Errori Comuni Frammento 1

Esercitazione di Laboratorio di Sistemi Operativi

12/04/2010

new_List()

- Non inizializzare `head` a `NULL`:

```
newList->compare = compare;
newList->copyk = copyk;
newList->copyp = copyp;
```

- Corretto:

```
newList->head = NULL;
newList->compare = compare;
newList->copyk = copyk;
newList->copyp = copyp;
```

add_ListElement()

- Assegnare direttamente `key` e `payload` senza copiarli:

```
newElem>key = key;
newElem->payload = payload;
```

- Corretto:

```
newElem>key = t->copyk(key);
newElem->payload = t->copyp(payload);
```

free_list()

- Non controllare che i puntatori siano diversi da `NULL`:

```
elem_t *cur = (*pt)->head;
```

- Corretto:

```
if(pt == NULL || *pt == NULL) //lazy: OK
    return;
elem_t *cur = (*pt)->head;
```

free_list()

- Non effettuare la `free()` di `key` e `payload`:

```
while(curr != NULL)
{
    elem_t *temp = curr;
    free(temp);
    temp = (curr->next);
}
free(*pt);
*pt = NULL;
```

- Corretto:

```
while(curr != NULL)
{
    elem_t *temp = curr;
    free(curr->key);
    free(curr->payload);
    free(temp);
    temp = (curr->next);
}
free(*pt);
*pt = NULL;
```

new_hashTable()

- Non eseguire la `malloc()` per l'array di liste:

```
newHash->table = (list_t **)(size * sizeof(list_t));
```

- Fare `malloc()` di dimensione sbagliata per l'array di liste:

```
newHash->table = (list_t **) malloc(size * sizeof(list_t));
```

- Corretto:

```
newHash->table = (list_t **) malloc(size * sizeof(list_t *));
```

new_hashTable()

- Non inizializzare i puntatori alle liste a NULL:

```
newHash->table = (list_t **) malloc(size * sizeof(list_t *));
newHash->size = size;
newHash->compare = compare;
newHash->copyk = copyk;
newHash->copyp = copyp;
newHash->hash = hashfunction;
```

- Corretto:

```
newHash->table = (list_t **) malloc(size * sizeof(list_t *));
for(i=0; i<size; i++)
   newHash->table[i] = NULL;
newHash->size = size;
newHash->compare = compare;
newHash->copyk = copyk;
newHash->copyp = copyp;
newHash->hash = hashfunction;
```

find_hashElement()

- Ritornare il puntatore all'elemento trovato invece della copia:

```
elem = find_ListElement(t->table[index], key);
....
return elem->payload;
```

- Corretto:

```
elem = find_ListElement(t->table[index], key);
...
return t->copyp(elem->payload);
```

find_hashElement()

- Non controllare che la lista all'indice calcolato sulla chiave sia diversa da NULL:

```
index = t->hash(key, t->size);
found = find_ListElement(t->table[index], key);
```

- Corretto:

```
index = t->hash(key, t->size);
if(t->table[index])
   found = find_ListElement(t->table[index], key);
```

free_hashTable()

- `free()` sbagliata delle liste:

```
for (i = 0; i < *pt->size; i++)
   free(*pt->table[i]);

for (i = 0; i < *pt->size; i++)
   freeList(*pt->table[i]);

for (i = 0; i < *pt->size; i++)
   free(&(*pt->table[i]));
```

- Corretto:

```
for (i = 0; i < *pt->size; i++)
   freeList(&(*pt->table[i]));
```

- Non deallocare la `table` dopo aver deallocato le liste:

```
free(t->table);
t->table = NULL;
```

Altro

- `free_list()` ricorsiva: perché?
- Funzioni di hash sbagliate:
  - usano indirizzi anziché valori;

    ```
    return index = abs(((int)(&key)%(int)(&size)));
    ```

  - non controllano che l'indice sia minore di `size`.
- Fare la `free()` sui puntatori a funzione.
- I parametri in C sono sempre passati per valore, quindi passare il doppio puntatore serve proprio per ovviare a questa situazione.
- `mtrace` indica le righe in cui vengono fatte le allocazioni e non le mancate `free()`.

Uso Artistico di Puntatori

```
for(i=0;i<size;i++){
   if((*(table->table+i)=new_List(compare,copyk,copyp))==NULL){
      while(i-1>=0)
         free(&*((table->table+(-i))));
      free(table);
   }
}
```