**Pintos Project 0-2: Pintos Data Structure**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20171666 |
| 이름 : | 이예은 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | Int main() |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 명령어를 받아 알맞게 처리해줌 |

|  |  |
| --- | --- |
| **Prototype** | Void listCommand(char[] command) |
| **Parameter** | Command: main함수에서 받은 명령어 |
| **Return** | 없음 |
| **Function** | List에 관련된 명령어들을 처리하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | Void hashCommand(char[] command) |
| **Parameter** | Command: main함수에서 받은 명령어 |
| **Return** | 없음 |
| **Function** | Hash에 관련된 명령어들을 처리하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmapCommand(char[] command) |
| **Parameter** | Command: main함수에서 받은 명령어 |
| **Return** | 없음 |
| **Function** | Bitmap에 관련된 명령어들을 처리하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | Struct list\* findList(char\* name) |
| **Parameter** | Name: 찾으려는 list의 이름 |
| **Return** | 찾은 list를 반환, 없으면 NULL을 반환 |
| **Function** | List의 모음인 testList를 탐색해 이름이 name인 list를 찾음. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem\* searchListIndex(struct list\* list, int index) |
| **Parameter** | List: elem을 찾으려는 대상이 되는 list  Index: 찾으려는 elem의 인덱스 |
| **Return** | 찾은 list\_elem을 반환 |
| **Function** | List의 맨 처음부터 받은 index만큼 이동해 찾으려는 index의 list\_elem을 찾음. |

|  |  |
| --- | --- |
| **Prototype** | Bool lessList(const struct list\_elem\* a, const struct list\_elem\* b, void\* aux) |
| **Parameter** | a와 b는 비교하려는 list\_elem, aux는 보조 변수 |
| **Return** | a의 data가 작으면 true, 크면 false를 반환 |
| **Function** | List\_elem을 list\_entry를 이용해 변환하여 내부 data를 비교 |

|  |  |
| --- | --- |
| **Prototype** | Bool lessHash(const struct hash\_elem\* a, const struct hash\_elem\* b, void\* aux) |
| **Parameter** | a와 b는 비교하려는 hash\_elem, aux는 보조 변수 |
| **Return** | a의 데이터가 작으면 true, 크면 false를 반환 |
| **Function** | Hash\_elem을 hash\_entry를 이용해 변환하여 내부 data를 비교 |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hashFunc(const struct hash\_elem\* e, void\* aux) |
| **Parameter** | e는 해시를 적용하려는 hash\_elem, aux는 보조 변수 |
| **Return** | Hash\_int를 적용한 값 |
| **Function** | Hash\_elm을 hash\_entry를 사용해 변환하여 얻은 데이터를 hash\_int에 적용. |

|  |  |
| --- | --- |
| **Prototype** | Struct hash\* findHash(char\* name) |
| **Parameter** | Name: 찾으려는 hashtable의 이름 |
| **Return** | 찾은 hashtable, 없다면 NULL을 반환 |
| **Function** | Hashtable의 모음인 testHash를 탐색해 이름이 name인 hashtable을 찾음 |

|  |  |
| --- | --- |
| **Prototype** | Void squareHashAction(struct hash\_elem\* e, void\* aux) |
| **Parameter** | e는 계산하려는 hash\_elem, aux는 보조 변수 |
| **Return** | 없음 |
| **Function** | e를 hash\_entry로 변환해 내부 data를 제곱한 값으로 갱신 |

|  |  |
| --- | --- |
| **Prototype** | Void tripleHashAction(struct hash\_elem\* e, void\* aux) |
| **Parameter** | e는 계산하려는 hash\_elem, aux는 보조 변수 |
| **Return** | 없음 |
| **Function** | e를 hash\_entry로 변환해 내부 data를 세제곱한 값으로 갱신 |

|  |  |
| --- | --- |
| **Prototype** | Void destructHashAction(struct hash\_elem\* e, void\* aux) |
| **Parameter** | e는 계산하려는 hash\_elem, aux는 보조 변수 |
| **Return** | 없음 |
| **Function** | e를 hash\_entry로 변환해 해당 메모리를 free |

|  |  |
| --- | --- |
| **Prototype** | Struct bitmap\* findBitmap(char\* name) |
| **Parameter** | Name: 찾으려는 bitmap의 이름 |
| **Return** | 찾은 bitmap을 반환. 없으면 NULL을 반환 |
| **Function** | Bitmap의 모음인 testBitmap을 탐색해 이름이 name인 bitmap을 찾음. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_swap(struct list\_elem\* a, struct list\_elem\* b) |
| **Parameter** | A와 b는 서로 바꾸려는 list\_elem |
| **Return** | 없음 |
| **Function** | A와 b가 연속한 elem이 아니라면, swap함수를 이용해 각자 연결되어 있는 포인터들을 바꿈. 연속하다면, list\_remove와 list\_insert를 이용해 뒤바꿈 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_shuffle(struct list\* list) |
| **Parameter** | List: shuffle하려는 list |
| **Return** | 없음 |
| **Function** | List 크기의 2배 횟수만큼 랜덤한 두 인덱스 값의 elem을 뒤바꿈. |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hash\_int\_2(int i) |
| **Parameter** | 해싱하려는 정수 i |
| **Return** | i의 해싱 값 |
| **Function** | Robert Jenkins’ 32 bit integer hash function을 사용해 해싱이 일어나도록 구현. (참조: http://web.archive.org/web/200712231  73210/http://www.concentric.net/~Ttwang/tech/inthash.htm) |

|  |  |
| --- | --- |
| **Prototype** | Struct bitmap\* bitmap\_expand(struct bitmap\* bitmap, int size) |
| **Parameter** | 비트맵 bitmap, 더 추가하려는 크기 size |
| **Return** | 크기를 늘린 bitmap을 반환 |
| **Function** | 이전 bitmap의 크기 + size 만큼의 크기를 가진 bitmap을 새로 생성 후 이전 bitmap의 값을 새 bitmap에 복사. 이후 새 생성된 것을 반환 |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem) |
| **Parameter** | Elem: 머리인지 확인하려는 list\_elem |
| **Return** | Elem이 head이면 true, 아니면 false를 반환 |
| **Function** | Elem이 head인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem) |
| **Parameter** | 내부 원소인지 확인하려는 list\_elem |
| **Return** | Elem이 interior면 true, 아니면 false를 반환 |
| **Function** | Elem이 내부 원소인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem) |
| **Parameter** | 꼬리인지 확인하려는 list\_elem |
| **Return** | Elem이 tail이면 true, 아니면 false를 반환 |
| **Function** | Elem이 tail인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_init (struct list \*list) |
| **Parameter** | 초기화 하려는 list |
| **Return** | 없음 |
| **Function** | List가 비어 있지 않으면, 새로운 list를 초기화해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_begin (struct list \*list) |
| **Parameter** | 시작지점을 찾으려는 list |
| **Return** | List의 시작지점의 list\_elem을 반환 |
| **Function** | List가 비어 있지 않으면, 해당 list의 시작부분을 반환(head.next) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_next (struct list\_elem \*elem) |
| **Parameter** | 다음 list\_elem을 찾으려는 list\_elem |
| **Return** | 해당 list\_elem의 다음 elem을 반환 |
| **Function** | 해당 list\_elem이 tail이 아닌 경우에 다음 elem을 찾음. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_end (struct list \*list) |
| **Parameter** | 끝 지점을 찾으려는 list |
| **Return** | List의 마지막 list\_elem을 반환 |
| **Function** | List가 비어 있지 않는 경우 list의 tail을 찾음. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_rbegin (struct list \*list) |
| **Parameter** | Reverse beginning을 찾으려는 list |
| **Return** | List의 reverse beginning elem을 반환 |
| **Function** | List가 비어 있지 않으면, list의 마지막 시작점을 찾음(tail.prev) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_prev (struct list\_elem \*elem) |
| **Parameter** | 이전 list\_elem을 찾으려는 list\_elem |
| **Return** | 해당 list\_elem의 이전 elem을 반환 |
| **Function** | 해당 list\_elem이 head가 아닌 경우에 이전 elem을 찾음. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_rend (struct list \*list) |
| **Parameter** | Reverse order의 끝을 찾으려는 list |
| **Return** | Reverse order의 끝 주소 |
| **Function** | List가 비어 있지 않으면, list의 reverse order의 끝이므로 list의 head주소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_head (struct list \*list) |
| **Parameter** | Head를 찾으려는 list |
| **Return** | Head의 주소를 반환 |
| **Function** | List가 비어 있지 않으면, list의 head 부분을 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_tail (struct list \*list) |
| **Parameter** | Tail을 찾으려는 list |
| **Return** | Tail의 주소를 반환 |
| **Function** | List가 비어 있지 않으면, list의 tail 부분을 반환 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | 넣을 자리인 before, 넣을 노드인 elem |
| **Return** | 없음 |
| **Function** | Before의 앞에 elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | 삽입할 위치인 before, 넣을 노드들의 첫 위치인 first, 마지막 위치인 last |
| **Return** | 없음 |
| **Function** | first부터 last까지를 원래 list에서 제거하고 이를 새 list의 before의 앞에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 넣을 노드인 elem, 노드를 넣을 list |
| **Return** | 없음 |
| **Function** | List의 가장 앞에 elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 넣을 노드인 elem, 노드를 넣을 list |
| **Return** | 없음 |
| **Function** | List의 가장 마지막에 elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_remove (struct list\_elem \*elem) |
| **Parameter** | 제거할 노드인 elem |
| **Return** | 제거한 elem의 다음 원소 |
| **Function** | Elem이 내부 원소이면 해당 노드를 삭제. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_pop\_front (struct list \*list) |
| **Parameter** | Pop 하려는 list |
| **Return** | Pop한 노드 |
| **Function** | List의 가장 처음 노드를 제거 후 해당 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_pop\_back (struct list \*list) |
| **Parameter** | pop하려는 list |
| **Return** | Pop한 노드 |
| **Function** | List의 가장 마지막 노드를 제거 후 해당 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_front (struct list \*list) |
| **Parameter** | Front 노드를 찾으려는 list |
| **Return** | Front 노드 |
| **Function** | List의 front 노드를 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_back (struct list \*list) |
| **Parameter** | Back 노드를 찾으려는 list |
| **Return** | Back 노드 |
| **Function** | List의 back 노드를 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | 크기를 찾으려는 list |
| **Return** | List의 크기 |
| **Function** | List의 시작부터 끝까지를 돌며 크기를 계산해 반환 |

|  |  |
| --- | --- |
| **Prototype** | Bool list\_empty (struct list \*list) |
| **Parameter** | 비어 있는지 확인하려는 list |
| **Return** | 비어 있으면 true, 아니면 false |
| **Function** | List의 begin과 end를 비교해 비어 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b) |
| **Parameter** | 뒤바꾸려는 list\_elem |
| **Return** | 없음 |
| **Function** | a와 b를 서로 뒤바꿈. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_reverse (struct list \*list) |
| **Parameter** | 순서를 거꾸로 하려는 list |
| **Return** | 없음 |
| **Function** | List의 처음부터 끝까지 돌며, list를 reverse 순서로 바꿈. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | 시작점 a, 끝점 b, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 정렬되어 있으면 true, 아니면 false |
| **Function** | a부터 b까지 less 함수에 따라 정렬되어 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem \* find\_end\_of\_run (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | 시작점 a, 끝점 b, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 정렬이 안된 노드를 반환 |
| **Function** | a부터 b까지 노드 중, less 함수에 따라 정렬되지 않은 가장 첫 노드를 찾음 |

|  |  |
| --- | --- |
| **Prototype** | inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux) |
| **Parameter** | 첫 range의 시작점 a0, 끝점이자 두번째 rage의 시작점 a1b0, 두번째 range의 끝점인 b1, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 없음 |
| **Function** | A0~a1b0, a1b0~b1 이 두 구간의 노드들을 정렬된 순서에 맞게 merge |

|  |  |
| --- | --- |
| **Prototype** | Void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬하려는 list, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 없음 |
| **Function** | List를 merge sort를 이용해 정렬 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | 넣으려는 list, 넣을 노드인 elem, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 없음 |
| **Function** | Elem 노드를 list가 정렬된 순서에 맞는 위치에 삽입. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | 중복이 있는지 검사할 list, 중복된 노드를 넣을 duplicates, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 없음 |
| **Function** | list에서 중복된 data를 가지는 노드를 하나만 남기고 제거, 이 data가 duplicates에 없다면 duplicates에 삽입. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | max값을 찾으려는 list, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | List의 가장 큰 값을 가지는 노드를 반환 |
| **Function** | List의 처음부터 끝까지 탐색하며, 가장 큰 data를 가지는 노드를 찾음. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | min값을 찾으려는 list, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | List의 가장 작은 값을 가지는 노드를 반환 |
| **Function** | List의 처음부터 끝까지 탐색하며, 가장 작은 data를 가지는 노드를 찾음. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | Bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | 초기화 할 hashtable h, hash 함수 hash, less를 판단해주는 함수 less, 보조 변수 aux |
| **Return** | 문제없으면 true, 아니면 false를 반환 |
| **Function** | h를 초기화 |

|  |  |
| --- | --- |
| **Prototype** | Void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 해시테이블인 h, 해시를 삭제하는 함수 destructor |
| **Return** | 없음 |
| **Function** | h의 모든 bucket안의 list를 삭제하고 bucket을 초기화 |

|  |  |
| --- | --- |
| **Prototype** | Void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 해시테이블인 h, 해시를 삭제하는 함수 destructor |
| **Return** | 없음 |
| **Function** | h의 모든 bucket을 초기화하고 메모리를 free |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 넣을 해시테이블 h, 넣을 노드 new |
| **Return** | 새로 넣었다면 NULL, 아니라면 new를 반환 |
| **Function** | 새 노드의 값이 이미 있는 값이라면 넣지 않고, 새로운 값이라면 삽입. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 넣을 해시테이블 h, 넣을 노드 new |
| **Return** | 새로 넣었다면 NULL, 아니라면 이전 노드를 반환 |
| **Function** | 새 노드의 값이 이미 있는 값이라면 이전 같은 값 노드를 삭제하고 새로 삽입. 새로운 값이라면 바로 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 넣을 해시테이블 h, 찾을 노드 e |
| **Return** | 찾은 노드를 반환 |
| **Function** | Find\_elem을 사용해 해시 테이블에서 e와 같은 값의 노드를 찾음 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 넣을 해시테이블 h, 삭제할 노드 e |
| **Return** | 삭제한 노드를 반환 |
| **Function** | Find\_elem을 사용해 해시 테이블에서 e와 같은 값의 노드를 찾고 해당 노드를 삭제. |

|  |  |
| --- | --- |
| **Prototype** | Void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | 해당 해시테이블 h, 어떤 연산을 해시 테이블에 적용할 지에 대한 함수 action |
| **Return** | 없음 |
| **Function** | 모든 해시 테이블의 노드에 대해 action 함수의 연산을 진행 |

|  |  |
| --- | --- |
| **Prototype** | Void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | 초기화 할 iterator i, 해당 해시테이블 h |
| **Return** | 없음 |
| **Function** | Iterator i를 h의 처음으로 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_next (struct hash\_iterator \*i) |
| **Parameter** | 다음 원소를 얻으려는 iterator i |
| **Return** | 다음 hash\_elem을 반환 |
| **Function** | i->elem의 다음 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | 현재 노드 정보를 갖고 있는 iterator i |
| **Return** | 현재 hash\_elem을 반환 |
| **Function** | 현재 iterator가 갖고 있는 hash\_elem 정보를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | 해시 테이블 h |
| **Return** | h의 크기 |
| **Function** | h에 저장되어 있는 h의 크기를 반환 |

|  |  |
| --- | --- |
| **Prototype** | Bool hash\_empty (struct hash \*h) |
| **Parameter** | 비어 있는지 확인할 해시 테이블 h |
| **Return** | 비어 있으면 true, 아니면 false |
| **Function** | h가 비어 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | 해시의 사이즈 |
| **Return** | 만들어진 byte size의 해시를 반환 |
| **Function** | Byte size의 해시를 만들어 반환 |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hash\_string (const char \*s\_) |
| **Parameter** | 해시로 만들 문자열 |
| **Return** | 만들어진 s의 해시를 반환 |
| **Function** | S 문자열의 해시를 만들어 반환 |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hash\_int (int i) |
| **Parameter** | 정수 i |
| **Return** | i 사이즈만큼의 byte 해시를 반환 |
| **Function** | Hash\_bytes 함수를 이용해 i 사이즈만큼의 byte 해시를 만듦 |

|  |  |
| --- | --- |
| **Prototype** | static struct list \* find\_bucket (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 해시 테이블 h, 찾으려는 노드 e |
| **Return** | 해당 bucket을 반환 |
| **Function** | e가 속해있는 bucket을 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \* find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 해시 테이블 h, 찾으려는 노드가 속해있는 bucket, 찾으려는 노드 e |
| **Return** | 해당 hash\_elem을 반환 |
| **Function** | 해당 bucket에서 e와 같은 data를 가지는 노드를 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h) |
| **Parameter** | 해시 테이블 h |
| **Return** | 없음 |
| **Function** | 해시 테이블이 ideal 하게 돌아갈 수 있도록 bucket을 갱신 |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 해시 테이블 h, 넣으려는 노드가 있는 bucket, 넣으려는 노드 e |
| **Return** | 없음 |
| **Function** | 해당 bucket에 e를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 해시 테이블 h, 삭제하려는 노드 e |
| **Return** | 없음 |
| **Function** | 노드 e를 삭제 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | Bit의 인덱스인 bit\_idx |
| **Return** | Elem의 인덱스 |
| **Function** | Bit\_idx를 ELEM\_BITS로 나눠 elem의 인덱스를 계산 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | Bit의 개수 bit\_cnt |
| **Return** | Bit\_cnt개수의 bit에 필요한 elements의 개수를 반환 |
| **Function** | Bit\_cnt개수의 bit에 필요한 elements의 개수를 계산 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | Bit의 개수 bit\_cnt |
| **Return** | Bit\_cnt개수의 bit에 필요한 byte의 개수를 반환 |
| **Function** | Bit\_cnt개수의 bit에 필요한 byte의 개수를 계산 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | Bit의 인덱스인 bit\_idx |
| **Return** | 생성된 Bit mask를 반환 |
| **Function** | Bit\_idx가 1로 켜진 bit mask를 생성 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | 해당 비트맵 b |
| **Return** | 생성된 bit mask를 반환 |
| **Function** | B의 마지막 비트로 사용된 값은 1, 나머지는 0으로 하여 bit mask를 생성 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | Bitmap의 사이즈인 bit\_cnt |
| **Return** | 초기화 된 bitmap을 반환 |
| **Function** | Bit\_cnt 크기만큼의 bitmap을 만들고 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ) |
| **Parameter** | Bitmap 사이즈인 bit\_cnt, 미리 할당된 block, block의 크기인 block\_size |
| **Return** | 만들어진 bitmap을 반환 |
| **Function** | block 안에 bit\_cnt 크기만큼의 bitmap을 만들고 초기화 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | Bitmap의 크기인 bit\_cnt |
| **Return** | Bit\_cnt만큼의 크기인 bitmap의 전체 크기를 반환 |
| **Function** | Bit\_cnt만큼의 크기인 bitmap의 전체 크기를 계산 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | Bitmap인 b |
| **Return** | 없음 |
| **Function** | 할당되었던 b의 메모리를 free |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | 크기를 얻으려는 bitmap b |
| **Return** | Bitmap의 크기 |
| **Function** | B의 정보인 bit\_cnt를 반환 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | 해당 bitmap b, 설정하려는 bitmap의 인덱스 idx, 설정하려는 값인 value |
| **Return** | 없음 |
| **Function** | b의 idx에 해당하는 값을 value로 세팅 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 설정하려는 bitmap b, 설정하려는 bit의 index인 bit\_idx |
| **Return** | 없음 |
| **Function** | b의 bit\_idx에 해당하는 값을 true로 마킹 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 설정하려는 bitmap b, 설정하려는 bit의 index인 bit\_idx |
| **Return** | 없음 |
| **Function** | b의 bit\_idx에 해당하는 값을 false로 설정 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 설정하려는 bitmap b, 설정하려는 bit의 index인 bit\_idx |
| **Return** | 없음 |
| **Function** | b의 bit\_idx에 해당하는 값이 true면 false로, false면 true로 변환 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | 설정하려는 bitmap b, 설정하려는 bit의 index인 bit\_idx |
| **Return** | Bit\_idx의 값에 따라 true, false 반환 |
| **Function** | b의 bit\_idx에 해당하는 값을 확인 후 반환 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | 해당 bitmap b, 설정하려는 값인 value |
| **Return** | 없음 |
| **Function** | b의 모든 값을 value로 세팅 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt, 설정하려는 값인 value |
| **Return** | 없음 |
| **Function** | b에서 start 부터 cnt개의 값을 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt, 설정하려는 값인 value |
| **Return** | Value인 값의 개수 |
| **Function** | b에서 start부터 cnt개의 값 중 value인 것의 개수를 세서 반환 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt, 설정하려는 값인 value |
| **Return** | Value인 비트가 있으면 true, 아니면 false를 반환 |
| **Function** | b에서 start부터 cnt개의 값 중 value인 것이 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt |
| **Return** | True인 값이 있으면 true, 아니면 false를 반환 |
| **Function** | b에서 start부터 cnt개의 값 중 true인 값이 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt |
| **Return** | True인 값이 있으면 false, 아니면 true를 반환 |
| **Function** | b에서 start부터 cnt개의 값 중 true인 값이 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt |
| **Return** | false인 값이 있으면 false, 아니면 true를 반환 |
| **Function** | b에서 start부터 cnt개의 값 중 false인 값이 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt, 설정하려는 값인 value |
| **Return** | 스캔 후 찾은 인덱스의 값을 반환, 없다면 BITMAP\_ERROR를 반환 |
| **Function** | b에서 start부터 cnt개의 값이 모두 value로 설정되어있는 첫번째 인덱스를 찾음. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 해당 bitmap b, 시작 index인 start, 설정하려는 개수인 cnt, 설정하려는 값인 value |
| **Return** | 스캔 후 찾은 인덱스의 값을 반환, 없다면 BITMAP\_ERROR를 반환 |
| **Function** | b에서 start부터 cnt개의 값이 모두 value로 설정되어있는 첫번째 인덱스를 찾고 해당 비트들을 모두 flip |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*b) |
| **Parameter** | 해당 비트맵 b |
| **Return** | 크기를 반환 |
| **Function** | b를 file에 저장하기 위해 필요한 크기를 계산 후 반환 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_read (struct bitmap \*b, struct file \*file) |
| **Parameter** | 해당 비트맵 b, 읽으려는 파일 file |
| **Return** | 성공적으로 읽으면 true, 아니면 false를 반환 |
| **Function** | file에서 b를 읽음 |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_write (const struct bitmap \*b, struct file \*file) |
| **Parameter** | 해당 비트맵 b, 쓰려는 파일 file |
| **Return** | 성공적으로 쓰면 true, 아니면 false를 반환 |
| **Function** | file에 b를 작성 |

|  |  |
| --- | --- |
| **Prototype** | Void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | 출력하려는 비트맵 b |
| **Return** | 없음 |
| **Function** | Hex\_dump를 사용해 b를 적절히 끊어 출력해줌 |