Data structure Analysis
Project #1: MyLib

CSE4100, AIE4050: System Programming

Professor: Youngjae Kim and Youngmin Yi (PhD)

TA: Kihwan Kim, Seonghoon Ahn
E-Mail: sogang_sp2025@googlegroups.com

목차

- 프로젝트 개요
- 배경지식
  1. List
  2. Hash table
  3. Bitmap
- 자료구조 함수 분석

프로젝트 개요

"일반적인 커널 코드에서 쓰이는 세개의 자료구조 분석 및 구현"

1. Data Structures in MyLib

- List, Hash Table, Bitmap 세개의 자료구조를 분석 및 추가 구현

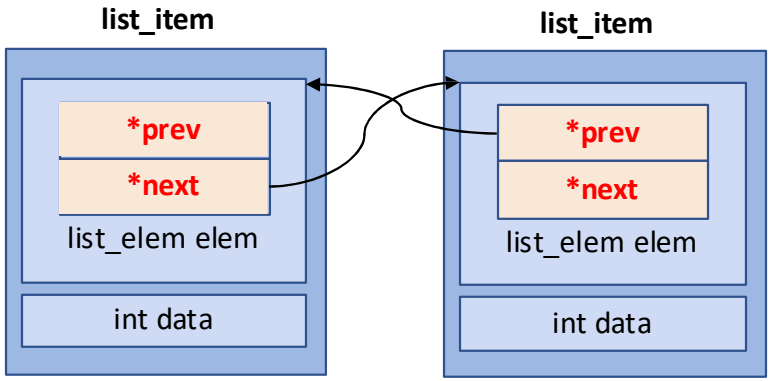
2. Interactive Program

- 명령어를 입력 받아 구현한 자료구조의 functionality를 확인할 수 있는 프로그램 개발

※ 해당 PPT에서 설명하는 내용은 MyLib에서 사용되는 자료구조에 대한 배경지식 설명입니다.
프로젝트 진행과 관련한 자세한 내용은 프로젝트 명세서를 확인 바랍니다.

배경지식 #1: List

- MyLib에서 사용되는 리스트의 종류는 **"doubly linked list"** 이다.
- 이는 일반적인 리스트 구조와 다르기 때문에, PPT에서 설명하는 구조를 이해하시고 코드를 짜는 것을 추천함
- **Mylib의 list entry (list_item)는 pointer (list_elem)와 데이터를 나누어서 저장함**
  (Described in next slide)



Split the pointer and data

※

```
struct list_elem
{
    struct list_elem *prev;
    struct list_elem *next;
}
```

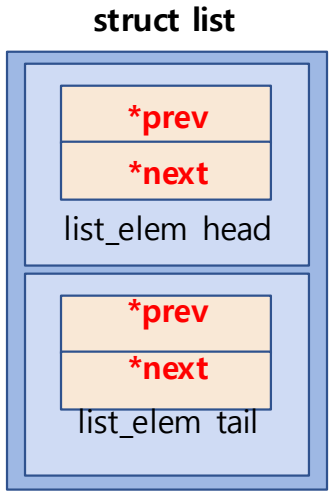
```
struct list_item
{
    struct list_elem elem;
    int data;
    /* Other members you want */
}
```

※ 'struct list_item' is not given in the source code, you need to implement it.

배경지식 #1: List

- **struct list_elem**
  - 리스트와 관련된 데이터를 저장하는 자료구조는 list_elem 자료구조를 포함하고 있어야 함.
  - List를 다루는 모든 함수들은 list_item 자료구조가 아닌 list_elem 자료구조를 통해 수행됨.
  - 현재 skeleton code에선 list_elem 자료구조만 구현되어 있음.
  - 따라서, list_elem과 data를 포함하는 list_item 자료구조를 추가 구현하는 것이 필요함.

- **Struct list**
  - List 자료구조는 list의 head와 tail을 저장하고 있음.
    이 head와 tail 역시 list_elem 자료구조로 이루어짐.



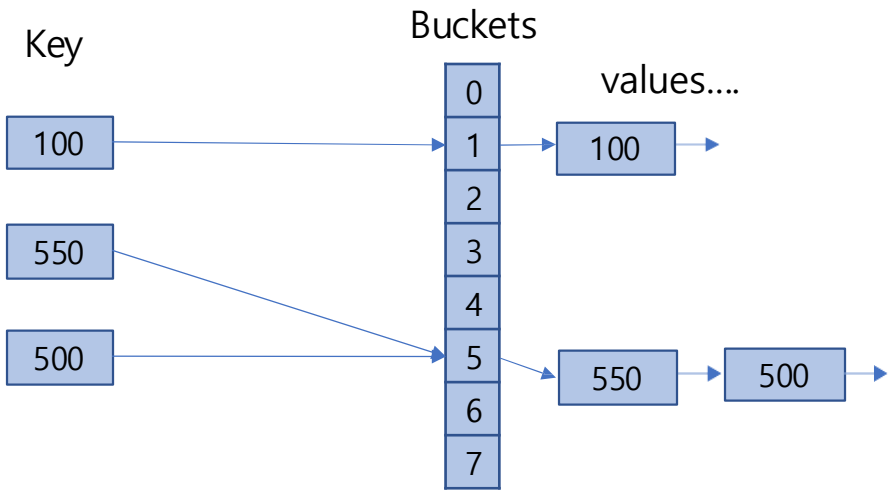
※ 'struct list_item' is not given in the source code, you need to implement it.

배경지식 #2: Hash Table

- A hash table은 keys 와 values 로 연관된 자료구조이다.
- The primary operation is a lookup.
  - Key가 주어지면, key에 해당하는 value를 찾는 것
- 특정 hash function을 이용하여 key를 hash로 바꿔주는 역할을 함

※ Assume that key and value are same.

hash(key)=key/100
=bucket ID

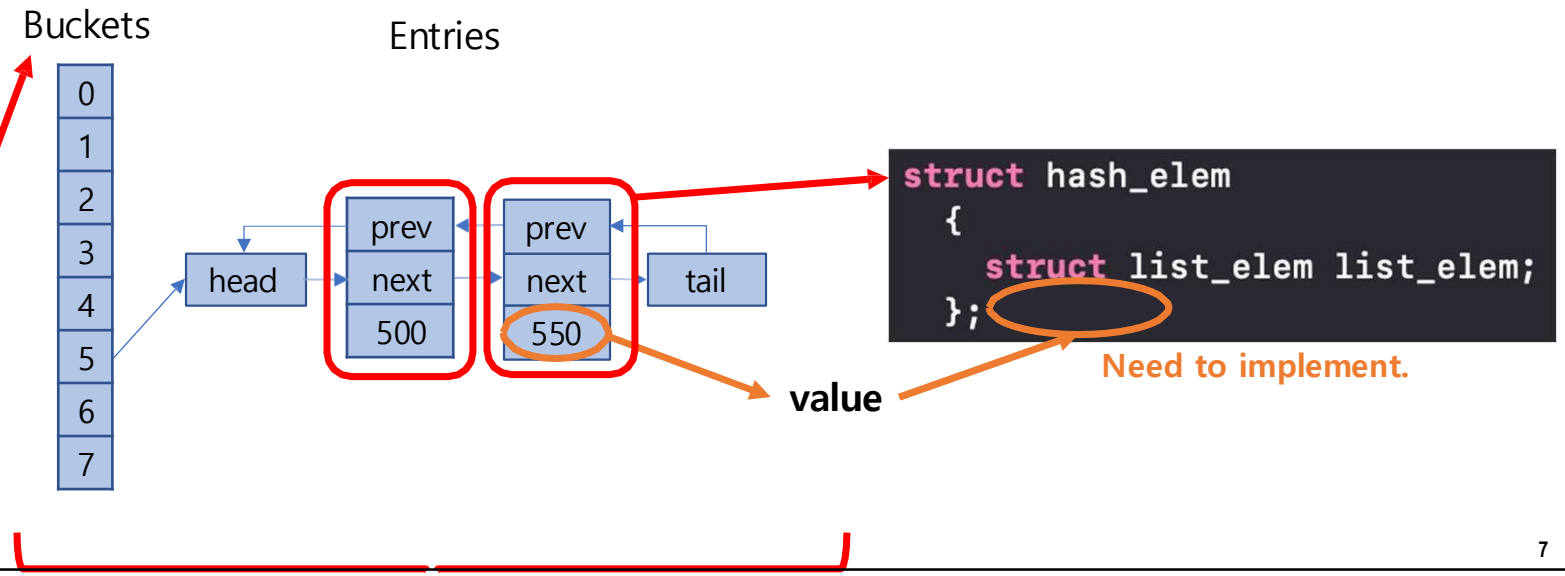


배경지식 #2: Hash Table

- 각 bucket은 list structure로 구성되어 있음(we covered in previous slides.)
- 각 value들은 entry로 구성되어 있음.
  - 이 때, entry는 hash_elem structure로 구성
- In hash_elem, splits list element pointers and data. (same as list)

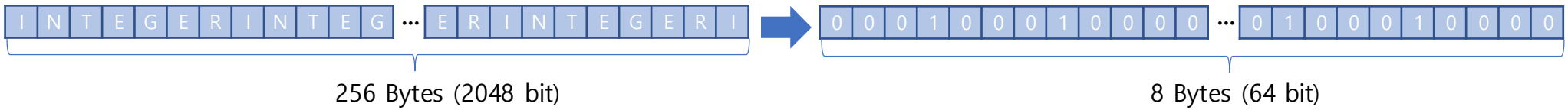
※ **Assume that key and value are same.**

```
struct hash
  {
    size_t elem_cnt;
    size_t bucket_cnt;
    struct list *buckets;
    hash_hash_func *hash;
    hash_less_func *less;
    void *aux;
  };
```



배경지식 #3: Bitmap

- A bit array(or bitmap, in some cases) 는 각각의 Boolean value로 저장되는 array이다.
- A bitmap은 memory space 낭비를 줄일 수 있음 (아래 그림 참고)



- MyLib에서 사용되는 bitmap 자료구조는 usual bitmap이랑 다르게 구성되어 있음
  - 기능은 동일
- Bitmap in MyLib

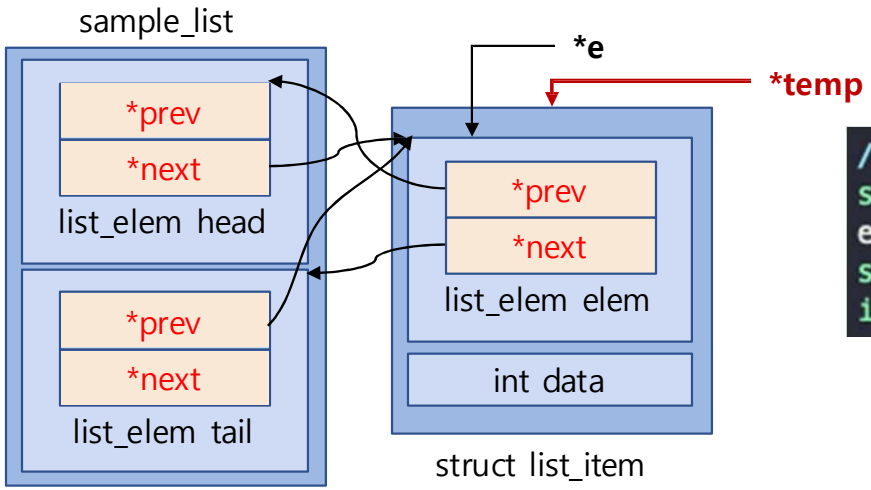
```
typedef unsigned long elem_type;
struct bitmap
{
    size_t bit_cnt;
    elem_type *bits;
};
```

자료구조 분석 #1: List Function in MyLib

- **void list_init(struct list *list)**
  - Initializes LIST as an empty list.
  - It should be executed before an element is inserted in LIST.
- **struct list_elem* list_begin(struct list *list)**
  - Returns the first element of LIST.
  - Usually used to iterate the LIST.
- **struct list_elem* list_next(struct list_elem *elem)**
  - Returns the next element of ELEM.
  - Usually used to iterate the LIST or search ELEM in the LIST.
- **struct list_elem* list_end(struct list *list)**
  - Returns the last ELEM in the LIST.
  - Usually used to iterate the LIST.

자료구조 분석 #1: List Function in MyLib

- **#define list_entry(list_elem, struct, member)**
  - Converts the pointer from list_elem into a pointer of list_item that list_elem is embedded.
  - Usually used to get address of list_item which embeds list_elem.




```
/* Assume that there already exists a list, sample_list */
struct list_elem *e;
e = list_begin (&sample_list);
struct list_item *temp = list_entry(e, struct list_item, elem);
int temp_data = temp->data;
```

By using list_entry, we can get address of list_item

자료구조 분석 #2: Hash Table Function in MyLib

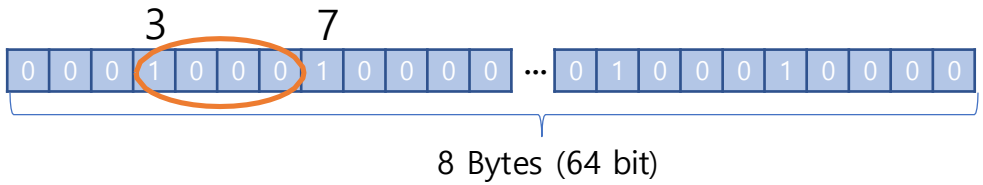
- **void hash_init(struct hash *h, hash_hash_func *hash, hash_less_func *less, void *aux)**
  - Initializes hash table H and sets hash function HASH and comparison function LESS.
  - You can see the example of hash function such as hash_int, hash_bytes, and hash_string.
    (You have to use hash_int function to pass the test.)
  - Comparison function LESS is used to compare two hash elements.

- **void hash_apply(struct hash *h, hash_action_func *action)**
  - You can apply any ACTION function which you made to hast table H.
  - Used for applying specific function to all elements in hash table.
    e.g.) square function.
  - You can learn the usage of it from 'hash_apply.in' and 'hash_apply.out' in tester directory.

- **#define list_elem_to_hash_elem(LIST_ELEM) list_entry(LIST_ELEM, struct hash_elem, list_elem)**
  - Converts pointer to LIST_ELEM into a pointer HASH_ELEM

자료구조 분석 #3: Bitmap Function in MyLib

- **struct bitmap *bitmap_create(size_t bit_cnt)**
  - Initializes a bitmap of BIT_CNT bits and sets all its bits to false.
- **void bitmap_set (struct bitmap *b, size_t idx, bool value)**
  - Atomically sets the bit numbered IDX in B to VALUE.
- **size_t bitmap_count (const struct bitmap *b, size_t start, size_t cnt, bool value)**
  - Returns the number of bits in B between START and START + CNT, exclusive, that are set to VALUE.



bitmap_count(b, 3, 4, 1) == 1