OS-Project7

常烁晨-521021910369

Task1: base部分

1、任务简介：本部分实现一个普通的内存分配器，用于将段页式的连续内存进行分配/释放功能，同时记录内存分配的进程。（6分）内存分配有三种模式，分别是F(first fit), B(best fit), W(worst fit) 。

2、代码实现：

（1）初始化定义：本实验用连续的数组进行内存的存储。实验要求的内存大小为 1M ，每个 Page 的大小是 256 字节，总共有 4096 个页。

```
int *Memory_Page;
const int PageSize = 4096;
const int PageNum = 256;
void init(){
    Memory_Page = (int*)malloc(PageNum *
sizeof(int));
    for(int i = 0; i < PageNum; i++)
        Memory_Page[i] = -1;}
```

（2）内存分配实现：根据输入的参数（F、W、B）来确定内存分配的规则。当输入 F 时，从数组头开始寻找第一个可以存下所需空间的位置，将相应的数组值保存为对应进程。输入 W 时，选择当前最大的连续空间进行分配，输入 B 时，则选择能存放相应进程的最小的空间。

```
if(j - i >= page_num){
for(int k = i; k < i + page_num; k++)
    Memory_Page[k] = process;}//F

while(j < PageNum && Memory_Page[j] == -1)j++;
if(j - i >= page_num && j - i > max){
    max = j - i; start = i;}i = j;//W

while(j < PageNum && Memory_Page[j] == -1)j++;
if(j - i >= memory_size && j - i < min){
    min = j - i; start = i;}i = j;//B
```

（3）内存释放实现：直接遍历数组，将对应进程设置为 -1。

（4）其他功能（输入 STAT：遍历数组打印对应内存空间的分配情况。输入 exit：退出循环）该部分代码比较简单，不再赘述。

Task2: bonus部分

1、任务简介：维护一个双链表，用于将所有的空闲内存块用双链表连接，并加速 W 和 B 的策略。从上文可以看出，在没有双链表的情况下，每次需要遍历整个数组空间，寻找最优情况或者最坏情况。维护双链表后，每次对链表节点进行操作后，维护双链表的有序性（递增），因此 B 策略从开头选取， W策略从结尾选取。

2、代码实现：

```
void sort_forward(MemoryNode *node)//node节点向前比较;
void sort_backward(MemoryNode *node)//node节点向后比较;
void division(MemoryNode *node, int num){
    if(num == node->end - node->start + 1)
    //删除节点
    else node->start += num; sort_forward(node);}
void merge(int start, int end){//start到end释放
    if …… elseif ……//特殊情况
    else{
    MemoryNode *node =malloc(sizeof(MemoryNode));
    node->start = start; node->end = end;
    node->next = head->next;
    node->prev = head; head->next->prev = node;
    head->next = node; sort_forward(node);}
void memory_request(int process, int memory_size, char
*allocation_type){
    if(allocation_type[0] == 'F'): 默认
    if(allocation_type[0] == 'W'):
        if tail->prev 满足条件，则保存，否则报错;
        division(tail->prev, page_num);
    if(allocation_type[0] == 'B')
        while(node 不满足要求) node=node->next;
        division(node, page_num);}
void memory_release(int process){
    merge (begin, end) ;}
```

实验结果展示：

```
parallels@ubuntu-linux-22-04-desktop:~/final-src-osc10e/ch10$ ./allocator_bonus

allocator>RQ P0 5000 W
Memory allocated successfully!

allocator>RQ P1 2048 W
Memory allocated successfully!

allocator>RQ P2 5000 B
Memory allocated successfully!

allocator>RL P1
Memory released successfully!

allocator>STAT
Addresses[0:8191] Process P0.
Addresses[8192:12287] Unused.
Addresses[12288:20479] Process P2.
Addresses[20480:1048575] Unused.

allocator>RQ P1 2048 W
Memory allocated successfully!

allocator>PQ P3 2048 B
Invalid command

allocator>RQ P3 2048 B
Memory allocated successfully!

allocator>STAT
Addresses[0:8191] Process P0.
Addresses[8192:12287] Process P3.
Addresses[12288:20479] Process P2.
Addresses[20480:24575] Process P1.
Addresses[24576:1048575] Unused.

allocator>exit
```