OS Lab6 Memory Management

1. Basic requirements (60)
   1. You should understand the given code and complete it. So that it can achieve basic memory management goals. More concretely, your program must has the function to allocate a given memory space to a process(15), kill a process and release the memory space the process kept(15) and show the memory usage(10).
   2. You should consider some situations such that your program cannot handle (10). Such as allocate a negative number of memory spaces. Final test will contain this kind of test case.
   3. Additionally, you are required to write at least two kinds of algorithms we talked in class (First fit, best fit and worst fit). Actually, you only need to change the compare function.(10)
2. Lab report(40)
   1. Write the report carefully and concretely. (20)
   2. We will give you a sequence of input operation. You can give a screenshot of your program at the end of lab report. If it works well, you will get the remaining points.(20)
3. Bonus: If you finish any one of the following requirements, you will get full 100 points. (In this occasion, your report should contain your ideas and details about how you write your code.).
   1. You achieve your own buddy system
   2. You achieve your own Segmentation or Paging
4. Things help you to do this lab:
   1. How to make your vim powerful?
      1. Auto complement: <https://www.linuxidc.com/Linux/2017-02/141088.htm>
      2. Supporting Chinese character: <https://www.zhihu.com/question/22363620>
      3. Write more code
   2. More explanation about code:
      1. My idea: Keep a list of free-block, which stores memory blocks we can use. And also keep a list of allocated-block, which stores memory blocks we have allocated. When we allocated a memory block to a process, we also need to store the pid of this process. So when we want to kill process, we can find the correct block.
      2. Two import functions you need to complete:
         1. int allocate\_mem(allocated\_block \*ab) which give a block memory space
         2. int free\_mem(allocated\_block \*ab) which release the memory space for this block

For allocate, you should use your own algorithm to allocate memory space. That means you need to do something on the free-block list.

For free, when we free the memory space, we need to add the block to the free-block list. At this time, you also need to change the list by your own algorithm.

* + 1. Some other things:
       1. typedef pair<int, string> My\_algo; This line define my own type called My\_algo. You can declare a variable of this type. The first component is an integer, the second component is a string. That means:

My\_algo algo;

algo = make\_pair(0, “FirstFit”);

printf(“%d %s\n”, algo.first, algo.second);

After running this code, you will get result: 0 FirstFit

make\_pair is a function, which you don’t need to write by your own.

* + - 1. NULL. When you deal with pointers, please pay attention to NULL. That is, only when a pointer is not NULL, it can has next element.

1. Some Test cases(You need to do screenshots!):
   1. (1, 2048), 5, (3, 1024), 3(1023), 5, (4, 1), 5, (4, 2), 233
   2. (3, 1024), 5, (3, 1), (4, 1), 5, (4, 2), 233
   3. (1, 700), (3, 100), (3, 200), (3, 300), (4, 2), 5, (3, 300), 5, (4, 3), 5, 233