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

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20161559 |
| 이름 : | 강민석 |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | bool compare\_list(const struct list\_elem\*, const struct list\_elem\*, void\*); |
| **Parameter** | const struct list\_elem \*a, const struct list\_elem \*b, void \*aux |
| **Return** | Boolean value |
| **Function** | list의 원소간 대소관계를 비교하는 LESS function이다. 함수 내부에서 list\_entry 매크로를 호출하여서 list\_elem을 원소로 갖는 list\_item을 만들고 list\_item에 있는 data(integer) 값을 비교하여 결과를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool compare\_hash(const struct hash\_elem\*, const struct hash\_elem\*, void\*); |
| **Parameter** | const struct hash\_elem \*a, const struct hash\_elem \*b, void\* aux |
| **Return** | Boolean value |
| **Function** | hash의 원소간 대소관계를 비교하는 LESS function이다. 함수 내부에서 hash\_entry 매크로를 호출하여서 hash\_elem을 원소로 갖는 hash\_item을 만들고 hash\_item에 있는 data(integer) 값을 비교하여 결과를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_func(const struct hash\_elem\*, void \*); |
| **Parameter** | const struct hash\_elem \*e, void \*aux |
| **Return** | unsigned int |
| **Function** | hash\_elem \*e에 대하여 hash\_entry 매크로를 호출하여 hash\_item을 만들고, 갖고 있는 data(integer)에 대하여 hashing 결과를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void square\_func(struct hash\_elem \*, void \*); |
| **Parameter** | struct hash\_elem \*e, void \*aux |
| **Return** | - |
| **Function** | hash action function으로 hash table에 있는 모든 원소의 data 값을 제곱해주는 역할을 한다. |

|  |  |
| --- | --- |
| **Prototype** | void triple\_func(struct hash\_elem\*, void\*); |
| **Parameter** | struct hash\_elem \*e, void \*aux |
| **Return** | - |
| **Function** | hash action function으로 hash table에 있는 모든 원소의 data 값을 세제곱해주는 역할을 한다. |

|  |  |
| --- | --- |
| **Prototype** | void destructor\_func(struct hash\_elem\*, void\*); |
| **Parameter** | struct hash\_elem \*e, void \*aux |
| **Return** | - |
| **Function** | hash action function으로 hash table의 동적으로 할당된 원소를 deallocate 하는 역할을 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem\*, struct list\_elem\*); |
| **Parameter** | struct list\_elem \*a, struct list\_elem \*b |
| **Return** | - |
| **Function** | parameter로 들어온 list내부의 두 원소의 위치를 바꿔주는 역할을 한다. 두 list\_elem에 대해 앞 뒤의 원소를 저장해주고 각각 prev와 next를 수정해주고 a, b의 prev, next를 수정해주는 코드를 구현하였다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list\*); |
| **Parameter** | struct list \*list |
| **Return** | - |
| **Function** | list내부의 원소를 random하게 shuffle하는 함수이다. Knuth shuffle을 구현하였다. 구체적으로 이중 반복문을 통하여 한 원소에 대해 아직 고정이 안된 뒤의 원소들 중 임의로 하나를 골라(rand함수 이용) list\_swap함수를 호출해주어 구현하였다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int); |
| **Parameter** | int i |
| **Return** | unsigned int |
| **Function** | hashing 함수이다. parameter로 들어온 정수에 대해 하나의 비트만큼 shift left를 수행한 후 parameter와 XOR한 값을 반환해주었다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand (struct bitmap\*, int); |
| **Parameter** | struct bitmap\* bitmap, int size |
| **Return** | struct bitmap\* |
| **Function** | parameter로 들어온 bitmap의 크기 + size 만큼의 크기를 갖는 bitmap으로 확장시키는 함수이다. 우선 확장된 크기를 갖는 bitmap을 bitmap\_create함수를 호출하여 생성하고 반복문으로 bitmap\_set을 호출하여 parameter bitmap의 값을 복사해주었다. 이후에 parameter bitmap은 deallocate 해주고 새로 생성된 bitmap을 return 해준다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list\*); |
| **Parameter** | struct list \*list |
| **Return** | - |
| **Function** | list를 초기화 하는 함수이다. parameter list에 대하여 head의 경우 앞을 가리키는 list\_elem이 있으면 안되므로 NULL로, next는 tail로 초기화하고 tail의 경우 prev는 head로, 뒤를 가리키는 list\_elem이 있으면 안되므로 NULL로 초기화 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_begin(struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list에 들어있는 첫 번째 element를 반환한다. (head 다음이 된다.) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_next(struct list\_elem\*); |
| **Parameter** | struct list\_elem\* elem |
| **Return** | struct list\_elem\* |
| **Function** | parameter로 들어온 elem의 다음 list원소를 반환한다. elem이 맨 끝에 있는 경우 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_end(struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list에 들어있는 마지막 element를 반환한다. list의 tail이 된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin(struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | reverse begin, 즉 list에 들어있는 마지막 element를 반환한다. (tail 바로 이전 값이 된다.) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_prev(struct list\_elem \*); |
| **Parameter** | struct list\_elem \*elem |
| **Return** | struct list\_elem\* |
| **Function** | parameter로 들어온 elem의 이전 list원소를 반환한다. elem이 맨 앞에 있는 경우 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_rend(struct list \*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | reverse end, 즉 list에 들어있는 첫 element를 반환한다. list의 head가 된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list\*); |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem\* |
| **Function** | list의 head를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list\*); |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem\* |
| **Function** | list의 tail을 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert(struct list\_elem\*, struct list\_elem\*); |
| **Parameter** | struct list\_elem \*before, struct list\_elem \*elem |
| **Return** | - |
| **Function** | parameter로 들어온 elem을 before자리에 삽입하는 역할을 하는 함수다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice(struct list\_elem\*, struct list\_elem\*, struct list\_elem\*); |
| **Parameter** | struct list\_elem\* before, struct list\_elem \*first, struct list\_elem \*last |
| **Return** | - |
| **Function** | first부터 last elem을 list로부터 삭제한 후, before위치에 insert 하는 역할을 수행하는 함수다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front(struct list\*, struct list\_elem\*); |
| **Parameter** | struct list\* list, struct list\_elem \*elem |
| **Return** | - |
| **Function** | list의 맨 앞 위치에 elem을 insert하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back(struct list\*, struct list\_elem\*); |
| **Parameter** | struct list\* list, struct list\_elem \*elem |
| **Return** | - |
| **Function** | list의 맨 끝 위치에 elem을 insert하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove(struct list\_elem\*); |
| **Parameter** | struct list\_elem\* elem |
| **Return** | struct list\_elem\* |
| **Function** | elem을 list에서 제거하는 원소이다. elem의 next, prev에 있는 element에 대해 pointer를 조정한 뒤 elem-next를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list의 맨 앞에 있는 list\_elem을 remove한 후 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list의 맨 끝에 있는 list\_elem을 remove한 후 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front(struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list의 맨 앞에 있는 list\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back(struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | struct list\_elem\* |
| **Function** | list의 맨 끝에 있는 list\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | struct list\* list |
| **Return** | size\_t |
| **Function** | list 내부에 있는 list\_elem의 원소의 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list\*); |
| **Parameter** | struct list\* list |
| **Return** | Boolean value |
| **Function** | list가 비었는지(list\_elem의 개수가 0인지) 확인한 후 비어있다면 true를, 그렇지 않다면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap(struct list\_elem \*\*, struct list\_elem \*\*); |
| **Parameter** | struct list\_elem \*\*a, struct list\_elem \*\*b |
| **Return** | - |
| **Function** | parameter로 들어온 a, b의 값을 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | struct list\* list |
| **Return** | - |
| **Function** | list 내부에 있는 list\_elem의 순서를 뒤집는다. (ex. 마지막에 있는 원소가 첫 원소가 된다.) |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted(struct list\_elem\*, struct list\_elem \*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\_elem\* a, struct list\_elem\* b, list\_less\_func \*less, void \*aux |
| **Return** | Boolean value |
| **Function** | parameter로 들어온 list 내부의 원소 a부터 b이전까지의 원소가 적합하게 정렬이 되어있는지 확인한다. 정렬이 되어 있다면 true를, 그렇지 않다면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem\* find\_end\_of\_run(struct list\_elem\*, struct list\_elem \*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux |
| **Return** | struct list\_elem\* |
| **Function** | list\_elem에 대응되는 list\_item의 data가 nondecreasing한 element를 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge(struct list\_elem\*, struct list\_elem \*, struct list\_elem\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\_elem\* a0, struct list\_elem\* a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux |
| **Return** | - |
| **Function** | a0 ~ a1b0 까지의 list\_elem과 a1b0 ~ b1까지의 list\_elem을 merge하는 함수이다. 이 때 list\_elem은 정렬된 상태가 되어야 한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort(struct list\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\* list, list\_less\_func \*less, void\* aux |
| **Return** | - |
| **Function** | parameter로 들어온 list를 less function에 의하여 정렬을 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered(struct list\*, struct list\_elem\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\* list, struct list\_elem\* elem, list\_less\_func\* less, void \*aux |
| **Return** | - |
| **Function** | list의 원소를 삽입하지만 정렬된 상태를 유지하게끔 알맞은 자리에 insert한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique(struct list\*, struct list\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\* list, struct list\* duplicates, list\_less\_func\* less, void\* aux |
| **Return** | - |
| **Function** | list를 순회하면서 인접한 원소들이 같으면 그 인접한 원소를 삭제한다. 만약 duplicate가 non-null인 경우, list에 있는 원소들이 duplicate에 첨가된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max(struct list\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\* list, list\_less\_func\* less, void\* aux |
| **Return** | struct list\_elem\* |
| **Function** | list 내부에 있는 원소중 가장 큰 값을 갖는 list\_elem을 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_mix(struct list\*, list\_less\_func\*, void\*); |
| **Parameter** | struct list\* list, list\_less\_func\* less, void\* aux |
| **Return** | struct list\_elem\* |
| **Function** | list 내부에 있는 원소중 가장 작은 값을 갖는 list\_elem을 찾아 반환한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init(struct hash\*, hash\_hash\_func \*, hash\_less\_func\*, void\*); |
| **Parameter** | struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux |
| **Return** | Boolean value |
| **Function** | h(hash table)을 초기화하는 역할을 한다. hashing에 사용되는 함수는 hash를, hash\_elem간 대소비교에 사용되는 함수는 less로 초기화해주고 기본적으로 element\_cnt(element 개수)는 0으로, bucket\_cnt(bucket 개수)는 4로 초기화해준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear(struct hash\*, hash\_action\_func \*); |
| **Parameter** | struct hash \*h, hash\_action\_func \*destructor |
| **Return** | - |
| **Function** | hash table을 비워주는 역할을 하는 함수이다. 적절한 action function을 이용하여 추가적인 역할을 수행할 수 있다. (hash\_elem만 제거한다) |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy(struct hash\*, hash\_action\_func\*); |
| **Parameter** | struct hash\* h, hash\_action\_func \*destructor |
| **Return** | - |
| **Function** | hash table을 삭제하는 역할을 하는 함수이다. hash\_clear와 다른 점은 bucket들 또한 deallocate함으로써, hash table을 말 그대로 삭제하는 기능을 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert(struct hash\*, struct hash\_elem\*); |
| **Parameter** | struct hash\* h, struct hash\_elem \*new |
| **Return** | struct hash\_elem \* |
| **Function** | hash table h에 new를 삽입하는 연산을 진행한다. 만약 같은 data를 갖는 hash\_elem이 table에 존재한다면, 삽입하는 대신 그 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace(struct hash\*, struct hash\_elem\*); |
| **Parameter** | struct hash\* h, struct hash\_elem \*new |
| **Return** | struct hash\_elem \* |
| **Function** | hash table h에 new를 삽입하는 연산을 진행한다. 만약 같은 data를 갖는 hash\_elem이 table에 존재한다면, 삽입하고 기존에 있던 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find(struct hash\*, struct hash\_elem\*); |
| **Parameter** | struct hash\* h, struct hash\_elem \*e |
| **Return** | struct hash\_elem \* |
| **Function** | hash\_elem\* e와 같은 elem이 있으면 그 elem에 대한 포인터를 반환한다. 없을 시 null-pointer를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete(struct hash\*, struct hash\_elem\*); |
| **Parameter** | struct hash\* h, struct hash\_elem \*e |
| **Return** | struct hash\_elem \* |
| **Function** | hash\_elem\* e와 같은 elem이 있으면 그 elem을 삭제한 후 반환한다. 같은 elem이 없다면 null-pointer를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply(struct hash\*, hash\_action\_func\*); |
| **Parameter** | struct hash \*h, hash\_action\_func \*action |
| **Return** | - |
| **Function** | hash table h 내부에 있는 hash\_elem에 대해 parameter로 들어온 hash action function을 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first(struct hash\_iterator \*, struct hash \*); |
| **Parameter** | struct hash\_iterator \*i, struct hash \*h |
| **Return** | - |
| **Function** | hash table을 순회하는 iterator i에 대하여 hash table에 대한 i를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next(struct hash\_iterator \*); |
| **Parameter** | struct hash\_iterator \*i |
| **Return** | struct hash\_elem\* |
| **Function** | 현재 i가 가리키고 있는 다음 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur(struct hash\_iterator \*); |
| **Parameter** | struct hash\_iterator \*i |
| **Return** | struct hash\_elem\* |
| **Function** | 현재 i가 가리키고 있는 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size(struct hash\*); |
| **Parameter** | struct hash \*h |
| **Return** | size\_t |
| **Function** | hash table h에 있는 hash\_elem의 개수를 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty(struct hash\*); |
| **Parameter** | struct hash\*h |
| **Return** | Boolean value |
| **Function** | hash table이 비었는지(hash\_elem의 개수가 0인지) 파악한다. 비어있다면 true를, 그렇지 않다면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes(const void \*, size\_t); |
| **Parameter** | cosnt void \*buf\_, size\_t size |
| **Return** | unsigned int |
| **Function** | size 크기의 byte를 parameter로 받아 이에 대한 hash 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string(const char \*); |
| **Parameter** | const char \*s\_ |
| **Return** | unsigned int |
| **Function** | 문자열 \*s에 대한 hash 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int(int); |
| **Parameter** | int i |
| **Return** | unsigned int |
| **Function** | 정수 i에 대한 hash 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list\* find\_bucket(struct hash \*, struct hash\_elem \*); |
| **Parameter** | struct hash\* h, struct hash\_elem \*e |
| **Return** | struct list\* |
| **Function** | hash\_elem이 속한 bucket을 hash table \*h로부터 탐색하여 대응되는 bucket을 반환한다. 만약 없을 경우 null-pointer를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem\* find\_elem(struct hash\*, struct list\*, struct hash\_elem\*); |
| **Parameter** | struct hash\* h, struct list\* bucket, struct hash\_elem \*e |
| **Return** | struct hash\_elem\* |
| **Function** | e와 같은 hash\_elem이 있는지 hash table \*h로부터 탐색한 후 있을 경우 해당 hash\_elem을 반환하고 그렇지 않을 경우 null-pointer를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit(size\_t); |
| **Parameter** | size\_t x |
| **Return** | size\_t |
| **Function** | 매개 변수 x에 대하여 1로 설정된 가장 lowest-order bit를 0으로 만든 후 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2 (size\_t x); |
| **Parameter** | size\_t x |
| **Return** | size\_t |
| **Function** | 매개변수 x가 2의 거듭제곱 형태이면 true를, 그렇지 않을 경우 false를 반환한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx(size\_t); |
| **Parameter** | size\_t bit\_idx |
| **Return** | size\_t |
| **Function** | bit\_idx값을 갖는 bit를 포험하는 element의 index를 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask(size \_t); |
| **Parameter** | size\_t bit\_idx |
| **Return** | elem\_type |
| **Function** | bit\_idx에 대응되는 부분만 켜진(1) elem\_type을 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt(size\_t); |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 비트들에 필요한 element의 수를 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt(size\_t); |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 비트들에 필요한 byte수를 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask(const struct bitmap\*); |
| **Parameter** | const struct bitmap\* b |
| **Return** | elem\_type |
| **Function** | b의 last element에 사용된 bit을 1로 set하고 나머지는 0으로 set한 bitmask를 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_create(size\_t); |
| **Parameter** | size\_t bit\_cnt |
| **Return** | struct bitmap\* |
| **Function** | bit\_cnt의 크기를 갖는 bitmap을 생성하여 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_create\_in\_buf(size\_t, void\*, size\_t); |
| **Parameter** | size\_t bit\_cnt, void\* block, size\_t block\_size |
| **Return** | struct bitmap\* |
| **Function** | block의 위치에 block\_size의 byte만큼 allocate된 공간에 bit\_cnt 크기의 bitmap을 생성하여 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size(size\_t); |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 크기의 bitmap을 생성하는데 필요한 byte수를 계산하여 return 하는 역할을 하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy(struct bitmap\*); |
| **Parameter** | struct bitmap\* b |
| **Return** | - |
| **Function** | 매개변수로 들어온 bitmap \*b에 대해 동적으로 할당된 부분을 모두 deallocate하고 파괴하는 역할을 한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap\*); |
| **Parameter** | const struct bitmap\* b |
| **Return** | size\_t |
| **Function** | bitmap \*b에 있는 bit수를 계산하여 return 한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set(struct bitmap\*, size\_t, bool); |
| **Parameter** | struct bitmap\* b, size\_t idx, bool value |
| **Return** | - |
| **Function** | bitmap b에서 idx에 위치한 값을 value(true or false)로 바꿔주는 역할을 한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark(struct bitmap\*, size\_t); |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | - |
| **Function** | bitmap b에서 idx에 위치한 값을 true로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset(struct bitmap\*, size\_t); |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | - |
| **Function** | bitmap b에서 idx에 위치한 값을 false로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip(struct bitmap\*, size\_t); |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | - |
| **Function** | bitmap b에서 idx에 위치한 값을 true면 false로, false면 true로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap\*, size\_t); |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | boolean value |
| **Function** | bitmap b에서 idx에 위치한 값(true or false)을 반환한다.\* |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all(struct bitmap\*, bool); |
| **Parameter** | struct bitmap\* b, bool value |
| **Return** | - |
| **Function** | bitmap b에 대하여 모든 bit값을 value로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple(struct bitmap\*, size\_t, size\_t, bool); |
| **Parameter** | struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | - |
| **Function** | start로부터 cnt개의 bit만큼 value로 값을 지정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count(const struct bitmap\*, size\_t, size\_t, bool); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | start로부터 cnt개수의 bit중 value와 같은 값을 갖는 것의 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains(const struct bitmap\*, size\_t, size\_t, bool); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | Boolean value |
| **Function** | start로부터 cnt개의 bit중 value 값으로 설정된 것이 있으면 true를, 그렇지 않은 경우 false를 반환하는 기능을 한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any(const struct bitmap\*, size\_t, size\_t); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | Boolean value |
| **Function** | start로부터 cnt개의 bit중 true로 설정된 것이 있으면 true를, 그렇지 않은 경우 false를 반환하는 기능을 한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none(const struct bitmap\*, size\_t, size\_t); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | Boolean value |
| **Function** | start로부터 cnt개의 bit중 true로 설정된 것이 단 한 개도 없으면 true, 그렇지 않으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all(const struct bitmap\*, size\_t, size\_t); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | Boolean value |
| **Function** | start로부터 cnt개의 bit가 모두 true로 설정된 경우 true를, 그렇지 않은 경우 false를 반환하는 기능을 한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan(const struct bitmap\*, size\_t, size\_t, bool); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | value 값이 처음 나타나는 bit의 index를 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip(const struct bitmap\*, size\_t, size\_t, bool); |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | value 값이 처음 나타나는 bit의 value 값을 반전시키고, 해당 bit의 index를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size(const struct bitmap\*); |
| **Parameter** | const struct bitmap \*b |
| **Return** | size\_t |
| **Function** | bitmap b를 file에 저장하는데 필요한 byte를 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read(struct bitmap\*, struct file\*); |
| **Parameter** | struct bitmap \*b, struct file \*file |
| **Return** | bool |
| **Function** | file로부터 bitmap b를 읽어들인다. 성공하면 true를, 그렇지 않은경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write(const struct bitmap\*, struct file\*); |
| **Parameter** | struct bitmap \*b, struct file \*file |
| **Return** | bool |
| **Function** | file에 bitmap b를 쓴다. 성공하면 true를, 그렇지 않은경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump(const struct bitmap \*); |
| **Parameter** | const struct bitmap\* b |
| **Return** | - |
| **Function** | bitmap debugging을 위한 함수이다. hexadecimal꼴로 console에 출력해줌으로써 bitmap 구조체 안에 저장된 bit값들을 확인할 수 있다. |