- 练习1：理解first-fit 连续物理内存分配算法（思考题）
- 练习2：实现 Best-Fit 连续物理内存分配算法（需要编程）
  - 1、内存初始化
    - 功能：
    - 实现过程：
  - 2、页框分配
    - 功能：
    - 实现过程：
  - 3、页框释放
    - 功能：
    - 实现过程：
  - 4、Best-Fit 算法的改进空间
- 扩展练习Challenge：buddy system（伙伴系统）分配算法（需要编程）
- 扩展练习Challenge：任意大小的内存单元slub分配算法（需要编程）
- 扩展练习Challenge：硬件的可用物理内存范围的获取方法（思考题）
  - 1. BIOS/UEFI信息读取
  - 2. 内存映射 I/O（MMIO）
  - 3. 设备树（Device Tree）
  - 4. 内存检测算法
  - 5. 系统管理模式（SMM）
  - 6. ACPI（高级配置和电源接口）
  - 7. 驱动程序与内核模块

练习1：理解first-fit 连续物理内存分配算法（思考题）

练习2：实现 Best-Fit 连续物理内存分配算法（需要编程）

1、内存初始化

功能：

初始化给定数量的物理内存页框（struct Page），清空它们的标志和属性，并将它们加入到空闲链表中。

实现过程：

- 使用循环遍历每一个页框，将它们的标志（flags）和属性（property）重置为0，并将引用计数设置为0。
- 将第一个页框的属性设置为总页框数。
- 更新空闲页框的总数（nr_free）。
- 在空闲链表中插入新初始化的页框，保持链表的有序性。

2、页框分配

功能：

根据请求的页框数量（n），从空闲链表中查找并分配合适的页框。

实现过程：

- 确保请求的页框数大于0，并且不超过当前可用的页框数（nr_free）。
- 遍历空闲链表，寻找最小的满足请求的空闲页框。使用min_size变量跟踪找到的最小空闲页框数量。
- 一旦找到合适的页框，就从空闲链表中删除它，并在必要时分割它。
- 更新总的空闲页框数量并清除已分配页框的属性。

3、页框释放

功能：

释放指定数量的页框，将它们标记为可用。

实现过程：

- 清除每个页框的标志和引用计数，设置属性为释放的页框数，更新空闲页框数量（nr_free）。
- 将释放的页框插入空闲链表，保持链表的顺序。
- 检查是否可以合并相邻的空闲页框，以减少内存碎片。分别检查前后相邻页框是否连续，进行合并操作。

4、Best-Fit 算法的改进空间

尽管上述实现已经提供了一种有效的内存分配方式，但 Best-Fit 算法本身在某些情况下可能存在性能问题和内存碎片的问题。以下是一些可能的改进方向：

- 内存碎片：Best-Fit 可能会导致内存碎片化，因为它会留下一些非常小的空闲块，这些小块可能无法满足后续的分配请求。可以考虑在分配时合并较小的空闲块，或使用更复杂的合并策略。
- 查找效率：在当前实现中，每次分配时都需要遍历整个空闲链表以找到最佳匹配。这种查找可能是线性的，导致性能下降。可以考虑使用更高效的数据结构（如平衡树或散列表）来存储空闲块，从而加速查找过程。
- 动态合并策略：结合其他算法（如最坏适应或首次适应）实现动态合并策略，以减少内存分配时的碎片化。动态选择分配策略可以根据当前的内存状态选择最合适的算法。
- 使用位图管理内存：采用位图管理内存，以便更快速地跟踪空闲和已分配的页框。这种方式可以减少搜索空闲块所需的时间。
- 固定大小的块：考虑将内存分配限制在固定大小的块（如4KB），这样可以减少分配和释放过程中可能出现的复杂性。

扩展练习Challenge：buddy system（伙伴系统）分配算法（需要编程）

扩展练习Challenge：任意大小的内存单元slub分配算法（需要编程）

扩展练习Challenge：硬件的可用物理内存范围的获取方法（思考题）

获取硬件的可用物理内存范围对于操作系统（OS）是一个重要任务，尤其是在启动过程中。因为操作系统需要了解系统可用的内存资源，以便有效地管理内存、分配资源以及优化性能。以下是几种可能的方法来获取可用物理内存范围，即使操作系统在启动时无法提前知道这些信息：

1. BIOS/UEFI信息读取

在计算机启动时，BIOS或UEFI固件会初始化硬件并提供有关系统配置的信息。操作系统可以通过以下方式获取可用内存范围：

- **BIOS中断调用**：在传统的BIOS系统中，可以通过调用特定的BIOS中断（如`INT 0x15`）来查询内存映射信息。
- **UEFI接口**：在UEFI系统中，操作系统可以使用EFI Boot Services中的`GetMemoryMap`函数，获取系统内存的映射，包括可用和保留的内存区域。

2. 内存映射 I/O（MMIO）

通过直接访问系统内存和硬件寄存器，操作系统可以获取系统内存的布局。以下是相关步骤：

- **查询内存控制器**：在现代系统中，内存控制器（如北桥芯片）通常会提供内存映射信息。通过与内存控制器通信，操作系统可以获取可用内存的信息。
- **读取内存区域**：直接读取特定地址（如`0x00000000`到`0xFFFFFFFF`范围内）以识别可用的内存区域。这通常涉及在启动阶段对内存进行探测。

3. 设备树（Device Tree）

在一些嵌入式系统或特定架构（如ARM架构）中，设备树是描述硬件组件的一种数据结构。操作系统可以解析设备树，获取内存信息。

- **解析设备树**：设备树中通常包含内存节点，操作系统可以通过解析这些节点来获取可用物理内存的范围。

4. 内存检测算法

在启动时，操作系统可以使用内存检测算法来识别可用内存：

- **内存检测**：操作系统可以通过尝试访问某一内存区域并检测是否会产生错误来识别内存的可用性。这种方法通常涉及遍历一定范围的内存地址，以识别哪些区域是可用的。

5. 系统管理模式（SMM）

在某些系统中，SMM可用于在系统运行时执行代码并访问硬件资源。操作系统可以通过SMM获取内存范围信息。

6. ACPI（高级配置和电源接口）

操作系统还可以通过ACPI表获取内存配置。ACPI表提供了系统的硬件配置和管理信息，包括可用内存范围。

- **解析ACPI表**：通过读取ACPI中的`Memory Mapped`相关表（如MADT、SRAT），操作系统可以获取内存的分布情况。

7. 驱动程序与内核模块

在某些情况下，操作系统的驱动程序和内核模块可以在系统运行时访问特定硬件资源，以获取更多的内存信息。