第二章 程序设计与操作系统

概念题

- 运行时系统

  编译程序在将高级语言代码生成可执行文件时,添加一些运行时代码,为用户程序的运行提供一定的管理工作,这部分代码就是运行时系统.

- 物理地址?虚拟地址?
- CPU内核模式:管理整个系统的模式,可以执行ISA的全部指令
- CPU用户模式:执行应用程序的模式,不允许对系统的关键资源直接操作,能执行部分指令,不能执行特权指令.
- 特权指令:能对系统中关键资源进行操作的指令.
- ISA: Instruction Set Architecture,指令集架构.软件通过指令集架构来指挥硬件如何运作.是程序员与计算机之间的界面,

装入和链接

装入分为:

- 绝对装入
- 静态重定位
- 动态重定位

链接分为:

- 静态链接(运行前已经链接完成
- 动态链接(运行中,使用到时才进行链接操作

各自的优缺点?

进行地址变换,都需要谁的参与?　　答:操作系统和MMU

运行时系统

- 编译程序在将高级语言代码生成可执行文件时,在obj->exe的过程中,添加相应的运行时代码,为用户程序的运行提供一定的管理工作.这部分代码就是运行时系统.例如JVM和沙箱
- 运行时系统相较于普通的用户程序
  - 其运行中 调用子程序时的压栈入栈,临时变量的内存分配与销毁,参数的传递都由运行时系统完成.
  - 用户程序全部出现在**编译后的目标文件**中,而运行时系统的代码仅出现在**可执行文件**中.
- 运行时系统和程序库的最大区别在于
  - 普通的程序库是用户程序去**主动调用**完成用户的意图,运行时系统是对用户程序的运行起支持作用,不是用户程序主动调用的.
  - 在OS眼中并不区分运行时系统和程序库.
- 与操作系统的不同在于
  - 用户程序运行时,其跟随进入内存,程序结束时,其**跟随退出**.但操作系统**一直都在内存中运行**.
  - 运行时系统只**为**特定程序设计语言的用户程序提供更加个性化的\跨平台的服务,而操作系统**面向整个计算机系统**
- 实现运行时系统和系统调用的程序的本质区别
  - 实现运行时系统的程序,服务于应用程序但也**属于应用程序**,在**用户态**下运行
  - 实现系统调用的应用程序,服务于应用程序,但是由**操作系统提供机制**,在**内核态**下运行.
- **库函数**和**系统调用**的区别?
  - 所属不同:库函数被包含在语言或者应用程序中;而系统调用是操作系统的一部分.

    - 库函数运行在用户态,系统调用运行在内核态.

- 运行时系统还支持:多线程支持,类型检查,堆栈的创建和释放,垃圾回收
- 高级语言中,运行时系统负责在程序末尾添加(停机的)系统调用

作业题

- 2.2 静态重定位和动态重定位之间的区别是什么?

  答：静态重定位是指在程序开始运行前，程序中的各个地址有关的项均已完成重定位，地址变换通常是在装入时一次完成的，以后不再改变。而动态重定位即在程序运行过程中要访问数据时再进行虚拟地址与物理地址的变换。也就是说区别在于，静态重定位在装入目标程序时，对目标程序中指令和数据地址修改，而动态重定位在执行目标程序时，才对目标程序中指令和数据地址修改。

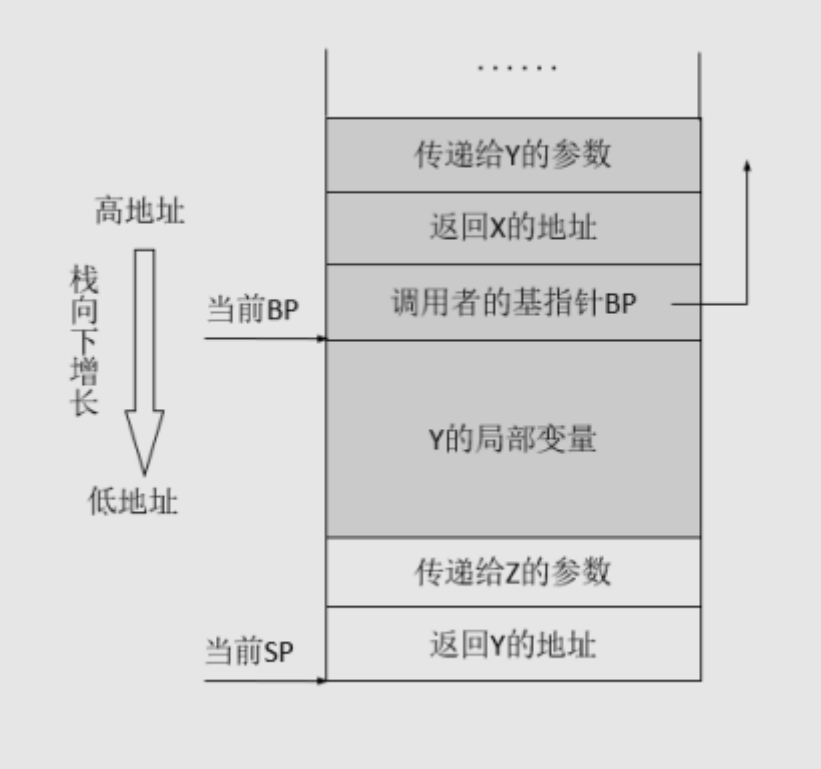
- 虚拟地址空间的大小是由**ISA**决定的
- 栈帧

  为每个子程序在栈内分配的数据结构，用于传递给子程序的参数，形成了调用者和被调用者之间的数据通道，还用于存储返回地址，局部运行环境，如局部变量、寄存器的内容。

  下图中，程序X调用Y，Y调用Z：X->Y->Z。

  此处只着重说明X->Y。

  1. 调用子程序Y时，将参数和返回地址压入栈中。同时还存储主程序X的BP指针，改变BP，SP指针内容，即SP--；BP=>[SP]；SP=>BP。
  2. 执行子程序Y时，栈帧存储为Y分配的局部变量。
  3. 子程序Y返回时，BP== >SP, 恢复主程序X的BP，即[BP]==>BP, 从SP取出返回地址，跳转到主程序X。



第三章 程序运行与操作系统

概念题

- 可执行文件的大小并不等于代码和数据在内存中的样子.(后者通常会更大)
- 指令流:

  指令序列及执行控制.(人

指令的执行序列,不断前进(CPU

- 指令流的运行环境

  上一条指令执行完毕后留下的全部信息(包括内存,CPU内寄存器和设备

- 异常

  指令执行过程中由CPU检测并发出警告的意外事件,例如除零,算术溢出,非法访问内存

- 可以修复的异常记住两个:

  由于精度问题导致除数变成0(结果转化为Nan

  页错误(页置换

- 异常处理意义:CPU可执行多个不同代码序列,从容地在多个代码序列间切换
- 中断请求:外设向CPU发送INTR信号(INTR:INT-REQUEST)
- 中断向量 `interrupt vector` :将异常,中断,陷入的入口地址统一编址.它们的中断处理程序入口地址或者存访中断服务程序的首地址.
- 中断嵌套:进入中断处理程序后,发生另一类中断,操作系统转去执行另一个中断
- 中断重入:中断处理程序执行过程中,又发生该中断事件,CPU暂停当前指令流,再次进入该程序
- 上下文:程序运行环境,指寄存器内容,内存,设备等
- 运行时库(RTL):运行时系统提供的一组供用户程序调用其功能的接口库
- 系统调用:操作系统向应用程序提供的一组服务机制,在内核态下运行

中断事件

不同类型中断事件的比较

属性	意外事件	指令流改变	内部中断	外部中断
异常	√	√	√	×
设备中断	√	√	×	√
陷入	×	√	√	×
子程序调用	×	×	×	×

- 陷入

  陷入的两个例子:断点调试,系统调用

- 陷入与中断的区别

  0. 中断硬件引起,陷入软件引起.
  1. 中断被动发生,陷入主动发出请求.
  2. 中断的发生具有随机性和突发性,而陷入是程序安排好的,不具有突发性.
  3. 中断处理程序提供的服务则不一定为了当前进程,而陷入处理程序提供的服务是被当前进程所使用的.

- 中断与子程序调用的区别

  1. 中断意外发生,由随机事件引发.子程序调用是按照指令流运行时发生.
  2. 中断处理会离开当前进程上下文环境和指令流;子程序调用不会.
  3. 任务不同.

- 陷入产生系统调用与API接口的区别(API接口是由一个或者几个系统调用实现的)

  两者的区别在于系统调用不可以跨平台,但是API可以跨平台.例如C中,fopen可以跨平台,但是open不可以.

关于关中断

- 作用:为中断处理程序提供安心不被打扰的环境
- 硬件实现
- 对不可屏蔽中断不起作用
- 不可长时间关中断

中断响应过程

1. 保存现场1(PC,PSW):因为如果此时不保存,后面再进行保存时的PC和PSW就与原本程序的不同了.
2. 查询中断号
3. 计算中断处理程序入口地址
4. 关中断
5. 调用中断处理程序(保护现场2:保存寄存器内容.因为要确定保存哪几个寄存器,而不是要全部都保存)

运行时系统

运行时系统相较于OS的不同?

1. 退出
2. 服务对象(只为特定程序设计语言的用户提供更加个性化的跨平台服务/面向整个计算机系统

运行时系统相较于用户代码的区别?

1. 出现时机(可执行文件/目标文件)

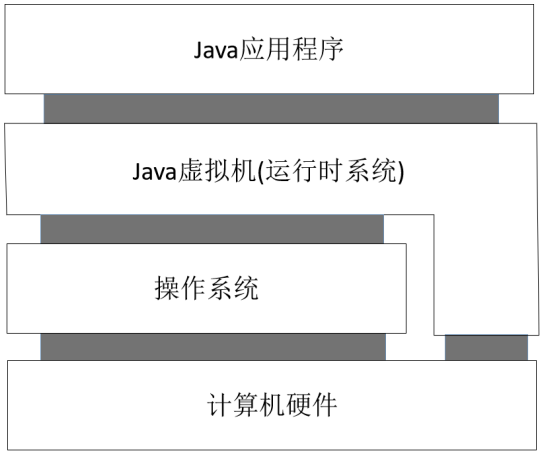
运行时系统与标准函数的区别?

1. 按照程序设计语言的规范行事/按用户程序的意志

提供的服务?

3. 动态内存管理,垃圾回收,堆栈管理
4. 子程序调用时的相关操作
5. 多线程支持
6. 数据类型检查等

地位图:

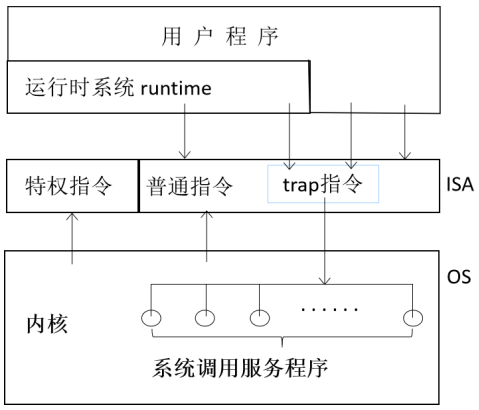


系统调用

类型:进程控制,文件管理,设备管理,信息维护,进程通信.

层次关系:系统命令>C库函数>系统调用>内核函数s

地位图:



停机

- 停机指令也是**特权指令**,直接影响CPU的运行.(暂停了CPU,并没有让整个计算机系统停下来.中断信号使CPU被激活.
- 一个应用程序结束时的动作:

  调用`exit()`,结束应用程序,将CPU控制权转交给操作系统.

作业题

- 应用程序在可执行文件中和在虚拟地址空间中的存放形式有什么区别？举例说明。

  一个可执行文件被执行的同时伴随着新进程的创建.OS为该进程创建虚拟内存空间,读取可执行文件的文件头,进行可执行文件到虚拟存储空间的映射.将CPU指令指针寄存器设置为可执行文件的入口地址.

  在可执行文件中,包含了可以被CPU执行的指令和待处理的数据,全部以二进制形式存储.其中又划分出了专门的段.代码段用来存放转化为二进制形式的指令代码,数据端存访完成初始化赋值的待处理数据,BSS段存访没有被赋值的变量.
- 程序在执行过程中，CPU在什么情况下离开当前程序的指令流去执行其他程序?

  回答时注意时机和时间.

  在**发生中断**,并且**开中断**,**优先级较高**时.会转去执行中断服务程序.

  发生异常时,会直接执行.

  产生陷入.

第四章 操作系统的形成和发展

概念

- 作业:用户在系统中完成一项相对完整的工作.
- 人机交互:操作员与计算机之间传递,交换信息的接口和方式
- CPU利用率:(CPU执行时间)/(CPU执行时间/CPU空闲时间). 被用来评价作业控制.
- 批处理:以作业流为前提,将一类作业(例如Fortran作业)放在同一个时间段进行处理,这样一次处理多个作业只需要一次编译程序的装入,人工操作仍然是限制条件.
- 分时:操作系统将CPU时间划分为若干个时间片,操作系统以时间片为单位,在用户间快速切换,轮流为每个终端用户提供服务,每次服务一个时间片.
- 实时系统: 当外界事件或者数据产生时,能够接受并且以足够快的速度予以处理,处理结果又能在规定的时间之内控制生产过程或者对处理系统作出快速响应,并控制所有实时任务协调运行.
- 第一个操作系统使用的是批处理系统.
- 并行:执行时间可以重叠,无论在宏观还是微观,两者都是同时在运行的.

在多CPU系统中,一个CPU执行一个进程时,另一个CPU可以同时执行另一个程序,互不抢占CPU资源,可同时进行.

- 并发:对于一个CPU而言,表面(宏观)看上去'同时'执行多个任务,但是实际上在操作系统中,指这个一时间段中有几个程序都处于已启动运行到运行完毕之间.但是同一时刻,系统中最多只有一个任务在执行.

  并且这几个程序都是在同一个处理机上运行.只是由于时间片的划分,导致CPU不停在不同任务之间切换,看起来像是"同时发生".

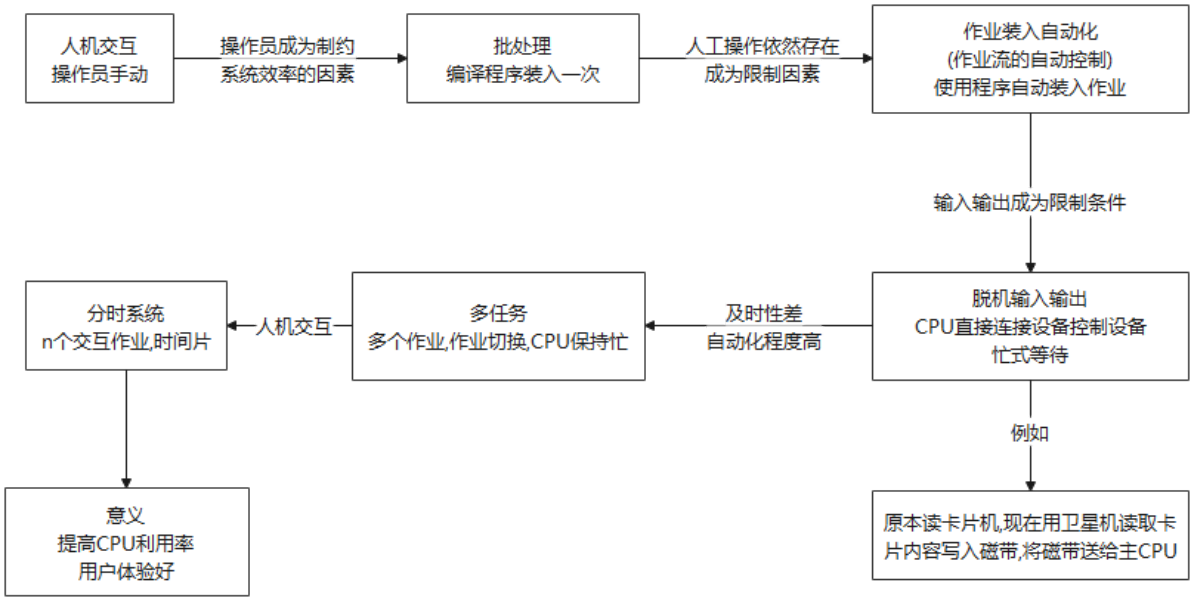
- **操作系统**:操作系统是一组控制和管理计算机硬件和软件资源、合理地对各类作业进行调

  度，以及方便用户的程序集合。操作系统是用户和计算机的接口，同时也是计算机硬

  件和其他软件的接口。

- **内核**:操作系统,或者说操作系统的主体部分代码;在CPU内核模式下执行的程序.
- 保护域:访问权的集合,描述一个对象能干的所有事
- 用户接口:特殊的应用程序,用户登录后,会接管所有交互设备,等待执行命令或者程序.
- 分层设计: 系统分为若干层,高层建立于低层的基础之上,只能由高层程序调用低层程序
- 微内核: 将内核功能尽量从内核空间移到用户空间,内核尽量小.小到仅负责模块间的通信.
- 策略:要做什么,是任务
- 机制:怎样做,是解决办法

发展



- 作业装入自动化

  使用的程序是`监控程序(monitor)`,计算机启动后,首先装入执行监控程序.(优点:避免了人机交互

- 批处理技术的优势和不足
  - 优点:

    CPU利用率高,自动化程度高,系统效率和吞吐量高.

    适用于大容量,高强度的计算和数据处理
  - 缺点:

    没有人机交互,难以进行程序调试

    一味追求CPU利用率的提升
- 脱机输入输出的缺点
  - 及时性差

- 多任务的实现
    - CPU切换于多任务之间
    - 内存中同时装入多个作业
    - 设备共享
    - 作业之间不相互影响
- 分时系统的意义

  提高了CPU利用率,具有足够快的响应时间和交互性,更加友好.

OS

- 提供服务,管理和控制

  (装入服务,输入输出服务),(多任务,分时)
- 目的:提高资源利用率和用户友好性
- 为什么操作系统需要安装?

  针对于特定的硬件和应用需求,生成特定的操作系统.

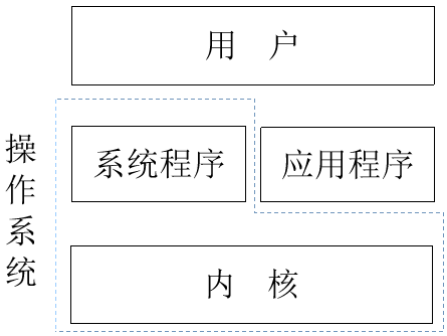
内核

- 内核拥有权限:

  对资源的分配,回收以及特权指令
- 如何区分内核和用户程序的权限?
- 内核之外:

  层次关系大致如下:用户-->应用程序-->内核.

  内核外的程序:



- 命令解释器:

  用户打开终端->登录->显示命令提示符->输入命令(判断注销?否)->解析命令,执行程序
- 微内核

  优点:容易扩充,移植;更加可靠和安全

  缺点:由于模块之间要通过内核进行通信,性能被降低,不容易得到整体的优化.内核与核外程序之间的通信降低性能.(并没有使得系统更加高效).
- 运行中的操作系统,内核负责处理中断事件和系统调用.

作业题

- 在多任务和分析系统环境中,部分用户共用操作系统.这种情况或导致不同的安全问题.问:这些安全问题都有啥?
    1. 窃取或复制其他用户的程序或者数据;
    2. 没有合理的预算来使用资源(CPU,内存,磁盘空间,外围设备)
- 我们能否确保分时机器与专用机器具有相同程度的安全性？解释你的答案

不能.因为分时机器本身就要比专用性机器的安全性差.并且人类设计的任何保护机制都会不可避免地被其他人所破译,而且机制越复杂,就越难对其能否正确实施感到自信.

- 中断的目的是什么?中断和陷入之间的区别?陷入能否由一个用户程序故意产生,如果有,目的是啥?
    - 当发生中断,CPU暂停它正在做的事,转到固定的位置去继续执行.这个固定位置通常是中断服务开始的地址.当该程序执行完毕后,返回中断发生时暂停的程序继续执行
    - 0-3(随机突发,硬件,被动,非当前进程
    - 可以.目的是将控制权转交给CPU进入内核模式,调用内核函数从而获得操作系统提供的服务.
- 操作系统提供的两类服务是啥?有啥区别?

  操作系统提供的一类服务是在系统中不同并发运行的进程之间强制保护。进程只允许访问与它们关联的内存位置。此外，进程不允许损坏与其他进程相关联的文件。进程也不允许在没有操作系统介入的情况下直接访问设备。操作系统提供的第二类服务是提供低层硬件不能直接支持的功能。虚拟内存和文件系统就是操作系统提供的两类服务的例子。
- 使用相同的系统调用接口来操作文件和设备有什么优点和缺点?(独立编码与统一编码)

  在设备和文件操作上使用相同的系统调用接口，可以像访问文件系统中的文件一样访问每个设备。因为大多数内核通过文件接口处理设备，通过执行硬件确定代码来支持这种抽象的文件接口来增加一个新的设备相对容易。这种方式有利于用户程序代码的发展，用户程序代码可以用相同的方式写入设备和文件，还有利于设备驱动程序代码，设备驱动程序代码可以支持规范定义的 API.

  使用相同接口的缺点是很难在文件访问API上下文中获得某些设备的功能，因此，结果或者是丢失功能或者是丢失性能。通过使用 ioctl 操作能够克服一些问题，这个操作为了进程在设备上援引操作提供一个通用接口 。
- 为什么要机制和策略分离?

  机制和策略必须分离，以确保系统方便更改。没有两个系统配置是相同的，因此每个配置要让操作系统配合它的需要。机制和策略分离的情况下，策略可以在机制不变的情况下随意更改，从而提供一个更灵活的系统。
- 内核和外核的区别?

  内核是操作系统最基本的部分。它是为众多应用程序提供对计算机硬件的安全访问的一部分软件，这种访问是有限的，并且内核决定一个程序在什么时候对某部分硬件操作多长时间。内核的分类可分为单内核和双内核以及微内核。严格地说，内核并不是计算机系统中必要的组成部分。

  外核不提供传统操作系统中进程、虚拟内存等抽象，而是专注于物理资源的隔离（保护）与复用，也就是说，这个非常小的外核仅负责保护系统资源，而硬件资源的管理则委托给应用程序和库操作系统。

王道题

- 中断处理过程
    - 关中断(硬件
    - 保存断点(硬件
    - 中断服务程序寻址(硬件
    - 保护现场和屏蔽字
    - 开中断
    - 中断服务程序
    - 关中断
    - 恢复现场和屏蔽字
    - 开中断
    - 中断返回
- 系统调用常见类别
    - 设备管理
- 文件管理
    - 进程管理
- 内存管理

    - 进程通信
- 操作系统的特征
    - 共享
    - 内存(两者是OS最基本的特征,相互为存在的条件)
    - 虚拟
    - 异步
- 在中断发生后,进入中断处理的程序属于:

  A. 用户程序　　B. 可能应用程序,也可能操作系统程序

  C. 操作系统程序　D. 既不是应用程序,也不是操作系统程序

  (此时已经进入内核态,选C)
- 计算机区分核心态和用户态指令后,从核心态到用户态的转换是由OS执行后完成的,而用户态到核心态的转换由( )完成.

  A. 硬件　　B. 核心态程序

  C. 用户程序　　D. 中断处理程序

  A(发生中断,进入内核态执行中断处理程序,由硬件将CPU寄存器某位置为0(核心态))
- 在操作系统中,只能在核心态下执行的指令是( )

  A. 读时钟　　B. 取数　　C. 广义指令　　C. 寄存器清'0'

  C(广义指令指的是系统调用命令,运行于核心态;但是**调用**广义指令可能在用户态下运行.)
- 下列错误地是:

  A. 内部异常的响应发生在指令执行过程中　　B. 内部异常处理后返回到发生异常的指令继续执行

  B(内部异常的响应不可屏蔽,一旦发生便要处理,所以A正确?; 内部异常处理后不一定回到发生异常的指令继续执行,例如当发生除零或者自行中断,都会跳过)
- 执行系统调用的过程包括操作如下:
    - 传递系统调用参数
    - 执行陷入指令
    - 执行相应的服务程序
    - 返回用户态
- 用户可以通过( )和( )两种方式来使用计算机.

  答:命令接口和系统调用.

  命令接口分为联机命令接口和脱机命令接口:

  联机命令接口是指一组键盘操作命令;

  脱机命令接口是指一组作业控制命令,用于批处理.
- 系统调用是操作系统提供给用户的,它( )

  A. 只能通过键盘交互方式使用　　B. 只能通过用户程序间接使用

  C. 是命令接口中的命令　　D. 与系统的命令一样

  答:B,只能通过用户程序请求调用,间接使用
- 下列选项中,不属于多道程序设计的基本特征的是: ( )

  A. 制约性　　B. 间断性　　C. 顺序性　　D. 共享性

  答: C(多道程序设计失去了封闭性和顺序性)
- 提高单机资源利用率的关键技术是( )

  A. 脱机技术　B. 虚拟技术　C. 交换技术　D. 多道程序设计

  答:D
- 操作系统的基本类型主要有:

  批处理操作系统+分时操作系统+实时操作系统.

- 批处理操作系统,分时操作系统和实时操作系统各有什么优缺点?
    - 批处理操作系统:用户脱机使用计算机,作业成批处理.多道程序并发执行.

      系统吞吐量大,资源利用率高;

      用户响应时间长,不提供人机交互能力.
    - 分时操作系统:多个用户可以同时使用计算机

      人机交互性强,系统响应较为及时;

      可靠性不强,对于实时性的紧急要求不够及时.
    - 实时操作系统:能对控制对象做出实时反应/

      可靠性强,响应及时;

      资源利用率低.