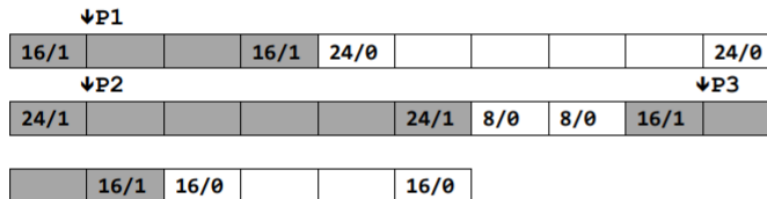


ICS EXE 12

May 27, 2020

1 Memory Allocation

Now we organize the heap as a sequence of **contiguous** allocated and free blocks, as shown below. **Allocated** blocks are shaded, and **free** blocks are blank (each block represents 1 word = 4 bytes). **Headers** and **footers** are labeled with the number of bytes and allocated bit. The allocator maintains **double-word** alignment. You are given the execution sequence of memory allocation operations (**malloc()** or **free()**) from 1 to 6.



- ```
1. P4 = malloc(8)
2. free(P1)
3. P5 = malloc(6)
4. free(P2)
5. P6 = malloc(3)
6. P7 = malloc(2)
```

Please answer the questions below. Assume that **immediate coalescing** strategy and **splitting free blocks** are employed. (NOTE: **DON'T** need consider P1, P2 and P3 when calculating **internal** fragments)

1. Assume **best-fit** algorithm is used to find free blocks. Please draw the **final** status of memory and mark with block size in headers and footers after the operation sequence is executed. Please also calculate the total bytes of the **internal** fragments.
2. Assume **first-fit** algorithm is used to find free blocks. Please draw the **final** status of memory and mark with block size in headers and footers after the operation sequence is executed. Please also calculate the total bytes of the **internal** fragments.
3. Suppose that we use the entire free block instead of plitting when doing allocation and coalesce immediately after freezing. The total size of heap we can use is shown in the picture. Please identify that whether the operation sequence will succeed or not for both **first-fit** and **best-fit** respectively. If yes, please give the total bytes of the **internal** fragments. If no, please point out **which step** causes the failure (out of memory).

## 2 Dynamic Linking

ICSTA wrote two C programs as shown below: `subvec.c` and `dynamic_line.c`. We compile `subvec.c` as a shared library (`linux > gcc -shared -fpic -o libvector.so subvec.c`):

```
1 /* subvec.c */
2 int delcnt;
3 void subvec(int *x, int *y, int *z, int n) {
4 for (int i = 0; i < n; i++) {
5 z[i] = x[i] - y[i];
6 }
7 }
```

```
1 /* dynamic_link.c */
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <dlfcn.h>
5
6 int x[2] = {1,2};
7 int y[2] = {3,4};
8 int z[2];
9 int main(void) {
10 /* we can call subvec() like */
11 /* any other function */
12 subvec(x,y,z,2);
13 printf("z = [%d %d]", z[0], z[1]);
14
15 return 0;
16 }
```

1. Please give two ways that we can link the shared libraries `libvector.so` (NOTE: You can modify the `dynamic_link.c`)
2. Why GOT is needed to relocate functions in `libvector.so`?
3. After the shared libraries `libvector.so` was loaded in memory, please fill in the address of **GOT entry of subvec** before and after first invocation. Suppose that the address of `PLT[0]` is `0x404360`, the address of `PLT` entry of `subvec` is `0x404560`, and the address of `subvec()` is `0x400128`, the value of `GOT[0]` is `0x406670`.  
Before:  
After: