Project #1: MyLib

Today: March 19, 2024
Submission Due: March 29, 2024
(Late submission: April 1, 2024, 10% per a day)

1 Introduction

In the MyLib project, students will explore various data structures within the kernel library. They will analyze these data structures and enhance their C programming skills by implementing structures such as lists, hash tables, and bitmaps. As a result, students will develop an interactive program capable of evaluating the functionalities of lists, hash tables, and bitmaps.

2 Materials

There are three items you must download to complete the MyLib Project.

- `Basics to know to implement MyLib`: To understand the basics of data structures required for MyLib, you can refer to 'sp_prj1_mylib.pdf', which is available for download on Cyber Campus.
- `Skeleton code`: You can begin the project by downloading 'sp_prj1_code.tar.gz', which is provided on Cyber Campus, to a directory where you plan to work on the project.
- `Tester code`: To verify your implementation, you need to download 'sp_prj1_tester.tar.gz'. **You must implement MyLib to ensure it operates with every command in the file format of '*.in'.** You can also execute the tester script with your interactive program ($ sh prj_tester.sh ../20xxxxxx/testlib). This will generate '.output' files that display the output contents of each test ('.output' 파일은 기존에 있는 '.out' 파일과 비교됩니다. '.out'파일을 삭제하지 않도록 유의하시기 바랍니다). Result files will show the outcome of each test, and 'Score.txt' will reveal your total score.
- Below are examples of commands used in the '*.in' file. Your interactive program should work properly with each command as illustrated in the example.
  - `create list <LIST>`: Creates LIST.
  - `create hashtable <HASH_TABLE>`: Creates HASH_TABLE.
  - `create bitmap <BITMAP> <BIT_CNT>`: Creates BITMAP with the size of BIT_CNT.
  - `delete <LIST | HASH_TABLE | BITMAP>`: Deletes the given data structure.
  - `dumpdata <LIST | HASH_TABLE | BITMAP>`: Prints the given datas in data structure to STDOUT.
  - `quit`: Terminates the interactive program.
- `Simulation Example`: Following is and example of running interactive program.

```
$~> ./testlib         //-------> excutable file name which will be putted in Makefile
create list list1
dumpdata list1        //-------> No output yet
list_push_front list1 1
list_push_back list1 4
list_puch_back list1 3
dumpdata list1
1 4 3                 //-------> Output of 'dumpdata'
list_max list1
4                     //-------> Output of 'list_max'
```

```
list_shuffle list1
dumpdata list1
4 1 3                  //-------> Output of 'dumpdata'
quit                   //-------> Terminate the program
$~>
```

---

3 Project Description

In this project, you are required to implement the following functions for three data structures used in MyLib.

- `List (in list.c)`:

```
void list_swap(struct list_elem *a, struct list_elem *b){
/*************************************************
- Functionality : Swap two list elements in parameters.
- Parameter     : Pointer of two list elements that will be swapped.
- Return value  : None.
*************************************************/
}
```

Figure 1: Description of list_swap function

```
void list_shuffle(struct list *list){
/*************************************************
- Functionality : Shuffle elements of LIST in the parameter.
- Parameter     : Pointer of list that will be shuffled.
- Return value  : None.
*************************************************/
}
```

Figure 2: Description of list_shuffle function

- `Hash Table (in hash.c)`:

```
unsigned hash_int_2 (int i){
/*************************************************
- Functionality : Implement this in your own way and describe it in the document.
- Parameter     : Integer that will be hashed.
- Return value  : Hash value of integer i.
*************************************************/
}
```

Figure 3: Description of hash_int_2 function

- `Bitmap (in bitmap.c)`:

```
struct bitmap *bitmap_expand(struct bitmap *bitmap,int size){
/*************************************************
- Functionality : Expand the given bitmap to the size (backward expansion).
- Parameter     : Pointer of bitmap that you want to expand and the required size of it.
- Return value  : Pointer of expanded bitmap if succeed, NULL if fail.
*************************************************/
}
```

Figure 4: Description of bitmap_expand function

After implementing the above functions, you have to write an interactive program that can test the functionalities of lists, hash tables, and bitmaps in the main function. (main function에서 sp_prj1_tester 폴더 내의 '*.in' 형식의 파일을 줄 단위로 읽고, 해당 줄의 명령어를 MyLib을 통해 수행한다고 생각하시면 됩니다. 따라서, sp_prj1_tester 폴더 내의 '*.in' 형식의 파일들의 모든 명령어가 올바르게 동작해야합니다.)
You can utilize various functions of the three data structures and fix them if needed, to ensure the interactive program works correctly.

Thus, an example of the contents in your working directory might look similar to Figure 5 (Note that you also need to create a **Makefile**).

```
cse20189999@cspro9:~/20189999$ ls -al
total 96
drwxr-xr-x 2 cse20189999 under  4096 Sep 18 09:20 .
drwx------ 6 cse20189999 under  4096 Sep 18 00:48 ..
-rw-r--r-- 1 cse20189999 under  9948 Sep 17 14:23 bitmap.c
-rw-r--r-- 1 cse20189999 under  1809 Sep 11  2008 bitmap.h
-rw-r--r-- 1 cse20189999 under   391 Sep 17 14:03 debug.c
-rw-r--r-- 1 cse20189999 under  1264 Sep 17 14:01 debug.h
-rw-r--r-- 1 cse20189999 under     0 Sep 17 14:27 document_20189999.docx
-rw-r--r-- 1 cse20189999 under 11845 Sep 13  2012 hash.c
-rw-r--r-- 1 cse20189999 under  3821 Sep 11  2008 hash.h
-rw-r--r-- 1 cse20189999 under  1518 Sep 17 14:04 hex_dump.c
-rw-r--r-- 1 cse20189999 under   185 Sep 17 14:04 hex_dump.h
-rw-r--r-- 1 cse20189999 under   705 Sep 13  2012 limits.h
-rw-r--r-- 1 cse20189999 under 14913 Sep 13  2012 list.c
-rw-r--r-- 1 cse20189999 under  5863 Sep 11  2008 list.h
-rw-r--r-- 1 cse20189999 under    25 Sep 17 14:33 main.c
-rw-r--r-- 1 cse20189999 under   428 Sep 17 14:32 Makefile
-rw-r--r-- 1 cse20189999 under   580 Sep 17 14:02 round.h
cse20189999@cspro9:~/20189999$
```

Figure 5: List of contents in project folder

Also, during implementing requirements, you have to follow the assumptions below.

1. All inputs are from standard input (STDIN).
2. All inputs and outputs are lower cases.
3. All the types used in the program are integer.
4. Use hash_int() as hash function for hash table, which is already given library (hash.c의 hash_int()를 사용할 것. 그래도 hast_int_2는 구현하셔야 합니다).
5. Use true or false when the return type is Boolean.
6. The number of list, hash table and bitmap is less than or equal to 10.
7. You can use any function in given source codes and you can implement your own code if it is needed.

Tip 1: You may find that your program becomes pended while testing list_swap. Many students struggle with swapping the 0th and 1st elements.

Tip 2: When printing 'size_t' type values with printf(), use the length sub-specifier, such as z, to preserve the data's length. Ex. size_t a=10; printf("%zu", a);

4 Submission Guideline

Hand-In Specifications: The submission should include only source code files, a Makefile, and documentation. No executable programs should be included. The TA responsible for grading your project will automatically rebuild your shell program from the provided source code. **Please adhere strictly to the submission instructions below to avoid penalties on your overall project score.**

1. You must submit a single archive file named '`prj1_`*your_student_id*`.tar.gz`'.
2. Upon extracting the file, the root directory name should be your student ID.
3. The executable file created by the Makefile must be named **testlib** (no other names are allowed).
4. The document should be named 'document_*your_student_id*.docx (no formats other than .docx are allowed).

Attachment File:

- `Makefile`: Failure to use a Makefile will result in no points being awarded.
- `Provided libraries`: Files in the 'sp_prj1_code' directory.
- `Your Source Code`: For example, main.c, etc.
- `Document` : Briefly explain all the library functions and the functions you wrote, including their functionality, parameters, and return value.

Scoring:

- 80% for test cases (implementation) and 20% for documentation.
- Compilation failures will result in zero points.
- Code Copy will incur penalties (1st time: 0 point and downgrade; 2nd time: F grade).

Following is and example of a submission.

```
$~> ls
prj1_20241234.tar.gz
$~> tar -zxvf prj1_20241234.tar.gz
$~> ls
20241234  prj1_20241234.tar.gz
$~> cd 20241234
$~> ls -al
document_20241234.docx
bitmap.c
bitmap.h
debug.c
debug.h
hash.c
hash.h
hex_dump.c
hex_dump.h
limits.h
list.c
list.h
main.c
Makefile
round.h
```

All students are requested to upload their archived source files on eclass (Cyber Campus).

Note: Please ensure that you compile your source code on the CSPRO server, as the TA will build and compile your submitted project on that server. If the submitted code does not compile, it cannot be scored!

5 Miscellaneous

5.1 Best practices the system programmers must comply to

WHAT YOU MUST DO:

1. You must thoroughly read the entire data structure documentation to fully understand, learn, and complete this project.
2. Ensure you read the project specifications before beginning your MyLib project programming.
3. Commenting on your source code is a professional practice for programmers and is required for this project as well.

WHAT YOU MUST NOT DO:

1. Do not submit any binary or object code files in your source code directory.
2. Avoid including any hardcoded paths in your Makefile.
3. Your Makefile must list all dependencies (libraries, etc.) required to build your program.
4. Do not copy or share your source code with others. Doing so is strictly prohibited and will result in penalties.