Operating System Project 7

潘梓丞 517030910349

The goal of the project is to create an allocator to show the steps when memory is allocated
The porject was done by VM VirtualBox 5.2.18
The code are written by C and the library needed will be shown in code

idea

Basicly, an allocator definitely knows all the process and the location it occupied. So we use a struct to store the $start-location, length, name, state$ of a process. Additionally, if an allocator only know the information of all the processes, it has to recalculate every time we do operations, so we use some extraordinary space to store the location of free memory.

problem

When we insert a process, it's easy, we just find the satisfy space we stored in the array $unuse$. But when we have to release, there are too many conditions which lead a lot of bugs. Using a link list will save lots of problems, but in this project, I just use arrays and classified discussion.

code

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

#define NUMBER_OF_PROCESSES 10

int main(int argc, char *args[])
{
    struct{
    int empty;
    int pos;
    int length;
    char name[5];
    } process[NUMBER_OF_PROCESSES];

    int i,j,s;
    int memory=atoi(args[1]);
    for(i=0;i<NUMBER_OF_PROCESSES;i++){
        process[i].empty=0;
        process[i].pos=0;
        process[i].length=0;
    }

    int cnt=1;
    int unuse[NUMBER_OF_PROCESSES+1][2];
    unuse[0][0]=0;
    unuse[0][1]=memory-1;
```

```
for(j=0;j<cnt-1;j++){
    if(unuse[j][1]+1==process[i].pos && unuse[j+1][0]-1==process[i].pos+process[i].length){
        unuse[j][1]=unuse[j+1][1];
        for(s=j+1;s<NUMBER_OF_PROCESSES+1;s++){
            unuse[s][0]=unuse[s+1][0];
            unuse[s][1]=unuse[s+1][1];
        }
        cnt--;
        flag=0;
    }
    if(unuse[j][1]+1==process[i].pos && unuse[j+1][0]-1!=process[i].pos+process[i].length){
        unuse[j][1]+=process[i].length;
        flag=0;
    }
    if(unuse[j][1]+1!=process[i].pos && unuse[j+1][0]-1==process[i].pos+process[i].length){
        unuse[j+1][0]-=process[i].length;
        flag=0;
    }
}
if(flag){
    if(unuse[0][0]>process[i].pos){
        if(unuse[0][0]==process[i].pos+process[i].length){
            unuse[0][0]=process[i].pos;
        }
        else{
            for(j=cnt;j>0;j--){
                unuse[j][0]=unuse[j-1][0];
                unuse[j][1]=unuse[j-1][1];
            }
            unuse[0][0]=process[i].pos;
            unuse[0][1]=process[i].pos+process[i].length-1;
            cnt++;
        }
    }
    else if(unuse[cnt][0]<process[i].pos){
            if(unuse[cnt][1]==process[i].pos-1){
                unuse[cnt][1]=process[i].pos+process[i].length-1;
            }
            else{
                unuse[cnt+1][0]=process[i].pos;
                unuse[cnt+1][1]=process[i].pos+process[i].length-1;
                cnt++;
            }
        }
    else if(unuse[0][0]<process[i].pos && process[i].pos<unuse[cnt][0]){
            for(j=0;j<cnt-1;j++){
                if(unuse[j][0]<process[i].pos && unuse[j+1][0]>process[i].pos){
                    for(s=j+1;s<NUMBER_OF_PROCESSES;s++){
                        unuse[s][0]=unuse[s+1][0];
                        unuse[s][1]=unuse[s+1][1];
                    }
```