# Project #0-2: Pintos Data Structure Analysis

[CSE4070]

Fall 2019

Hyeongu Kang

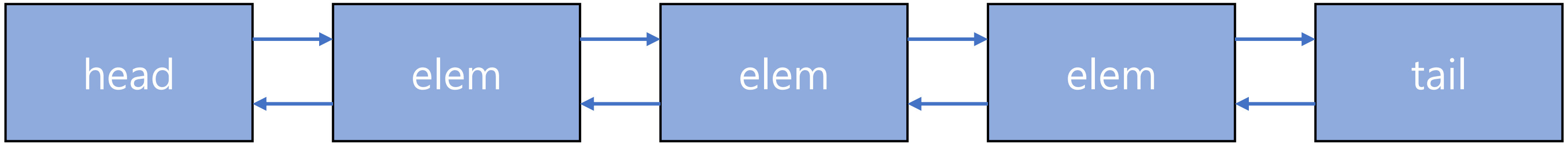
# Data Structures

# Data Structures in Pintos Kernel

- Before we dive into Pintos project, we will practice Pintos data structures
- 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**

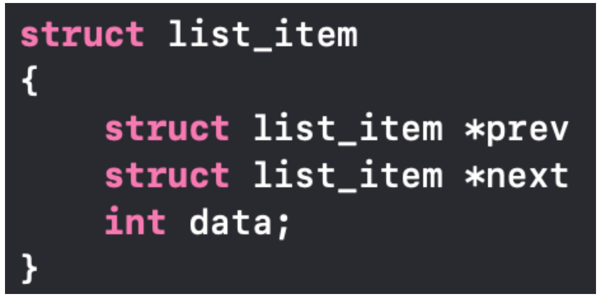
# List

- 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 of the list functions operate on `struct list_elem` not the list item

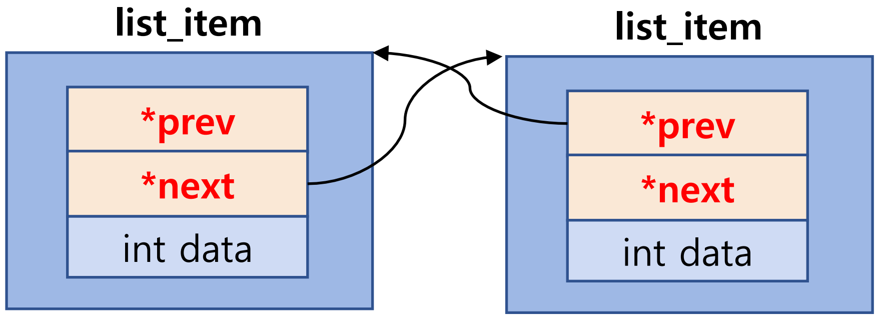


# List

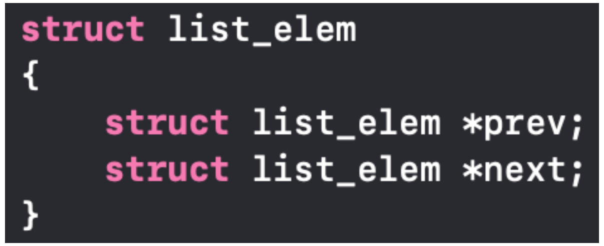
- Linked List: Usual way



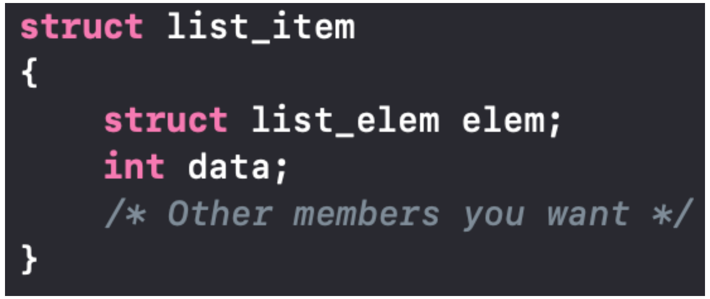
```
struct list_item
{
    struct list_item *prev
    struct list_item *next
    int data;
}
```



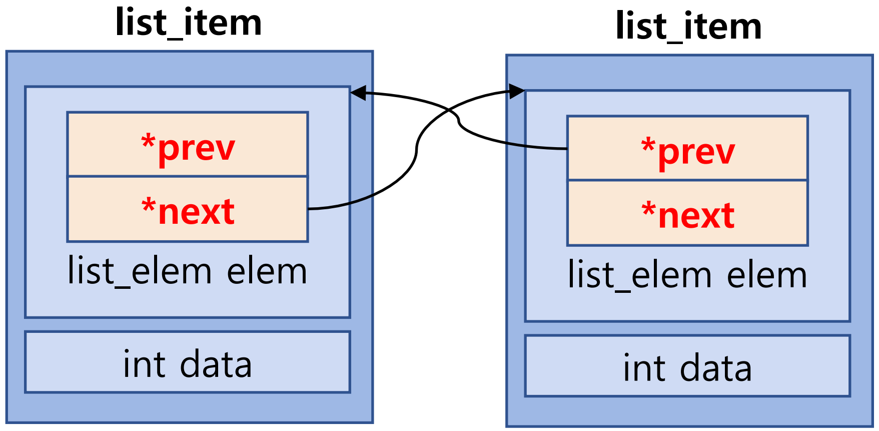
- Linked List: Pintos kernel



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



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



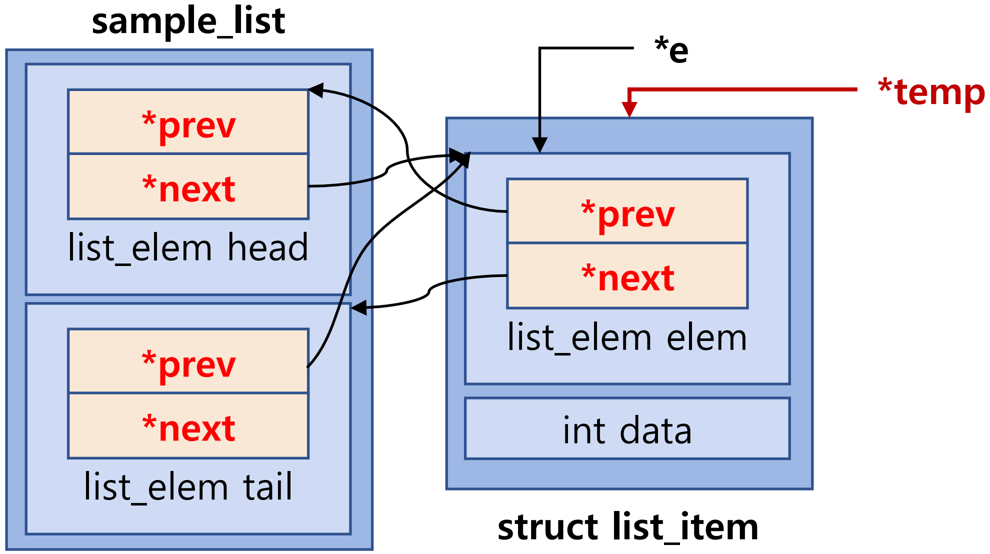
Split the pointer and data

# List Function Analysis

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

# List Function Analysis

- `struct list_elem* list_end(struct list *list)`
  - Return the last ELEM in LIST
  - Usually used to iterate LIST
- `#define list_entry(list_elem, struct, member)`
  - Converts 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

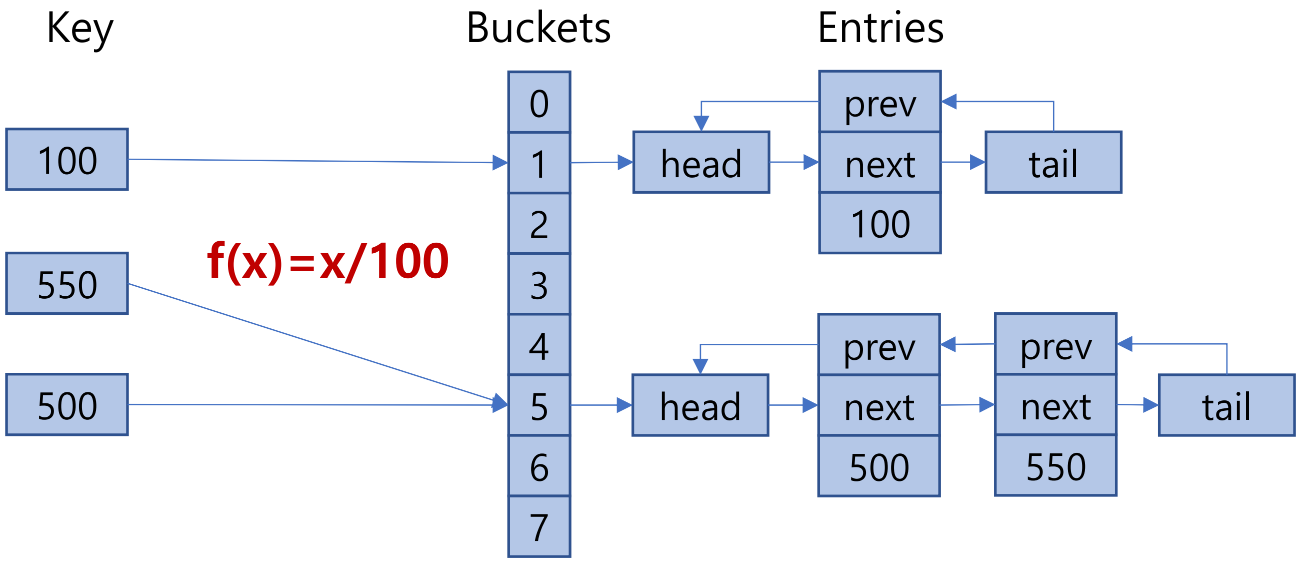


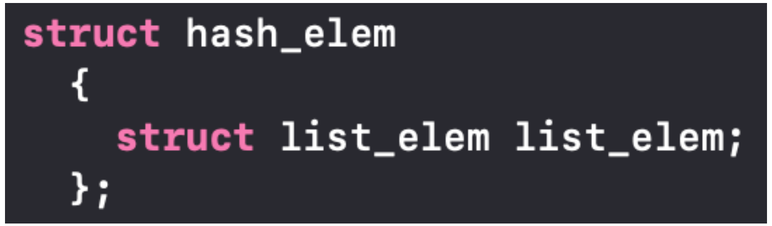
```
/* Assume that there already exists a list, sample_list */
struct list_elem *e;
e = list_begin (&sample_list);
struct list_item *temp = list_entry(e, struct list_item, elem)
int temp_data = temp->data
```

**By using list_elem, we can get address of list_item**

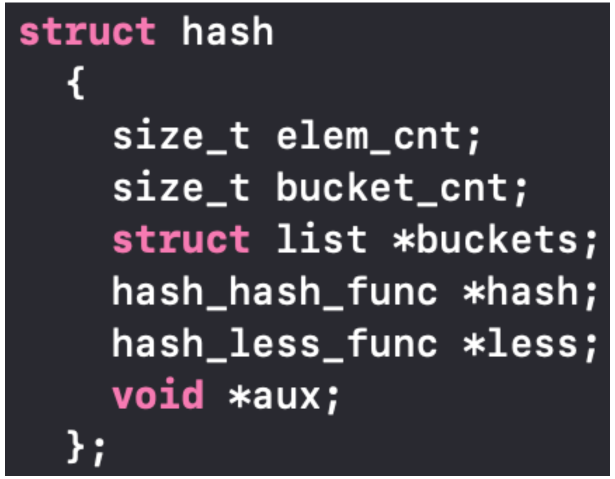
# Hash Table

- A hash table is a data structure that associates keys with values
- **We assume that key and value are same in this project**
- The primary operation it supports efficiently 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;
  };
```



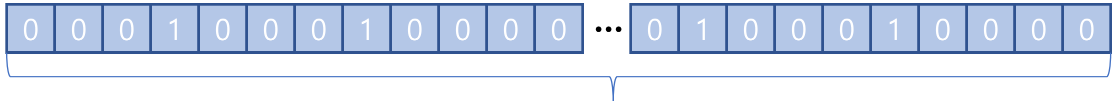
```
struct hash
  {
    size_t elem_cnt;
    size_t bucket_cnt;
    struct list *buckets;
    hash_hash_func *hash;
    hash_less_func *less;
    void *aux;
  };
```

# Hash Table Function Analysis

- `void hash_init(struct hash *h, hash_hash_func *hash, hash_less_func *less, void *aux)`
  - Initialize hash table H and set hash function HASH and comparison function LESS
  - You can see the example hash function such as `hash_bytes`, `hash_string` and `hash_int`
  - Comparison function LESS is used to compare two hash elements
  - You need to make your own hash function and comparison function
- `void hash_apply(struct hash *h, hash_action_func *action)`
  - You can apply any ACTION function which you make to hast table H
  - You can learn the usage of it from 'hash_apply.in' and 'hash_apply.out' in tester directory
- `#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

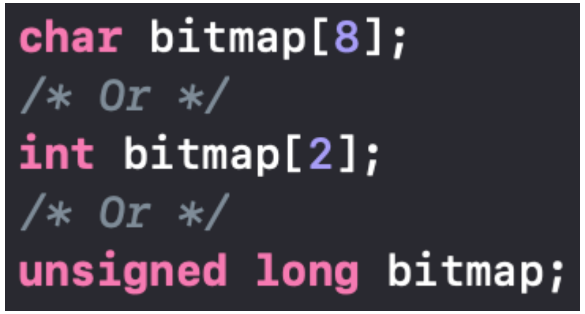
- A bit array(or bitmap, in some cases) is an array which compactly stores individual bits (Boolean values)
- A bitmap can reduce the waste of memory space



0 0 0 1 0 0 0 1 0 0 0 0 … 0 1 0 0 0 1 0 0 0 0

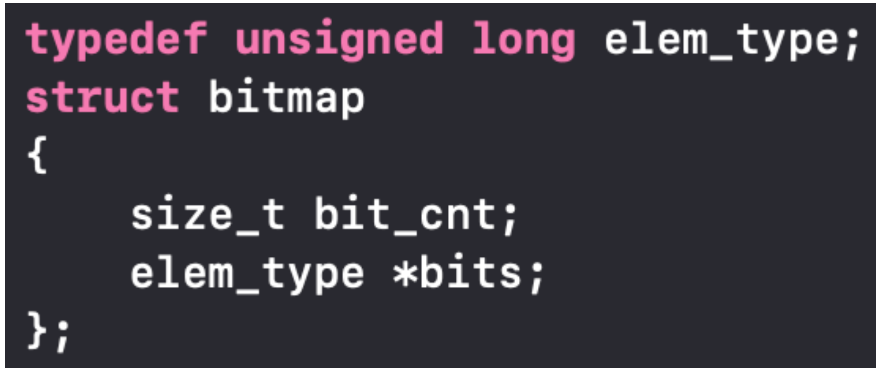
8 Bytes (64 bit)

- 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)`
  - Initialize a bitmap of BIT_CNT bits and sets all of 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

# Requirements

# Project #0-2

- **Write the interactive program that can check functionalities of list, hash table and bitmap in Pintos kernel**
- Assumptions
  - All inputs are from standard input (STDIN)
  - All inputs and outputs are lower cases
  - All the type used in the program is integer
  - Use `hash_int( )` as hash function for hash table
  - Print `true` or `false` when the return type is `Boolean`
  - The number of list, hash table and bitmap is less than 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 need to implement the following functions as well**

- 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

# Project #0-2: List of Functions to Implement

## You need to implement the following functions as well

※ You must not use this function in your code, just implement hash_int_2() in your code

※ Please use hash_int() as hash function to pass the test program

- Hash table

  1) `unsigned hash_int_2(int i)`

     ▪ Parameter : Integer that will be hashed

     ▪ Return value : Hash value of integer I

     ▪ Functionality : You can implement this function in your own way and describe what you implement 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)

# Project #0-2

- These are small part of commands used in interactive program
- **You should check the tester file (*.in) to see what commands are used for the test**
  - create list <LIST>
    - Create LIST
  - create hashtable <HASH_TABLE>
    - Create HASH_TABLE
  - create bitmap <BITMAP> <BIT_CNT>
    - Create BITMAP with the size of BIT_CNT
  - delete <LIST | HASH_TABLE | BITMAP>
    - Delete the given data structure
  - dumpdata <LIST | HASH_TABLE | BITMAP>
    - Print the given data structure to standard out (STDOUT)
  - quit
    - Terminate 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

# 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.ssh ../20179999/testlib)
  5) It will produce **.output** files which show you the output contents of each test, **.result** files which show you the result of each test and **Score.txt** will show you the total score

# 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 (files in lib_hw1 directory)
  ③ Source code you made
  ④ document (softcopy and hardcopy)
  → Explain briefly library functions which you used and functions which you wrote (functionality, parameter, return value)

- **Form and way to submit**
  1) Form of the file
     - document: **document.doc** or **document.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 20179999, makefile, provided libraries, source code you made and document file should be in '20179999' directory.
     - Compress the 'ID' directory into **'os#0_2_[ID].tar.gz'**
       ✓ You should use -zcf options for using tar
  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, main is not allowed)
     - You should 'make clean' before you compress the directory
     - **Submit Hardcopy to AS916 (You should submit softcopy and hardcopy both, if not 3% of point will be deducted)**
  3) Due date: 2019. 10. 6  23:59

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

❖ **Late submission is allowed up to 3 days (~10/9) and 10% of point will be deducted per day**

# Project Schedule

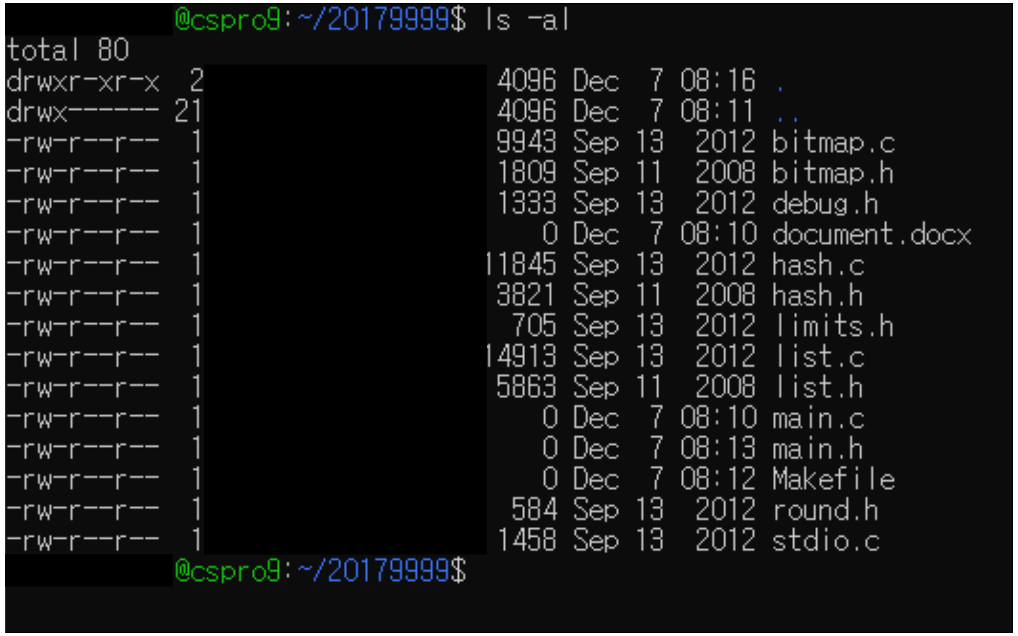
| Projects | Points | Contents | Periods | Lectures |
|---|---|---|---|---|
| Project 0-1 | 1 | Installing Pintos | 9/16 – 9/22 | Manual will be provided |
| **Project 0-2** | **3** | **Pintos Data Structures** | **9/21 – 10/6** | **9/21 (Sat.)** |
| Project 1 | 6 | User Programs (1) | 10/5 – 11/3 | 10/5 (Sat.) |
| Project 2 | 4 | User Programs (2) | 11/2 – 11/17 | 11/2 (Sat.) |
| Project 3 | 6 | Threads | 11/16 – 12/8 | 11/16 (Sat.) |

**※ Once you copy other's codes, you will get F grade**

# Appendix

# Way to Submit Your Work

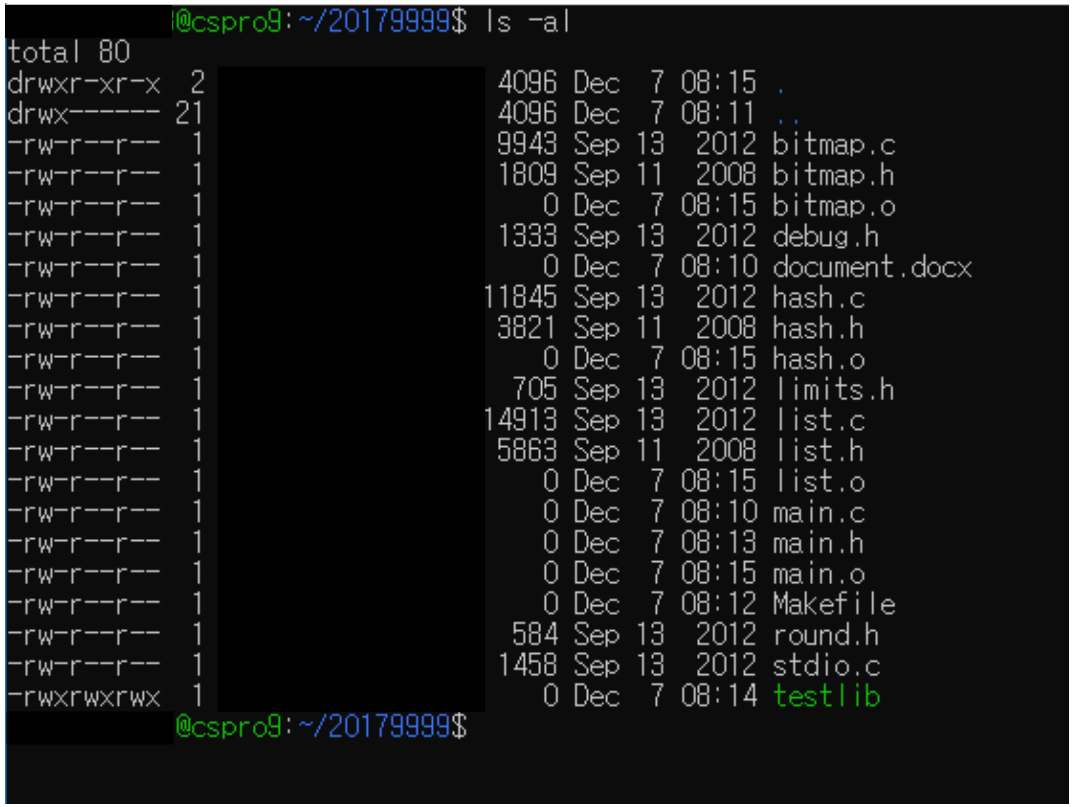
- Correct case
  - All the files should be in your 'ID' directory



```
[redacted]@cspro9:~/20179999$ ls -al
total 80
drwxr-xr-x  2 [redacted]  4096 Dec  7 08:16 .
drwx------ 21              4096 Dec  7 08:11 ..
-rw-r--r--  1              9943 Sep 13  2012 bitmap.c
-rw-r--r--  1              1809 Sep 11  2008 bitmap.h
-rw-r--r--  1              1333 Sep 13  2012 debug.h
-rw-r--r--  1                 0 Dec  7 08:10 document.docx
-rw-r--r--  1             11845 Sep 13  2012 hash.c
-rw-r--r--  1              3821 Sep 11  2008 hash.h
-rw-r--r--  1               705 Sep 13  2012 limits.h
-rw-r--r--  1             14913 Sep 13  2012 list.c
-rw-r--r--  1              5863 Sep 11  2008 list.h
-rw-r--r--  1                 0 Dec  7 08:10 main.c
-rw-r--r--  1                 0 Dec  7 08:13 main.h
-rw-r--r--  1                 0 Dec  7 08:12 Makefile
-rw-r--r--  1               584 Sep 13  2012 round.h
-rw-r--r--  1              1458 Sep 13  2012 stdio.c
[redacted]@cspro9:~/20179999$
```

# Way to Submit Your Work

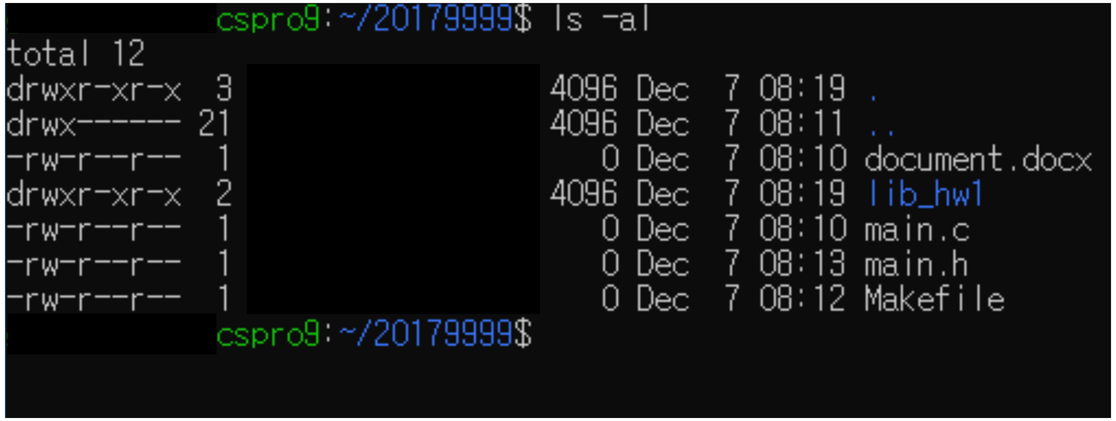
- Wrong case (1)
  - 'make clean' is not performed in this case (There are *.o and testlib files)



```
[redacted]@cspro9:~/20179999$ ls -al
total 80
drwxr-xr-x  2 [redacted]  4096 Dec  7 08:15 .
drwx------ 21             4096 Dec  7 08:11 ..
-rw-r--r--  1             9943 Sep 13  2012 bitmap.c
-rw-r--r--  1             1809 Sep 11  2008 bitmap.h
-rw-r--r--  1                0 Dec  7 08:15 bitmap.o
-rw-r--r--  1             1333 Sep 13  2012 debug.h
-rw-r--r--  1                0 Dec  7 08:10 document.docx
-rw-r--r--  1            11845 Sep 13  2012 hash.c
-rw-r--r--  1             3821 Sep 11  2008 hash.h
-rw-r--r--  1                0 Dec  7 08:15 hash.o
-rw-r--r--  1              705 Sep 13  2012 limits.h
-rw-r--r--  1            14913 Sep 13  2012 list.c
-rw-r--r--  1             5863 Sep 11  2008 list.h
-rw-r--r--  1                0 Dec  7 08:15 list.o
-rw-r--r--  1                0 Dec  7 08:10 main.c
-rw-r--r--  1                0 Dec  7 08:13 main.h
-rw-r--r--  1                0 Dec  7 08:15 main.o
-rw-r--r--  1                0 Dec  7 08:12 Makefile
-rw-r--r--  1              584 Sep 13  2012 round.h
-rw-r--r--  1             1458 Sep 13  2012 stdio.c
-rwxrwxrwx  1                0 Dec  7 08:14 testlib
[redacted]@cspro9:~/20179999$
```

# Way to Submit Your Work

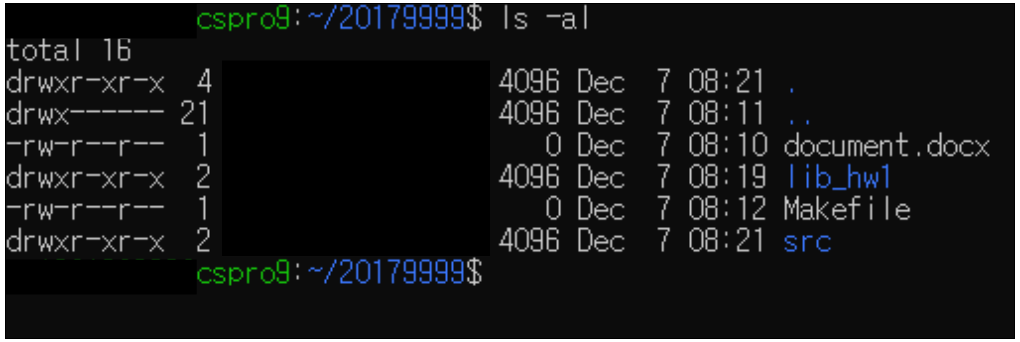
- Wrong case (2)
  - Library files are in 'lib_hw1' directory in this case
  - Don't contain libraries in other directory, just contain it in 'ID' directory



```
[redacted]cspro9:~/20179999$ ls -al
total 12
drwxr-xr-x  3 [redacted] 4096 Dec  7 08:19 .
drwx------ 21            4096 Dec  7 08:11 ..
-rw-r--r--  1               0 Dec  7 08:10 document.docx
drwxr-xr-x  2            4096 Dec  7 08:19 lib_hw1
-rw-r--r--  1               0 Dec  7 08:10 main.c
-rw-r--r--  1               0 Dec  7 08:13 main.h
-rw-r--r--  1               0 Dec  7 08:12 Makefile
[redacted]cspro9:~/20179999$
```

# Way to Submit Your Work

- Wrong case (3)
  - Source codes are in 'src' directory and also libraries are in 'lib_hw1'



```
[redacted]cspro9:~/20179999$ ls -al
total 16
drwxr-xr-x  4 [redacted] 4096 Dec  7 08:21 .
drwx------ 21            4096 Dec  7 08:11 ..
-rw-r--r--  1               0 Dec  7 08:10 document.docx
drwxr-xr-x  2            4096 Dec  7 08:19 lib_hw1
-rw-r--r--  1               0 Dec  7 08:12 Makefile
drwxr-xr-x  2            4096 Dec  7 08:21 src
[redacted]cspro9:~/20179999$
```

# Way to Submit Your Work

- How to use tar (Assume that your ID is 20179999)
  - **Compress: tar -zcvf os#0_2_20179999.tar.gz 20179999**
  - **Extract: tar -zxvf os#0_2_20179999.tar.gz**
  - Don't compress your directory in Windows, compress it in Linux
  - 20179999 directory, not os#0_2_20179999, should be shown after extracting tar.gz file
  - If you did 'tar -zcvf os#0_2_20179999.tar.gz os#0_2_20179999', you will get os#0_2_20179999 directory after extracting os#0_2_20179999.tar.gz ← **This is wrong case!**