名词解释

1. 地址映射：指将程序使用的逻辑地址（虚拟地址）转换为内存中的物理地址的过程。地址映射由硬件（如MMU）或操作系统完成。
2. 动态重定位：在程序执行期间，由硬件（如重定位寄存器）或操作系统动态地把程序中的逻辑地址转换为物理地址，使得程序可以在内存中的任意位置装入和执行。
3. 虚拟存储器：一种将程序所需的大量地址空间映射到有限的物理内存的方法。通过页式或段式管理机制，实现按需调入、置换，提供比实际物理内存更大的地址空间。
4. 静态链接：在程序编译连接阶段，将程序的各个模块及所需库函数在生成可执行文件时就链接在一起，执行时不再需要重新链接。
5. 对换：把暂时不用的进程或程序从内存调出到外存（如磁盘），以腾出内存空间给其他进程使用，当需要时再调回内存继续执行。
6. 设备驱动程序：操作系统中负责管理和控制具体硬件设备的程序，它为上层提供一致的接口，屏蔽硬件细节。
7. SPOOLing：假脱机技术，用于实现外设操作与CPU并行工作。先将用户输出写到磁盘（中间文件），由专门进程负责打印等，避免慢速外设阻塞用户进程。
8. I/O通道：一种独立的输入/输出处理单元，能自主管理数据在内存与外设间的传输，减轻CPU负担，实现并行I/O。
9. 文件系统：操作系统中用于组织、管理和存取文件的程序和数据结构，主要包括文件、目录、文件控制块、空闲空间管理等。
10. 目标文件：由编译器或汇编器生成的机器代码文件，还未完全链接成可执行文件，通常包含机器指令和符号表。
11. 文件的逻辑结构：指文件内容对用户呈现的组织方式，如流式文件（无结构）、顺序文件、索引文件等，侧重信息内容的组织。
12. 有结构文件：指具有特定内部组织格式的文件，如顺序文件、索引文件、顺序索引文件等，通常包含记录、字段等结构。
13. 位示图：用一组二进制位表示资源（如磁盘块、内存块）的使用情况，0表示空闲，1表示已占用，常用于空闲空间管理。
14. 程序接口：操作系统提供给用户程序调用的接口，如库函数、API等，用来访问操作系统服务。
15. 系统调用：用户程序请求操作系统内核提供服务的接口，常见如文件操作、进程控制、I/O管理等。
16. I/O中断：I/O设备完成或需要服务时向CPU发出的中断信号，使CPU暂停当前任务转去处理中断请求，实现设备与CPU的并行工作。
17. 文件管理系统：操作系统中负责对文件进行创建、删除、读写、保护、存取控制等管理的软件模块。
18. 文件：在操作系统中用于存储数据和信息的逻辑单位，是用户和程序进行信息交换和长期保存的载体。
19. 文件的逻辑结构：指文件内容对用户呈现的组织方式，如流式文件（无结构）、顺序文件、索引文件等，侧重信息内容的组织。
20. 文件的物理结构：文件在外存上的实际物理存放方式，如连续结构、链接结构、索引结构等，决定文件存取的效率和实现方式。

填空题

1. 重定位
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. 紧凑
6. 固定的、操作系统、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. 分配所用的数据结构，动态分区分配算法，分区分配与回收
10. 外部碎片
11. 分页式存储，分段式存储、段页式存储
12. 快表（或快TLB）
13. 逻辑地址空间和物理地址空间
14. 方便编程，信息共享，信息保护，动态链接，动态增长
15. 地址变换机构
16. 局部性、时间局部性和空间局部性
17. 处理器、操作指令类型单一，没有自己的内存，与cpu共享
18. 顺序，索引，顺序索引
19. 单极索引，多级索引，增量式索引
20. 动态重定位
21. 设备驱动程序，与设备无关的I/O软件，中断处理程序
22. 连续结构、链接结构、链式结构；索引结构
23. cpu寄存器、主储存器、辅助储存器
24. 连续分配、离散分配
25. 储存介质的储存性能，外存的分配方式
26. 顺序，索引，索引顺序
27. 连续分配、链式分配、索引分配
28. 绝对路径、相对路径
29. 空闲盘块链，空闲盘区链
30. 有向无环图，符号链接
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、环形缓冲
34. 寻道时间、传输时间

简答题

1. 计算机中的存储器系统层次结构通常包括cpu寄存器，主储存器，辅助寄存器。各层次包括：  
   cpu寄存器：寄存器

主储存器：高速缓存，主储存器，磁盘缓存

辅助储存器：固定磁盘，可移动存储介质

计算机存储系统呈现出“金字塔结构”，层次越高，速度越快、成本越高、容量越小；层次越低，容量越大、成本越低、速度越慢

速度：寄存器最快，Cache 次之，主存再次之，外存最慢。

容量：外存最大，主存中等，Cache 和寄存器最小。

成本：单位存储成本从寄存器到外存依次递减。

2.

**程序设计与编写**  
程序员使用高级程序设计语言（如 C、Java、Python 等）根据需求编写源程序代码。

**从磁盘中取出**

**编译或汇编**  
将源程序（高级语言）通过编译器翻译成目标程序（机器指令），有些低级语言如汇编则需要汇编器转成机器指令。

**链接**  
编译生成的目标文件与所需的库文件、模块文件通过链接程序合并，生成完整的可执行文件（如 .exe）。

**装入（装载）**  
由装入程序或操作系统把可执行文件从外存调入内存，为程序分配内存空间，完成地址重定位。

**执行**  
CPU 按指令顺序执行程序，程序通过系统调用访问操作系统服务，进行输入/输 出、文件操作等。

**运行结束与回收**  
程序执行完毕后，由操作系统回收分配的内存、关闭打开的文件和设备，释放相关资源。

3.

**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：**  
静态链接、装入时动态链接和运行时动态链接这三种方式的共同点是，它们都是为了将程序和所需的库代码连接起来，使程序能够正确调用库函数和外部模块，保证程序能够正常运行。

**不同点：**

**链接时机不同**：静态链接是在编译阶段完成，生成包含所有代码的完整可执行文件；装入时动态链接是在程序加载到内存时由操作系统完成链接；运行时动态链接则是在程序运行过程中，根据需要动态加载和链接库函数。

**文件大小不同**：静态链接生成的可执行文件较大，因为包含所有依赖库代码；动态链接方式生成的可执行文件较小，库代码在运行时单独加载。

**加载速度和运行效率不同**：静态链接的程序加载最快，运行效率最高；装入时动态链接的程序加载速度居中，运行效率较高；运行时动态链接由于需要在运行过程中动态查找和加载库，加载速度和运行效率相对较低。

**更新维护灵活性不同**：静态链接不方便维护，库更新需要重新链接程序；动态链接方式更加灵活，库文件更新不影响程序运行，支持库的独立更新和替换。

**共享性不同**：静态链接生成的程序包含库的副本，不支持多个程序共享库代码；动态链接允许多个程序共享同一个库文件，节省内存和存储空间。

4.

分为顺序分配算法，索引式分配算法

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。

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。

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。

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。

基于索引

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。

伙伴算法:将内存划分为大小为 2^k（k 为整数）的块，每次分配时，从足够大的块开始切分成两个“伙伴”块，直到得到满足请求大小的最小块。

哈希算法：利用哈希函数对请求的内存块大小或起始地址进行哈希，快速定位空闲区，减少搜索时间。

5

1. 典型内存动态分区算法：

首次适应算法（First Fit）

循环首次适应算法（Next Fit）

最佳适应算法（Best Fit）

最差适应算法（Worst Fit）

快速适应算法

伙伴算法（Buddy System）

哈希算法

1. 典型CPU调度算法：

先来先服务（FCFS, First-Come, First-Served）

短作业优先（SJF, Shortest Job First）

优先级调度（Priority Scheduling）

时间片轮转（Round Robin, RR）

多级队列

多级反馈队列（Multilevel Feedback Queue）

相同点：

用于管理有限资源的重要策略。它们都需要在多个请求者之间动态分配资源，保证系统资源得到合理利用和公平分配，从而提升系统整体性能和效率。

不同点：

内存动态分区分配算法主要关注的是对物理内存这一空间资源的分配和回收，重点解决如何选择合适的空闲内存块以减少内存碎片和提高内存利用率；而 CPU 调度算法则侧重于时间资源的管理，通过合理安排进程运行顺序和时间片，优化系统的响应时间、周转时间和吞吐量。两者的资源类型不同，分配时机和关注的优化目标也有所区别。

6

1. 典型页面置换算法：

先进先出算法（FIFO, First-In First-Out）

最近最少使用算法（LRU, Least Recently Used）

最佳置换算法（OPT, Optimal）

时钟算法（Clock）

页面缓冲

1. 典型CPU调度算法：

先来先服务（FCFS, First-Come, First-Served）

短作业优先（SJF, Shortest Job First）

优先级调度（Priority Scheduling）

时间片轮转（Round Robin, RR）

多级队列

多级反馈队列（Multilevel Feedback Queue）

相同点：

两者都是操作系统中用于资源管理的调度策略。

都涉及对有限资源的动态分配与管理。

都通过算法优化系统性能，提高响应速度和效率。

不同点：

页面置换算法管理的是内存中的页面空间资源，目标是减少缺页率，提高内存使用效率。

CPU调度算法管理的是CPU时间资源，目标是优化进程调度顺序，提高系统响应和吞吐量。

页面置换算法多关注历史访问信息，决定哪个页面被替换；CPU调度算法多关注进程状态和优先级，决定哪个进程运行。

7.

典型页面置换算法：

先进先出算法（FIFO, First-In First-Out）

最近最少使用算法（LRU, Least Recently Used）

最佳置换算法（OPT, Optimal）

时钟算法（Clock）

页面缓冲

1. 典型磁盘调度算法：

典型的磁盘调度算法

先来先服务（FCFS）

最短寻道时间优先（SSTF）

扫描算法（SCAN，又称电梯算法）

循环扫描算法（C-SCAN）

cLOOK算法

相同点：

都是操作系统中用于优化有限硬件资源访问效率的调度算法。

都需要根据请求的访问顺序或历史访问情况，合理选择下一次的操作目标。

都以减少系统开销（缺页中断次数或磁头寻道时间）为目标，提高系统性能。

不同点：

页面置换算法针对的是内存管理中的页面替换问题，核心是决定哪个内存页被换出以腾出空间。

磁盘调度算法针对的是磁盘I/O请求的调度问题，核心是优化磁头移动路径以减少寻道时间。

页面置换更多关注访问时间的先后顺序和访问频率；磁盘调度更多关注空间上的物理位置及磁头移动距离。

8.分别简述分页存储、分段存储和段页存储的地址变换过程。

### 分页存储的地址变换过程

**逻辑地址结构**：逻辑地址由页号（Page Number）和页内偏移（Offset）组成。

取出逻辑地址中的页号部分，查页表，找到对应的物理页框号。

将页框号与页内偏移组合，形成物理地址。

### 分段存储的地址变换过程

**逻辑地址结构**：逻辑地址由段号（Segment Number）和段内偏移（Offset）组成。

用段号索引段表，获得段基址和段限长。

检查偏移是否在段限长范围内，合法则将段基址与偏移相加，得到物理地址。

### 段页存储的地址变换过程

**逻辑地址结构**：逻辑地址由段号、页号和页内偏移三部分组成。

用段号查段表，得到该段的页表基址和段限长。

用页号查页表，得到对应页框号。

将页框号和页内偏移组合，得到物理地址。

9.

**保存现场**  
当中断发生时，CPU首先保存当前正在执行程序的状态（如程序计数器PC、状态寄存器等），以便中断处理完成后能恢复继续执行。

**确定中断源**  
中断控制器或CPU确定引发中断的具体设备或中断类型。

**响应中断**  
根据中断类型，跳转到对应的中断处理程序（中断服务程序ISR）的入口地址。

**执行中断服务程序**  
处理中断请求，完成相应的服务任务，如数据传输、状态更新、错误处理等。

**恢复现场**  
中断处理完成后，恢复之前保存的现场信息，确保程序能够从中断点继续执行。

**返回主程序**  
执行返回中断指令（如IRET），恢复程序正常运行。

10.

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：

源文件，目标文件，可执行文件

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。

11.

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：

从根目录开始，按路径名逐层进入子目录，最终定位文件。

查询时需要遍历树结构，路径名解析较复杂。

1. 无环图目录文件：

查询时根据路径，按层次访问目录项。

由于存在多个父目录，查询时可能存在多条路径指向同一文件。

必须在遍历时检测并防止循环访问（通过标记访问节点等机制），确保查询终止。

12.

### 顺序文件组织（Sequential Organization）

优点是结构简单，存储利用率高，非常适合批处理和大量顺序访问的场景。

缺点是随机访问效率低，不适合需要频繁插入和删除记录的情况，因为要查找或修改特定记录时，需要顺序扫描整个文件。

### 2. 索引文件组织（Indexed Organization）

优点是支持快速的随机访问，能够快速定位记录，适合在线查询和频繁更新的场合。

缺点是需要维护额外的索引结构，增加了存储空间开销和管理复杂度，同时索引维护也会带来一定的性能开销。

### 3. 索引顺序文件组织（Indexed Sequential Organization）

优点是结合了顺序文件和索引文件的优势，既能实现顺序访问也支持快速随机访问，适用范围较广。

缺点是结构较为复杂，索引的维护成本较高，尤其是在频繁插入和删除时，管理起来比较麻烦。

13.

1. 内存：

**连续分配**  
内存被划分为若干连续的内存区，进程获得一块连续的内存空间。分配和回收简单，但容易产生外部碎片。

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：

两者都需要合理分配有限的存储空间，保证数据和程序的有效存放。

都存在连续分配和非连续分配两种基本策略。

都面临碎片问题，需要采用相应算法减少碎片。

不同点：

内存空间分配更注重速度和实时性，采用分页、分段等机制提高管理效率和灵活性。

外存空间分配更注重存储容量和数据的持久性，采用索引和链接等方法支持文件的存储和访问。

内存多用于程序执行和临时数据存储，分配粒度较小且频繁；外存用于长期存储，分配粒度较大且访问相对较慢。

14.

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. 虚拟盘;由操作系统或专门的软件创建，它将一部分物理存储空间映射为一个虚拟硬盘。
6. 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。