操作系统实验一 可变分区存储管理

518021910879 陈瀚嵩

1 实验题目

编写一个 C 语言程序，模拟 UNIX 的可变分区内存管理，使用**循环首次适应法**实现对一块内存区域的分配和释放管理。

2 算法思想及概要设计

2.1 可变分区存储管理

与固定分区法不同，可变分区存储管理法并不预先将内存划分为分区，而是等到作业运行需要内存时，向系统申请空闲内存区中的一块内存，其大小等于作业所需内存大小，避免产生“内零头”。

2.2 循环首次适应法

可变分区存储管理有四种内存分配策略，本次实验使用循环首次适应法。

该算法把空闲表设计成链接结构的循环队列，各空闲区按地址从低到高的次序存入空闲表。同时设置一个查找指针，初始时指向空闲表的第一项。

分配时，从查找指针所指的项开始，第一次找到满足要求的空闲区时，就分配所需大小的空闲区，修改表项，调整查找指针使其指向被分配的空间之后的空闲区。

释放时，从查找指针所指的项开始，找到指定地址，将指定大小的内存释放，视情况修改指针值，使其指向正确的空闲表项。

3 重要模块的功能、详细设计以及接口说明

3.1 初始化模块

3.1.1 功能

初始化各变量，申请内存空间。

3.1.2 详细设计

该模块被调用时，为 `ptr` 指针分配空间，初始化其所指结构体成员，对于其中的地址项分配预设大小 (1 KB) 空间，设定初始的空闲块状态。记录分配的内存起止位置。

之后，打印初始化成功提示，及所分配内存的起止地址。

3.1.3 接口说明

调用时不带参数；无返回值。被调用时即执行预设功能。

3.2 内存分配模块

3.2.1 功能

依据循环首次适应分配方法，尝试分配指定大小内存。

3.2.2 详细设计

该模块被调用时，首先检查申请大小是否为零，若为零则报错；再检查是否还有空闲块，若无则返回内存不足提示。

然后，从 `ptr` 指针所指位置开始，依次循环查找是否有符合要求大小的空闲内存块，若找到，则分配预设大小空间，改变该指针所指空闲内存块起始位置及大小；若该指针所指空闲内存块大小为零，则删除该内存块的记录，并释放对应内存，若所有内存都已被分配，则将 `ptr` 指针置空。

若循环一圈，都没有找到符合要求大小的空闲块，则返回内存不足提示。

3.2.3 接口说明

调用时带有 `size` 参数，表示申请内存大小；返回值为 `char*` 类型，指向地址为所申请到内存块的起始地址，若未申请到符合要求的内存，返回 `NULL` 。

3.3 内存释放模块

3.3.1 功能

依据循环首次适应分配方法，尝试释放指定大小内存。

3.3.2 详细设计

首先判断要求释放的内存是否越界（超出初始分配的内存地址范围），若越界则返回越界提示。

然后，从 `ptr` 指针所指位置开始，依次循环查找指定起始地址的内存块，使得预释放地址位于 `ptr` 与 `ptr->next` 指针所指地址（即前空闲块、后空闲块）之间（包括 `ptr->next` 指针所指地址经历循环，从头部重新开始的情况）。

若释放区域与前空闲块（或后空闲块）相连，则将它们合并；若释放区域与前、后空闲块都相连，则将前后空闲块与释放区域合为一个空闲块；否则，新建一个空闲块，插入到前后空闲块之间。

特别地，若释放区域覆盖了后空闲块，则进行迭代，考虑后空闲块之后的其他空闲块是否也能被合并。

3.3.3 接口说明

调用时带有 `size` 、 `addr` 参数，分别表示释放内存大小、释放内存起始地址；无返回值。被调用时尝试释放指定起始地址、指定大小的内存块。

3.4 空闲内存块打印模块

3.4.1 功能

从 `ptr` 指针所指位置开始，打印当前所有空闲块信息。

3.4.2 详细设计

该模块被调用时，首先检查是否还有空闲块，若无则返回无空闲内存块提示。

然后，从 `ptr` 指针所指位置开始，循环一圈查找空闲的内存块，打印其起始地址和大小。

3.4.3 接口说明

调用时不带参数；无返回值。被调用时即执行预设功能。

4 重要数据结构及变量说明

4.1 数据结构

本次实验管理空闲内存区的数据结构采用链接法，每一个空闲分区用一个 `map` 结构管理：

```
struct map {
    unsigned m_size;
    char m_addr;
    struct map *next, *prior;
}
```

该结构存储空闲分区的大小，首地址，以及前一块 / 后一块空闲分区。

4.2 变量

- 查找指针 `*ptr` 为 `map` 类型
- 记录初始分配内存地址起始、结束位置的指针 `*begin_addr` 、 `*end_addr` 为 `map` 类型

5 测试方法及结果

5.1 编译程序

在测试之前，使用 `gcc main.c -o main` 命令编译程序。

5.2 测试方法

在Windows环境下，使用交互式测试方法，以 `./main.exe` 运行程序，用户指定分配或释放指定内存块，查看程序运行结果。

5.3 测试结果

在用户输入时，着重考虑极端情况，例如分配内存时空间不足、释放的内存地址覆盖多个现有空闲块、释放内存时地址越界等情况。对于测试语句，程序运行正常，结果正确。

一次具体测试的控制台输入、输出实例如下：

```
Memory init succeed!
The begin address of the memory is 10432800.
The end address of the memory is 10433824.
```

```
m 1024
10432800
Now there is no free memory.
f 1 10432800
Now the free momory blocks are:
Address:10432800, Size:1
f 1 10432820
Now the free momory blocks are:
Address:10432820, Size:1
Address:10432800, Size:1
f 1 10430000
Error! No such address!
Now the free momory blocks are:
Address:10432820, Size:1
Address:10432800, Size:1
f 1 10432840
Now the free momory blocks are:
Address:10432840, Size:1
Address:10432800, Size:1
Address:10432820, Size:1
f 1 10432860
Now the free momory blocks are:
Address:10432860, Size:1
Address:10432800, Size:1
Address:10432820, Size:1
Address:10432840, Size:1
f 100 10432840
Now the free momory blocks are:
Address:10432840, Size:100
Address:10432800, Size:1
Address:10432820, Size:1
m 1
10432840
Now the free momory blocks are:
Address:10432841, Size:99
Address:10432800, Size:1
Address:10432820, Size:1
m 100
Error! Memory not enough!
Now the free momory blocks are:
Address:10432841, Size:99
Address:10432800, Size:1
Address:10432820, Size:1
f 1 10432800
Now the free momory blocks are:
Address:10432800, Size:1
Address:10432820, Size:1
Address:10432841, Size:99
m 1
10432800
Now the free momory blocks are:
Address:10432820, Size:1
Address:10432841, Size:99
m 1
10432820
Now the free momory blocks are:
Address:10432841, Size:99
```

```
m 1
10432841
Now the free momory blocks are:
Address:10432842, Size:98
m 8
10432842
Now the free momory blocks are:
Address:10432850, Size:90
f 95 10432850
Now the free momory blocks are:
Address:10432850, Size:100
m 100
Now there is no free memory.
q
Program succefully run.
```

6 结果及错误的分析

6.1 结果分析

本次实验我完成了对 UNIX 可变分区存储管理——循环首次适应法分配、管理内存区域的模拟。在实验过程中，我对可变分区存储管理的理解得到加强，进一步熟悉了 C 语言指针的使用、链表上的基本操作等疑难问题。在完成实验后，我使用 C 语言编写大型系统程序的能力有了很大提升。

考虑到时间限制，本次实验所编写程序还有若干不足之处。日后可改进的方面包括：

- `lfree()` 函数使用相对位置进行管理，节省测试过程中所花费精力
- 完善文件测试功能及自动化测试，更全面地发现程序存在的问题
- 增加对于其他三种可变分区存储管理的内存分配策略的支持，进一步增强对可变分区存储管理的理解
- 进一步优化代码细节

6.2 错误分析

本次实验过程中所犯错误主要是由于对极端情况考虑不足所导致的，例如所释放的内存位于最后一个空闲块之后，或是第一个空闲块之前、释放内存较大，覆盖了多个现有空闲内存块，需逐一合并等情况。对此，我进行了全面、完整的测试，确保极端情况被充分考虑到，程序得以正确执行。