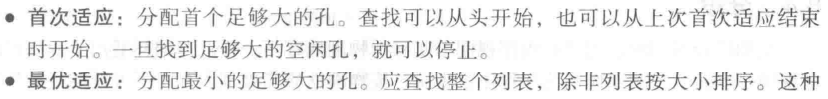
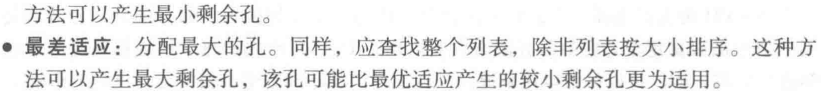
**Project 7: Contiguous Memory Allocation**

（一）问题分析

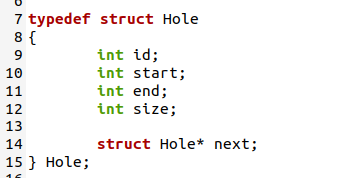
需要实现一个连续内存分配模型，支持4种操作：分配内存，释放内存，整合内存，查看内存使用情况。其中，分配内存可选三种策略：首次适应、最优适应或最差适应。





（二）实现细节

一、数据结构



用链表来模拟连续内存空间，结点有五个属性：

1、id：记录结点类型——id = -1表示该结点为孔，id = x 表示该结点为进程Px。

2、start：结点在内存中的开始位置。

3、end：结点在内存中的结束位置。

4、size：结点在内存中占多少空间。

5、\*next：指向下一个结点的指针。

二、全局变量

1、\*memory：指向连续内存链表头结点的指针。

2、MAX：连续内存空间的大小。

三、函数

1、request()

（1）检查传入的参数是否合法：size是否为正，id是否非负。非法则在控制台输出错误信息，退出函数。

（2）根据type调用对应的策略的函数。如果没有对应的策略，则在控制台输出错误信息。

（3）退出函数。

2、first\_fit()

（1）用指针p来遍历链表。

（2）如果p指向的结点为孔：

①如果该孔的空间大小等于进程申请的空间的大小，则直接整块分配，即根据申请空间的进程的id更新该孔的id。退出函数。

②如果该孔的空间大小大于进程申请的空间的大小，则逻辑上需要将该孔一分为二，前一部分根据申请空间的进程修改各属性，后一部分依据剩余空间修改属性。实现上，p指向的结点依据申请空间的进程修改属性，还需要新建一个结点，依据剩余空间初始化属性。最后将该新结点插入到p指向的结点之后。退出函数。

③如果该孔的空间大小小于进程申请的空间的大小，则继续遍历。

（3）如果p指向的结点不为孔，则继续遍历。

（4）遍历结束后，仍然没有退出函数，说明没有符合条件的空间可供分配，在控制台输出错误信息，退出函数。

3、best\_fit()

（1）用指针p来遍历链表，指针best来记录最优孔。

（2）如果p指向的结点为孔：

①如果该孔为遍历过程中遇到的第一个孔，则用best记录它；

②如果该孔的空间大于等于进程申请的空间，且小于目前best指向的孔的空间，则用best记录它；

③如果该孔的空间小于进程申请的空间，则继续遍历。

（3）如果p指向的结点不为孔，则继续遍历。

（4）遍历结束后，如果best不为孔，明没有符合条件的空间可供分配，在控制台输出提示信息，退出函数。如果best为孔：

①如果该孔的空间大小等于进程申请的空间的大小，同理first\_fit()（2）①。

②如果该孔的空间大小大于进程申请的空间的大小，同理first\_fit()（2）②。

4、worst\_fit()

（1）用指针p来遍历链表，指针worst来记录最大孔。

（2）如果p指向的结点为孔：

①如果该孔为遍历过程中遇到的第一个孔，则用worst记录它。

②如果该孔的空间大于等于进程申请的空间，且大于目前worst指向的孔的空间，则用worst记录它。

③如果该孔的空间小于进程申请的空间，则继续遍历。

（3）如果p指向的结点不为孔，则继续遍历。

（4）遍历结束后，如果worst不为孔，明没有符合条件的空间可供分配，在控制台输出提示信息，退出函数。如果worst为孔：

①如果该孔的空间大小等于进程申请的空间的大小，同理first\_fit()（2）①。

②如果该孔的空间大小大于进程申请的空间的大小，同理first\_fit()（2）②。

5、release()

（1）检查传入的参数是否合法：id是否非负。

（2）首先处理头结点为目标结点的情况，对于头结点存在三种情况：

①head->NULL：直接释放空间，即修改id为 -1，退出函数。

②head->process：直接释放空间，即修改id为 -1，退出函数。

③head->hole：逻辑上释放空间后还需要将空间合并。实现上，求出合并后有多少空间，据此修改head指向的结点的属性。还需要释放掉后一个结点。退出函数。

（3）如果头结点不是目标结点，则用指针p遍历链表，指针prev记录当前结点的前一个结点。

（4）p为目标结点有六种情况：

①process->target->NULL：直接释放空间，即修改id为 -1，退出函数。

②process ->target->process：直接释放空间，即修改id为 -1，退出函数。

③process ->target->hole：逻辑上释放空间后还需要将空间合并。实现上，求出合并后有多少空间，据此修改target指向的结点的属性，然后释放掉后一个结点。退出函数。

④hole->target->NULL：逻辑上释放空间后还需要将空间合并。实现上，求出合并后有多少空间，据此修改prev指向的结点的属性，然后释放掉target指向的结点。退出函数。

⑤hole->target->process：逻辑上释放空间后还需要将空间合并。实现上，求出合并后有多少空间，据此修改prev指向的结点的属性，还要将prev的next指针指向process，然后释放掉target指向的结点。退出函数。

⑥hole->target->hole：逻辑上释放空间后还需要将空间合并。实现上，求出合并后有多少空间，据此修改prev指向的结点的属性，还要将prev的next指针指向target后的第二个结点，然后释放掉target指向的结点和target后的第一个结点。退出函数。

（5）遍历结束后，仍然没有退出函数，说明没有符合条件的进程，在控制台输出错误信息，退出函数。

6、compact()

（1）如果内存里没有进程或全部为一个进程，则不需要整合。退出函数。

（2）用指针p遍历链表，指针prev记录上一个为process的结点，初始化为空。用unused\_space记录剩余空间总和。

（3）如果p指向的结点为孔，则更新unused\_space，释放掉这个结点。

（4）如果p指向的结点为进程：

①如果prev为空即这是第一个进程，则用prev记录这个结点。

②如果这是下一个进程，则将p指向的结点连接到prev指向的结点之后，修改p指向的结点的属性，然后让prev记录p指向的结点，p继续遍历。

（5）遍历结束后，内存中所有的进程对应的结点已相邻，所有孔对应的结点都被释放。新建一个结点，根据prev和unused\_space修改其属性，然后把它连接到链表末尾。

（6）退出函数。

7、stat()

（1）用指针p遍历链表。

（2）根据各结点属性，输出相应信息。

（3）退出函数。

四、主函数

1、从命令行参数获取MAX。

2、初始化memory。

3、仿照project2-1：Unix Shell写命令行“allocator>”主循环代码框架。

4、input[]存储输入的指令。

5、RQ指令

处理input[]得到进程编号id，进程申请的空间size，采用的分配策略type，调用函数request()。

6、RL指令

处理input[]得到进程编号id，调用函数release()。

7、C 指令

调用函数compact()。

8、STAT指令

调用函数stat()。

9、X指令

退出allocator命令行，即程序终止

（三）运行结果

