計算機概論與程式設計

LAB 11
2022/12/12
黃奕軒

Lab 11-1 Combine two sorted linked lists

- For details, please refer to the description of Formosa OJ
  - Given two sorted lists
  - Merge these two lists into a one sorted list.

Sample Input #1

```
2 4 39 45 67 80
3 36 78
```

Sample Output #1

```
2 3 4 36 39 45 67 78 80
```

- You should implement this LAB with linked list.

```
struct ListNode {
    int val;
    struct ListNode *next;
 };

struct ListNode* mergeTwoLists(struct ListNode* list1, struct ListNode* list2) {
}

struct ListNode* printList(struct ListNode* list) {
    struct ListNode *p = list;

    while(p) {
        printf("%d ",p->val);
        p = p->next;
    }

    printf("\n");
}
```

Lab 11-2 function pointer: callback function

- Download reference code from E3.
  - There are two TA function in libta.so
    - `ta_register_callback`
    - `ta_run`

- You may have two doubts or not about this reference code.
  - Why can TA's library run the functions that I implemented?
  - Why can a specific function (`ta_run`) redirect to a different function?

```
void example_add(_CUSTOM_PARAMS_T params)
{
    printf("Your answer is %d\n", params.arg1 + params.arg2);
}

void example_mul(_CUSTOM_PARAMS_T params)
{
    printf("Sorry, I don't want to do anything\n");
}
```

```
ta_register_callback("add", example_add);
ta_register_callback("mul", example_mul);
```

```
ta_run("add", m_params);
ta_run("mul", m_params);
```

```
===== Normal example =====
Action 1: Your answer is 3
Action 2: Sorry, I don't want to do anything
```

Lab 11-2 function pointer: callback function

- In fact, you can treat your function like instance variable A (int A)
  - void test(int a) { ... }
    - The "test" function exists like an instance variable in your program
    - “test" is similar to the pointers you've used before.
  - How to declare a variable to describe "test" function?
    - void (*FPTR)(int a)
      - FPTR is your function pointer, so you can assign "test" directly.
        - FPTR = test
      - When you call FPTR(3), it is the same as test(3)
- How to manage when there are multiple test like functions?
  - void test_1(int a)
  - void test_2(int a)
  - void test_N(int a)

```
const struct test pseudo_table[] =
{
    {keyword_1, test_1_FUNC},
    {keyword_2, test_2_FUNC},
        ...
    {keyword_N, test_N_FUNC},
}
```

Organized into structured array

```
for(int i = 0; i < table_len; i++)
{
    if(keyword match)
    {
        call your function pointer like FPTR
    }
}
```

Search callback function from array and execute callback

What should you do of Lab 11-2

- Implement these two function by yourself and rename
  - `ta_register_callback => student_register_callback`
  - `ta_run => student_run`

- Complete “add / sub / mul / div” by
  - “`student_register_callback`” + “`student_run`”
    - `student_run("add", m_params);` 3
    - `student_run("sub", m_params);` -1
    - `student_run(“mul", m_params);` 2
    - `student_run(“div", m_params);` 0.5

```
_CUSTOM_PARAMS_T m_params;
m_params.arg1 = 1;
m_params.arg2 = 2;
```

You should prove to the TAs that the result comes from your callback function

```
void example_add(_CUSTOM_PARAMS_T params)
{
    printf("HAHA my answer is %d\n", params.arg1 + params.arg2);
}
```

Hint: Lab 11-2

- You can implement callback function of “add / sub / mul / div” with TA’s library firstly.
  - `ta_register_callback`
  - `ta_run`
- And then start to implement your core function
  - `student_register_callback`
  - `student_run`

Grading

- Lab11-1 (100%)
  - Finish on Formosa OJ and demo to TAs.
  - https://oj.nctu.edu.tw/
- Bonus: Lab11-2 (15%)
  - Finish on local PC and demo to TAs
- Total: 100%