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

|  |  |
| --- | --- |
| 과목 명 : | [CSE4070] 운영체제 1반 |
| 담당 교수 : | 박성용 교수님 |
| 학번 : | 20171601 |
| 이름 : | 강주형 |

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

**제출일 : 2019년 10월 6일**

1. **List**
2. **void list\_init (struct list \*);**

parameter로 넘겨받은 list를 empty list로 만들어 준다. (list 새로 만들 때도 이 함수 사용)

1. **struct list\_elem \*list\_begin (struct list \*);**

parameter로 넘겨받은 list의 가장 첫 번째 element(list\_elem) 주소를 반환해준다.

1. **struct list\_elem \*list\_next (struct list\_elem \*);**

parameter로 넘겨받은 list\_elem의 다음(paramter의 next가 가리키는) element의 주소를 반환해준다.

1. **struct list\_elem \*list\_end (struct list \*);**

parameter로 넘겨받은 list의 가장 마지막 element(list\_elem) 주소를 반환해준다.

1. **struct list\_elem \*list\_rbegin (struct list \*);**

parameter로 넘겨받은 list의 끝에서 첫 번째 element 주소를 반환한다. (list를 reverse order로 돌아볼 때 사용)

1. **struct list\_elem \*list\_prev (struct list\_elem \*);**

parameter로 넘겨받은 list\_elem의 prev가 가리키는 element의 주소를 반환한다.

1. **struct list\_elem \*list\_rend (struct list \*);**

parameter로 넘겨받은 list의 head를 반환한다.

1. **struct list\_elem \*list\_head (struct list \*);**

list\_rend와 마찬가지로 parameter로 넘겨받은 list의 head를 반환한다.

1. **struct list\_elem \*list\_tail (struct list \*);**

parameter로 넘겨받은 list의 tail을 반환한다.

**10 . void list\_insert (struct list\_elem \* before, struct list\_elem \*);**

before의 바로 이전(prev)에 새로운 list\_elem을 넣어주는 함수이다.이 함수 수행 후, 기존의 list는 “ ~ - new\_elem – before ~” 형태로 바뀐다.

1. **void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct l ist\_elem \*last);**

first~last까지의 list\_elem들을before 바로 전에(prev)끼워넣는 함수이다. 이 함수 수행 후, 기존의 list는 “~ before “ 에서 “~ first-last before “ 형태로 바뀐다.

1. **void list\_push\_front (struct list \*, struct list\_elem \*);**

parameter로 받은 list의 맨 앞에(begin)list\_elem을 추가한다.

1. **void list\_push\_back (struct list \*, struct list\_elem \*);**

parameter로 받은 list의 맨 뒤에(end)list\_elem을 추가한다.

1. **struct list\_elem \*list\_remove (struct list\_elem \*);**

parameter로 받은 list\_elem을 리스트에서 제외시키고, 해당 elem을 다시 반환한다. (parameterlist\_elem앞 뒤의 원소들을, 서로를 가리키게 수정해준다. )

1. **struct list\_elem \*list\_pop\_front (struct list \*);**

parameter로 받은 list의 가장 첫번째(begin)list\_elem을 리스트에서 제외시키고 그 list\_elem을 반환한다.

1. **struct list\_elem \*list\_pop\_back (struct list \*);**

parameter로 받은 list의 가장 마지막(end) list\_elem을 리스트에서 제외시키고 그 list\_elem을 반환한다.

1. **struct list\_elem \*list\_front (struct list \*);**

list의 가장 첫번째(begin)list\_elem을 반환한다.

1. **struct list\_elem \*list\_back (struct list \*);**

list의 가장 마지막(end)list\_elem을 반환한다.

1. **size\_tlist\_size (struct list \*);**

list의 list\_elem개수를 반환한다.

1. **bool list\_empty (struct list \*);**

list가 비어 있는 경우 true, 그렇지 않은 경우에는 false를 반환한다.

1. **void list\_reverse (struct list \*);**

list의 연결 순서를 거꾸로 만들어주는 함수이다. A-B-C-D로 연결되어 있던 list의 경우, D-C-B-A로 만들어준다.

1. **Static bool is\_sorted(struct list\_elem\*a, struct list\_elem\*b, list\_less\_func \* less, void \*aux);**

list\_elem a부터 b까지 모든 elem들이 less function 대로 정렬이 되어 있는 경우에는 true, 그렇지 않은 경우에는 false를 반환하는 함수이다.

( a, a1,a2,a3,b 이런 상태였다면, a <a1<a2<a3<b 를 만족하면 true가 반환된다)

1. **void list\_sort (struct list \*,list\_less\_func \*, void \*aux);**

less function을 바탕으로 list를 정렬시켜주는 함수이다.(오름차순으로 정렬).

1. **void list\_insert\_ordered (struct list \*, struct list\_elem \*,list\_less\_func \*, void \*aux)**

Parameter로 받은 list\_elem을 less function 기준으로list에 정렬하여 삽입하는 함수이다.

1. **void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux);**

list에서 값이 같은 element가 있으면 하나를 삭제하고, duplicates가 null이 아니면 그 element를 duplicates의 맨 뒤에 추가한다.

1. **struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux);**

list에서 less function 기준으로 제일 큰 element를 반환해준다.

1. **struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux);**

list에서 less function 기준으로 제일 작은 element를 반환해준다.

1. **void list\_swap(struct list\_elem \*, struct list\_elem \*);**

list\_elem두개의 위치를 서로 바꿔주는 함수이다.List\_elema,b의 prev와 next를 수정해주고,a->next/a\_>prev/b->next/b->prev도 알맞게 수정해준다.

1. **void list\_shuffle(struct list\*list);**

입력받은list의 원소들을 무작위로 섞어주는 함수이다.

List의 모든 원소들을 random idx와 swap 해주는 방식으로 섞어주었다.

1. **Hash Table**
2. **bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux);**

parameter로 받은 hashfunction을 사용하여 hash 값을 얻어온 뒤, hash less function을 바탕으로 hashtable을 초기화해주는 함수이다.

1. **void hash\_clear (struct hash \*, hash\_action\_func \*);**

hashtable의 모든 원소들에 parameter로 받은 action function = destructor 함수를 적용해준다. (destructor가 NULL이 아닌경우에)

1. **void hash\_destroy (struct hash \*, hash\_action\_func \*);**

destructor가 null이 아니면 hash\_clear함수를 호출한뒤, hash의 buckerts을 free해준다.

1. **struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*);**

parameter로 넘겨받은 hash\_elem을 hash에 넣어주는 함수이다. 만약 기존 hash에 hash\_elem가 가지는 값과 동일한 값을 가지는 element가 있었으면 그 hash\_elem을 반환하고, 그렇지 않으면 NULL을 반환해준다.

1. **struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*);**

parameter로 넘겨받은 hash\_elem을 hash에 넣어주는 함수이다. 하지만 기존 hash에 hash\_elem가 가지는 값과 동일한 값을 가지는 element가 있었으면 기존의 element를 삭제한 뒤, 해당 element를 연결하고이전 hash\_elem을 반환한다.

1. **struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*);**

parameter로 넘겨받은 hash\_elem과 동일한 값을 가지는 element가 hashtable에 있으면 그 element를 반환하고, 그렇지 않으면 NULL을 반환하는 함수이다.

1. **struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*);**

parameter로 넘겨받은 hash\_elem을 hash에서 삭제해주는 함수이다.

1. **void hash\_apply (struct hash \*, hash\_action\_func \*);**

hashtable의 모든 원소들에 parameter로 받은 action function을 적용시켜주는 함수이다. Dumpdata hash를 할 때 사용한 함수이다.

1. **void hash\_first (struct hash\_iterator \*, struct hash \*);**

parameter로 넘겨받은 hash\_iterator가 hash를 가리키도록 해주는 함수이다.

1. **struct hash\_elem \*hash\_next (struct hash\_iterator \*);**

parameter로 넘겨받은 hash\_iterator가 next로 가리키고 있는 element를 반환해주는 함수이다.

1. **struct hash\_elem \*hash\_cur (struct hash\_iterator \*);**

parameter로 넘겨받은 hash\_iterator가현재 가리키고 있는 element를 반환해준다. 만약 hashtable의 마지막을 가리키고 있는 경우에는 NULL을 반환한다.

1. **size\_thash\_size (struct hash \*);**

hash안의element 개수를 반환해준다.

1. **bool hash\_empty (struct hash \*);**

hash안에 아무런 element도 없는 경우 true, 그렇지 않은 경우에는 false를 반환한다.

1. **unsigned hash\_bytes (const void \*, size\_t);**

buf\_ 안의 hash가 차지하는 byte 수를 반환하는 함수이다.

1. **unsigned hash\_string (const char \*);**

parameter로 넘겨받은 문자열의 hash 값을 반환하는 함수이다 .

1. **unsigned hash\_int (int);**

parameter로 넘겨받은 integer의 hash 값을 반환하는 함수이다 .

1. **unsigned hash\_int\_2(int );**

입력 받은 integer를 hash\_byte함수를 통해 hash 값을 만들어 반환한다.

1. **Bitmap**
2. **struct bitmap \*bitmap\_create (size\_tbit\_cnt);**

bit\_cnt개의 bit을 가지는 새로운 bitmap을 만들어서 반환해주는 함수이다.

1. **struct bitmap \*bitmap\_create\_in\_buf (size\_tbit\_cnt, void \*, size\_tbyte\_cnt) ;**

BLOCK에 미리 할당된 byte에 bit\_cnt개의 bit를 가진 새로운 bitmap을 생성하고, 이를 반환한다.

1. **size\_tbitmap\_buf\_size (size\_tbit\_cnt);**

bit\_cnt를 가지는 bitmap을 담기 위해 필요한 byte 수를 반환한다.

1. **void bitmap\_destroy (struct bitmap \*);**

입력 받은 bitmap을 없애 주는 함수이다. ( 모두 free 시킴 )

1. **size\_tbitmap\_size (const struct bitmap \*);**

parameter로 받은 bitmap의 사이즈를 반환하는 함수이다.

1. **void bitmap\_set (struct bitmap \*, size\_tidx, bool);**

parameter로 받은 bitmap의 idx번째 bit를 value(0/1)로 설정해주는 함수이다.

1. **void bitmap\_mark (struct bitmap \*, size\_tidx);**

parameter로 받은 bitmap의 idx번째 bit를 true(1)로 설정해주는 함수이다.

1. **void bitmap\_reset (struct bitmap \*, size\_tidx);**

parameter로 받은 bitmap의 idx번째 bit를 false(0)로 설정해주는 함수이다.

1. **void bitmap\_flip (struct bitmap \*, size\_tidx);**

parameter로 받은 bitmap의 idx번째 bit를 기존의 값과 반대로 설정해주는 함수이다.

**10.bool bitmap\_test (const struct bitmap \*, size\_tidx);**

parameter로 받은 bitmap의 idx번째 bit값을 반환해주는 함수이다. 이 함수를 통해 dumpdata bitmap 기능을 수행할 수 있다.

1. **void bitmap\_set\_all (struct bitmap \*, bool);**

parameter로 받은 bitmap의 모든 bit를 value로 설정해주는 함수이다.

**12. void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_tcnt, bool);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 bit를 value로 설정해 주는 함수이다.

**13. size\_tbitmap\_count (const struct bitmap \*, size\_t start, size\_tcnt, bool);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 bit 중에 값이 value인 bit 수를 반환하는 함수이다.

**14. bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_tcnt, bool);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 bit 중에 값이 value인 bit가 있으면 true, 그렇지 않으면 false를 반환한다.

**15. bool bitmap\_any (const struct bitmap \*, size\_t start, size\_tcnt);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 bit 중에 값이 true인 bit가 있으면 true, 그렇지 않으면 false를 반환한다.

**16. bool bitmap\_none (const struct bitmap \*, size\_t start, size\_tcnt);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 bit 중에 값이 true인 bit가 없으면 true, 그렇지 않으면 false를 반환한다.

1. **bool bitmap\_all (const struct bitmap \*, size\_t start, size\_tcnt);**

parameter로 받은 bitmap의 start부터 cnt개수만큼의 모든 bit들의 값이 true이면 true, 그렇지 않으면 false를 반환한다.

1. **size\_tbitmap\_scan (const struct bitmap \*, size\_t start, size\_tcnt, bool);**

parameter로 받은 bitmap의 start부터 연속적으로 값이 value인 cnt개의 bit가 나타나는 그룹의 시작 index를 반환해주는 함수이다.

1. **size\_tbitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_tcnt, bool) ;**

parameter로 받은 bitmap의 start부터 연속적으로 값이 value인 cnt개의 bit가 나타나는 그룹의 시작 index를 반환하고, 이 그룹의 모든 비트들의 값을 반대로 설정해준다.

1. **void bitmap\_dump (const struct bitmap \*);**

bitmap안의 내용을 16진수로 출력해주는 함수이다.

1. **struct bitmap\* bitmap\_expand(struct bitmap\*, int );**

Bitmap을 size만큼 확장해주는 함수이다.

Size만큼의 새로운 bitmap을 하나 생성한 뒤,이 bitmap의 bits를 기존 bitmap의 bits와 같게 해줌으로서 확장 기능을 수행한다.