减少等待时间
- [ ] 增加吞吐量

解释：系统处理单任务的速度不能通过 CPU 调度策略来改善，只能通过改善硬件性能和改良系统架构来提高。

有 5 个批处理作业(A,B,C,D,E)几乎同时到达一个计算中心,估计运行时间分别为 2,4,6,8,10 分钟,在使用时间片轮转作法（时间片为 2 分钟）,作业的平均周转时间为（）

- [x] 18 分钟
- [ ] 6 分钟
- [ ] 14 分钟
- [ ] 22 分钟

解释：进程 A 在第一次时间片轮转后就完成了，所以等待时间为 0；进程 B 在第二次时间片轮转后完成，等待时间为 2+2=8；进程 C 在第三次时间片轮转后完成，等待时间为 2+2+2+2+2*2=14；进程 D 在第四次时间片轮转后完成，等待时间为 2+2+2+2+2+2+2+2=18；进程 E 在第五次时间片轮转后完成，等待时间为 2+2+2+2+2+2+2+2+2+2=20；因此总的周转时间为 2+0+4+8+6+14+8+18+10+20=90，所以平均周转时间为 90/5=18。

多选题

对上下文切换的描述正确的是（）

- [x] 切换 CPU 的当前任务到另一个任务

- [ ] 不需要保存当前进程在 PCB/TCP 中的执行上下文
- [x] 需要读取下一个进程的上下文
- [ ] 只能读取没有被执行过的进程

解释：上下文切换的相关概念：（1）切换 CPU 的当前任务，从一个进程到另一个进程；（2）保存当前进程在 PCB/TCP 的执行上下文；（3）读取下一个进程的上下文。被切换的进程可以是新来的，也可以是之前没有执行完的。

可以作为进程调度算法的有（）。

- [x] 先来先服务调度算法
- [x] 时间片轮转调度算法
- [ ] 最高优先级调度算法
- [x] 最高响应比优先调度算法
- [ ] 均衡调度算法

解释：不存在最高优先级调度算法和均衡调度算法。

下面可以作为比较调度算法的指标有（）

- [x] CPU 使用率
- [x] 吞吐量
- [x] 周转时间
- [x] 等待时间
- [x] 响应时间

解释：衡量调度算法的 5 个方面：CPU 使用率，吞吐量，周转时间，等待时间和响应时间。

CPU 调度策略可以通过哪几种方式增加系统的吞吐量（）

- [x] 减少操作系统开销
- [x] 减少上下文切换次数
- [ ] 加快系统处理单个任务的速度
- [x] 高效利用系统资源

解释：增加吞吐量可以从两个方面入手：（1）减少开销（操作系统开销，上下文切换）；（2）系统资源的高效利用（CPU，I/O 设备）。

下面对 FFS 公平共享调度控制用户对系统资源的访问的描述中，正确的是（）

- [ ] 所有的用户组都是平等的
- [x] 能够保证不重要的用户组无法垄断资源
- [x] 未使用的资源按照每个组所分配的资源的比例来分配
- [x] 没有达到资源使用率目标的组可以获得更高的优先级

解释：公平共享调度控制用户对系统资源的访问：（1）一些用户组比其他用户组重要；（2）保证不重要的组无法垄断资源；（3）未使用的资源按照每个组所分配的资源的比例来分配；（3）没有达到资源使用率目标的组获得更高的优先级。

判断题

作业调度选择一个作业装入主存后，该作业能否占用处理器必须由作业控制来决定。

- [ ] 对
- [x] 错

解释：作业能够占用处理器是由进程调度来决定的。

在进行作业调度时，要想兼顾作业等待时间和计算时间，可选取响应比高者优先算法。

- [x] 对
- [ ] 错

解释：最高响应比优先算法的公式为 R＝（w+s)/s，其中 w 为等待时间，s 为计算时间，所以兼顾作业的等待时间和计算时间。

在作业调度时，采用最高响应比优先的作业调度算法可以得到最短的作业平均周转时间。

- [ ] 对
- [x] 错

解释：短进程优先算法的平均等待时间最小。

轮循算法的时间量子越大越好。（错）

- [ ] 对
- [x] 错

解释：轮循算法的时间量子太大的话会导致进程等待的时间过长，极限情况下会退化成 *FCFS*。

可抢占式的调度算法比不可抢占式的调度算法开销要小。（错）

- [ ] 对
- [x] 错

解释：可抢占式的调度算法比不可抢占式的调度算法开销要大，因为其上下文切换比不可抢占式的要多。

同步

单选题

操作系统中，两个或多个并发进程各自占有某种资源而又都等待别的进程释放它们所占有的资源的现象叫做什么（）

- [ ] 饥饿
- [x] 死锁
- [ ] 死机
- [ ] 死循环

解释：饥饿状态的进程不会进入等待状态，死锁是指两个或多个进程各自占有某种资源而又等待别的进程释放其所占有的资源。

临界资源是什么类型的共享资源（）

- [ ] 临界资源不是共享资源
- [ ] 用户共享资源
- [x] 互斥共享资源

- [ ] 同时共享资源

解释：临界资源是指能够被多个进程共享，但是同一时间只能由一个进程访问的资源，因此是互斥的。

要想进程互斥地进入各自的同类资源的临界区，需要（）

- [ ] 在进程间互斥使用共享资源
- [ ] 在进程间非互斥使用临界资源
- [x] 在进程间互斥地使用临界资源
- [ ] 在进程间不使用临界资源

解释：临界资源位于临界区，共享资源不一定位于临界区，因此无法保证进程进入临界区；非互斥使用临界资源和不使用临界资源均无法保证进程互斥地进入临界区，因为不存在临界资源的互斥使用的话个进程之间不存在互斥关系。

一个进程由阻塞队列进入就绪队列，可能发生了哪种情况（）

- [x] 一个进程释放一种资源
- [ ] 系统新创建了一个进程
- [ ] 一个进程从就绪队列进入阻塞队列
- [ ] 一个在阻塞队列中的进程被系统取消了

解释：一个进程释放了一种资源后，可能该资源正是位于阻塞队列中的一个进程所必需的资源，因此该进程便可以从阻塞队列进入就绪队列；其余三种情况均不会使某个进程从阻塞队列进入就绪队列。

设两个进程共用一个临界区的互斥信号量 mutex，当一个进程进入了临界区，另一个进程等待时，mutex 应该等于多少（）

- [x] -1
- [ ] 0
- [ ] 1
- [ ] 2

解释：两个进程共用一个临界区的互斥信号量 mutex，那么 mutex 的取值范围应该是 1 到-1，1 表示没有进程进入临界区并且也没有进程等待，0 表示有一个进程进入临界区，-1 表示有一个进程进入临界区并且另一个进程等待。

共享变量是指（）访问的变量

- [ ] 只能被系统进程
- [ ] 只能被多个进程互斥
- [ ] 只能被用户进程
- [x] 可被多个进程

解释：共享变量可以被多个进程访问，并且不需要互斥访问，可以访问的进程既可以是系统进程，也可以是用户进程。

临界区是指并发进程中访问共享变量的（）段

- [ ] 管理信息
- [ ] 信息存储
- [ ] 数据
- [x] 程序

解释：临界区是指进程中的一段需要访问共享资源并且当另一个进程处于相应代码区域时便不会被执行的代码区域。

假定在一个处理机上执行以下五个作业，其中采用 *HRN*（最高响应比优先）算法时第三个被选择的作业号是（）

作业号	到达时间	运行时间
A	0	4
B	1	3
C	2	5
D	3	2
E	4	4

- [ ] A
- [ ] B
- [ ] C
- [x] D
- [ ] E

解释：A 到达时没有其他进程到达，所以先处理 A，当 A 处理完时其余所有进程都到达，此时计算各自的响应比：B=(3+3)/3=2，C=(2+5)/5=1.4，D=(1+2)/2=1.5，E=4/4=1。所以第二个被选择的是 B，B 完成后再次计算响应比：C=(5+5)/5=2，D=(4+2)/2=3，E=(3+4)/4=1.75，所以第三个被选择的是 D。

下面关于 Bakery 算法的描述错误的是（）

- [ ] 进入临界区前，每个进程都会得到一个数字
- [ ] 得到数字最小的进程可以进入临界区
- [x] 如果 P2 和 P4 两个进程得到的数字相同，那么 P4 先进入临界区
- [ ] 数字是按照从小到大生成的\

解释：Bakery 算法描述：（1）进入临界区之前，进程接收一个数字；（2）得到的数字最小的进入临界区；（3）如果进程 Pi 和 Pj 收到相同的数字，那么如果 i<j，Pi 先进入临界区，否则 Pj 先进入临界区；（4）编号方案总是按照枚举的增加顺序生成数字

Peterson 算法是解决 Pi 和 Pj 之间互斥的经典的（）的解决方法

- [ ] 基于中断禁用
- [x] 基于软件
- [ ] 基于硬件
- [ ] 基于原子操作

解释：Peterson 算法是满足进程 Pi 和 Pj 之间互斥的经典的基于软件的解决方法（1981 年）。

如果有 5 个进程共享同一程序段，每次允许 3 个进程进入该程序段，若用 PV 操作作为同步机制则信号量 S 为-1 时表示什么（）

- [ ] 有四个进程进入了该程序段
- [ ] 有一个进程在等待
- [x] 有三个进程进入了程序段，有一个进程在等待
- [ ] 有一个进程进入了该程序段，其余四个进程在等待

解释：S 初始为 3，当有一个进程进入程序段或等待时，S 减一. S 为 -1，意味着有四次减 1 的操作，也即 3 个进程获准进入，1 个在等待。

多选题

产生死锁的必要条件（1345）

- [x] 互斥
- [ ] 可抢占
- [x] 不可抢占
- [x] 占有且申请
- [x] 循环等待

解释：产生死锁的四个必要条件：（1）互斥--一个资源每次只能给一个进程使用（2）不可抢占--资源申请者不能强行的从资源占有者手中夺取资源，资源只能由占有者自愿释放（3）占有且申请--一个进程在申请新的资源的同时保持对原有资源的占有（只有这样才是动态申请，动态分配）（4）循环等待--存在一个进程等待队列 {P1，P2，…，Pn}，其中 P1 等待 P2 占有的资源，P2 等待 P3 占有的资源，…，Pn 等待 P1 占有的资源，形成一个进程等待环路。

锁的实现方法有哪几种（124）

- [x] 禁用中断
- [x] 软件方法
- [ ] 添加硬件设备
- [x] 原子操作指令

解释：实现锁机制的三种方法：禁用中断（仅限于单处理器），软件方法（复杂）和原子操作指令（单处理器或多处理器均可）。锁机制是高等级的编程抽象，因此无法使用硬件设备实现。

判断题

产生死锁的根本原因是供使用的资源数少于需求资源的进程数。

- [x] 对
- [ ] 错

解释：死锁是指两个或多个进程各自占有某种资源而又等待别的进程释放其所占有的资源，因此根本原因就是提供的资源少于需求的资源。

一旦出现死锁，所有进程都不能运行。

- [ ] 对
- [x] 错

解释：出现死锁后，处于死锁状态的进程无法继续运行，但是其他无关的进程可以继续运行。

所有进程都挂起时，系统陷入死锁。

- [ ] 对
- [x] 错

解释：死锁是指两个或多个进程各自占有某种资源而又等待别的进程释放其所占有的资源。所有进程都挂起并不代表其无法被完成。

参与死锁的所有进程都占有资源。

- [ ] 对
- [x] 错

解释：应该是参与死锁的所有进程都等待资源。不占有资源的进程也可能进入死锁。

有 m 个进程的操作系统出现死锁时，死锁进程的个数为 $1<k\leq m$。

- [x] 对
- [ ] 错

解释：死锁并不会将所有的进程都牵扯进去，但出现死锁时一定会有进程参与。

进程间的互斥是一种特殊的同步关系。

- [x] 对
- [ ] 错

解释：基本概念，互斥是实现同步的一种方式，因此也代表一种同步关系。

所有进程都进入等待状态时，系统陷入死锁。

- [ ] 对
- [x] 错

解释：产生死锁的四个必要条件：（1）互斥--一个资源每次只能给一个进程使用（2）不可抢占--资源申请者不能强行的从资源占有者手中夺取资源，资源只能由占有者自愿释放（3）占有且申请--一个进程在申请新的资源的同时保持对原有资源的占有（只有这样才是动态申请，动态分配）（4）循环等待--存在一个进程等待队列 {P1 , P2 , … , Pn}, 其中 P1 等待 P2 占有的资源，P2 等待 P3 占有的资源，…，Pn 等待 P1 占有的资源，形成一个进程等待环路。

死锁和进程间通信

单选题

若 P,V 操作的信号量 S 初值为 4,当前值为-1,则表示有（）进程处于等待状态。

- [ ] 0
- [x] 1
- [ ] 2
- [ ] 3

p 操作会使 s 减 1，如果 s<0，则 p 操作进程进入等待；v 操作会使 s 加 1，如果 s<=0,则会唤醒一个等待的程序。处于等待状态的进程的数目只和信号量当前值有关，而和信号量的初始值无关。

任何两个并发进程之间（）。

- [ ] 一定存在互斥关系
- [ ] 一定存在同步关系
- [ ] 一定彼此独立无关
- [x] 可能存在同步或互斥关系

如果两个并发程序为互斥关系，则必定存在临界区，但是实际上不是多有的并发程序之间都存在临界区；我们把异步环境下的一组并发进程因

直接制约而互相发送消息、进行互相合作、互相等待，使得各进程按一定的速度执行的过程称为进程间的同步，实际上不是所有的程序间都存在直接制约关系。所有两个并发的程序之间只是有可能存在同步或互斥关系。

银行家算法是一种（）算法。

- [ ] 死锁解除
- [x] 死锁避免
- [ ] 死锁预防
- [ ] 死锁检测

银行家算法是一种最有代表性的避免死锁的算法。在避免死锁方法中允许进程动态地申请资源，但系统在进行资源分配之前，应先计算此次分配资源的安全性，若分配不会导致系统进入不安全状态，则分配，否则等待。

在为多道程序所提供的可共享的系统资源不足时，可能出现死锁。但是，不适当的（）也可能产生死锁。

- [ ] 进程优先权
- [ ] 资源的线性分配
- [x] 进程推进顺序
- [ ] 分配队列优先权

不合理的进程推进顺序可能产生死锁的原因是相互等待资源。如进程 p1 申请资源的顺序是资源 1 和资源 2，进程 p2 申请资源的顺序是资源 2 和资源 1；这时当两个进行都申请成功了第一个资源后，在申请第二个资源的时候就会出现死锁。

产生死锁的四个必要条件是：互斥、( )、循环等待和不剥夺。

- [ ] 请求与阻塞
- [x] 请求与保持
- [ ] 请求与释放
- [ ] 释放与阻塞

互斥是一次只有一个进程可以使用一个资源，其他进程不能访问已分配给其他进程的资源；请求与保持是当一个进程等待其他进程时，继续占有已经分配的资源；不剥夺是不能强行抢占进程已占有的资源；循环等

待指存在一个封闭的进程链，使得每个进程至少占有此链中下一个进程所需要的一个资源。这四个是产生死锁的必要条件。

在下列解决死锁的方法中，属于死锁预防策略的是（）。

- [ ] 银行家算法（死锁避免）
- [x] 资源有序分配法
- [ ] 死锁检测法
- [ ] 资源分配图化简法

资源有序分配法将资源按某种规则系统中的所有资源统一编号，申请的时候必须按照编号的顺序申请。对进行必须使用的同类资源，必须一次申请；不同类的资源必须按照资源编号顺序申请，这样就破坏了死锁环路。

采用资源剥夺法可以解除死锁，还可以采用（）方法解除死锁。

- [ ] 执行并行操作
- [x] 撤销进程
- [ ] 拒绝分配新资源
- [ ] 修改信号量

当检测到系统中已发生死锁时，须将进程从死锁状态中解脱出来。常用的实施方法是撤销或挂起一些进程，以便回收一些资源，再将这些资源分配给已处于阻塞状态的进程，使之转为就绪状态，以继续运行。

进程从运行态进入阻塞态可能是由于（）。

- [ ] 现运行进程运行结束
- [x] 现运行进程执行了 P 操作
- [ ] 现运行进程执行了 V 操作
- [ ] 现运行进程时间片用完

p 操作使信号量减 1，表明程序申请了资源，当信号量小于 0 时，表明没有可供使用的资源，程序将从运行态进入阻塞态。

在（)情况下，系统出现死锁。

- [ ] 计算机系统发生了重大故障
- [ ] 有多个封锁的进程同时存在
- [x] 若干进程因竞争而无休止地相互等待他方释放已占有的资源

- [ ] 资源数远远小于进程数或进程同时申请的资源数量远远超过资源总数

死锁产生的必要条件是互斥、请求与保持、不可抢占、循环等待。所以当若干进程因竞争而无休止地相互等待他方释放已占有的资源时，系统会产生死锁。

若信号量 S 的初值为 2，且有三个进程共享此信号量，则 S 的取值范围是( )。

- [ ] [-3,2]
- [ ] [-2,2]
- [x] [-1,2]
- [ ] [0,2]

因为 s 的初始值为 2，而有 3 个进行共享信号量，所以 s 的最小值为 2-3=-1，最大值为 2-0=2。

对于记录型信号量，在执行一次 P 操作(wait 操作)时，信号量的值应当为减 1；当其值为( )时，进程应阻塞。

- [ ] 大于 0
- [x] 小于 0
- [ ] 大于等于 0
- [ ] 小于等于 0

在执行 p 操作时，如果信号量的值小于 0，则表明没有资源可以分配了，进行应进入等待状态。

预防死锁的论述中，（）条是正确的论述。

- [ ] 由于产生死锁的基本原因是系统资源不足，因而预防死锁的有效方法，是根据系统规模，配置足够的系统资源。
- [ ] 由于产生死锁的另一种基本原因是进程推进顺序不当，因而预防死锁的有效方法，是使进程的推进顺序合法。
- [ ] 因为只要系统不进入不安全状态，便不会产生死锁，故预防死锁的有效方法，是防止系统进入不安全状态。
- [x] 可以通过破坏产生死锁的四个必要条件之一或其中几个的方法，来预防发生死锁。

死锁是因为若干进程因竞争而无休止地相互等待他方释放已占有的资源，通过破坏四个必要条件，可以预防死锁产生。通过增加系统资源的

方法相当于增加了信号量的初始值，可以一定程度上减少死锁的出现，但是当众多进程申请同一资源时，还是会出现死锁的情况；合理的进程推进顺序可以降低死锁出现的可能，但是当四个必要条件存在时，还是有出现死锁的可能；让系统不进入安全状态是可以预防死锁，但是因为破坏了互斥关系，导致程序执行错误。

操作系统中，进程与程序的重要区别之一是( )。

- [ ] 程序有状态而进程没有
- [x] 进程有状态而程序没有
- [ ] 程序可占有资源而进程不可
- [ ] 进程能占有资源而程序不能

进程是一个“执行中的程序”，所以进程是有状态的，而程序只是一个静态的概念，并不存在状态。

进程从阻塞状态进入就绪状态可能是由于( )。

- [ ] 现运行进程运行结束
- [ ] 现运行进程执行了 P 操作
- [x] 现运行进程执行了 V 操作
- [ ] 现运行进程时间片用完

v 操作会使信号量加 1，表明有程序释放了资源，而需要该资源的一个进程会被唤醒，成阻塞状态转换成就绪状态，准备获取资源并执行。

发生死锁的必要条件有四个，要防止死锁的发生，可以破坏这四个必要条件，但破坏( )条件是不太实际的。

- [x] 互斥
- [ ] 不可抢占
- [ ] 占有且等待
- [ ] 循环等待

互斥是不可能被禁止的，因为如果需要对资源进行互斥访问，那么操作系统必须支持互斥；预防占有且等待可以要求进行一次性请求所有需要的资源，并且阻塞这个进程直到所有请求都同时满足；对于不可抢占可以要求如果占有某些资源的一个进程进一步申请资源时被拒绝，则该进行必须释放它最初占有的资源；循环等待可以通过定义资源类型的线性顺序来预防。

(北京理工大学)资源的有序分配策略可以破坏死锁的（）条件。

- [ ] 互斥
- [ ] 请求和保持
- [ ] 不剥夺
- [x] 循环等待

资源的有序分配策略属于死锁预防的一种，死锁预防是通过破坏 4 个必要条件中的 1 个或者多个以确保系统不会发生死锁。采用资源有序分配法是破坏了“环路”条件，即破坏了循环等待。

(南京理工大学)一进程在获得资源后，只能在使用完资源后由自己释放，这属于死锁必要条件的（）。

- [ ] 互斥条件
- [ ] 请求和释放条件
- [x] 不剥夺条件
- [ ] 环路等待条件

死锁的必要条件包括互斥、请求和保持、不可剥夺、循环等待。如果一个程序占有的资源只能由其使用完后自己释放，则满足其中的不剥夺条件。

(四川大学)死锁产生的原因之一是：（）。

- [ ] 系统中没有采用 Spooling 技术
- [ ] 使用 PV 操作过多
- [ ] 有共享资源存在
- [x] 资源分配不当

任何系统的资源都是有限的，所以不恰当的资源分配可能会导致死锁的产生。

(南京理工大学)计算机系统产生死锁的根本原因是（）。

- [ ] 资源有限
- [ ] 进程推进顺序不当
- [ ] 系统中进程太多

- [x] A 和 B

产生死锁的原因有两个，一是系统提供的资源不能满足每个进程的使用需求；二是在多道程序运行时，进程推进顺序不合法。

(上海交通大学)某系统中有 11 台打印机，N 个进程共享打印机资源，每个进程要求 3 台，当 N 不超过（）时，系统不会死锁。

- [ ] 4
- [x] 5
- [ ] 6
- [ ] 7

考虑下面的极端情况，每个进程都刚好分到了 2 台打印机，则只需要再分到一台打印机，某个进程就可以获得该打印机，完成自己的工作，并释放所有的打印机。其他的进程就可以完成，这样，N*2+1=11,所以 N=5

(电子科技大学)死锁定理是用于处理死锁的哪一种方法（）。

- [ ] 预防死锁
- [ ] 避免死锁
- [x] 检测死锁
- [ ] 解除死锁

死锁定理是操作系统中用于检测死锁的充分必要条件的方法，所以死锁定理属于检测死锁。

(青岛大学)通常，（）是预防系统死锁的主要策略。

- [ ] 动态分配与静态分配相结合
- [ ] 静态分配与银行家算法相结合
- [ ] 死锁检测与死锁解除相结合
- [x] 静态分配、剥夺式分配和按序分配

死锁预防是通过破坏 4 个必要条件中的 1 个或者多个以确保系统不会发生死锁。采用资源的静态预分配策略，破坏了保持和请求；运行进程剥夺使用其他进程占有的资源，从而破坏不剥夺性条件，采用资源有序分配法，破坏了循环等待条件。

(兰州大学)死锁检测检查的是（）。

- [x] 资源分配图
- [ ] 前趋图
- [ ] 搜索树
- [ ] 安全图

如果资源分配图中不存在环路，则系统不存在死锁；反之如果资源分配图中存在环路，则系统可能存在死锁，也可能不存在死锁。

(兰州大学)采用资源剥夺法可以解除死锁，还可以采用（）方法解除死锁。

- [ ] 执行并行操作
- [x] 撤销进程
- [ ] 拒绝分配资源
- [ ] 修改信号量

解除死锁通常的做法有三种：一是撤销处于死锁状态的进程并收回它们的资源；二是资源剥夺法；三是进程回退。所以这里选择撤销进程。

(四川大学)当进程 A 使用磁带机时，进程 B 又申请该磁带机，这种情况（）。

- [ ] 是不可能出现的
- [ ] 是没法解决的
- [ ] 就是死锁
- [x] 以上均不正确

首先，这种情况在多道程序系统中是可能出现的，甚至是会经常出现的。同时，死锁是指多个进程因竞争资源而形成的一种僵局，若无外力作用，这些进程都将永远不能再向前推进。通常情况下，进程都在等待彼此已经占据的资源。本题中的情况没有构成死锁。

(电子科技大学)下面关于检测死锁的正确描述是（）。

- [ ] 银行家算法是典型的检测死锁算法
- [ ] 检测死锁中系统需要反复检测各个进程资源申请和分配情况
- [ ] 检测死锁是预防卷入了死锁
- [x] 检测死锁方法对系统吱呀 u 年的分配不加限制，只要有则可以进行分配

银行家算法是死锁避免算法，死锁检测方法是对资源分配不加限制，即允许死锁发生。但是系统定时地运行一个死锁检测程序，判断系统是否已发生死锁，若检测到死锁发生，则设法加以解除。

判断题

死锁与程序的死循环一样。

- [ ] 对
- [x] 错

死锁是因为若干进程因竞争而无休止地相互等待他方释放已占有的资源，造成程序不能顺利执行；而程序的死循环的程序在执行逻辑上存在缺陷，导致程序不能够结束循环。

信号量机制中，P、V 操作必须成对出现。

- [x] 对
- [ ] 错

p 操作为申请资源操作，p 操作成功执行后，信号量会减 1；v 操作为释放资源操作，v 操作执行成功后，信号量会加 1；所有资源的申请和释放必须成对出现。

当系统同时具备了死锁的四个必要条件时就肯定会产生死锁。

- [ ] 对
- [x] 错

在系统存在死锁的四个必要条件只是表明该系统可能会出现死锁，而不是肯定会产生死锁。在存在这四个必要条件的时候可以通过明智的选择，确保永远都不会到达死锁点。

死锁是指两个或多个进程都处于互等状态而无法继续工作。

- [x] 对
- [ ] 错

死锁是因为若干进程因竞争而无休止地相互等待他方释放已占有的资源。

死锁避免比死锁预防对系统条件限制更严格，所以使得系统资源利用率不高。

- [ ] 对

- [x] 错

  死锁预防是通过破坏死锁的四个必要条件来预防死锁，但这样会导致低效的资源使用和低效的进程执行；死锁避免则相反，它允许死锁的互斥、占有且等待、不可抢占三个条件，但通过明智的选择，确保永远不会到达死锁点，因此死锁避免比死锁预防允许更多的并发，所以其资源利用率要高于死锁预防。

文件系统

单选题

文件系统的主要目的是（）。

- [x] 实现对文件的按名存取
- [ ] 实现虚拟存贮器
- [ ] 提高外围设备的输入输出速度
- [ ] 用于存贮系统文档

  在现在操作系统中，几乎都有一个文件管理系统，这个系统的目的主要实现对文件的按名存取。

按逻辑结构划分，文件主要有两类，UNIX 中的文件系统采用（）。

- [ ] 网状文件
- [ ] 只读文件
- [ ] 读写文件
- [ ] 记录式文件
- [ ] 索引文件
- [x] 流式文件

  按文件的逻辑结构可分为记录文件和流式文件，而 UNIX、DOS、WINDOWS 系统中的普通文件都是流式文件。

通常，文件的逻辑结构可以分为两大类：无结构的（）和有结构的记录式文件。

- [ ] 堆文件
- [x] 流式文件

- [ ] 索引文件
- [ ] 直接（Hash）文件

按文件的逻辑结构可分为记录文件和流式文件。

链接文件解决了顺序结构中存在的问题，它（）。

- [x] 提高了存储空间的利用率
- [ ] 适合于随机存取方式
- [ ] 不适用于顺序存取
- [ ] 指针存入主存，速度快

顺序结构 ：把逻辑文件的记录（内容）按其本身的顺序（逻辑记录的顺序）在磁盘上也按序存放在连续的块中。读取时也从第一个记录开始按顺序进行。在文件目录中指出文件名，存放的起始块号和占用块数。其最大优点是存取速度快。而问题主要是存储空间利用率不高、输出文件时难以估计需要多少磁盘块、影响文件扩展。链接结构：如果逻辑文件中的各个逻辑记录任意存放到一些磁盘块中，再用指针把各个块按逻辑记录的顺序链接起来，在文件目录中只记录第一块的地址和最后一块的地址，那么这种文件组织方式就是链接结构。链接结构解决了顺序结构中的所有问题，所有空闲块都可以被利用，在顺序读取时效率较高但需要随机存取时效率低下（因为要从第一个记录开始读取查。

文件管理实际上是对（2）的管理。

- [ ] 主存空间
- [x] 辅助存储空间
- [ ] 逻辑地址空间
- [ ] 物理地址空间

从系统角度看，文件系统是一个负责文件存储空间的管理机构，文件管理实际是对辅助存储空间的管理。

下面关于索引文件的论述中，第（）条是正确的论述。

- [ ] 索引文件中，索引表的每个表项中含有相应记录的关键字和存放该记录的物理地址。
- [x] 对顺序文件进行检索时，首先从 FCB 中读出文件的第一个盘块号；而对索引文件进行检索时，应先从 FCB 中读出文件索引表始址。
- [ ] 对于一个具有三级索引表的文件，存取一个记录通常要访问三次磁盘。

- [ ] 在文件较大时，无论是进行顺序存取还是随机存取，通常都是以索引文件方式为最快。

索引结构 ：索引结构是实现非连续存储的另一种方法，索引结构为每个文件建立一张“索引表”，把指示每个逻辑记录存放位置的指针集中在索引表中。（最直观的索引结构就比如我们的网站，首页就相当于一个索引表，每个链接记录了一个文件的位置，当我们点击时，就可以找到那个文件）。文件目录中指出文件名的索引表位置，而索引表中每个项指出一个逻辑记录的存放位置。存取文件时根据索引表中的登记项来查找磁盘上的逻辑记录。索引结构既适合顺序存取记录，也可以方便地随机存取记录，并且容易实现记录的增删和插入，所以索引结构被广泛应用。但是索引结构增加了索引表，要占用部分空间并增加读写索引表的时间。当索引项很多时，还要考虑采用多级索引结构。

下面关于顺序文件和链接文件的论述中错误的论述是（）。

- [x] 顺序文件适于建立在顺序存储设备上，而不适合建立在磁盘上。
- [ ] 在链接文件中是在每个盘块中设置一链接指针，用于将文件的所有盘块链接起来。
- [ ] 顺序文件必须采用连续分配方式，而链接文件和索引文件则都可采取离散分配方式。
- [ ] 在 MS-DOS 中采用的是链接文件结构。
- [ ] 链接文件解决了顺序结构中存在的问题，它提高了存储空间的利用率。

看关于顺序文件、链接文件和索引文件的介绍。

在文件系统中，（）要求逻辑记录顺序与磁盘块顺序一致。

- [x] 顺序文件
- [ ] 链接文件
- [ ] 索引文件
- [ ] 串联文件

看 4 关于顺序文件、链接文件和索引文件的介绍。

下列文件中，（）的物理结构不便于文件的扩充。

- [x] 顺序文件
- [ ] 链接文件
- [ ] 索引文件
- [ ] 多级索引文件

看关于顺序文件、链接文件和索引文件的介绍。

()的物理结构对文件随机存取时必须按指针进行，效率较低。

- [ ] 连续文件
- [x] 链接文件
- [ ] 索引文件
- [ ] 多级索引文件

看关于顺序文件、链接文件和索引文件的介绍。

一个采用二级索引文件系统，存取一块盘块信息通常要访问（）次磁盘。

- [ ] 1
- [ ] 2
- [x] 3
- [ ] 4

二级索引取需要访问 2 次，存需要一次，共需要三次。

设有一个包含 1000 个记录的索引文件，每个记录正好占用一个物理块。一个物理块可以存放 10 个索引表目。建立索引时，一个物理块应有一个索引表目，试问索引及其文件本身应占（）个物理块？

- [ ] 1000
- [ ] 1001
- [ ] 1011
- [x] 1111

共 1000 个记录，即有 1000 个索引表目，索引级数：lg1000=3；一个物理块可以放 10 个索引表目，三级索引需 1000/10=100 个物理块；二级索引：100/10=10；一级索引 10/10=1。所以索引及文件共需 1000+100+10+1=1111 物理快。

打开文件操作的使用是（）。

- [ ] 把整个文件从磁盘拷贝到内存
- [x] 把文件目录项(FCB)从磁盘拷贝到内存
- [ ] 把整个文件和文件目录项(FCB)从磁盘拷贝到内存

- [ ] 把磁盘文件系统的控制管理信息从辅存读到内存

文件目录项是系统管理文件的必须信息结构，是文件存在的唯一标志，打开文件把文件目录项(FCB)从磁盘拷贝到内存。

如果文件系统中有两个文件重名，不应采用（1）。

- [x] 单级目录结构
- [ ] 树型目录结构
- [ ] 二级目录结构
- [ ] 单级和二级目录结构

一级目录结构，要求所有的文件名均不相同，一般只适用于微机的单用户系统。

文件系统采用二级文件目录可以（）。

- [ ] 缩短访问存储器的时间
- [ ] 实现文件共享
- [ ] 节省内存空间
- [x] 解决不同用户间的文件命名冲突

二级目录结构 则增加一级主文件目录，此目录是为用户建立的独立文件目录，用户访问文件时先要找到用户自己的目录再查找该目录下的指定文件。实际上，二级目录结构中，文件系统把用户名和文件名合起来作为文件标识。

【2010 年计算机联考真题】设当前工作目录的主要目的是（）。

- [ ] 节省外存空间
- [ ] 节省内存空间
- [x] 加快文件的索引速度
- [ ] 加快文件的读/写速度

当文件系统含有多级目录时，每访问一个文件，都要使用从树根开始到树叶为止、包含中间节点名的全路径名。当前目录又称工作目录，进程对各个文件的访问都是相对于当前目录进行，而不需要一层一层的检索，加快了文件的检索速度。 选项 12 都是与相对目录无关；选项 4 加快文件的读/写速度取决于磁盘的性能。

【2009 年计算机联考真题】文件系统中，文件访问控制信息存储的合理位置是（）。

- [x] 文件控制块
- [ ] 文件分配表
- [ ] 用户口令表
- [ ] 系统注册表

> 为了实现“按名存取”，文件系统为每个文件设置用于描述和控制文件的数据结构，称为文件控制块（FCB）。在文件控制块中，通常包含三类信息，即基本信息、存取控制信息级使用信息。

【2012 年计算机联考真题】若一个用户进程通过 read 系统调用读取一个磁盘文件中的数据，则下列关于此进程的叙述中，正确的是（）。 I. 若文件的数据不在内存中，则该进程进入睡眠等待状态 II. 请求 read 系统调用会导致 CPU 从用户态切到核心态 III. read 系统调用的参数应包含文件的名称

- [x] 仅 I、II
- [ ] 仅 I、III
- [ ] 仅 II、III
- [ ] I、II 和 III

> 对于 I，当所读文件的数据不在内存是，产生中断（缺页中断），原进程进入阻塞状态，知道所需数据从外存调入内存后，才将该进程唤醒。对于 II，read 系统调用通过陷入将 CPU 从用户态进入核心态，从而获取操作系统提供的服务。对于 III，读一个文件首先要用 open 系统调用打开该文件。open 参数包含文件的路径名与文件名，read 只需要 open 返回的文件描述符，不用文件名作为参数。read 要求三个输入参数：1 文件描述符 fd；2buf 缓冲区首地址；3 传送的字节数 n。read 的功能试图从 fd 所指示的文件中读入 n 个字节的数据，并将它们送到 buf 所指示的缓冲区中。

【2013 年计算机联考真题】用户删除某文件的过程中，操作系统不可能执行的操作是（）。

- [x] 删除文件所在的目录
- [ ] 删除与此文件关联的目录项
- [ ] 删除与此文件对应的文件控制块
- [ ] 释放与此文件关联的内存缓冲区

此文件的目录下可能还存在其他的文件，因此删除文件不能删除文件所在的目录，而与此文件关联的目录项和文件控制块需要随着文件一同删除，同时释放文件关联的内存缓冲区。

【2009 年计算机联考真题】 设文件 F1 的当前引用计数值为 1，先建立文件 F1 的符号链接（软链接）文件 F2，在建立文件 F1 的硬链接 F3，然后删除文件 F1.此时，文件 F2 和文件 F3 的引用技术支持分别是（）。

- [ ] 0,1
- [x] 1,1
- [ ] 1,2
- [ ] 2,1

建立符号链接时，引用计数直接复制；建立硬链接时，引用计数加 1.删除文件时，删除操作对于符号链接是不可见的，这并不影响文件系统，当以后通过符号链接访问文件时，发现文件不存咋，直接删除符号链接；但对于硬链接则不可以直接删除，引用计数值减 1，若值不为 0，则不能删除文件，因为还有其他的硬链接指向此文件。当建立 F2 时，F1 和 F2 的引用计数值都为 1.当建立 F3 时，F1 和 F3 的引用计数值都为 2 了。删除 F1，F3 的引用计数值为 2-1=1，F2 的引用计数值不变。

【河北大学】文件系统采用两级索引分配方式，如果每个磁盘块的大小为 1KB，每个盘块号占 4 个字节，则在该系统中，文件的最大长度是（）。

- [x] 64MB
- [ ] 128MB
- [ ] 32MB
- [ ] 以上都不对

每个磁盘块大小 1KB，每个盘块号占 4 个字节，则一个盘块可以存放 1KB/4B=256 个盘块，则 2 级索引文件的最大长度是 2562561KB=64MB。

【2013 年统考真题】为支持 CD-ROM 中视频文件的快速随机播放，播放性能最好的文件数据块组织方式是（）。

- [x] 连续结构
- [ ] 链式结构
- [ ] 直接索引结构
- [ ] 多级索引结构

视频文件属于有结构文件中的定长记录文件，适合用连续分配来组织，连续分配的优点主要有顺序访问容易，顺序访问速度快。为了实现快速随机播放，要保证最短时间查询，不宜选取链式和索引结构。

【2013 年统考真题】若某文件系统索引节点（inode）中有直接地址项和间接地址项，则下列选项中，与单个文件长度无关的因素是（）。

- [x] 索引节点的总数
- [ ] 间接地址索引级数
- [ ] 地址项的个数
- [ ] 文件块大小

一个文件对应一个索引节点，索引节点的总数只能说明有多少个文件，跟单个文件的长度没有关系。而间接地址索引的级数、地址项的个数和文件块大小都跟单个文件长度相关。

【燕山大学，2006 年】在磁盘上容易导致存储碎片的物理文件结构式（）。

- [ ] 链接
- [x] 连续
- [ ] 索引
- [ ] 索引和链接

连续文件的优点是在顺序存取时速度较快。存在如下缺点：1 要求建立文件时就确定它的长度，2 不便于文件的动态扩充。3 可能出现外部碎片，就是在存储介质上存在很多空闲块，但不连续，无法被连续文件使用。

【南昌大学，2006 年】采用直接存取法来读写磁盘上的物理记录时，效率最高的是（）。

- [x] 连续结构的文件
- [ ] 索引结构的文件
- [ ] 链接结构文件
- [ ] 其它结构文件

在直接存取法下，连续文件只要知道文件在存储设备上的起始地址和文件长度，就能很快的进行存取。适合随机存取的程度总结为：连续>索引>链接。

多选题

按用途分类，文件主要能分为（）

- [x] 系统文件
- [ ] 档案文件
- [x] 用户文件
- [x] 库文件

按用途 ： 系统文件、库文件、用户文件 按保护级别 ： 可执行文件、只读文件、读写文件 按信息流向： 输入文件、输出文件、输入输出文件 按存放时限： 临时文件、永久文件、档案文件 按设备类型： 磁盘文件、磁带文件、卡片文件、打印文件 按文件组织结构：逻辑文件、物理文件（顺序文件、链接文件、索引文件）

允许多个用户同时使用同一个共享文件时，下列（）做法是正确的。

- [x] 允许多个用户同时打开共享文件执行读操作
- [ ] 允许读者和写者同时使用共享文件
- [x] 不允许读者和写者同时使用共享文件
- [x] 不允许多个写者同时对共享文件执行写操作

进行文件读写要保持文件的正确性，所以不能进行同时读写，多个同时写等操作。

文件系统的功能有（）

- [x] 文件系统实现对文件的按名存取
- [x] 负责实现数据的逻辑结构到物理结构的转换

- [ ] 提高磁盘的读写速度
- [x] 提供对文件的存取方法和对文件的操作

文件系统的功能主要有：1、管理文件的存储介质 2、实现文件名到物理地址的映射 3、提供用户对文件和目录的操作命令 4、 提供用户共享文件机制 5、提供文件存取机制，保证文件安全性。

文件的物理结构可分为（）

- [x] 顺序结构
- [x] 链表结构
- [x] 索引结构
- [ ] 目录结构

文件的物理结构 ：由文件系统在存储介质上的文件构造方式称为文件的物理结构。不论用户看来是什么文件，在存储介质上存储时，按何种构造方式记录呢，因为介质上的存储单位是物理块，那么这些物理快是顺序存放，还是链式结构，或者索引结构，都要由文件系统结构来实现。

从对文件信息的存取次序考虑，存取方法可分为（）。

- [x] 顺序存取
- [x] 随机存取
- [ ] 索引存取
- [ ] 连续存取

文件的存取方式有顺序存取和随机存取两种。磁带上的文件只能顺序存取，磁盘上的文件既可采用顺序方式也可用随机方式存取。

I/O 子系统

单项选择题

在操作系统中，用户在使用 I/O 设备时，通常采用（）。

- [ ] 物理设备名
- [x] 逻辑设备名

- [ ] 虚拟设备名
- [ ] 设备号

用户程序提出使用设备申请时，使用系统规定的设备类型号和自己规定的设备相对号（即逻辑设备名）由操作系统进行地址转换，变成系统的设备绝对号（物理设备号）。

操作系统中采用缓冲技术的目的是为了增强系统（）的能力。

- [ ] 串行操作
- [ ] 控制操作
- [ ] 重执操作
- [x] 并行操作

为了提高 CPU 和设备之间的并行程度。

操作系统采用缓冲技术，能够减少对 CPU 的（）次数，从而提高资源的利用率。

- [x] 中断
- [ ] 访问
- [ ] 控制
- [ ] 依赖

I/O 中断：是指中央处理器和通道协调工作的一种手段。通道借助 I/O 中断请求 CPU 进行干预，CPU 根据产生的 I/O 中断事件了解输入输出操作的执行情况，I/O 中断事件是由于通道程序的执行或其他外界原因引起的，采用缓冲技术可以减少 CPU 中断。

CPU 输出数据的速度远远高于打印机的打印速度，为了解决这一矛盾，可采用（）。

- [ ] 并行技术
- [ ] 通道技术
- [x] 缓冲技术
- [ ] 虚存技术

在设备 I/O 中引入缓存技术是为了改善 CPU 与设备 I/O 直接速度不匹配的矛盾。

缓冲技术用于（）。

- [x] 提高主机和设备交换信息的速度
- [ ] 提供主、辅存接口
- [ ] 提高设备利用率
- [ ] 扩充相对地址空间

在设备 I/O 中引入缓存技术是为了改善 CPU 与设备 I/O 直接速度不匹配的矛盾。

通道是一种（）。

- [ ] I/O 端口
- [ ] 数据通道
- [x] I/O 专用处理机
- [ ] 软件工具

通道：计算机系统中能够独立完成输入输出操作的硬件装置，也称为“输入输出处理机”，能接收中央处理机的命令，独立执行通道程序，协助中央处理机控制与管理外部设备。一个独立于 CPU 的专门 I/O 控制的处理机，控制设备与内存直接进行数据交换。

设备管理的主要程序之一是设备分配程序，当进程请求在内存和外设之间传送信息时，设备分配程序分配设备的过程通常是（）。

- [x] 先分配设备，再分配控制器，最后分配通道
- [ ] 先分配控制器，再分配设备，最后分配通道
- [ ] 先分配通道，再分配设备，最后分配控制器
- [ ] 先分配通道，再分配控制器，最后分配设备

主机对外部设备的控制可分为四个层次：主机、通道、控制器、设备。所以，设备分配过程是先设备、在分控制器、最后是通道。

下列描述中，不是设备管理的功能的是（）。

- [ ] 实现外围设备的分配与回收
- [ ] 缓冲管理与地址转换
- [x] 实现按名存取
- [ ] 实现 I/O 操作

设备管理的功能有：1、缓冲管理。为达到缓解 CPU 和 I/O 设备速度不匹配的矛盾，达到提高 CPU 和 I/O 设备利用率，提高系统吞吐量的目的，许多操作系统通过设置缓冲区的办法来实现。2、设备分配。设备分配的基本任务是根据用户的 I/O 请求，为他们分配所需的设备。如果在 I/O 设备和 CPU 之间还存在设备控制器和通道，则还需为分配出去的设备分配相应的控制器和通道。3、设备处理。设备处理程序又称设备驱动程序。其基本任务是实现 CPU 和设备控制器之间的通信。4、设备独立性和虚拟设备。用户向系统申请和使用的设备与实际操作的设备无关。 按名存取是文件管理的功能。

用户编制的程序与实际使用的物理设备无关是由（）功能实现的。

- [ ] 设备分配
- [ ] 设备驱动
- [ ] 虚拟设备
- [x] 设备独立性

设备独立性是设备管理要达到的目的之一，就是，用户程序应与实际使用的物理设备无关，由操作系统来考虑因实际设备不同使用不同的驱动等问题。采用“设备类、相对号”方式使用设备时，用户编程就不必指定特定设备，在程序中由“设备类、相对号”定义逻辑设备。程序执行时由系统根据用户指定的逻辑设备转换成与其对应的具体物理设备。所以，用户编程时使用的设备与实际使用哪台设备无关，这就是“设备独立性”

SPOOLing 技术利用于（）。

- [ ] 外设概念
- [x] 虚拟设备概念
- [ ] 磁带概念
- [ ] 存储概念

SPOOLing 技术有 3 个特点：(1)提高了 I/O 速度。从对低速 I/O 设备进行的 I/O 操作变为对输入井或输出井的操作,如同脱机操作一样,提高了 I/O 速度,缓和了 CPU 与低速 I/O 设备速度不匹配的矛盾。(2)设备并没有分配给任何进程。在输入井或输出井中,分配给进程的是一存储区和建立一张 I/O 请求表。(3)实现了虚拟设备功能。多个进程同

时使用一个独享设备,而对每一进程而言,都认为自己独占这一设备,从而实现了设备的虚拟分配。不过,该设备是逻辑上的设备。

【2010 年计算机联考真题】本地用户通过键盘登陆系统时，首先获得键盘输入信息的程序是（）。

- [ ] 命令解释程序
- [x] 中断处理程序
- [ ] 系统调用服务程序
- [ ] 用户登陆程序

键盘是典型的通过中断 I/O 方式工作的外设，当用户输入信息时，计算机响应中断并通过中断处理程序获得输入信息。

【2011 年计算机联考真题】操作系统的 I/O 子系统通常有 4 个层次组成，每一层明确定义了与邻近层次的接口，其合理的层次组织排列顺序是（）。

- [x] 用户级 I/O 软件、设备无关软件、设备驱动程序、中断处理程序
- [ ] 用户级 I/O 软件、设备无关软件、中断处理程序、设备驱动程序
- [ ] 用户级 I/O 软件、设备驱动程序、设备无关软件、中断处理程序
- [ ] 用户级 I/O 软件、中断处理程序、设备无关软件、设备驱动程序

设备管理软件一般分为四个层次：用户层、与设备无关的软件层、设备驱动程序以及中断处理程序。

【2013 年计算机联考真题】用户程序发出磁盘 I/O 请求后，系统的处理流程是：用户程序-系统调用处理程序-设备驱动程序-中断处理程序。其中，计算数据所在磁盘的柱面号、磁头号、扇区号的程序是（）。

- [ ] 用户程序
- [ ] 系统调用处理程序
- [x] 设备驱动程序
- [ ] 中断处理程序

计算数据所在磁盘的柱面号、磁头号、扇区号的工作是由设备驱动程序完成的。题中的功能因设备硬件的不用而不同，因此应有厂家提供的设备驱动程序完成。

【2011 年计算机统考真题】某文件占用 10 个磁盘块，现在要把该文件磁盘块逐个读入主缓冲区，并送用户区进行分析。假设一个缓冲区与一个磁盘块大小相同，把一

个磁盘块读入缓冲区的时间为 100μs，将缓冲区的数据传送到用户区的时间是 50μs，CPU 对一块数据进行分析的时间为 50μs。在单缓冲区和双缓冲区结构下，读入并分析完该文件的时间分别是（）。

- [ ] 1500μs，1000μs
- [x] 1550μs，1100μs
- [ ] 1550μs，1550μs
- [ ] 2000μs，2000μs

单缓冲区下，当上一个磁盘块从缓冲区读入用户区完成时下一磁盘块才能开始读入，所以当最后一块磁盘块读入用户区完毕时，所用时间为１５０×１０＝１５００，加上处理最后一个磁盘块的 cpu 处理时间５０为１５５０.双缓冲区下，读入第一个缓冲区之后可以立刻开始读入第二个缓冲区，读完第二个缓冲区之后，第一个缓冲区的数据已经传送到用户区，因此不存在等待磁盘块从缓冲区读入用户区的问题，也就是１００×１０＝１１００，再加上最后一个缓冲区的数据传输到用户区并有 CPU 处理的时间 50+50=100，总的时间是 1000+100=1100。

【2005 年，北京理工大学】（）是操作系统中采用的以空间换取时间的技术。

- [x] Spooling 技术
- [ ] 虚拟存储技术
- [ ] 覆盖与交换技术
- [ ] 通道技术

SPOOLing 技术是操作系统中采用的以空间换取时间的技术；虚拟存储技术和覆盖交换技术是为了扩充内存容量，是以时间换空间技术；通道技术是为了提高设备速度，增加了硬件，不属于这两者的任何一个。

【2012 年统考真题】下列选项中，不能改善磁盘设备 I/O 性能的是（）。

- [ ] 重排 I/O 请求次序
- [x] 在一个磁盘上设置多个分区
- [ ] 预读和滞后写
- [ ] 优化文件物理的分布

重排 I/O 请求次序就是将磁盘请求访问序列进行重新排序，就是有关磁盘访问调度策略的选择对 I/O 有性能的影响，不同的调度策略有不同的寻道时间;磁盘分区实质上就是对磁盘的一种格式化。但磁盘设备 I/O 性能是由调用顺序和磁盘本身性质决定的，和分区的多少无太大关系，如果设置过多分区，还会导致一个 I/O 需要启动多个分区，反而会减低效率;预读和滞后写是常见的提升磁盘设备 I/O 速度的方法，具有重复性及阵发性的 I/O 进程和改善磁盘 I/O 性能很有帮助;优化文件物理块的分布可以减少寻找时间与延迟时间，从而提高磁盘性能。

【2009 年全国统考】假设磁头当前位于第 105 道，正在向磁道序号增加的方向移动。现有一个磁道访问请求序列为 35，45，12，68，110，180，170，195，采用 SCAN 调度（电梯调度）算法得到的磁道访问序列是()

- [x] 110，170，180，195，68，45，35，12
- [ ] 110，68，45，35，12，170，180，195
- [ ] 110，170，180，195，12，35，45，68
- [ ] 12，35，45，68，110，170，180，195

SCAN 调度算法就是电梯调度算法，顾名思义就是如果开始时磁头往外就一直要到最外面，然后再返回向里（磁头编号一般是最外面为 0 号往里增加），就像电梯若往下则一直要下到最底层才会再上升一样。

【西安电子科技大学，2000 年】在关于 SPOOLing 的叙述中，（）描述不正确。

- [ ] SPOOLing 系统必须使用独占设备
- [ ] SPOOLing 系统加快了作业执行的速度
- [ ] SPOOLing 系统使独占设备变成共享设备
- [x] SPOOLing 系统利用了处理器与通道并行工作的能力

SPOOLing 是操作系统中采用的一种将独占设备改造为共享设备的技术，它有效减少了进程等待读入读出信息的时间，加快了作业执行的速度。不过，无论有没有通道，SPOOL 系统都可以运行，因此 4 选项不对。

【河北大学】系统设备是通过一些数据结构来进行的，下面的（）不属于设备管理数据结构。

- [x] FCB

- [ ] DCT
- [ ] SDT
- [ ] COCT

FCB 是文件控制块，与设备管理无关。DCT 是设备控制表，SDT 是系统设备表，COCT 控制器控制表，这 3 者都死设备管理中的重要数据结构。

【电子科技大学】 不属于 DMA 控制器的是（）

- [ ] 命令/状态寄存器
- [ ] 内存寄存器
- [ ] 数据寄存器
- [x] 堆栈指针寄存器

DMA 控制器与 CPU 的接口有 3 类信号线：数据线、地址线和控制线。通常与：数据寄存器、命令/状态寄存器两类寄存器相连。为了将数据送到内存，DMA 控制器还需要内存地址寄存器。