**운영체제**

**학번 : 20141602**

**이름 : 황기덕**

1. list에서 사용된 함수
   1. void list\_insert (struct list\_elem \*, struct list\_elem \*);
      * list의 첫번째 argument의 위치에 2번째 argument를 넣는다.
   2. void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last);
      * parameter로 넘어옴 elem을 first부터 last까지 잘라 before앞에 넣는다.
   3. void list\_push\_front (struct list \*, struct list\_elem \*);
      * 받은 elem을 list 맨 앞에 추가한다.
   4. void list\_push\_back (struct list \*, struct list\_elem \*);
      * 받은 elem을 list 맨 뒤에 추가한다.
   5. struct list\_elem \*list\_remove (struct list\_elem \*);
      * 받은 elem을 삭제하고, 삭제된 다음 elem을 return한다.
   6. struct list\_elem \*list\_pop\_front (struct list \*);
      * list의 맨 앞부분을 pop하여 삭제하고, 그 elem을 return한다.
   7. struct list\_elem \*list\_pop\_back (struct list \*);
      * list의 맨 뒷부분을 pop하여 삭제하고, 그 elem을 return한다.
   8. struct list\_elem \*list\_front (struct list \*);
      * list의 맨 앞부분의 elem을 return한다.
   9. struct list\_elem \*list\_back (struct list \*);
      * list의 맨 뒷부분의 elem을 return한다.
   10. size\_t list\_size (struct list \*);
       * list의 size를 return한다.
   11. bool list\_empty (struct list \*);
       * list가 비어있다면 true, 아니라면 false를 return한다.
   12. void list\_reverse (struct list \*);
       * list의 elem부분을 역순으로 뒤집는다.
   13. void list\_sort (struct list \*, list\_less\_func \*, void \*aux);
       * list를 function pointer를 이용하여 정렬한다.
   14. void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux);
       * list를 정렬한 후, elem을 적절한 위치에 삽입한다.
   15. void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux);
       * list를 돌면서 중복된 elem이 있는지 확인하고, 있으면 제거한 후, duplicate에 중복된 값들을 제거하여 넣는다.
   16. struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux);
       * list에서 list\_less\_func을 이용해 최대값을 찾아 return해준다.
   17. struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux);
       * list에서 list\_less\_func을 이용해 최솟값을 찾아 return해준다.
   18. void list\_init (struct list \*);
       * list를 초기화해준다.
   19. struct list\_elem \*list\_begin (struct list \*);
       * list의 시작부분 elem을 return해준다.
   20. struct list\_elem \*list\_next (struct list\_elem \*);
       * elem의 다음 elem을 return해준다.
2. hash에서 사용된 함수
   1. bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux);
      * hash를 초기화해준다.
   2. void hash\_destroy (struct hash \*, hash\_action\_func \*);
      * hash를 없애준다.
   3. struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*);
      * hash에 elem을 적절한 위치에 추가시켜준다. 같은 데이터가 있다면 추가하지 않는다.
   4. struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*);
      * hash 안에 새로운 elem을 추가한다. 이미 같은 data를 가지고 있다면, 기존에 있던 데이터를 새 데이터로 대체한다.
   5. struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*);
      * hash에서 elem과 같은 값을 가진 elem을 찾아 이를 return해주고, 없으면 NULL을 return해준다.
   6. struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*);
      * hash에서 elem과 같은 값을 가진 elem을 삭제시켜주고, 삭제한 elem을 return시켜 준다. 만약 존재하지 않는다면 NULL을 return시켜준다.
   7. void hash\_apply (struct hash \*, hash\_action\_func \*);
      * hash에 대해 hash\_action\_func을 수행한다.
   8. void hash\_first (struct hash\_iterator \*, struct hash \*);
      * hash에서 첫부분을 hash\_iterator 부분에 넣어준다.
   9. size\_t hash\_size (struct hash \*);
      * hash의 size를 return해준다.
   10. bool hash\_empty (struct hash \*);
       * hash가 비어있으면 true, 아니면 false를 return해준다.
3. bitmap에서 사용된 함수
   1. void bitmap\_destroy (struct bitmap \*);
      * bitmap을 free시켜준다.
   2. size\_t bitmap\_size (const struct bitmap \*);
      * bitmap의 size를 return시켜준다.
   3. void bitmap\_set (struct bitmap \*, size\_t idx, bool);
      * bitmap에서 넘겨받은 idx를 bool에 맞춰서 설정해준다.
   4. void bitmap\_mark (struct bitmap \*, size\_t idx);
      * bitmap의 idx에 있는 부분을 true로 set한다.
   5. void bitmap\_reset (struct bitmap \*, size\_t idx);
      * bitmap의 idx에 있는 부분을 false로 set한다.
   6. void bitmap\_flip (struct bitmap \*, size\_t idx);
      * bitmap의 idx번째 값을 다른 값으로 바꾼다.
   7. bool bitmap\_test (const struct bitmap \*, size\_t idx);
      * bitmap의 idx번째 값을 return해준다.
   8. void bitmap\_set\_all (struct bitmap \*, bool);
      * bitmap을 bool값으로 전부 set해준다.
   9. void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool);
      * bitmap의 start bit부터 cnt만큼 bool값으로 변경해준다.
   10. size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool);
       * bitmap의 start부터 cnt값까지 bit를 비교하여 bool로 set되어있는 bit수를 return해준다,
   11. bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool);
       * bitmap의 start부터 cnt까지의 bit중 bool을 포함하는게 있으면 true, 아니면 false를 return해준다.
   12. bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt);
       * bitmap의 start부터 cnt까지 bit중 하나라도 true면 true를, 아니면 false를 return해준다.
   13. bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt);
       * bitmap의 start부터 cnt까지 모든 bit가 false면 true를, 아니면 false를 return해준다.
   14. bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt);
       * bitmap의 start부터 cnt까지 모든 bit가 true면 true를, 아니면 false를 return해준다.
   15. size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool);
       * bitmap의 start부터 bool값이 연속적으로 cnt만큼 반복되는 첫 번째 구간을 찾아 그 위치를 return 해주고, 없다면 BITMAP\_ERROR를 return해준다.
   16. size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool);
       * bitmap의 start부터 bool값이 연속적으로 cnt만큼 반복되는 첫 번째 구간을 찾아 그 구간의 값들을 flip(not)시켜주고, 없으면 BITMAP\_ERROR를 return해준다.
   17. void bitmap\_dump (const struct bitmap \*);
       * bitmap의 내용을 16진수로 출력해준다.
4. 추가 구현한 함수
   1. int find\_datastruct\_number(int type, char\*name);

* type에 맞는 data struct의 번호를 찾아 return해준다.
  1. bool list\_less(const struct list\_elem \*, const struct list\_elem \*, void\*);
* list sort에서 사용되는 list less 함수, 대소관계를 비교해 앞에것이 작으면 true, 아니면 false를 return해준다.
  1. void list\_swap(struct list\_elem \*, struct list\_elem \*);
* 두 개의 elem을 서로 교환해준다.
  1. list\_shuffle(struct list \*);
* list에 있는 elem을 무작위로 섞어준다.
  1. unsigned hash\_int\_2(int );
* int를 받아, 각 i자리에 그 자리수(1, 2, 3….)를 곱한 것을 모두 더한 값을 142857 로 나눈 나머지를 사용한다.
  1. bool hash\_less (const struct hash\_elem \*, const struct hash\_elem\*, void\*);
* hash\_elem 두 개중 왼쪽 값이 작으면 true, 아니면 false를 return한다.
  1. unsigned hash\_func(const struct hash\_elem \*, void\*);
* hash를 사용하기 위해 만들어 둔 function
  1. void hash\_destruct(struct hash\_elem\*, void\*);
* hash를 free시키기 위해 사용해준다.
  1. void hash\_square(struct hash\_elem, void\*);
* elem으로 들어온 값을 제곱해준다.
  1. void hash\_triple(struct hash\_elem, void\*);
* elem으로 들어온 값을 세제곱해준다.
  1. struct bitmap \*bitmap\_expand(struct bitmap\*, int)
* bitmap의 size를 int만큼 더해 그 결과 주소를 return해준다.