ECPS 203
Discussion - 2

TA: Zhongqi Cheng

Agenda

- Review of Assignment 1
    - text editor
    - function call tree generation
- Introduction to Assignment 2
    - task 1: change two lines of code
    - task 2: change C to C++
    - task 3,4: remove dynamic memory allocations
    - compile and submit

Assignment 1

- 100% successful in writing C code

- How to edit the code ?

Assignment 1

- Editing code:
  - **vi** command
  - windows user:
    - WinSCP
  - mac os user:
    - scp command to copy file from remote server

Assignment 1

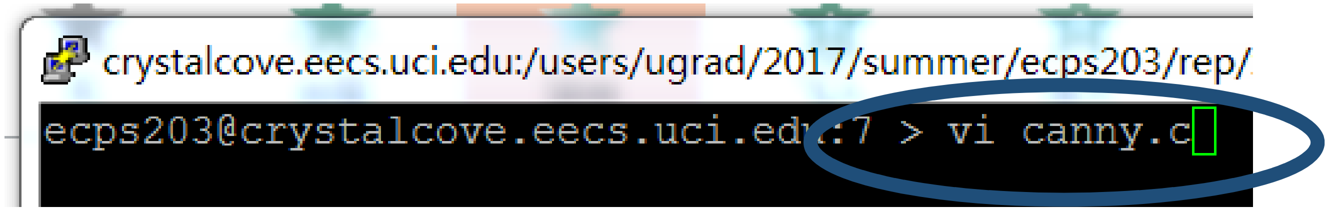
- vi
  - Build-in linux text editor
    - installed on the server
  - So you are using it remotely
    - and if your Internet connection is not stable, it will be slow

Assignment 1

- Open a file:

- for example:

  type **vi canny.c**, and press enter



crystalcove.eecs.uci.edu:/users/ugrad/2017/summer/ecps203/rep/

```
ecps203@crystalcove.eecs.uci.edu:7 > vi canny.c
```

Assignment 1

- Start editing:

Press A

Assignment 1

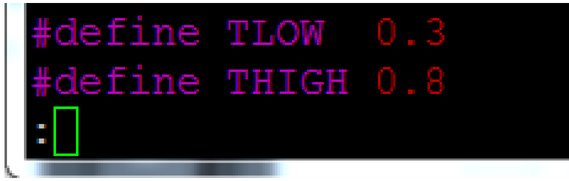
- quit vi

First, quit editing (press esc)

press

+

(enter the colon, :)



```
#define TLOW  0.3
#define THIGH 0.8
:
```

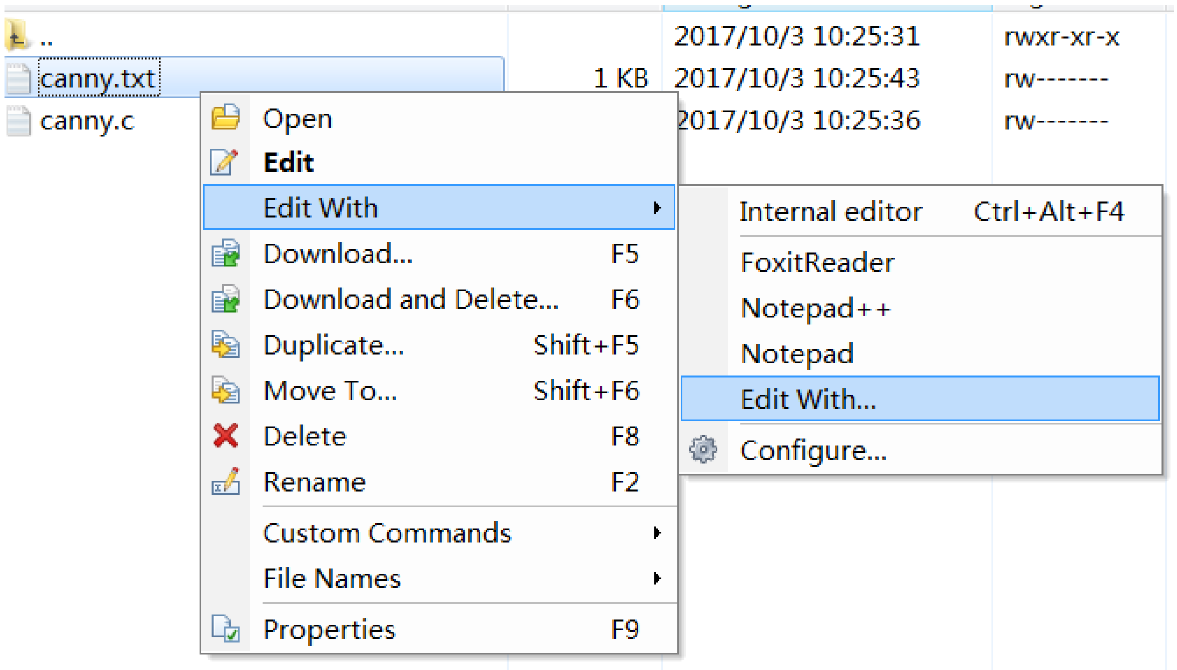
and type

1) q! -- quit without change
2) wq – save and quit

then press enter

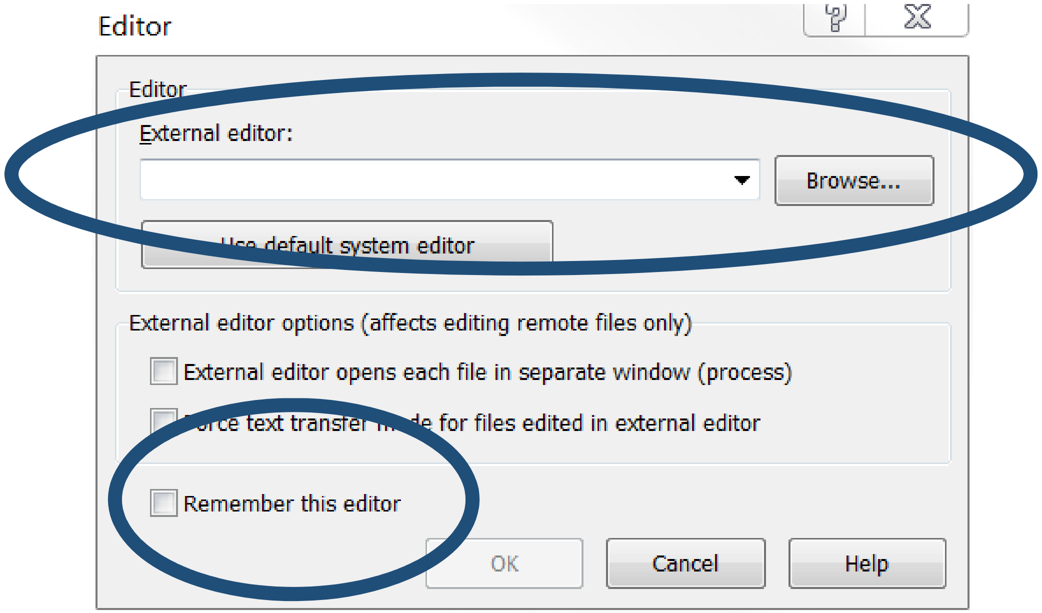
Assignment 1

- install WinSCP
- edit the file
  1) Right click the file
  2) edit with your favorite editor



Assignment 1

- install WinSCP
- edit the file
  1) Right click the file
  2) edit with your favorite editor



Editor

Editor

External editor:

Browse...

Use default system editor

External editor options (affects editing remote files only)

□ External editor opens each file in separate window (process)

□ Force text transfer mode for files edited in external editor

□ Remember this editor

OK Cancel Help

personally I recommend Notepad++

Assignment 1

- scp command : copying files

```
scp <source> <destination>
```

1) copy file from server to your computer

```
scp usr@bondi.eecs.uci.edu:~/project/hw1/canny.cpp .
```

(don’t forget the . in the end)

Assignment 1

- scp command

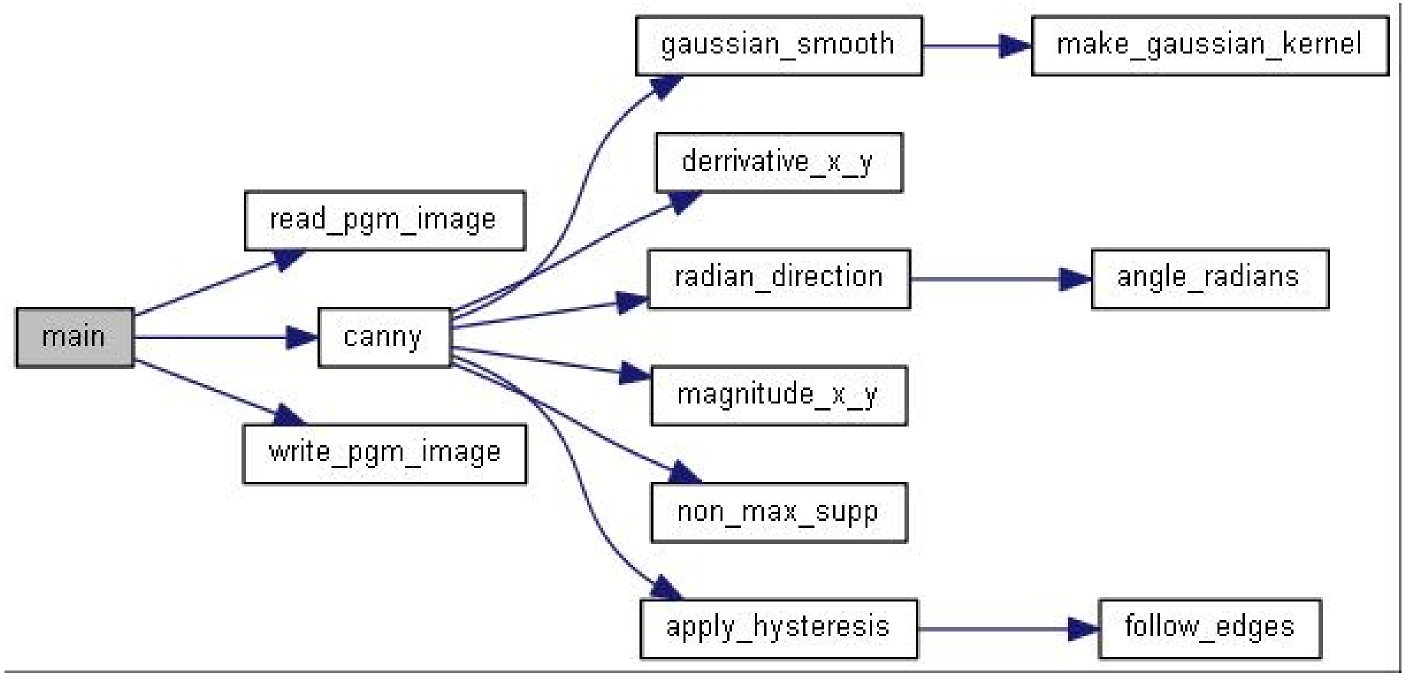
```
scp source destination
```

2) copy file from your computer to server

```
scp ./canny.cpp usr@bondi.eecs.uci.edu:~/project/hw1/canny.cpp
```

Assignment 1

- function call tree



Assignment 1

- **doxygen**, a free software on all platforms

  automatically calculates the relationship between functions

- **graphviz**, another free software

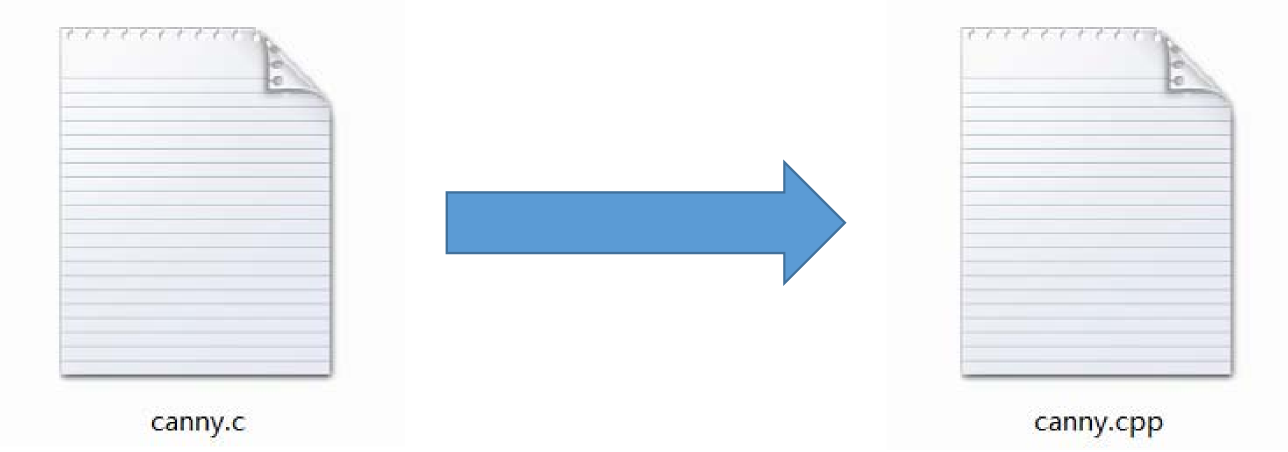
  doxygen uses it to draw diagrams

Assignment 2

due: Wednesday 18:00 next week

Assignment 2

- Rewrite your canny.c into canny.cpp
- SystemC model is in C++
- Convert the C program to C++
- Make some modifications



Assignment 2

- Task 1

  - Bug fix in non_max_sup(), about matrix manipulation
  - only two lines of code
  - Already fixed in the solution file for assignment 1
  - **~ecps203/public/canny.c**

  So, you can copy the reference file to your local directory, and build your canny.cpp based on this file

Assignment 2

- Task 2

Convert the C program into C++

Assignment 2

- Hint 1
- Add return type to function calls

✗

```
foo(){
        print("hello world\n");
}
```

Valid C code, non-valid C++ code

```
void foo(){
        print("hello world\n");
}
```

Valid C and C++ code

Assignment 2

- Hint 2
- Add function declarations

✗

```
void a(){
        foo();
}
void foo(){
        print("hello world\n");
}
```

Valid C code, non-valid C++ code

```
void foo();
void a(){
        foo();
}
void foo(){
        print("hello world\n");
}
```

Valid C code, non-valid C++ code

Assignment 2

- Task 3

Remove dynamic memory operations

malloc(), calloc(), free()

Assignment 2

- Why **NO** dynamic memory allocation?
  - In the future, we will do SystemC modeling
  - SystemC simulation is to imitate hardware
  - In hardware, dynamic memory allocation is not available.

Assignment 2

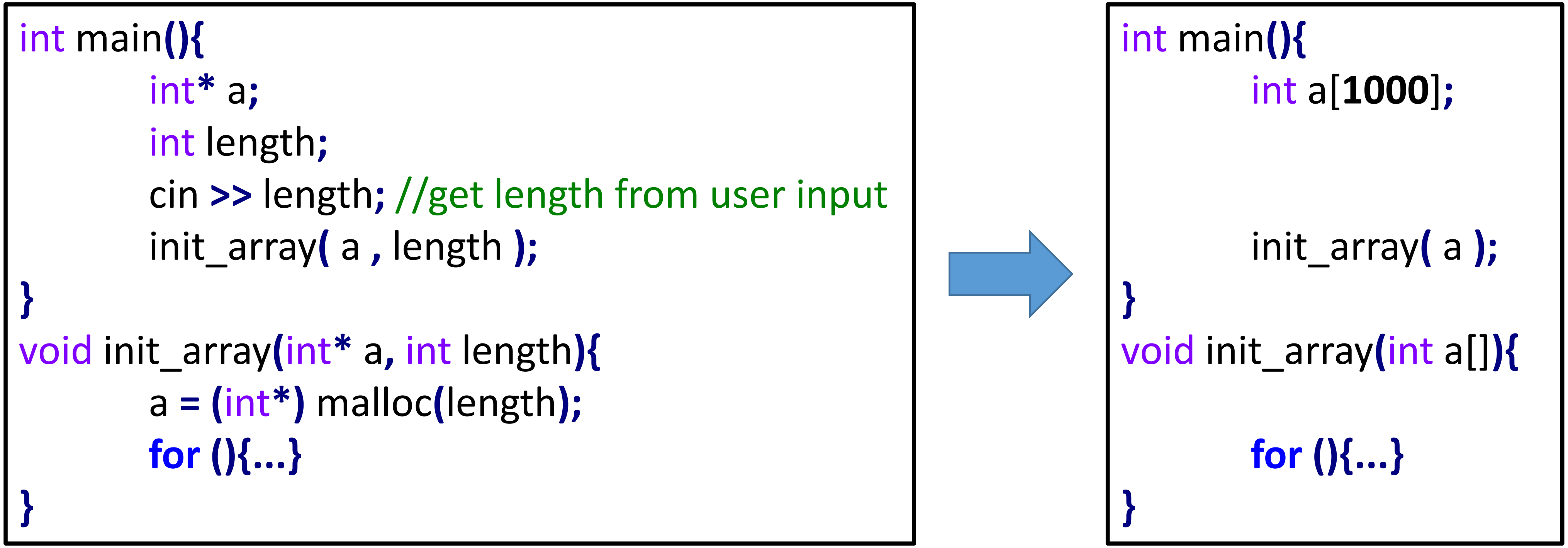
- what does malloc() do?
- for example, you want to create an array, whose length is determined by an input from keyboard

```
int main(){
        int* a;
        int length;
        cin >> length; //get length from user input
        init_array( a , length );
}
void init_array(int* a, int length){
        a = (int*) malloc(length);
        for (){initialize the array …}
}
```

create an array with dynamic length

Assignment 2

- hint: how to remove malloc
- need a fixed value of array length



```
int main(){
        int* a;
        int length;
        cin >> length; //get length from user input
        init_array( a , length );
}
void init_array(int* a, int length){
        a = (int*) malloc(length);
        for (){...}
}
```

```
int main(){
        int a[1000];


        init_array( a );
}
void init_array(int a[]){

        for (){...}
}
```

Assignment 2

- why we used malloc in canny edge detector?

- Because before reading the image, the algorithm does not know the size of it
- Only upon reading the image, the program knows the image size, and starts to create a corresponding matrix with enough size to store the image

Assignment 2

- why we don’t need malloc now?
- Because we now fix the size of input image to 240*320
- That is, the matrix size is now 240*320

Assignment 2

- task 4

Hard-code the parameters in your program

Assignment 2

- The parameters include:
  - rows = 240
  - cols = 320
  - sigma = 0.6
  - tlow = 0.3
  - thigh = 0.8

- in Assignment 1
  - rows and cols were read from file
  - sigma tlow thigh were user input

Assignment 2

- an example

```
int main(int argc, char *argv[]){
        //read tlow from command line in the terminal
        int tlow = atof(argv[1]);
        print(tlow);
}
```

before

```
#define TLOW 240
int main(int argc, char *argv[]){


        print(TLOW);
}
```

after

Assignment 2

- last task

compile your code

Assignment 2

- use g++ this time

-
  ```
  g++ canny.cpp -Wall -pedantic -O2 -o canny
  ```

show all warnings

show whether your code is in strict ANSI style

compiler will do some optimizations

Assignment 2

- submit canny.cpp canny.txt

- canny.txt: write anything you like in it, to show if you meet any difficulties
- It doesn’t matter if it’s empty, that simply implies that everything was successful