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

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | int find\_index(char \*s); |
| **Parameter** | index를 찾기 위한 s |
| **Return** | 해당하는 index 리턴 |
| **Function** | main에서 list, hashtable, bitmap의 각 번호들을 구분하기 위해 사용하는 함수이다. 예를 들어 lsit0의 경우 list\_Arr의 0번째 인덱스에 속하므로 0을 리턴한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*list); |
| **Parameter** | 빈 list로 초기화해줄 list 포인터 |
| **Return** | - |
| **Function** | list를 빈 리스트로 초기화시킨다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list elem 첫번째 주소값 |
| **Function** | 리스트의 첫번째 elem 값을 리턴해주는 함수이다. 만약 elem이 없다면 tail 값이 리턴된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*elem); |
| **Parameter** | 다음 elem의 주소 값을 찾을 list\_elem |
| **Return** | elem의 다음 list\_elem의 주소 값 |
| **Function** | elem의 next가 가리키는 list\_elem의 주소값을 리턴해주는 함수이다. 만약 리스트의 마지막 elem라면 tail를 리턴한다. 만약 elem이 tail이라면 정의되지 않는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list의 tail 주소값 |
| **Function** | list의 tail의 주소값을 리턴해주는 함수이다. 정방향으로 iterate할 때 사용되기도 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list의 마지막 elem의 주소값 |
| **Function** | list\_begin과 반대로 마지막 elem 주소값을 리턴한다. 역방향으로 iterate할 때 사용된다. 만약 list가 비어있다면 head를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*elem); |
| **Parameter** | 이전 elem의 주소값을 찾을 list\_elem |
| **Return** | elem의 이전 list\_elem 주소값 |
| **Function** | elem의 이전 list\_elem의 주소값을 리턴한다. 만약 elem이 리스트의 첫번째 element라면 head를 리턴한다. elem이 head인 경우이는 정의되지 않는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list의 head 주소값 |
| **Function** | list의 head의 주소값을 리턴해주는 함수이다. 역방향으로 iterate할 때 list\_rbegin 과 함께 사용된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list의 head 주소값 |
| **Function** | list의 head 의 주소값을 리턴해주는 함수이다. while문 사용 시 list 포인터의 초기 값으로 이용될 수 있다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list); |
| **Parameter** | NULL이 아닌 list |
| **Return** | list의 tail 주소값 |
| **Function** | list의 tail 주소값을 리턴해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem); |
| **Parameter** | list에 삽입할 elem와 삽입될 위치의 이전 elem 주소값 before |
| **Return** | - |
| **Function** | elem을 before의 위치에 삽입한다. ele이 before의 앞에 삽입되어 elem -> before 순서가 된다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before,  struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | 삽입합 list\_elem 구간의 first, last 주소값과 삽입할 위치 begin |
| **Return** | - |
| **Function** | list에서 first 부터 last 까지 구간을 잘라내어 before 전에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem); |
| **Parameter** | 삽입할 elem와 elem이 들어갈 list |
| **Return** | - |
| **Function** | elem을 list의 맨 앞(list\_begin)에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem); |
| **Parameter** | 삽입할 elem와 elem이 들어갈 list |
| **Return** | - |
| **Function** | elem을 list의 맨 뒤(list\_end)에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem); |
| **Parameter** | list에서 삭제할 elem |
| **Return** | 삭제된 elem의 다음 list\_elem |
| **Function** | list에서 elem를 삭제한 후 다음 list\_elem의 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list); |
| **Parameter** | 첫번째 elem를 pop할 list의 주소값 |
| **Return** | list의 첫 번째 elem 주소값 |
| **Function** | list의 첫 번째 elem(list\_front)를 제거한 뒤 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list); |
| **Parameter** | 마지막 elem를 pop할 list의 주소값 |
| **Return** | list의 마지막 elem 주소값 |
| **Function** | list의 마지막 elem(list\_back)를 제거한 뒤 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list); |
| **Parameter** | 비어있지 않은 list의 주소값 |
| **Return** | list의 첫 번째 elem 주소값 |
| **Function** | list의 첫 번째 elem의 주소값을 리턴한다. list가 비어있는 경우는 정의되지 않는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list); |
| **Parameter** | 비어있지 않은 list의 주소값 |
| **Return** | list의 마지막 elem의 주소값 |
| **Function** | list의 마지막 eleml의 주소값을 리턴한다. list가 비어있는 경우는 정의되지 않는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list); |
| **Parameter** | 크기를 알고싶은 list의 주소값 |
| **Return** | list의 elem 개수 |
| **Function** | list의 elem 개수를 세어 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list); |
| **Parameter** | 비어있는지 알고싶은 list의 주소값 |
| **Return** | list가 비어있으면 true, 비어있지 않으면 false |
| **Function** | list가 비어있는지 확인하여 만약 비어있다면 true를 리턴하고, 비어있지 않다면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b); |
| **Parameter** | 포인터를 swap할 list\_elem a와 b |
| **Return** | - |
| **Function** | list\_elem a와 b가 각각 가리키는 것을 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | 역순으로 바꾸고 싶은 list 주소값 |
| **Return** | - |
| **Function** | 만약 list가 비어있지 않다면 list의 elem를 역순으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux); |
| **Parameter** | 정렬할 list와 elem를 비교할 함수 less와 이를 위한 aux |
| **Return** | - |
| **Function** | list의 elem를 less를 이용해 비교하여 정렬하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem,  list\_less\_func \*less, void \*aux); |
| **Parameter** | 삽입할 elem와 list, order를 맞추기 위한 less 함수와 aux |
| **Return** | - |
| **Function** | elem을 less 함수를 이용해 list order에 맞는 위치에 삽입한다 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates,  list\_less\_func \*less, void \*aux); |
| **Parameter** | 중복된 값을 제거할 list와 제거된 값을 저장할 duplicate list, 중복 여부를 판단하기 위한 less와 aux |
| **Return** | - |
| **Function** | list의 elem에 중복된 값이 있다면 제거한 후 제거한 elem을 duplicate에 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list,  list\_less\_func \*less, void \*aux); |
| **Parameter** | list와 대소 비교를 위한 less, aux |
| **Return** | list에서 최대값을 가지는 elem |
| **Function** | list에서 최대값을 가지는 elem를 찾아 리턴해주는 함수이다. 만약 최대값을 가지는 elem이 여러개 있다면 먼저 나온 값을 리턴한다. 만약 list가 비어있다면 tail를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list,  list\_less\_func \*less, void \*aux); |
| **Parameter** | list와 대소 비교를 위한 less, aux |
| **Return** | list에서 최소값을 가지는 elem |
| **Function** | list에서 최소값을 가지는 elem를 찾아 리턴해주는 함수이다. 만약 최소값을 가지는 elem이 여러개 있다면 먼저 나온 값을 리턴한다. 만약 list가 비어있다면 tail를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void dump\_list(struct list \*list); |
| **Parameter** | 출력할 list의 주소값 |
| **Return** | - |
| **Function** | list의 data를 출력해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_find\_nth(struct list\*, int index); |
| **Parameter** | list의 주소값과 list에서 찾을 index 값 |
| **Return** | list의 index 번째 elem |
| **Function** | list에서 index 번째에 위치한 elem를 찾아서 리턴해준다. |

|  |  |
| --- | --- |
| **Prototype** | void delete\_list(struct list \*list); |
| **Parameter** | 삭제할 list의 주소값 |
| **Return** | - |
| **Function** | list의 각 elem를 삭제한 뒤 list에 할당된 메모리를 free해준다. |

|  |  |
| --- | --- |
| **Prototype** | bool less\_func(const struct list\_elem \*a, const struct list\_elem \*b,  void\* aux); |
| **Parameter** | 대소를 비교할 elem a,b와 aux |
| **Return** | 대소 비교한 boolean값 |
| **Function** | a와 b의 data 값을 비교한 뒤 a가 b보다 작으면 true, 크거나 같으면 false를 리턴한다. list\_less\_func에 사용된다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem\*b); |
| **Parameter** | swap할 elem a와 b |
| **Return** | - |
| **Function** | list의 elem a와 b를 swap해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | shuffle할 list |
| **Return** | - |
| **Function** | list의 elem를 무작위로 shuffle해준다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash,  hash\_less\_func \*less, void \*aux) |
| **Parameter** | 초기화할 hash table 주소값 h와 사용할 hash함수, elem의 대소를 비교하기 위한 less함수와 aux |
| **Return** | boolean값 |
| **Function** | hashtable h를 초기화시켜주는 함수이다. 만약 h의 bucket이 제대로 할당이 안되었다면 false를 리턴하고 제대로 됐다면 true를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | clear할 hash table과 이에 사용할 함수 destructor |
| **Return** | - |
| **Function** | destructor를 이용하여 h의 모든 elem를 삭제한 후 다시 bucket을 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | destroy할 hash table과 이에 사용할 destructor |
| **Return** | - |
| **Function** | h의 elem를 모두 clear해주고 bucket도 free한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 삽입할 hash\_elem와 hash table |
| **Return** | elem와 같은 값을 elem이 있다면 해당 elem를 리턴 |
| **Function** | 새로운 elem을 hash table에 삽입하고 NULL 포인터를 리턴한다. 이미 같은 elem이 table에 있다면 그대로 두고 기존의 elem의 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h,  struct hash\_elem \*new); |
| **Parameter** | 대체할 hash\_elem와 hash table |
| **Return** | elem와 같은 값이 이미 있다면 해당 elem 리턴 |
| **Function** | elem와 같은 elem이 있다면 새로운 elem로 대체 후 기존의 elem 주소값을 리턴한다. 같은 값이 없다면 그냥 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | hash table과 찾을 hash\_elem |
| **Return** | 찾은 hash\_elem 주소값 |
| **Function** | 만약 hash\_elem와 같은 elem가 있다면 찾아 리턴해주고, 없다면 NULL을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h,  struct hash\_elem \*e); |
| **Parameter** | 삭제할 hash\_elem와 elem이 들어있는 hash table |
| **Return** | 삭제한 elem 혹은 NULL 포인터 |
| **Function** | e와 같은 elem가 h에 있다면 찾아 삭제해주고 주소값을 리턴한다. 만약 없다면 NULL 포인터를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action); |
| **Parameter** | 함수를 적용할 hash table과 적용할 함수 action |
| **Return** | - |
| **Function** | h의 각 elemnet에 action 함수를 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h); |
| **Parameter** | iterate에 사용할 hash\_iterator와 hash table |
| **Return** | - |
| **Function** | h를 iterate하기 위한 i를 초기화해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i); |
| **Parameter** | hash\_iterator인 i |
| **Return** | 다음 hash\_elem 주소값 |
| **Function** | i를 이용하여 다음 hash elem 값을 리턴해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i); |
| **Parameter** | hash\_iterator인 i |
| **Return** | 현재 hash\_elem 주소값 |
| **Function** | hash table iteration에서 현재 element를 리턴해준다. 만약 table의 끝이라면 NULL값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h); |
| **Parameter** | 크기를 알고싶은 hash table |
| **Return** | h의 크기 |
| **Function** | hash table h의 element 개수를 리턴해준다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h); |
| **Parameter** | 비어있는지 알고자 하는 hash table |
| **Return** | 비어있는지 boolean값 |
| **Function** | h가 비어있다면 true를, 비어있지 않다면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*, size\_t); |
| **Parameter** | buf와 그 size |
| **Return** | unsiged hash 값 |
| **Function** | BUF의 size byte의 hash 값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s); |
| **Parameter** | hash 값으로 바꿀 string |
| **Return** | unsiged hash 값 |
| **Function** | s를 hash 값으로 바꿔 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i); |
| **Parameter** | hash 값으로 바꿀 integer |
| **Return** | unsiged hash 값 |
| **Function** | 정수 i를 hash\_bytes를 이용해 hash값으로 바꿔 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i); |
| **Parameter** | hash 값으로 바꿀 integer |
| **Return** | unsiged hash 값 |
| **Function** | hash\_int와는 다른 방법으로 hash 값을 구하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list \* find\_bucket (struct hash \*, struct hash\_elem \*) ; |
| **Parameter** | hash table과 해당하는 bucket을 찾고자 하는 hash\_elem |
| **Return** | elem이 속한 bucket 리스트의 주소값 |
| **Function** | elem이 속해있는 bucket을 hash에서 찾아서 리턴해준다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \* find\_elem (struct hash \*h,  struct list \*bucket, struct hash\_elem \*e); |
| **Parameter** | 찾을 elem와 hash table과 이에 속해있는 bucket |
| **Return** | 찾은 elem의 주소값 |
| **Function** | hash table의 bucket에 elem와 같은 element가 있는지 찾아서 리턴한다. 만약 없다면 NULL을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | static void rehash(struct hash \*h); |
| **Parameter** | rehash할 hash table |
| **Return** | - |
| **Function** | hash table의 bucket을 바꿔 rehash를 진행한다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h,  struct list \*bucket, struct hash\_elem \*e); |
| **Parameter** | hash table, bucket과 삽입할 elem |
| **Return** | - |
| **Function** | e를 h의 bucket에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | hash table과 삭제할 elem |
| **Return** | - |
| **Function** | elem을 hash table에서 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem\*, const struct hash\_elem\*, void\*); |
| **Parameter** | 대소를 비교할 a, b와 대소비교에 사용할 aux |
| **Return** | 대소 비교한 boolean값 |
| **Function** | a와 b의 data를 비교하여 a가 b보다 더 작다면 ture를, 크거나 같다면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_func(const struct hash\_elem\*, void\*); |
| **Parameter** | hash elem와 aux |
| **Return** | unsinged hash 값 |
| **Function** | e를 hash\_int를 이용해 해쉬 값을 구하여 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_dump(struct hash \*hash); |
| **Parameter** | 출력할 hash table |
| **Return** | - |
| **Function** | hash의 값들을 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void destructor\_func(struct hash\_elem\*, void\*); |
| **Parameter** | destrut할 elem와 aux |
| **Return** | - |
| **Function** | elem를 free해준다. |

|  |  |
| --- | --- |
| **Prototype** | void square\_func(struct hash\_elem\*,void\*); |
| **Parameter** | action 함수를 취할 elem와 aux |
| **Return** | - |
| **Function** | elem 값을 제곱하여 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | void triple\_func(struct hash\_elem\*,void\*); |
| **Parameter** | action 함수를 취할 elem와 aux |
| **Return** | - |
| **Function** | elem 값을 세제곱하여 저장한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | 생성할 bitmap의 크기 bit\_cnt |
| **Return** | 생성한 bitmap 주소값 |
| **Function** | bit\_cnt 크기를 가지는 bitmap을 생성하여 0으로 초기화 한 후 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*, size\_t byte\_cnt); |
| **Parameter** | 할당되어 있는 block과 그 크기, 생성할 bitmap의 크기 |
| **Return** | 생성한 bitmap 주소값 |
| **Function** | 이미 buf에 할당되어 있는 block에 bitmap을 생성하여 0을 초기화 한 후 주소값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt); |
| **Parameter** | bitmap의 크기 |
| **Return** | bitmap의 메모리 사이즈 |
| **Function** | bitmap을 할당하기 위해 필요한 메모리의 크기를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | destroy할 bitmap |
| **Return** | - |
| **Function** | bitmap에 할당된 메모리를 free한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | 크기를 알고자하는 bitmap |
| **Return** | bitmap의 크기 |
| **Function** | bitmap의 크기를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool value); |
| **Parameter** | bitmap과 비트를 설정할 index와 설정할 값 |
| **Return** | - |
| **Function** | bitmap의 idx에 해당하는 값을 value 값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | bitmap과 비트를 1로 바꿀 index |
| **Return** | - |
| **Function** | bitmap의 idx에 해당하는 값을 1(true)로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| **Parameter** | bitmap과 비트를 0로 바꿀 index |
| **Return** | - |
| **Function** | bitmap의 idx에 해당하는 값을 0(false)로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | bitmap과 비트를 바꿀 index |
| **Return** | - |
| **Function** | bitmap의 idx에 해당하는 비트의 값을 toggle한다. 즉 만약 현재 비트가 0이라면 1로 바꾸고, 1이라면 0으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | bitmap과 값을 판단하려는 index |
| **Return** | index에 해당하는 bit 값 |
| **Function** | bitmap의 idx에 해당하는 bit 값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | 값을 setting하려는 bitmap 주소값 |
| **Return** | - |
| **Function** | bitmap의 모든 bit를 value 값으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool value); |
| **Parameter** | 값을 setting하려는 bitmap 주소값, setting 범위 시작과 범위에 해당하는 비트 수와 바꾸려는 value값 |
| **Return** | - |
| **Function** | bitmap의 start부터 start + cnt - 1에 해당하는 비트들을 value 값으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool value); |
| **Parameter** | bitmap과 count하려는 범위의 start와 cnt, 세려는 value |
| **Return** | 범위에 해당하는 value값 개수 |
| **Function** | bitmap의 start부터 cnt 개수 만큼의 범위 중 value에 해당하는 값이 몇개 있는지 세어 개수를 리턴한다. |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool value); |
| **Parameter** | bitmap과 검사 범위 및 value |
| **Return** | 범위 내에 value를 포함하는지 boolean 값 |
| **Function** | bitmap의 start부터 cnt 개수만큼의 범위 내에 value값이 포함되어있는지 확인하여 포함된다면 true, 아니면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | bitmap과 검사 범위 |
| **Return** | 검사한 boolean 값 |
| **Function** | start부터 cnt만큼의 범위 내에 1이 하나라도 있다면 true를, 아니라면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | bitmap과 검사 범위 |
| **Return** | 검사한 boolean 값 |
| **Function** | start부터 cnt만큼의 범위 내에 1이 하나라도 있다면 false를, 아니라면 true를 리턴한다. 즉 범위 내 값이 모두 false라면 true를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | bitmap과 검사 범위 |
| **Return** | 검사한 boolean 값 |
| **Function** | start부터 cnt만큼의 범위 내의 모든 값이 1이라면 true를, 아니라면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool value); |
| **Parameter** | bitmap과 scan할 범위 및 value |
| **Return** | 해당하는 index가 있다면 리턴, 없다면 error 리턴 |
| **Function** | bitmap의 start 이후부터 cnt 개수 만큼 value가 연속되는 구간이 있다면 시작 index를 리턴한다. 만약 연속되는 구간이 없다면 BITMAP\_ERROR를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | bitmap과 scan할 범위 및 value |
| **Return** | 해당하는 index가 있다면 리턴, 없다면 error 리턴 |
| **Function** | bitmap의 start 이후부터 cnt 개수 만큼 value가 연속되는 구간이 있다면 시작 index를 리턴하고 해당 구간의 값들을 toggle한다. 만약 연속되는 구간이 없다면 BITMAP\_ERROR를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*); |
| **Parameter** | 파일에 저장할 bitmap |
| **Return** | bitmap의 메모리 크기 |
| **Function** | bitmap을 파일에 저장하기 위해 필요한 byte 수를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*, struct file \*); |
| **Parameter** | bitmap 주소값과 파일 포인터 |
| **Return** | 제대로 읽어왔는지 boolean 값 |
| **Function** | file로 부터 bitmap값을 읽어온다. 만약 제대로 읽어왔다면 true를, 아니라면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*, struct file \*); |
| **Parameter** | bitmap 주소값과 파일 포인터 |
| **Return** | 제대로 write 했는지 boolean 값 |
| **Function** | bitmap읠 file에 저장한다. 제대로 저장되었다면 true를, 아니라면 false를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*); |
| **Parameter** | dump할 bitmap |
| **Return** | - |
| **Function** | bitmap의 content를 hexadecimal로 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void dump\_bitmap(struct bitmap\*); |
| **Parameter** | 출력할 bitmap |
| **Return** | - |
| **Function** | bitmap의 각 bit의 value들을 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap\*, int size); |
| **Parameter** | expand할 bitmap과 그 크기 |
| **Return** | expand한 bitmap |
| **Function** | bitmap의 크기를 size만큼 더 늘려 리턴한다. 만약 bitmap이 NULL이었다면 NULL을 리턴한다. |