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

|  |  |
| --- | --- |
| 과목 명 : | [CSE4070] 운영체제 2반 |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20141515 |
| 이름 : | 김연후 |

**프로젝트 제목 : pintos project 0-2 Pintos Data Structure**

**제출일 : 2019년 10월 5일**

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list\*) |
| **Parameter** | 초기화 시킬 list |
| **Return** | - |
| **Function** | parameter로 들어온 list를 초기화 한다 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin(struct list \*) |
| **Parameter** | begin값을 찾고 싶은 list |
| **Return** | list의 제일 앞 부분 list\_elem\* |
| **Function** | parameter로 들어온 list의 제일 앞 부분의 list\_elem \*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next(struct list\_elem \*) |
| **Parameter** | 다음 list\_elem\* 를 구하고 싶은 list\_elem\* |
| **Return** | 다음 list\_elem\* |
| **Function** | parameter로 들어온 list\_elem\* 의 다음에 위치하는 list\_elem\*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end(struct list \*) |
| **Parameter** | end값을 찾고 싶은 list |
| **Return** | list의 제일 뒷 부분 list\_elem\* |
| **Function** | parameter로 들어온 list의 제일 뒷 부분의 list\_elem \*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin(struct list \*) |
| **Parameter** | reverse begin값을 찾고 싶은 list |
| **Return** | list의 reverse begin list\_elem\* |
| **Function** | parameter로 들어온 list의 reverse begin의 list\_elem \*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev(struct list\_elem \*) |
| **Parameter** | 이전 list\_elem\* 를 구하고 싶은 list\_elem\* |
| **Return** | 이전 list\_elem\* |
| **Function** | parameter로 들어온 list\_elem\* 의 이전에 위치하는 list\_elem\*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend(struct list \*) |
| **Parameter** | reverse end값을 찾고 싶은 list |
| **Return** | list의 reverse end list\_elem\* |
| **Function** | parameter로 들어온 list의 reverse end의 list\_elem \*를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head(struct list \*) |
| **Parameter** | 제일 앞 부분 list\_elem \*를 구하고 싶은 list |
| **Return** | list의 제일 앞 부분 list\_elem \* |
| **Function** | list의 제일 앞 부분 list\_elem \* 를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail(struct list \*) |
| **Parameter** | 제일 뒷 부분 list \*를 구하고 싶은 list |
| **Return** | list의 제일 뒷 부분 list\* |
| **Function** | list의 제일 뒷 부분 list\* 를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*, struct list\_elem \*) |
| **Parameter** | 삽입할 index 위치의 list\_elem \*before, 삽입할 list elem \*elem |
| **Return** | - |
| **Function** | 삽입할 위치에 list elem \*elem을 넣고, 그 자리와 뒷 값들은 뒤로 한칸씩 옮긴다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | 값을 가져올 list\_elem \*before, 시작 list\_elem\*first, 끝 list\_elem\*last |
| **Return** | - |
| **Function** | 존재하는 list의 first ~ last 범위의 값을 가져와 before에 붙인다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*) |
| **Parameter** | 값을 push할 list, 넣어줄 값을 갖는 list\_elem\* |
| **Return** | - |
| **Function** | list의 앞부분에 list\_elem\* 를 push한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*) |
| **Parameter** | 값을 push할 list, 넣어줄 값을 갖는 list\_elem\* |
| **Return** | - |
| **Function** | list의 뒷부분에 list\_elem\* 를 push한다. |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*) |
| **Parameter** | 지울 값의 list\_elem \* |
| **Return** | 지운 값의 list\_elem \* |
| **Function** | list에서 parameter로 받아온 값을 찾아서 지워준다 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*) |
| **Parameter** | 값을 pop할 list |
| **Return** | pop한 값을 갖는 list\_elem \* |
| **Function** | list의 앞부분의 list\_elem\* 를 pop한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*) |
| **Parameter** | 값을 pop할 list |
| **Return** | pop한 값을 갖는 list\_elem \* |
| **Function** | list의 뒷부분의 list\_elem\* 를 pop한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*) |
| **Parameter** | 맨 앞을 구하고 싶은 list |
| **Return** | list의 맨 앞 list\_elem \* |
| **Function** | list의 맨 앞의 list\_elem \* 를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*) |
| **Parameter** | 맨 뒤를 구하고 싶은 list |
| **Return** | list의 맨 뒤 list\_elem \* |
| **Function** | list의 맨 뒤의 list\_elem \* 를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*) |
| **Parameter** | size를 구하고 싶은 list |
| **Return** | list의 size |
| **Function** | list의 size를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*) |
| **Parameter** | empty 여부를 파악하고 싶은 list |
| **Return** | empty면 true, 아니면 false |
| **Function** | list의 empty 여부를 파악한다. |
| **Prototype** | void list\_reverse (struct list \*) |
| **Parameter** | reverse하고 싶은 list |
| **Return** | reverse된 list |
| **Function** | list를 reverse 시킨다. |

|  |  |
| --- | --- |
| **Prototype** | typedef bool list\_less\_func (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | 비교할 elem 2개 list\_elem \*a, list\_elem \*b,. |
| **Return** | a<b 이면 true, 아니면 false |
| **Function** | a와b의 값을 비교하여 a<b이면 true 아니면 false 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | 정렬할 list, elem 값 비교해주는 함수, |
| **Return** | - |
| **Function** | list를 list\_less\_func를 이용하여 정렬함 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*,list\_less\_func \*, void \*aux) |
| **Parameter** | 정렬하며 삽입할 list, 삽입할 list\_elem \*, elem 값 비교 함수 |
| **Return** | - |
| **Function** | 정렬된 list에 elem을 정렬 되도록 삽입함 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux) |
| **Parameter** | 사용할 list, 삭제하는 elem을 저장할 list, elem값 비교함수 |
| **Return** | - |
| **Function** | list에서 중복된 elem들을 삭제하고, 삭제된 elem은 duplicates에 저장함 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | 최댓값을 구할 list, elem값 비교함수 |
| **Return** | list에서의 최댓값을 갖는 list\_elem \* |
| **Function** | list에서 최댓값을 구함 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | 최솟값을 구할 list, elem값 비교함수 |
| **Return** | list에서의 최솟값을 갖는 list\_elem \* |
| **Function** | list에서 최솟값을 구함 |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 서로 바꿔줄 list\_elem \*a, list\_elem \*b |
| **Return** | - |
| **Function** | a와 b를 서로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | shuffle할 list |
| **Return** | - |
| **Function** | list를 무작위로 섞는다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | typedef unsigned hash\_hash\_func (const struct hash\_elem \*e, void \*aux) |
| **Parameter** | 해쉬값 갖는 hash\_elem\* e |
| **Return** | hash\_elem 에 대한 hash 값을 반환함 |
| **Function** | hash elem의 hash값을 구해줌 |

|  |  |
| --- | --- |
| **Prototype** | typedef bool hash\_less\_func (const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | 값을 비교할 hash\_elem \*a, hash\_elem \*b |
| **Return** | a<b이면 true 아니면 false |
| **Function** | a<b이면 true 아니면 false를 반환함 |

|  |  |
| --- | --- |
| **Prototype** | typedef void hash\_action\_func (struct hash\_elem \*e, void \*aux) |
| **Parameter** | 이용할 hash\_elem \*e |
| **Return** | - |
| **Function** | hash\_elem을 이용하여 각각 정해진 action을 취함 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux) |
| **Parameter** | 초기화할 hashtable, hash값 구해주는 함수, hash값 비교함수 |
| **Return** | 초기화 여부 (true/false) 반환 |
| **Function** | hashtable을 초기화 한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*) |
| **Parameter** | 사용할 hash, hash\_action\_func 함수 |
| **Return** | - |
| **Function** | hash의 모든 elem을 지움 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*) |
| **Parameter** | 사용할 hash, hash\_action\_func 함수 |
| **Return** | - |
| **Function** | hash 자체를 free 해버려서 지워버린다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*) |
| **Parameter** | 사용할 hash, 추가할 elem |
| **Return** | 추가 된 경우 NULL, 이미 같은 것이 있는 경우 elem 그대로 반환 |
| **Function** | hash에 elem을 추가해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*) |
| **Parameter** | 사용할 hash, 추가할 elem |
| **Return** | 대체하고 나서 hash에서 나온 오래된 값의 elem 반환 |
| **Function** | hash에 elem으로 대체해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*) |
| **Parameter** | 사용할 hash, 찾을 elem |
| **Return** | 찾을 elem \* |
| **Function** | hash에서 elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*) |
| **Parameter** | 사용할 hash, 삭제할 elem |
| **Return** | 삭제 된 elem \* |
| **Function** | hash에서 elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*) |
| **Parameter** | 사용할 hash, hash\_action\_func |
| **Return** | - |
| **Function** | hash에 hash\_action\_func의 action을 수행함 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*, struct hash \*) |
| **Parameter** | hash의 반복자, 사용할 hash |
| **Return** | - |
| **Function** | hash에 사용할 hash의 반복자의 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*) |
| **Parameter** | hash의 반복자 |
| **Return** | 다음 elem \* |
| **Function** | 다음 elem \*를 구함 |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*) |
| **Parameter** | hash의 반복자 |
| **Return** | 현재의 elem \* |
| **Function** | hash의 반복자에서 현재의 elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*) |
| **Parameter** | 사용할 hash |
| **Return** | hash 크기 |
| **Function** | hash 크기를 구함 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*) |
| **Parameter** | 사용할 hash |
| **Return** | hash의 empty 여부 (true/false) |
| **Function** | hash의 empty 여부 파악 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*, size\_t) |
| **Parameter** | 버퍼, 크기 |
| **Return** | hash |
| **Function** | hash\_bytes 값을 구함 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*) |
| **Parameter** | 문자열 |
| **Return** | hash |
| **Function** | hash\_string 값을 구함 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int) |
| **Parameter** | 정수 |
| **Return** | 정수의 hash 값 |
| **Function** | 정수의 hash\_bytes 값을 구함 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int) |
| **Parameter** | 정수 |
| **Return** | 정수의 hash 값 (새로운 규칙) |
| **Function** | 정수의 hash\_bytes 값을 구함 (새로운 규칙 : 각 자리수의 합) |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | bit 크기 값 |
| **Return** | 만든 bitmap |
| **Function** | bitmap을 bit\_cnt 크기만큼 초기화 함 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*, size\_t byte\_cnt) |
| **Parameter** | bit 크기 값, byte 크기 값 |
| **Return** | 만든 bitmap |
| **Function** | 최소 block\_size 이상으로 미리 block에 allocated된 bitmap을 만듦 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | 확장할 bitmap과 size |
| **Return** | 성공 시 확장된 bitmap 실패시 NULL |
| **Function** | bitmap을 size만큼 확장함 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | bitmap 크기 |
| **Return** | byte size |
| **Function** | bitmap을 수용 가능한 byte 사이즈를 구함 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*) |
| **Parameter** | 삭제할 bitmap |
| **Return** | - |
| **Function** | bitmap을 free 시켜줌 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*) |
| **Parameter** | 사용할 bitmap |
| **Return** | bitmap 크기 |
| **Function** | bitmap 크기를 구해줌 |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool value) |
| **Parameter** | bitmap, index, 값 |
| **Return** | - |
| **Function** | bitmap의 idx 위치에 value를 넣음 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx) |
| **Parameter** | bitmap, index 값 |
| **Return** | - |
| **Function** | bitmap의 idx 위치에 true 넣음 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx) |
| **Parameter** | bitmap, index 값 |
| **Return** | - |
| **Function** | bitmap의 idx 위치에 false 넣음 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx) |
| **Parameter** | bitmap, index 값 |
| **Return** | - |
| **Function** | bitmap의 idx 위치에 true면 false, false면 true 넣음 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx) |
| **Parameter** | bitmap, index 값 |
| **Return** | bitmap의 idx 위치의 value - |
| **Function** | bitmap의 idx 위치의 value 반환함 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool value) |
| **Parameter** | bitmap, value |
| **Return** | - |
| **Function** | bitmap의 모든 bit들을 value로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap, 시작위치, 개수, 값 |
| **Return** | - |
| **Function** | bitmap의 start부터 cnt만큼 value로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap, 시작위치, 개수, 값 |
| **Return** | value를 갖는 개수 반환 |
| **Function** | bitmap에서 시작위치부터 개수만큼의 범위에서 value값을 갖는 개수 구함 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap, 시작위치, 개수, 값 |
| **Return** | value를 가지면 true 아니면 false |
| **Function** | bitmap에서 시작위치부터 개수만큼의 범위에서 value값을 갖는 여부 파악 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt) |
| **Parameter** | bitmap, 시작위치, 개수 |
| **Return** | true가 있으면 true 아님 false |
| **Function** | bitmap에서 시작위치부터 개수만큼의 범위에서 true값을 갖는 여부 파악 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt) |
| **Parameter** | bitmap, 시작위치, 개수 |
| **Return** | true가 있으면 false 아님 true |
| **Function** | bitmap에서 시작위치부터 개수만큼의 범위에서 false값을 갖는 여부 파악 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt) |
| **Parameter** | bitmap, 시작위치, 개수 |
| **Return** | 모두 true면 true 아님 false |
| **Function** | bitmap에서 시작위치부터 개수만큼의 범위에서 모두 true인지 파악 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool) |
| **Parameter** | bitmap, 시작위치, 개수, 값 |
| **Return** | value 갖는 그룹의 시작 index값, 없으면 BITMAP\_ERROR |
| **Function** | value 값을 갖는 그룹의 첫번째 index 값을 구함 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool) |
| **Parameter** | bitmap, 시작위치, 개수, 값 |
| **Return** | value 갖는 그룹의 시작 index값, 없으면 BITMAP\_ERROR |
| **Function** | value 값을 갖는 그룹을 찾아 전부 반대의 값으로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*) |
| **Parameter** | bitmap |
| **Return** | byte의 크기 |
| **Function** | bitmap을 file에 저장하기 위해 필요한 byte의 크기 값 구함 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*, struct file \*) |
| **Parameter** | bitmap, 읽어올 파일 |
| **Return** | 읽는데 성공 시 true, 아니면 false |
| **Function** | bitmap 을 읽어옴 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*, struct file \*) |
| **Parameter** | bitmap, 쓸 파일 |
| **Return** | 쓰는데 성공 시 true, 아니면 false |
| **Function** | bitmap 을 file에 씀 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*) |
| **Parameter** | 사용할 bitmap |
| **Return** | - |
| **Function** | bitmap의 값을 16진수로 출력함 |