Project #0-2: Pintos Data Structures

[CSE4070]

Instructors

Prof. Sungyong Park

Teaching Assistants

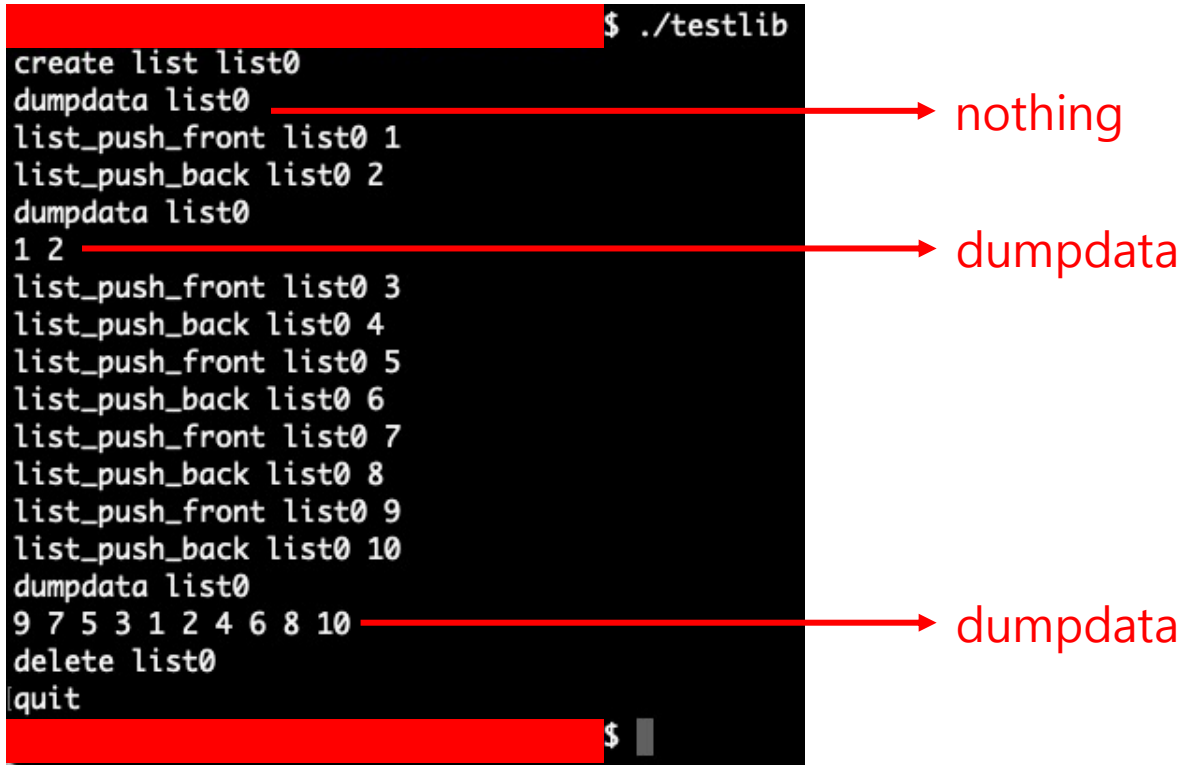
Hongsu Byun

Fall 2021

Project Outline

Outline

- Before we dive into Pintos project, we will practice Pintos data structures.
- Write the interactive program that can check functionalities of list, hash table and bitmap in Pintos kernel
- Ex) list push



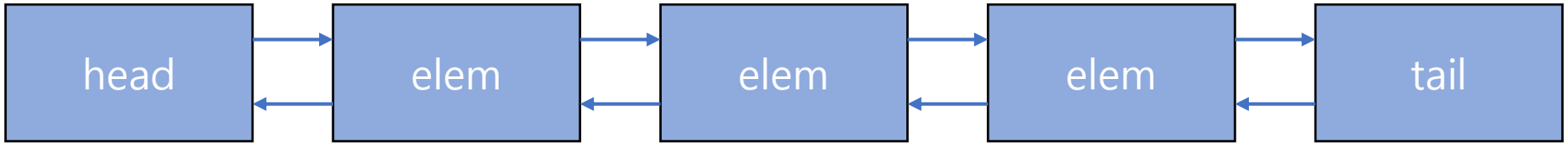
Data Structures

Data Structures in Pintos Kernel

- Pintos provides kernel and user libraries.
- You can find it in "pintos/src/lib/kernel" and "pintos/src/lib/user"
- In this project, we will cover data structures of Pintos kernel libraries.
  → **List, Hash table and Bitmap**

List

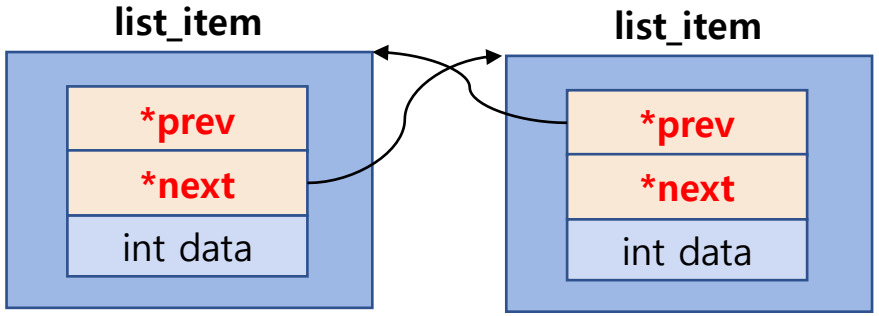
- List in Pintos is a **doubly linked list.**
- It is different from usual list structure.
- It splits list element pointers and data.
- struct list_elem
  - Each structure that will be a list item must embed a struct list_elem member.
  - All the list functions operate on list_elem, not the list item.
  - Only list_elem structure is given in the source code.
  - You must implement new structure that consists of list_elem and data.



List

- Linked List: Usual way

```
struct list_item
{
    struct list_item *prev
    struct list_item *next
    int data;
}
```

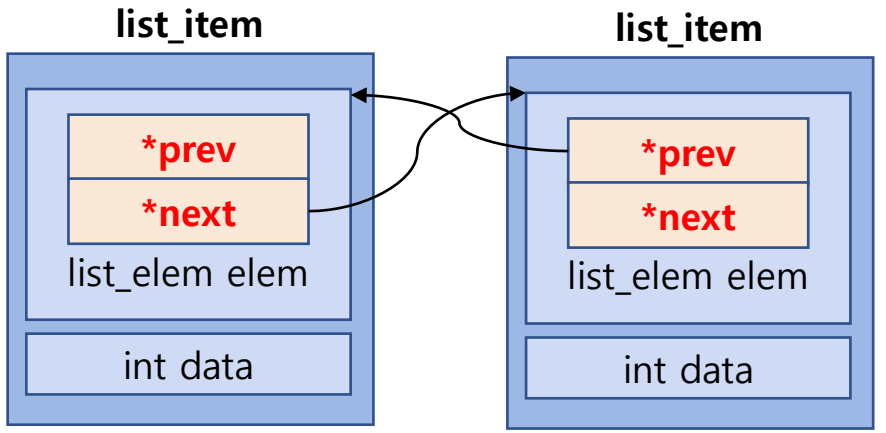


- Linked List: Pintos kernel

```
struct list_elem
{
    struct list_elem *prev;
    struct list_elem *next;
}
```

※

```
struct list_item
{
    struct list_elem elem;
    int data;
    /* Other members you want */
}
```



Split the pointer and data

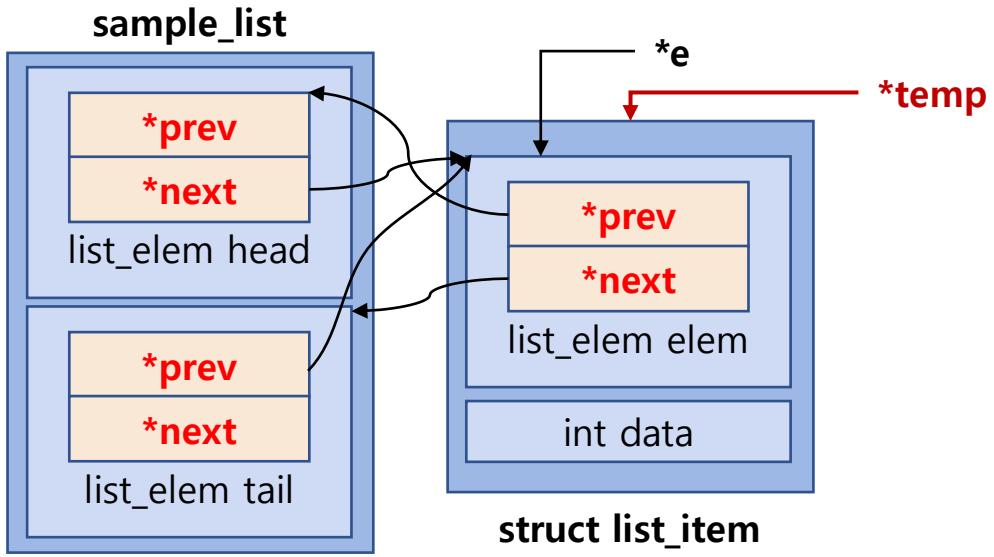
※ 'struct list_item' is not given in the source code, you need to implement it.

List Function Analysis

- **void list_init(struct list *list)**
  - Initializes LIST as an empty list.
  - It should be executed before an element is inserted in LIST.
- **struct list_elem* list_begin(struct list *list)**
  - Returns the first element of LIST.
  - Usually used to iterate the LIST.
- **struct list_elem* list_next(struct list_elem *elem)**
  - Returns the next element of ELEM.
  - Usually used to iterate the LIST or search ELEM in the LIST.

List Function Analysis

- **struct list_elem* list_end(struct list *list)**
  - Returns the last ELEM in the LIST.
  - Usually used to iterate the LIST.
- **#define list_entry(list_elem, struct, member)**
  - Converts the pointer to LIST_ELEM into a pointer to STRUCT that LIST_ELEM is embedded inside.
  - Usually used to get address of STURCT which embeds LIST_ELEM.

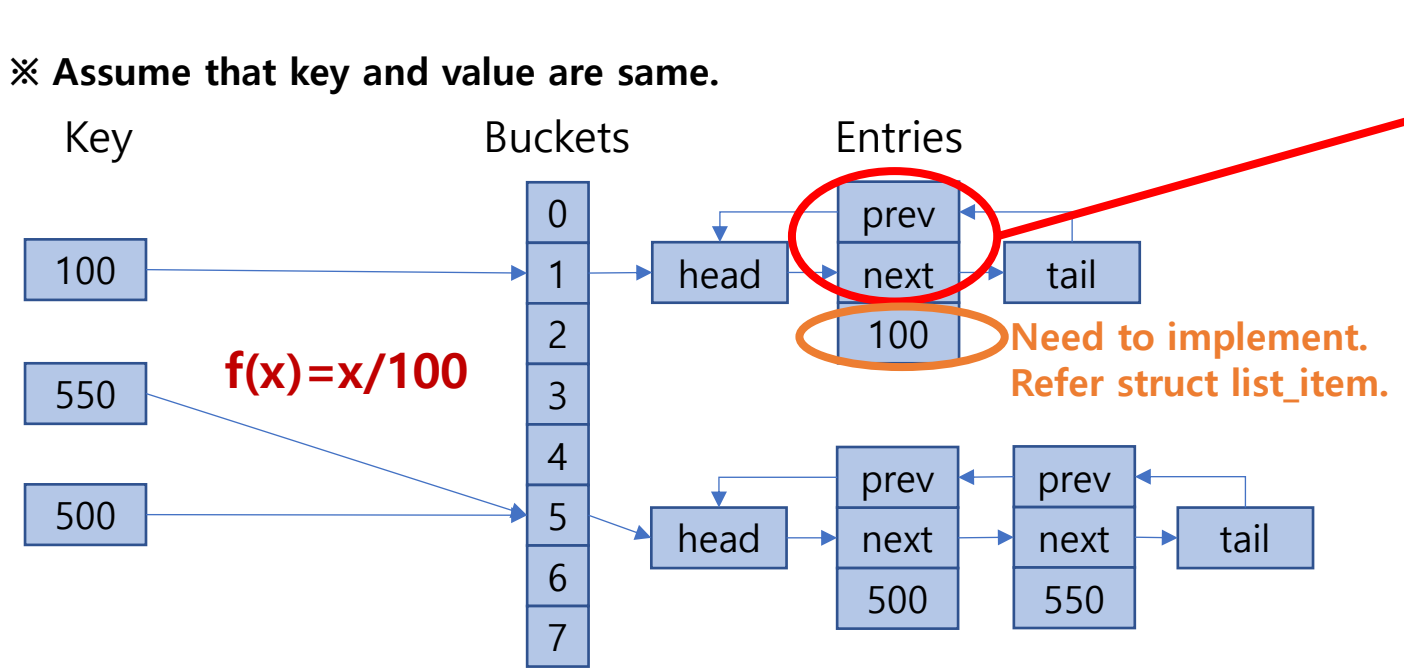



```
/* Assume that there already exists a list, sample_list */
struct list_elem *e;
e = list_begin (&sample_list);
struct list_item *temp = list_entry(e, struct list_item, elem)
int temp_data = temp->data
```

By using list_elem, we can get address of list_item

Hash Table

- A hash table is a data structure that associates keys with values.
- The primary operation is a lookup.
  - Given a key, find the corresponding value.
- It works by transforming the key using a **hash function** into a hash.




```
struct hash_elem
  {
    struct list_elem list_elem;
  };
```

```
struct hash
  {
    size_t elem_cnt;
    size_t bucket_cnt;
    struct list *buckets;
    hash_hash_func *hash;
    hash_less_func *less;
    void *aux;
  };
```

Hash Table Function Analysis

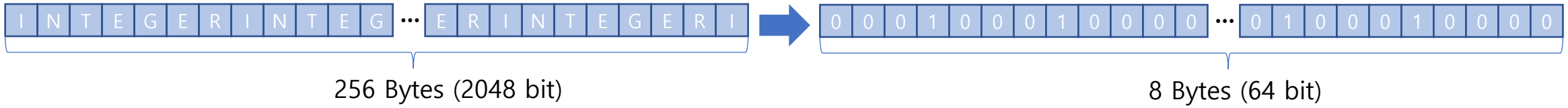
- **void hash_init(struct hash *h, hash_hash_func *hash, hash_less_func *less, void *aux)**
  - Initializes hash table H and sets hash function HASH and comparison function LESS.
  - You can see the example of hash function such as hash_int, hash_bytes, and hash_string.
    (You have to use hash_int function to pass the test.)
  - Comparison function LESS is used to compare two hash elements.
- **void hash_apply(struct hash *h, hash_action_func *action)**
  - You can apply any ACTION function which you made to hast table H.
  - Used for applying specific function to all elements in hash table.
    e.g.) square function.
  - You can learn the usage of it from 'hash_apply.in' and 'hash_apply.out' in tester directory.

Hash Table Function Analysis

- **#define hash_entry(hash_elem, struct, member)**
  - Converts pointer to HASH_ELEM into a pointer to STRUCT that HASH_ELEM is embedded inside.
  - Usually used to get address of STURCT which embeds HASH_ELEM.

Bitmap

- A bit array(or bitmap, in some cases) is an array which stores individual bits (Boolean values).
- A bitmap can reduce the waste of memory space.



- Bitmap: Usual way

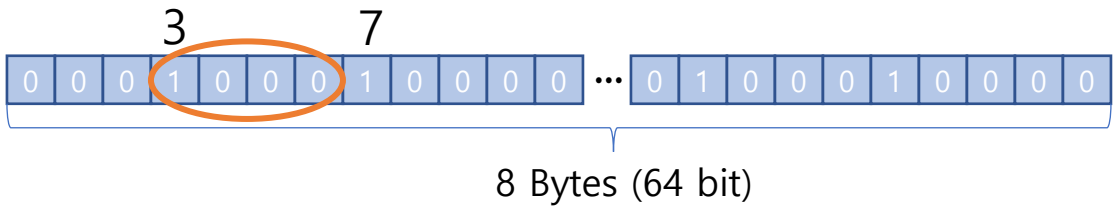
```
char bitmap[8];
/* Or */
int bitmap[2];
/* Or */
unsigned long bitmap;
```

- Bitmap: Pintos kernel

```
typedef unsigned long elem_type;
struct bitmap
{
    size_t bit_cnt;
    elem_type *bits;
};
```

Bitmap Function Analysis

- **struct bitmap *bitmap_create(size_t bit_cnt)**
  - Initializes a bitmap of BIT_CNT bits and sets all its bits to false.
- **void bitmap_set (struct bitmap *b, size_t idx, bool value)**
  - Atomically sets the bit numbered IDX in B to VALUE.
- **size_t bitmap_count (const struct bitmap *b, size_t start, size_t cnt, bool value)**
  - Returns the number of bits in B between START and START + CNT, exclusive, that are set to VALUE.



bitmap_count(b, 3, 4, 1) == 1

Requirements

Project #0-2

- **Write the interactive program that can check functionalities of list, hash table and bitmap in Pintos kernel**
- Assumption
  - All inputs are from standard input (STDIN).
  - All inputs and outputs are lower cases.
  - All the types used in the program are integer.
  - Use hash_int() as hash function for hash table.
    - **Use hash_int() in given library. (라이브러리의 hash_int()를 사용할 것.)**
  - Use true or false when the return type is Boolean.
  - The number of list, hash table and bitmap is less than or equal to 10.
  - You can use any function in given source codes and you can implement your own code if it is needed.

Project #0-2: List of Functions to Implement

You should implement the following functions.

- List

1) void list_swap(struct list_elem *a, struct list_elem *b)

- Parameter: Two list elements that will be swapped.
- Return value: None.
- Functionality: Swap two list elements in parameters.

2) void list_shuffle(struct list *list)

- Parameter: List that will be shuffled.
- Return value: None.
- Functionality: Shuffle elements of LIST in the parameter.

Implement list_swap() and list_shuffle() in list.c

Project #0-2: List of Functions to Implement

You should implement the following functions.

※ Do not use this function in your code, just implement hash_int_2() in your code
※ Use hash_int() as hash function to pass the test program
테스트 프로그램 통과를 위해 hash function은 hash_int()를 사용할 것.

- Hash table
  1) unsigned hash_int_2(int i)
     - Parameter: Integer that will be hashed
     - Return value: Hash value of integer I
     - Functionality: Implement this in your own way and describe it in the document.
- Bitmap
  1) struct bitmap *bitmap_expand(struct bitmap *bitmap, int size)
     - Parameter: Bitmap that you want to expand and the size of it
     - Return value: Expanded bitmap if succeed, NULL if fail
     - Functionality: Expand the given BITMAP to the SIZE (backward expansion)

Implement hash_int_2() in hash.c and bitmap_expand() in bitmap.c

Project #0-2

- These are only few examples of command used in interactive program you will make.
- **You should check the tester file (*.in) to see what commands are used for the test.**
  어떤 명령어가 사용되는지 *.in 테스터 파일을 통해 직접 확인할 것.
  - 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 data structure to standard out (STDOUT)s
  - quit
    - Terminates the interactive program.

Project #0-2

Example (List)

❖ Note that this is just the example of the test case.
You should test your program by yourself or by test program!

```
$ ./testlib
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 list1'
list_max list1
4                       <-- Output of 'list_max list1'
list_shuffle list1
dumpdata list1
4 1 3                   <-- Output of 'dumpdata list1'
quit
$
```

Project #0-2

- Kernel Library source files are in ‘pintos/src/lib/kernel’
  - list.h, list.c, hash.h, hash.c, bitmap.h, bitmap.c
- Some files are dependent on Pintos source codes.
- **You should use the source files in 'lib_hw1.tar.gz' which we provide.**
  Don't copy the files from Pintos source code.
  핀토스 소스코드에서 카피한 파일을 사용하지 마시오.

Project #0-2: Tester Program

- You can use tester program (hw1_tester.sh) to test your implementation.
  1) Download os_hw1_tester.tar.gz from e-class.
  2) Extract it. ($ tar -zxvf os_hw1_tester.tar.gz).
  3) Go to os_hw1_tester directory. ($ cd os_hw1_tester).
  4) Run tester script with your interactive program. ($ sh hw1_tester.sh ../20189999/testlib).
  5) It will produce **.output** files which show you the output contents of each test. **.result** files will show you the result of each test and **Score.txt** will show you the total score.
- Tips
  - You may face that your program is being pended while testing list_swap.
  - Many students failed to swap 0th element and 1st element.

Project #0-2: Cautions

- If you use printf() to print size_t type values, use length sub-specifier, such as z, not to harm the length of data.
- Ex:

  size_t a=10;

  printf("%zu", a);
- Refer the webpage http://www.cplusplus.com/reference/cstdio/printf/ for more. information.

Submission

- **Contents**
  - ① Makefile **(You will get no point if you do not use Makefile).**
  - ② Provided libraries (given files in lib_hw1 directory).
  - ③ Source code you made.
  - ④ Document (softcopy).
    → Explain briefly all the library functions and functions which you wrote (functionality, parameter, return value)
- **Form and way to submit (Check it twice before submission. 제출 전, 두 번 이상 확인할 것)**
  1) Form of the File
     - document: **document_[ID].docx** (Other format is not allowed such as .hwp)
     - Submission contents should be contained in the directory that has ID as directory name
       - ✓ For example, if your ID is 20189999, Makefile, provided libraries, source code you made, and document file should be in '20189999' directory.
     - Compress the 'ID' directory into **'os#0_2_[ID].tar.gz'**
       - ✓ You should use -zcf options for using tar. e.g.) **tar -zcf os#0_2_20189999.tar.gz 20189999**
  2) Way to Submit: Upload the tar.gz file to e-class
     - **The name of interactive program (produced by 'make' command) should be "testlib"** (Other name such as a.out, output, and main is not allowed)
     - **Do 'make clean' before you compress the directory**
     - **No Hardcopy.**
  3) Due Date: 2020. 10. 4 23:59

[ID] 작성 시 대괄호는 포함하면 안됩니다.
20189999 (O), [20189999] (X)

make로 산출되는 실행 파일 이름은
testlib이어야 합니다.

압축 전 make clean 하세요.

❖ **5% of point will be deducted for a wrong form and way to submit**

❖ **Late submission is allowed up to 3 days (~10/7) and 10% of point will be deducted per day**

Submission

- Score
  - 80% for test cases (implementation) and 20% for documentation.
- **If compile fails, you will get no point.**
- **Copy will get a penalty (1st time: 0 Point and downgrading, 2nd time: F grade)**

Appendix: Way to Submit Your Work

Way to Submit Your Work

- Correct case
  - All the files should be in your 'ID' directory

```
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$
```

Way to Submit Your Work

- **Incorrect case (1)**
  - 'make clean' is not performed in this case (There are *.o and testlib files)

```
cse20189999@cspro9:~/20189999$ ls -al
total 140
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  9152 Sep 18 09:20 bitmap.o
-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  2064 Sep 18 09:20 debug.o
-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  2616 Sep 18 09:20 hex_dump.o
-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  1360 Sep 18 09:20 main.o
-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
-rwxr-xr-x 1 cse20189999 under 18440 Sep 18 09:20 testlib
cse20189999@cspro9:~/20189999$
```

Way to Submit Your Work

- **Incorrect case (2)**
  - Library files are in 'lib_hw1' directory in this case
  - Don't contain libraries in other directory, just contain it in 'ID' directory

```
cse20189999@cspro9:~/20189999$ ls -al
total 20
drwxr-xr-x 3 cse20189999 under 4096 Sep 18 09:19 .
drwx------ 6 cse20189999 under 4096 Sep 18 00:48 ..
-rw-r--r-- 1 cse20189999 under    0 Sep 17 14:27 document_20189999.docx
drwxr-xr-x 2 cse20189999 under 4096 Sep 17 14:35 lib_hw1
-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
cse20189999@cspro9:~/20189999$
```

Way to Submit Your Work

- **Incorrect case (3)**
  - Source codes are in 'src' directory and libraries are in 'lib_hw1'

```
cse20189999@cspro9:~/20189999$ ls -al
total 20
drwxr-xr-x 4 cse20189999 under 4096 Sep 18 09:19 .
drwx------ 6 cse20189999 under 4096 Sep 18 00:48 ..
-rw-r--r-- 1 cse20189999 under    0 Sep 17 14:27 document_20189999.docx
drwxr-xr-x 2 cse20189999 under 4096 Sep 17 14:35 lib_hw1
-rw-r--r-- 1 cse20189999 under  428 Sep 17 14:32 Makefile
drwxr-xr-x 2 cse20189999 under 4096 Sep 18 09:19 src
cse20189999@cspro9:~/20189999$
```

Way to Submit Your Work

- How to use tar (Assume that your ID is 20189999)
  - **Compress: tar -zcvf os#0_2_20189999.tar.gz 20189999**
  - **Extract: tar -zxvf os#0_2_20189999.tar.gz**
  - Don't compress your directory in Windows, compress it in Linux (on CSPRO).
  - **20189999 directory, not os#0_2_20189999**, should be shown after extracting tar.gz file.
  - If you try 'tar -zcvf os#0_2_20189999.tar.gz os#0_2_20189999', you will get os#0_2_20189999 directory after extracting os#0_2_20189999.tar.gz ← **This is wrong case!**