운영체제 project #0-2 20121641 정유진

Library function 설명

<<list>>

1. void list\_swap (struct list\_elem \*a, struct list\_elem \*b)
   1. parameter a와 b를 swap한다.
2. void list\_shuffle(struct list \*list)
   1. parameter 로 넘어온 list 를 무작위로 섞는다.
3. void list\_init (struct list \*)
   1. parameter 로 넘어온 list 를 empty list 로 초기환한다.
4. void list\_insert (struct list\_elem \* before, struct list\_elem \* elem)
   1. parameter 로 넘어온 elem을 parameter로 넘어온 before 바로 앞에 삽입한다.
5. void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last)
   1. parameter 로 넘어온 first 에서 last 까지, 해당 list 에서 삭제한다.
   2. Parameter 로 넘어온 fist 에서 last 까지, before 의 바로 앞에 삽입한다.
6. void list\_push\_front (struct list \*, struct list\_elem \*)
   1. parameter 로 넘어온 list 의 시작 부분에 parameter 로 넘어온 elem 을 삽입한다.
7. void list\_push\_back (struct list \*, struct list\_elem );
   1. parameter 로 넘어온 list 의 끝 부분에 parameter 로 넘어온 elem 을 삽입한다.
8. struct list\_elem \*list\_remove (struct list\_elem \*)
   1. parameter 로 넘어온 elem을 list 에서 삭제한다.
9. struct list\_elem \*list\_pop\_front (struct list \*)
   1. parameter 로 넘어온 list 의 시작 elem을 list에서 삭제하고 return 한다.
10. struct list\_elem \*list\_pop\_back (struct list \*)
    1. parameter 로 넘어온 list 의 끝 elem을 list에서 삭제하고 return 한다.
11. struct list\_elem \*list\_front (struct list \*)
    1. parameter 로 넘어온 list 의 시작 elem을 return 한다.
12. struct list\_elem \*list\_back (struct list \*)
    1. parameter 로 넘어온 list 의 끝 elem을 return 한다.
13. size\_t list\_size (struct list \*)
    1. parameter 로 넘어온 list 의 크기를 return 한다.
14. bool list\_empty (struct list \*)
    1. parameter 로 넘어온 list가 empty 일 경우 true 를, 아닐 경우 false를 return 한다.
15. void list\_reverse (struct list \*)
    1. parameter 로 넘어온 list를 반전시킨다.
16. void list\_sort (struct list \*,list\_less\_func \*, void \*aux)
    1. parameter 로 넘어온 list 를 parameter 로 넘어온 less function에 따라 parameter 로 넘어온 auxiliary data 에 대하여 정렬한다.
17. void list\_insert\_ordered (struct list \*, struct list\_elem \*,list\_less\_func \*, void \*aux)
    1. parameter 로 넘어온 elem 을 parameter 로 넘어온 list의 알맞은 위치에 삽입한다. parameter 로 넘어온 less function에 따라 parameter 로 넘어온 auxiliary data 에 대하여 정렬하여 알맞은 위치를 결정한다.
18. void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux)
    1. parameter 로 넘어온 list 내에 반복되는 원소를 삭제하고, 해당 원소를 parameter 로 넘어온 duplicates 에 저장한다.
    2. 원소가 중복되는지 여부(동일성) 은 parameter로 넘어온 less function 에 따라 auxiliary data 에 대한 값으로 결정한다.
19. struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux)
    1. parameter 로 넘어온 list의 최대값을 가지는 원소의 elem 을 return 한다.
    2. 이때 list 내 원소들의 크기는 parameter 로 넘어온 less function 에 따라 주어진 auxiliary data에 대하여 결정한다.
20. struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux)
    1. parameter 로 넘어온 list의 최소값을 가지는 원소의 elem 을 return 한다.
    2. 이때 list 내 원소들의 크기는 parameter 로 넘어온 less function 에 따라 주어진 auxiliary data에 대하여 결정한다.

<<hash>>

1. bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux)
   1. parameter 로 넘어온 hash table 을 초기화 한다.
   2. Hashing 을 위해 parameter 로 넘어온 hash\_hash\_func 를 사용한다.
   3. Hash element 간 크기 비교를 위해 parameter 로 넘어온 hash\_less\_fucntion 을 parameter 로 넘어온 auxiliary data 에 대하여 사용한다.
2. void hash\_clear (struct hash \*, hash\_action\_func \*)
   1. parameter 로 넘어온 hash table 의 모든 원소를 삭제한다.
3. struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*)
   1. parameter 로 넘어온 elem 을 parameter 로 넘어온 hash table 에 삽입한다.
   2. parameter 로 넘어온 hash table 에 이미 삽입하고자 하는 elem 과 동일한 원소가 없을 경우 null을, 아닐 경우 삽입하지 않고 해당 elem 을 return 한다.
4. struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*)
   1. parameter 로 넘어온 hash table 에 parameter 로 넘어온 elem 을 삽입한다.
   2. parameter 로 넘어온 elem 과 동일한 원소가 이미 parameter 로 넘어온 hash table 에 존재 할 경우 덮어쓰고 해당 원소를 return 한다.
5. struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*)
   1. parameter 로 넘어온 elem 과 동일한 원소를 parameter 로 넘어온 hash table 에서 찾고, 해당 elem 을 return 한다.
6. struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*)
   1. parameter 로 넘어온 elem 과 동일한 원소를 parameter 로 넘어온 hash table 에서 찾고, 삭제한 후 해당 elem 을 return 한다. 동일한 원소가 없을 경우 null pointer를 리턴한다.
7. void hash\_apply (struct hash \*, hash\_action\_func \*)
   1. parameter 로 넘어온 hash table의 원소 각각에 대하여 parameter 로 넘어온 action 을 call 한다.
8. size\_t hash\_size (struct hash \*)
   1. parameter 로 넘어온 hash table 의 크기를 리턴한다.
9. bool hash\_empty (struct hash \*)
   1. parameter 로 넘어온 hash table 의 empty 여부를 리턴한다.
10. unsigned hash\_int (int)
    1. parameter 로 넘어온 int 의 hashing 한 값을 return 한다.
11. unsigned hash\_int\_2 (int)
    1. parameter 로 넘어온 int 의 hashing 한 값을 return 한다.
    2. Robert Jenkins' 32 bit integer hash function 이다.
    3. This version of integer hash function uses operations with integer constants to help producing a hash value. I suspect the actual values of the magic constants are not very important. Even using 16 bit constants may still work pretty well.
    4. These magic constants open up the construction of perfect integer hash functions. A test program can vary the magic constants until a set of perfect hashes are found.
    5. 출처 : <http://web.archive.org/web/20071223173210/http://www.concentric.net/~Ttwang/tech/inthash.htm>

<<bitmap>>

1. struct bitmap \*bitmap\_create (size\_t bit\_cnt)
   1. parameter 로 넘어온 size를 가지는 bitmap 을 생성한다.
2. size\_t bitmap\_size (const struct bitmap \*)
   1. parameter 로 넘어온 bitmap 의 크기를 리턴한다.
3. void bitmap\_set (struct bitmap \*, size\_t idx, bool)
   1. parameter 로 넘어온 bitmap의 parameter 로 넘어온 index 의 값을 parameter 로 넘어온 bool 값으로 set한다.
4. void bitmap\_mark (struct bitmap \*, size\_t idx)
   1. parameter 로 넘어온 bitmap의 parameter 로 넘어온 index 의 값을 true로 set한다.
5. void bitmap\_reset (struct bitmap \*, size\_t idx)
   1. parameter 로 넘어온 bitmap의 parameter 로 넘어온 index 의 값을 false l 값으로 set한다.
6. void bitmap\_flip (struct bitmap \*, size\_t idx)
   1. parameter 로 넘어온 bitmap의 parameter 로 넘어온 index 의 값을 toggle 한다.
7. bool bitmap\_test (const struct bitmap \*, size\_t idx)
   1. parameter 로 넘어온 bitmap의 parameter 로 넘어온 index 의 값을 return 한다.
8. void bitmap\_set\_all (struct bitmap \*, bool)
   1. parameter 로 넘어온 bitmap의 모든 값을 parameter 로 넘어온 bool 값으로 set 한다.
9. void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool)
   1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 의 값을 parameter 로 넘어온 bool 값으로 set 한다.
10. size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 의 값 중 parameter 로 넘어온 bool 값의 개수를 리턴한다.
11. bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 의 값 중 parameter 로 넘어온 bool 값의 존재 여부를 리턴한다.
12. bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 의 값 중 true 값의 존재 여부를 리턴한다.
13. bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼이 전부 false 인지 여부를 리턴한다.
14. bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼이 전부 true 인지 여부를 리턴한다.
15. size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 parameter로 넘어온 bool 값이 연속되는 group의 첫번째 index 를 리턴한다.
16. size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool)
    1. parameter 로 넘어온 bitmap의 start index부터 count 만큼 parameter로 넘어온 bool 값이 연속되는 group의 첫번째 index 를 리턴하고 값들을 flip 한다.
17. void bitmap\_dump (const struct bitmap \*)
    1. bitmap 의 값을 16진수로 출력한다.