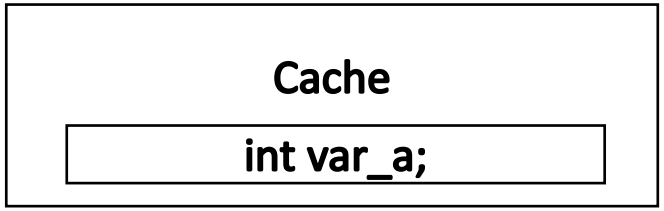
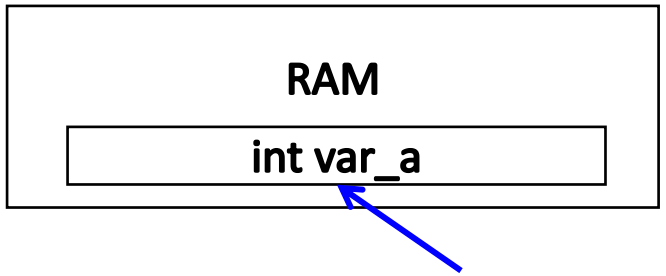
volatile

- Usage → volatile int var_a;
- Read data/variable from physical memory space instead of cache
- Tell the compiler do not do optimization on this variable



3. Your program is going to access the var_a
3.1 without volatile, fetch from cache → Error!
3.2 with volatile, fetch from RAM → correct

2. The var_a in cache is not updated accordingly



1. var_a changes because of interrupt

const

- Take a variable as a constant (cannot change its value)
- **const** int a = 30;
- **const** int a[5] = {1, 2, 3, 4, 5};
- **const** int *p = a (use as function argument)
  - p is a pointer that direct to a constant (you cannot change the content of a)
  - *p = 3 → Error !
- int * **const** p = a (use as function argument)
  - The location that p points to is not changeable
  - p = b → Error, p++ → Error
- **const** int * **const** p = a (use as function argument)
  - Have the characteristics of the above two

static/extern

- static
  - Static function → can not call by the procedure located in other file
    - static int swap (int *a, int *b);
  - Static variable
    - Declare inside a function → always exists
    - Declare outside a function → A global variable but cannot change by the procedure in other file
- extern
  - in 1.c → int var_a;
  - in 2.c → extern int var_ a;
  - You can use the same var_a variable in 2.c (1.c and 2.c share the common var_a)

extern

```c
#include <stdio.h>

extern int b;

int main (void)
{
   int a = 10;
   int *p;

   p = &a;
   printf ("*p= %d \n", *p);

   changeP(&p);
   printf ("*p= %d \n", *p);

   p = &b;
   b = 2000;
   printf ("*p= %d, p: %x \n", *p, p);
}
```

in change.c

```c
#include <stdio.h>

int b = 100;

void changeP (int **pp)
{
   *pp = &b;
   **pp = 1000;
   printf ("changeP: %x\n", &b);
}
```

```
*p= 10
changeP: 80495e4
*p= 1000
*p= 2000, p: 80495e4
```

union

```c
#include <stdio.h>

union StateMachine {
    char character;
    int number;
    char *str;
};
```

```c
int main(void) {

    union StateMachine machine;

    machine.number = 1;
    printf("sizeof: %d\n", sizeof(machine));
    printf("number: %d\n", machine.number);

    return 0;
}
```

http://caterpillar.onlyfun.net/Gossip/CGossip/union.html

union

```c
#include <stdio.h>

#defined NOT_SEL 0xFF
typedef unsigned char bool;

typedef struct stu {
    int ID;
    int mathScore;
    union
    {
        bool selected;
        int hisScore;
    } his;
} tStu;
```

```c
int main(void) {
    tStu stu;

    stu.ID = 1;
    stu.mathScore = 90;

    stu.his.hisScore = 20;
 // stu.his.selected = NOT_SEL;

    printf("sizeof: %d\n", sizeof(stu.his));
    printf(“selected: %d\n",
                  stu.his.selected); //the result?

    return 0;
}
```

- Union is commonly integrated in a structure
- The content of the union may be different types of internal signals in an OS

enum

http://caterpillar.onlyfun.net/Gossip/CGossip/enum.html

- Declaration
  - **enum Action {stop, sit, stand, walk, run};**
  - **enum Action {stop = 1, sit, stand, walk, run};**

  - **enum Action {stop = 1, sit, stand=2, walk, run};**
    - **sit and stand will be both 2**

- Usage
  - **enum Action action = stop;**

```c
#include <stdio.h>

typedef enum test
{
   #include "enumm.h"
   NUMBER
} testEnum;

int main (void)
{
    int a = 10;
    int *p;
    testEnum x = 5;

    if (x >= NUMBER)
        printf (“Larger than %d\n“, NUMBER);
    else
        printf (“OK! NUMBER: %d \n", NUMBER);
    return 0;
}
```

enumm.h

```c
stop = 0,
sit,
walk,
run,
stand,
```

W16-assignment

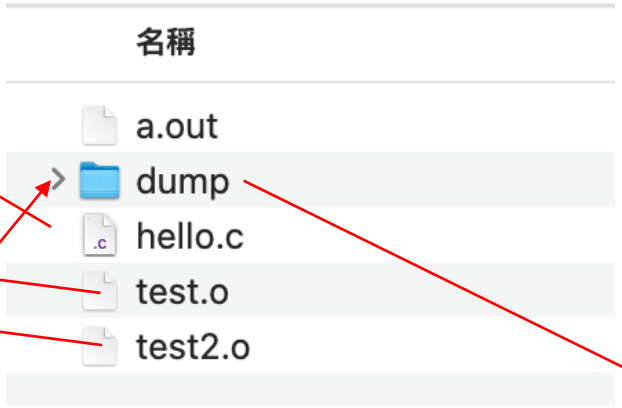
- Write a program to allow user to enter “Name” and “Phone number”
  - Store the name by char name[10]
  - Phone number may have two types: “home” or “cellular” (integrated by union)
    - For the home number, you should record the area code and number
    - For the cellular phone number, you should record operator’s name by enum (CHT, FET, TWN) and number
- The inputted information should be maintained by a linked list
- After finishing enter a new user, you should print all information

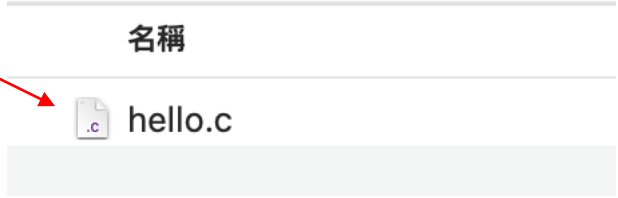
Final project
A customized command line and file system

```
ryanpan@Ryan-Mac-mini-M2 demo % ./a.out
options:
  1, loads from file
  2. create new partition in memory
2

Input size of a new partition (example 102400
2048000
partition size = 2048000

Make new partition successful !
List of commands
'ls' list directory
'cd' change directory
'rm' remove
'mkdir' make directory
'rmdir' remove directory
'put' put file into the space
'get' get file from the space
'cat' show content
'status' show status of the space
'help'
'exit and store img'

/ $
```

Safari

```
/ $ put hello.c
/ $ ls
hello.c
/ $ mkdir test
/ $ ls
test hello.c
/ $ cd test
/test/ $ ls

/test/ $ put test1.o
failed to open file 'test1.o'
/test/ $ put test.o
/test/ $ put test2.o
/test/ $ ls
test2.o test.o
/test/ $ cd ..
/ $ ls
test hello.c
/ $ status
partition size: 2048000
total inodes:   221
used inodes:    5
total blocks:   2000
used blocks:    52
files' blocks:  21
block size:     1024
free space:     1994752
/ $ cat hello.c
#include <stdio.h>

int main (void)
{
    printf("HELLO!!");
}/ $ get hello.c
```





```
List of commands
'ls' list directory
'cd' change directory
'rm' remove
'mkdir' make directory
'rmdir' remove directory
'put' put file into the space
'get' get file from the space
'cat' show content
'status' show status of the space
'help'
'exit and store img'
```

- a.out
- dump
- hello.c
- my_fs.dump
- test.o
- test2.o

```
ryanpan@Ryan-Mac-mini-M2 demo % ./a.out
options:
  1, loads from file
  2. create new partition in memory
```

Extra credits

- Security
  - Use password to protect the dump file
  - Encrypt the dump file
  - Decrypt when loading the dump file

- Create/Edit text files in your file system

How to deliver

- Record a YouTube video to
  - Demonstrate
  - Present your design and detailed flows
  - Your teamwork (e.g., student A was responsible for OO function)
  - Percentage of individual contribution (e.g., A → 20%, B→ 15%)
  - Video length should be more than 20 minutes

- Submit the source code and URL to the NTUT iSchool+
  - Submit by the group leader
- Deadline 2024/1/10 23:59