**os#0\_2\_20131549**

**1반 류혜원 20131549**

**(1)새로 만든 libary function 함수에 대한 설명**

**-List <**(list.c)에 구현>

|  |  |
| --- | --- |
| Prototype | Void **list\_swap** (struct list\_elem \*a, struct list\_elem \*b) |
| Parameter | Swap 대상인 두개의 list\_elem |
| Return | None |
| function | **Parameter로 넘어온 2개의 element를 swap한다** |

|  |  |
| --- | --- |
| Prototype | Void **list\_shuffle** (struct list \*b) |
| Parameter | shuffle하려는 list |
| Return | None |
| function | **Parameter로 넘어온 list의 element 를 무작위로 섞는다** |

**-Hashtable <hash.c>에 구현**

|  |  |
| --- | --- |
| Prototype | Unsigned **hash\_int\_2**(int i) |
| Parameter | hash값을 구하려는 interger |
| Return | Integer i 의 hash값 |
| function | **Parameter로 넘어온 정수의 각 자리수를 digit을 더해서 hash값을 구한다**  Ex) 123 의 hash값은 (1+2+3=) 7  152 의 hash값은 (1+5+2=) 8 이된다 |

**-bitmap <bitmap.c>에 구현**

|  |  |
| --- | --- |
| Prototype | Struct bitmap \***bitmap\_expand** (Struct bitmap \*bitmap, int size) |
| Parameter | 확장하고 싶은 bitmap과 size |
| Return | 성공시 확장된 bitmap, 실패시 NULL |
| function | **기존의 bitmap을 size만큼 더 확장해주는 함수이다** |

**기타함수**

|  |  |
| --- | --- |
| void create(char \*input); | 이름별로 list, hash, bitmap인지를 구분하여  Datastructure에 대한 create를 진행한다 |
| void **create\_list**(char \*name); | List 생성한다 |
| void **create\_hashtable**(char \*name); | Hashtable 생성한다 |
| void **create\_bitmap** (char \*name, char \*cnt); | Bitmap 생성한다 |
| bool **delete**(char \*name); | 주어진 이름의 자료구조를 삭제한다 |
| void **dumpdata**( char \*name); | 주어진 이름의 자료구조를 stdout에 출력한다 |
| void display\_list(struct list\*); | List 정보를 출력한다 |
| void display\_bitmap(struct bitmap\* ); | bitmap정보를 출력한다 |
| void display\_hash(struct hash \*); | Hash 정보를 출력한다 |
| void **command**(char \*input); | 주어진 input에 따라 command를 내린다 |
| static bool **value\_less**  (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux); | list에서 두 element의 값을 비교하는 함수 |
| static bool  **value\_less\_hash**  (const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); | hash에서 두 element의 값을 비교하는 함수 |
| unsigned **hash\_func\_in**  (const struct hash\_elem \*a, void \*aux); | Hash func으로 hash\_int()를 이용하여 만들었다 |
| unsigned **hash\_func\_square**  (const struct hash\_elem \*e,void \*aux); | Hash func으로 hash\_int\_2()를 사용하여 구현하였다 |
| void **hash\_act\_sq**(struct hash\_elem \*a ,void \*aux); | Hash act 함수로 값을 제곱해준다 |
| void **hash\_act\_tr**(struct hash\_elem \*a ,void \*aux); | Hash act 함수로 값을 세제곱해준다 |
| int quit(char\*) | 입력이 "quit"일 때 프로그램을 종료를 알려주는 함수 |

**-사용한 구조체 및 전역 변수**

|  |  |
| --- | --- |
| struct int\_data{  struct list\_elem elem;  int data;  }; | List로 연결되어 있고 정수형 data를 표현하는 구조체 |
| struct int\_hash{  struct hash\_elem elem;  int data;  }; | 해쉬로 관리하고 정수형 data를 포현하는 구조체 |
| char\* ds\_tab[30]; | list\_name, hashtable\_name, bit\_name이 저장되어있다 |
| struct list \*list\_tab[10]; | List들의 주소를 저장하는 배열 |
| struct bitmap \*bitmap\_tab[10]; | Bitmap들의 주소를 저장하는 배열 |
| struct hash \*hash\_tab[10]; | Hash들의 주소를 저장하는 배열 |
| int cnt\_list=0;  int cnt\_hash=0;  int cnt\_bitmap=0; | List\_tab, bitmap\_tab, hash\_tab의 현재 사용개수를 나타내는 변수이다 |

**- debugging시 이용한 함수**

|  |  |
| --- | --- |
| void info\_tab(); | Ds\_tab을 출력해주는 함수로 현재 생성된 모든 함수의 이름을 출력하게 하였다. |
| void list\_info(struct list \* list); | List를 정보를 출력해주는 함수이다. head, begin, tail값을 확인할 때 사용하였다. |

**(2) 기존에 있는 함수 analysis**

**-List**

|  |  |
| --- | --- |
| Prototype | Void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| function | Before 앞에 elem 을 추가한다 |

|  |  |
| --- | --- |
| Prototype | Void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| function | 기존의 list에서 first~last 사이의 값을 slice로 잘라내어 before앞에 위치하게 만든다. |

|  |  |
| --- | --- |
| Prototype | void  list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| function | List 맨 앞에 element를 추가한다 |

|  |  |
| --- | --- |
| Prototype | void  list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| function | List 맨 뒤에 element를 추가한다 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_remove (struct list\_elem \*elem) |
| function | Parameter로 넘겨받은 element를 list에서 삭제하고, 삭제한 element다음 의 element를 return 한다 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_pop\_front (struct list \*list) |
| function | List의 맨 앞element를 지우고 이를 return한다 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_pop\_back (struct list \*list) |
| function | List의 맨 마지막 element를 지우고 이를 return한다 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_front (struct list \*list) |
| function | List의 맨 처음 element를 return한다. (삭제 없이) |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_back (struct list \*list) |
| function | List의 맨 마지막 element를 return한다. (삭제 없이) |

|  |  |
| --- | --- |
| Prototype | size\_t  list\_size (struct list \*list) |
| function | List 안의 element개수를 return 해준다 |

|  |  |
| --- | --- |
| Prototype | bool  list\_empty (struct list \*list) |
| function | List가 비어있으면 true를 ,element가 하나라도있으면 false를 return 한다 |

|  |  |
| --- | --- |
| Prototype | void  list\_reverse (struct list \*list) |
| function | List를 역순으로 만든다 |

|  |  |
| --- | --- |
| Prototype | void  list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| function | list들을 less func의 기준에 따라서 sort한다. |

|  |  |
| --- | --- |
| Prototype | void  list\_insert\_ordered (struct list \*list, struct list\_elem \*elem,  list\_less\_func \*less, void \*aux) |
| function | 정렬된 list에 element를 규칙(list\_less\_func)에 따라 삽입하는 함수이다 |

|  |  |
| --- | --- |
| Prototype | void  list\_unique (struct list \*list, struct list \*duplicates,  list\_less\_func \*less, void \*aux) |
| function | List에 중복된 element를 없애는 함수이다.  이때 list에서 삭제된 element들은 duplicates라는 list에 append해서 저장한다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| function | Less, aux라는 비교기준에서 가장 큰 값을 return 한다 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*  list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| function | Less, aux라는 비교기준에서 가장 작은 값을 return 한다 |

**-hash table**

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*  hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| Function | 파라미터로 넘겨받은 hash 구조에,새로운 요소 new를 추가하는 함수이다.  만약 기존 hash 구조에 동일한 요소가 있다면 insert가 일어나지 않는다 |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*  hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| Function | 파라미터로 넘겨받은 hash 구조에,새로운 요소 new를 추가하는 함수이다.  만약 기존 hash에 동일한 요소가 있다면, 기존의 data를 지우고, 파라미터로 넘겨받은 new로 대체한다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*  hash\_find (struct hash \*h, struct hash\_elem \*e) |
| Function | hash에서 e를 찾아 그 요소를 return 해준다  만약 일치하는 element가 hash에 없다면 null을 return한다 |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*  hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| Function | hash에서 element를 찾아 이를 지우고 rehash(hash\_최적화)를 시킨다 |

|  |  |
| --- | --- |
| Prototype | void  hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| Function | 파라미터로 넘겨받은 h의 모든 element를 삭제한다 |

|  |  |
| --- | --- |
| Prototype | size\_t  hash\_size (struct hash \*h) |
| Function | 파라미터로 넘어온 hash의 크기를 return 한다 |

|  |  |
| --- | --- |
| Prototype | bool  hash\_empty (struct hash \*h) |
| Function | 파라미터로 넘어온 hash의 empty여부를 확인하여 return 한다 |

|  |  |
| --- | --- |
| Prototype | void  hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| Function | hash에다가 action을 기하는 함수이다. |

**-bitmap**

|  |  |
| --- | --- |
| prototype | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| function | value가 1이면 해당 bit map의 위치값을 1로 mark하고  value가 0 이명 해당 bitmap의 위치값을 0으로 reset한다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| function | parameter로 넘겨받은 bitmap의 idx번째 값을 1로 mark시켜준다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| function | parmater로 남겨 받은 bitmap의 idx번째 값을 0으로 reset시켜준다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| function | parmater로 남겨 받은 bitmap의 idx번째 값을 반전시켜준다 (1을 0으로 0을 1로) |

|  |  |
| --- | --- |
| prototype | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| function | parmater로 남겨 받은 bitmap의 idx번째 값을 보여준다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_set\_all (struct bitmap \*b, bool value); |
| function | bitmap b의 모든 값을 value로 만든다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| function | bitmap[start] ~ bitmap[start+cnt-1]까지 value로 바꿔준다 |

|  |  |
| --- | --- |
| prototype | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| function | b[start] ~ b[start+cnt-1] 까지 value이 몇개 있는지 세준 값을 return 한다 |

|  |  |
| --- | --- |
| prototype | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| function | b[start]~ b[start+cnt-1] 중에 value값이 있는지 여부를 확인한다 |

|  |  |
| --- | --- |
| prototype | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| function | b[start]~ b[start+cnt-1]에 1이 있는지를 확인해주는 함수 |

|  |  |
| --- | --- |
| prototype | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| function | b[start]~ b[start+cnt-1]dl 모두 0인지를 확인하는 함수 |

|  |  |
| --- | --- |
| prototype | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| function | b[start]~ b[start+cnt-1]dl 모두 1인지를 확인하는 함수 |

|  |  |
| --- | --- |
| prototype | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| function | start부터 시작하여 value값이 연속적으로 cnt번 나오는 위치를 return 한다 |

|  |  |
| --- | --- |
| prototype | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| function | bit\_map\_scan함수와 반전 기능을 동시에 수행하는 함수이다 |

|  |  |
| --- | --- |
| prototype | void bitmap\_dump (const struct bitmap \*); |
| function | bitmap b에 저장된 data를 16진수로 출력해준다 |