**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20211523 |
| 이름 : | 김종원 |
|  |  |

1. **Additional Implementation**

**// main.c 전역변수/ 구조체**

|  |  |
| --- | --- |
| **Prototype** | typedef struct ds\_entry; |
| **Parameter** | char title[32];  ds\_type type;  void \*ds; |
| **Function** | 리스트, 해시테이블, 비트맵 자료구조를 관리하기 위해 만든 구조체이다. title은 각 자료 구조의 이름인 식별자이고, type은 ds\_type을 사용해 자료구조의 타입을 구분하며, ds는 해당 자료구조의 실제 포인터이다. |

|  |  |
| --- | --- |
| **Prototype** | typedef enum { DS\_LIST, DS\_HASH, DS\_BITMAP } ds\_type; |
| **Function** | 자료구조의 타입을 지정하는 열거형으로, 리스트, 해시테이블, 비트맵 중 하나를 나타낸다. |

|  |  |
| --- | --- |
| **Prototype** | ds\_entry ds\_array[MAX\_DS]; |
| **Function** | 각 자료구조의 이름, 타입, 포인터를 저장하는 배열. |

|  |  |
| --- | --- |
| **Prototype** | int ds\_count = 0; |
| **Function** | 현재 생성된 자료구조의 개수이고, 처음엔 0으로 초기화 되어있다. |

|  |  |
| --- | --- |
| **Prototype** | void process\_command(char \*line); |
| **Parameter** | line: 처리할 명령 줄이다. |
| **Return** | 없음(void 반환함) |
| **Function** | 사용자 입력 문자열을 파싱하여 명령어를 if-else if-else로 조건을 걸어 분기 실행하는 중앙 명령 처리 함수이다. strtok으로 파싱된 명령어를 분석하고 적절한 명령 함수를 불러서 위임한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_~\_~ ( ~ ~ ~); |
| **Parameter** | 현재 받은 명령에 따라 알맞은 파라미터를 받음 |
| **Return** | 없음(void 반환함)(cmd\_ 함수는 모두 반환하는 것이 없다.) |
| **Function** | process\_command에서 불리면, 거기서 받은 char\*로 된 정보를 , 즉, 문자열 string으로 되어있는 정보를 읽을 수 있는 size\_t, list, hash 또는 bitmap 형태로 바꿔서 각 cmd\_~의 ~부분이 나타내는 함수를 호출해서 명령을 수행한다. |

**// Hash 관련 추가/변경 구조체**

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem |
| **Parameter** | struct list\_elem list\_elem;  int real; // 비교를 위해 추가함 |
| **Function** | 비교용 값(real)을 넣고 처리하기 위해 추가했다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_item |
| **Parameter** | struct hash\_elem elem;  int real; |
| **Function** | 해시 테이블 요소에 실제 값을 넣기 위해 추가로 만들었다. |

**// Hash 관련 추가/변경 함수**

|  |  |
| --- | --- |
| **Prototype** | bool really\_less  (const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); |
| **Parameter** | a: 비교할 해시 요소 포인터 a  b: 비교할 해시 요소 포인터 b  aux: 부가 데이터 |
| **Return** | true: a->real < b->real  false: a->real >= b->real |
| **Function** | real 필드를 기준으로 두 해시 요소를 비교하는 함수이다.  사용자 정의 비교 함수로 hash\_init()에서 사용된다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\* create\_Hash(); |
| **Parameter** | 없음 |
| **Return** | 생성된 해시 테이블의 포인터(struct hash\*) |
| **Function** | calloc()을 사용해 메모리를 동적 할당하고, no\_change\_hash\_int() 및 really\_less()를 기반으로 hash\_init()을 호출하여 새로운 해시 테이블을 생성한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned no\_change\_hash\_int  (const struct hash\_elem \*h, void \*aux); |
| **Parameter** | h: 해시 요소  aux: 부가 데이터 |
| **Return** | real 필드를 기반으로 계산된 해시값 |
| **Function** | 해시값을 단순히 real 값으로 변환하기 위해 hash\_int()를 그대로 호출하는 래퍼 함수이다.  create\_Hash()에서 사용된다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* make\_Hash\_elem(int real\_data); |
| **Parameter** | real\_data: 저장할 실제 데이터 값 |
| **Return** | 생성된 해시 요소의 포인터 |
| **Function** | hash\_item 구조체를 동적으로 생성하고 real 필드를 설정하여 hash\_elem 포인터로 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void print\_Hash(struct hash\* h); |
| **Parameter** | h: 출력할 해시 테이블 포인터 |
| **Return** | 없음(void 반환함) |
| **Function** | hash\_apply()를 사용해 모든 요소를 real 값 기준으로 출력하고 줄바꿈한다(다음 명령어 입력을 위함). |

|  |  |
| --- | --- |
| **Prototype** | void print\_Hash\_elem(struct hash\_elem\* h, void\* aux); |
| **Parameter** | h: 출력 대상 해시 요소  aux: 부가 데이터 |
| **Return** | 없음(void 반환함) |
| **Function** | 해시 요소의 real 값을 printf()로 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void Hash\_square(struct hash\_elem\* h, void\* aux); |
| **Parameter** | h: 제곱 처리할 해시 요소  aux: 부가 데이터 |
| **Return** | 없음(void 반환함) |
| **Function** | 해시 요소의 real 값을 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void Hash\_triple(struct hash\_elem\* h, void\* aux); |
| **Parameter** | h: 세제곱 처리할 해시 요소  aux: 부가 데이터 |
| **Return** | 없음(void 반환함) |
| **Function** | 해시 요소의 real 값을 세제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void remove\_Hash\_elem(struct hash\_elem\* h, void\* aux); |
| **Parameter** | h: 제거할 해시 요소  aux: 부가 데이터 |
| **Return** | 없음(void 반환함) |
| **Function** | 해당 해시 요소의 메모리를 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | void apply\_Hash(struct hash\* h, char\* com); |
| **Parameter** | h: 연산 대상 해시 테이블  com: 연산 명령어 문자열 ("square" 또는 "triple") |
| **Return** | 없음 |
| **Function** | com에 따라 Hash\_square 또는 Hash\_triple을 적용하는 래핑 함수이다.  그래서 이후에 main에서 사용된다. |

**// Bitmap 관련 추가/변경 함수**

|  |  |
| --- | --- |
| **Prototype** | void print\_bitmap(struct bitmap \*\*b, char \*b\_name); |
| **Parameter** | b: 비트맵 포인터 배열  b\_name: 출력할 비트맵 이름 (예: "bm0", "bm1") |
| **Return** | 없음 (void 반환함) |
| **Function** | 주어진 이름의 인덱스(b\_name)을 기준으로 bitmap을 콘솔에 출력한다.  존재하지 않는 비트맵일 경우, 아무 작업도 하지 않고 종료한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | a: 교환할 첫 번째 리스트 요소 포인터  b: 교환할 두 번째 리스트 요소 포인터 |
| **Return** | 없음 (void 반환됨) |
| **Function** | 리스트 내 두 list\_elem인 a와 b의 위치를 바꾼다.  인접한 경우와 떨어진 경우를 구분해서, 작업을 수행한다.  a == b인 경우는 수행하지 않는다 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | list: 셔플할 대상 리스트 포인터 |
| **Return** | 없음 (void 반환됨) |
| **Function** | 기존 함수들과 rand() 함수를 사용하여 구현한 Fisher-Yates 알고리즘을 사용해 리스트의 요소들을 무작위로 셔플한다.  리스트의 요소를 배열에 저장한 뒤, 인덱스를 랜덤으로 섞은 후 리스트를 재구성한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i); |
| **Parameter** | i: 해시 연산 대상이 되는 정수 |
| **Return** | unsigned: 변환된 해시 값 |
| **Function** | 정수형 입력값에 대해 더 복잡한 해시 연산을 수행하는 hash\_int()를 대체할 수 있는 함수로 비트 연산과 곱셈을 통해 충돌을 줄이는 해시값을 생성한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_expand(struct bitmap \*b, size\_t new\_bit\_cnt); |
| **Parameter** | b: 확장할 비트맵 포인터  new\_bit\_cnt: 확장 후 비트 수 |
| **Return** | true: 확장 성공  false: 메모리 재할당 실패 |
| **Function** | 기존 비트맵의 크기보다 클 경우, realloc()을 이용하여 메모리를 재할당하고 확장한다.  새로운 영역은 0으로 초기화되고, bit\_cnt도 최신화 해준다.  확장이 실패하면, false를 반환해준다. |