名词解释

1. 地址映射：为了保证CPU执行指令时可正确访问存储单元，需将用户程序中的逻辑地址转换为运行时由机器直接寻址的物理地址。
2. 动态重定位：动态重定位的目的是为了让程序能够在不同的内存地址空间中运行，从而避免内存地址的硬编码。
3. 虚拟存储器：虚拟存储器是通过内存管理技术，使得应用程序能够使用比物理内存更大的地址空间。
4. 静态链接：在编译程序时将程序所依赖的库文件的代码和数据复制到可执行文件中，生成一个完全独立的可执行文件。
5. 对换：把内存中暂时不能运行的进程或者暂时不用的程序和数据换出到外存上。
6. 设备驱动程序：可以使计算机和设备通信的特殊程序。
7. SPOOLing：同时联机外围操作技术，慢速字符设备与计算机主机间数据交换的技术。
8. I/O通道：I/O通道是用于处理I/O操作的硬件或软件，可以独立于CPU进行I/O操作。
9. 文件系统：操作系统用于明确存储设备或分区上的文件的方法和数据结构。
10. 目标文件：编译器编译源代码后生成的文件。
11. 文件的逻辑结构：文件的逻辑结构是指文件从用户角度看起来的组织形式，如记录的顺序等。
12. 有结构文件：指文件内部有特定的组织结构，如数据库文件。
13. 位示图：利用二进制的一位来表示磁盘中一个盘块的使用情况。
14. 程序接口：程序接口是操作系统提供给应用程序的一组调用方法，用于请求操作系统服务。
15. 系统调用：操作系统提供给应用程序使用的接口。
16. I/O中断：I/O中断通过中断处理器执行中断操作。
17. 文件管理系统：负责管理和存储文件信息的软件机构。
18. 文件：操作系统中的文件是一种抽象机制，提供在磁盘上保存信息且方便以后读取的方法。
19. 文件的逻辑结构：文件的逻辑结构是指文件从用户角度看起来的组织形式，如记录的顺序等。
20. 文件的物理结构：文件在外存上的存储组织形式。

填空题

1. 地址映射
2. 绝对装入方式、可重定位装入方式、动态运行时装入方式
3. 静态链接、装入时动态链接、运行时动态链接
4. 空闲分区表、空闲分区链
5. 内存紧缩
6. 固定的、系统、可变的、用户程序
7. 对换性、虚拟性
8. 文件、目录、管理文件的软件
9. 分区的分配、分区的保护、分区的扩充
10. 内部碎片和外部碎片
11. 固定分区分配、动态分区分配、离散分配
12. 快表（或快TLB）
13. 地址空间和物理内存
14. 信息的逻辑单位、信息的共享、信息的保护、信息的动态增长、方便使用
15. 地址映射（或动态重定位）
16. 局部性、时间局部性和空间局部性
17. 处理器、I/O通道可以独立于CPU执行I/O操作、I/O通道可以执行I/O指令序列
18. 流式文件、记录式文件、索引式文件
19. 线性索引、树形索引、倒排索引
20. 动态重定位
21. 用户级、驱动级、硬件级
22. 连续结构、索引结构、链式结构；连续结构
23. 高速缓存、主存、外存
24. 连续分配、离散分配
25. 文件的物理结构、存储设备的特性
26. 流式文件、记录式文件、索引式文件
27. 连续分配、链式分配、索引分配
28. 绝对路径、相对路径
29. 空闲链表、位示图
30. 直接共享、间接共享
31. 块设备和字符设备；独占设备、共享设备、虚拟设备
32. 通道方式、DMA方式
33. 单缓冲、双缓冲、缓冲池、循环缓冲
34. 寻道时间、传输时间

简答题

计算机中的存储器系统层次结构通常包括寄存器、高速缓存、内存和外部存储器。

寄存器：内容为CPU内部的高速存储单元，用于存储指令、数据和地址等。

缓存：位于CPU和主存之间的高速存储器，用于暂存CPU频繁访问的数据和指令。

内存：用于存储当前运行的程序和数据内容。

外部存储器：包括硬盘驱动器（HDD）、固态驱动器（SSD）、光盘、USB闪存等

从存储性能角度来看，存储系统遵循“速度容量成本”的权衡原则。寄存器速度最快但容量最小、成本最高；外部存储器速度最慢但容量最大、成本相对较低；缓存和内存处于中间位置，分别在速度、容量和成本之间取得平衡。这种层次结构使得系统能够在保证一定性能的同时，以较低的成本实现大容量的存储需求。

计算机程序从代码编写到运行完毕主要经历预处理、编译、汇编和链接四个阶段。预处理替换指令等内容，编译转换为汇编代码，汇编生成机器码文本表示，链接生成可执行文件并运行。

1. 主要方式：

静态链接、装入时动态链接、运行时动态链接

相同点：都是将程序的不同部分（模块）组合成一个完整的可执行程序。

不同点：

静态链接在编译时完成，生成独立可执行文件，体积较大且更新需重编译。装入时动态链接在装入内存时进行，可减小体积但增加装入时间。运行时动态链接在运行时按需进行，能进一步减少初始代码量，提高内存利用率，却增加了运行时复杂性和错误风险。

1. 首次适应算法：从内存的起始位置开始查找，找到第一个能够满足进程需要的空闲分区进行分配。
2. 最佳适应算法：从所有空闲分区中找到最小的能够满足进程需要的空闲分区进行分配。
3. 最坏适应算法：从所有空闲分区中找到最大的能够满足进程需要的空闲分区进行分配。
4. 循环首次适应算法：类似首次适应，但每次搜索都是从上次搜索结束的位置开始。
5. 快速适应算法：将内存分为若干个大小相等的分区，每个分区维护一个空闲链表，根据进程需要的大小在相应的链表中查找空闲分区进行分配。
6. 典型内存动态分区算法：

首次适应算法：从内存的起始位置开始搜索，分配第一个足够大的空闲分区。

最佳适应算法：在整个内存中搜索，找到能够满足需求且大小最小的分区。

最坏适应算法：找最大的分区，即使它比需求大得多。

循环首次适应算法：类似首次适应，每次搜索从上次结束位置开始。

1. 典型CPU调度算法：

先来先服务：按照进程到达的顺序进行调度。

短作业优先：优先调度估计执行时间最短的进程。

优先级调度：根据进程的优先级进行调度，优先级高的进程先执行。

时间片轮转：每个进程被分配一个固定的时间片，按顺序轮流执行。

相同点：

都是为了有效利用系统资源（内存空间或CPU时间），提高系统性能。

都需要根据一定的策略进行资源分配决策。

都可能受到系统状态动态变化的影响，需要灵活调整。

不同点：

资源类型：内存动态分区算法分配的是内存空间，CPU调度算法分配的是CPU时间。

目标：内存动态分区算法目标是有效利用内存空间，减少外部碎片；CPU调度算法目标是公平、高效地分配CPU时间，提高系统吞吐量。

性能影响：内存动态分区算法影响程序的加载时间和内存利用率；CPU调度算法影响进程的响应时间和等待时间。

算法选择依据：内存动态分区算法的选择取决于系统对内存利用率和分配速度的需求；CPU调度算法的选择取决于系统对响应时间和公平性的需求。

动态性：虽然两者都需要动态调整，但内存动态分区算法的调整更多与内存分配释放操作相关，而CPU调度算法的调整与进程状态变化（如就绪、运行、等待）密切相关。

1. 典型页面置换算法：

最近最少使用：淘汰最长时间未被访问的页面。

先进先出：按照页面进入内存的顺序进行淘汰。

最佳置换：理想情况下，淘汰未来最长时间不会被访问的页面。

时钟置换：使用类似时钟的数据结构，按顺序检查页面，淘汰最老的页面。

随机置换：随机选择一个页面进行淘汰。

1. 典型CPU调度算法：

先来先服务：按到达顺序调度。

短作业优先：优先调度估计执行时间短的进程。

优先级调度：根据优先级调度，高优先级先执行。

时间片轮转：分配固定时间片，按顺序轮转执行。

相同点：

都是为了优化系统资源的使用效率，提高系统整体性能。

都需要在多个选项中做出选择，以决定哪个页面被置换或哪个进程被调度。

都可以基于时间或使用频率等因素做出决策。

不同点：

资源类型与管理目标：页面置换算法管理内存中的页面，目标是减少页面错误，提高内存利用率；CPU调度算法管理CPU时间，目标是提高CPU利用率、公平性和系统吞吐量。

决策依据：页面置换算法主要依据页面的访问历史（如最近使用情况、进入时间等）；CPU调度算法依据进程的到达时间、预计执行时间、优先级等。

对系统的影响：页面置换算法影响程序执行效率和响应时间，频繁的页面置换可能导致抖动；CPU调度算法影响进程响应时间、等待时间以及系统的吞吐量。

算法实现复杂度：页面置换算法如OPT具有较高理论性能但实现复杂；CPU调度算法如时间片轮转相对简单，但优先级调度可能需要维护复杂的数据结构。

对公平性的考量：页面置换算法通常不直接涉及公平性问题，而CPU调度算法需要考虑进程间的公平性，避免饥饿现象。

1. 典型页面置换算法：

最近最少使用：淘汰最长时间未被访问的页面。

先进先出：按页面进入内存的顺序进行淘汰。

最佳置换：理想情况下，淘汰未来最长时间不会被访问的页面。

时钟置换：使用类似时钟的数据结构，按顺序检查页面，淘汰最老的页面。

随机置换：随机选择一个页面进行淘汰。

1. 典型磁盘调度算法：

先来先服务：按请求到达顺序进行服务。

最短寻道时间优先：优先服务距离当前磁头位置最近的请求。

扫描：磁头在磁盘两端之间来回扫描，每次移动到离当前位置最近的请求。

循环扫描：类似扫描算法，但磁头完成一次扫描后直接移动到另一端，而非返回。

最近最少使用：类似内存中的LRU算法，优先服务最近最少被访问的请求。

相同点：

都是用于管理共享资源（内存中的页面或磁盘I/O请求），提高资源使用效率。

都需要根据一定的策略选择资源分配对象（页面或磁盘请求）。

都对系统的性能产生重要影响，良好的算法能显著提高响应时间和吞吐量。

不同点：

操作对象不同：页面置换算法针对内存页面，关注页面换入换出；磁盘调度算法针对磁盘磁道扇区，关注磁头移动和数据访问顺序。

性能衡量指标不同：页面置换算法以缺页率和置换次数衡量性能，两者越低越好；磁盘调度算法以平均寻道时间和磁头移动距离衡量，越小越佳。

算法设计侧重点不同：页面置换算法重在预测页面使用减缺页；磁盘调度算法重在优化请求顺序降磁头移动时间。

1. 分别简述分页存储、分段存储和段页存储的地址变换过程。
2. 分页存储：

地址变换通过提取虚拟地址中的页号索引页表找到物理页号，再与页内偏移组合形成物理地址。

1. 分段存储：

地址变换通过提取虚拟地址中的段号索引段表获取基地址，与段内偏移相加形成物理地址。在访问之前，还需要检查段内偏移是否超出段长度，以防止越界访问。

1. 段页存储：

地址变换通过提取虚拟地址中的段号，索引段表获取页表基地址，与页号组合找到物理页号，再结合页内偏移形成物理地址，同时检查偏移是否越界。

中断处理程序的处理步骤如下：

1. 中断请求：当硬件设备或软件需要CPU注意时，发出中断请求信号。
2. 中断优先级判定（中断判优）：操作系统根据中断请求的优先级决定是否立 即响应该中断。当前正在处理更高优先级中断时，低优先级中断请求需等待。
3. 中断响应：CPU决定响应中断请求后，进入中断响应阶段，涉及硬件和软件协作，确保CPU正确处理即将来临的中断。
4. 中断处理：

保存当前任务的上下文，如寄存器状态和程序计数器。

识别中断源，可能通过查询中断向量表。

执行中断服务，如数据读取、硬件状态更新、错误处理等。

1. 中断返回：

恢复之前保存的上下文，使CPU能从中断发生的位置继续执行。

如果需要，重新开启中断，允许新的中断请求被处理。

可能涉及调度其他任务，如果中断处理改变了任务的优先级或状态。

1. 按用途分类：

系统文件：由系统软件构成的文件。大多数系统文件只允许用户调用，不允许用户读取，更不允许修改。

用户文件：由用户的源代码、目标文件、可执行文件或数据等构成的文件。用户将这些文件委托给系统保管。

库文件：由标准子例程及常用的例程等构成的文件。这类文件允许用户调用，但不允许修改。

1. 按数据类型分类：

文本文件：以字符形式存储数据，可直接读取和编辑，如源文件（源程序和数据构成，通常由终端或输入设备输入）。

二进制文件：以二进制代码形式存储数据，通常包括目标文件（源程序编译后但未链接的目标代码，后缀名常为“.obj”）和可执行文件（目标代码链接后形成）。

1. 按组织形式和处理方式分类：

普通文件：由ASCII码或二进制码组成的字符文件。一般用户建立的源程序文件、数据文件、目标代码文件及操作系统自身代码文件、库文件等都是普通文件，通常存储在外存储设备上。

目录文件：由文件目录组成的，用来管理和实现文件系统功能的系统文件，通过目录文件可以对其他文件的信息进行检索。由于目录文件也是由字符序列构成，因此对其可进行与普通文件一样的种种文件操作。

特殊文件：特指系统中的各类I/O设备。为了便于统一管理，系统将所有的输入/输出设备都视为文件，按文件方式提供给用户使用。

1. 一级目录结构：

采用单级目录形式，所有文件都列在一张目录表中。

查询方式为遍历检索整个目录表查找文件。

1. 二级目录结构：

包含主目录和用户目录两级。主目录列出所有用户目录，每个用户目录包含该用户的文件。

查询方式先在主目录中找到相应的用户目录，再在用户目录中查找文件。

1. 树结构：

根结点为主目录，分支结点为用户目录，子分支结点为文件，形成树状结构。

查询方式需要指出文件所在路径名，即从根目录到该文件的路径上各级目录名的组合，也称为文件全名。

1. 多级目录文件：

在树结构基础上可进一步扩展为多级目录，结构更复杂。

查询方式同样需要提供完整的路径名以定位文件。

1. 顺序结构：

优点：存储与管理简单，容易实现；支持顺序存取和随机存取；顺序存取速度快；所需的磁盘寻道次数和寻道时间最少。

缺点：需为每个文件预留物理块，不利于文件增长；不利于文件插入和删除。

1. 链式结构：

优点：提高磁盘空间利用率，无需为每个文件预留物理块；有利于文件插入和删除；有利于文件动态扩充。

缺点：存取速度慢，不适于随机存取；物理块间连接指针出错时数据丢失；寻道次数和时间多；链接指针占用空间，降低空间利用率。

1. 索引结构：

优点：无需为每个文件预留物理块；既能顺序存取，又能随机存取；满足文件动态增长、插入删除的要求。

缺点：寻道次数和时间较多；索引表本身带来系统开销，包括内外存空间和存取时间等。

1. 内存：

静态分配：在程序编译时确定内存分配，用于操作系统内核和关键系统程序。

动态分配：在程序运行时根据需要分配和释放内存，如C语言中的`malloc`和`free`。

分页：内存分为固定大小的页，每个页可独立加载到内存中。

分段：内存根据程序的逻辑结构分为不同的段，如代码段、数据段等。

段页式：结合分页和分段的特点，更灵活的内存管理。

1. 外存

连续分配：为文件分配连续的存储空间，如旧的文件系统。

链式分配：文件的每个块分散存储，通过链表结构连接。

索引分配：使用一个索引块记录文件块的位置，适用于大型文件。

相同点：

两者都需要有效的空间管理策略，确保存储资源合理利用。

都可能使用索引或表跟踪存储空间使用情况。

都面临如何高效分配和回收空间的问题。

不同点：

访问速度：内存访问速度远高于外存，内存分配注重快速访问和上下文切换。

存储介质：内存通常使用RAM，是易失性的；外存使用硬盘、固态硬盘等，是非易失性的。

使用场景：内存分配关注当前运行的程序和数据，外存分配需要考虑长期存储和数据备份。

1. 采用磁盘高速缓存：利用磁盘高速缓存技术提高I/O速度，是目前主要技术。
2. 提前读：对于顺序访问文件，可提前将下一盘块的数据读入高速缓存。
3. 延迟写：将可能被其他进程访问的数据挂到空闲缓冲队列末尾，而非立即写回磁盘。
4. 优化物理块的分布：尽量将一个文件存储在相邻盘块上，减少磁头切换磁道。
5. 使用更快的磁盘：采用更高性能的磁盘设备，如固态硬盘（SSD）替代传统机械硬盘，提升I/O速度。