

# ECPS 203 Discussion Week2

TA: Emad Arasteh

[emalekza@uci.edu](mailto:emalekza@uci.edu)  
[ecps203@eecs.uci.edu](mailto:ecps203@eecs.uci.edu)

Office Hours: Fri, 10:00-11:00am

EH 3404 [Zoom 989 2181 4881](https://zoom.us/j/98921814881)

Center for Embedded and Cyber-Physical Systems

University of California, Irvine



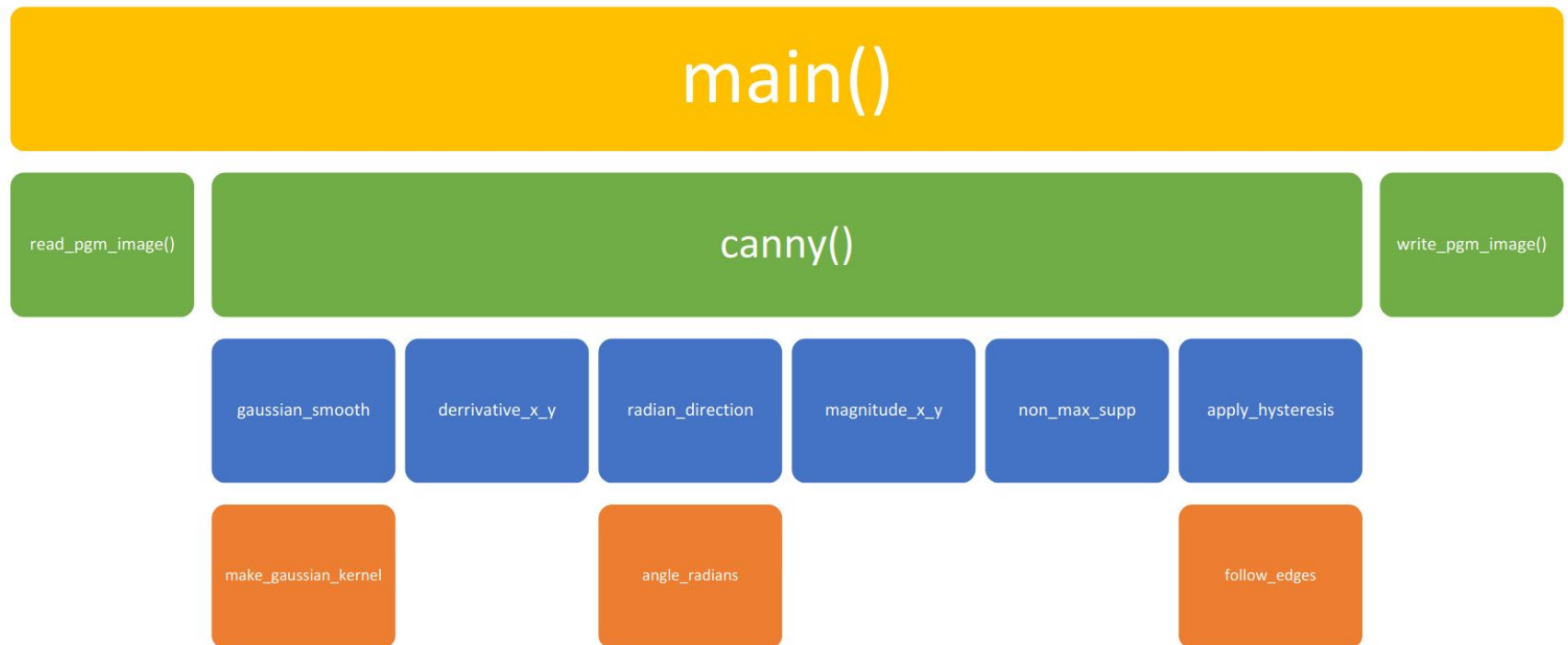
# Reminder

- Develop and test your codes on EECS servers
- Assignments are due on Wednesdays at 6pm PST
- Assignment 2 due date
  - **Wednesday, 13 October at 6pm**
- Hard deadline

# Outline

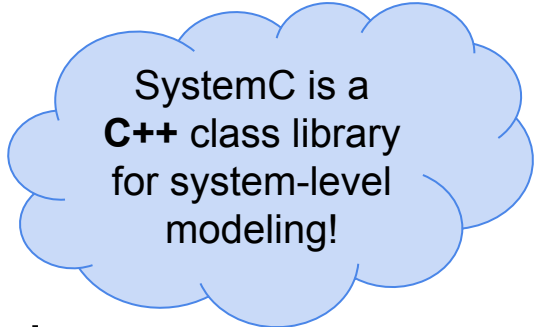
- Review assignment 1
- Assignment 2
- Hints
- Homework submission
- Questions

# Canny function call hierarchy



# Assignment 2

- **Create a clean C++ code with static memory allocation**
- Apply bug fix in the `non_max_supp` function
  - Copy the patched source code file from server
  - Inspect and compare with your own source code
- Clean-up source code
  - Compile using **g++**
  - Fix compiler errors/warnings
- Refine user-adjustable parameters to become hard-coded constants
- Remove or replace all dynamic memory allocations



SystemC is a  
**C++** class library  
for system-level  
modeling!

# Fix compiler warnings

- Compile **canny.cpp** with **-Wall -pedantic -O2** flags
  - `g++ canny.cpp -Wall -pedantic -O2 -o canny`
- **-Wall**

## **-Wall**

This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros. This also enables some language-specific warnings described in **C++ Dialect Options** and **Objective-C and Objective-C++ Dialect Options**.

- **-pedantic**

## **-pedantic**

Issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any **-std** option used.

- **-O2**

**-O2** Optimize even more. GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to **-O**, this option increases both compilation time and the performance of the generated code.

# C vs C++ function return type

- ANSI C (C89)
  - In C89, the default return type is **int**. This default behaviour was removed in C99 and compilers now warn you about this obsolete behavior
- ISO C++ (C++98)
  - C++ specification requires that functions must have explicit return type

C style

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

C++ style

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

# Excerpts from language standards

- ANSI C (C89)
  - “int, signed, signed int, or **no type specifiers** [...] designates the same type”
- ISO C++ (C++98)

## 7.1.5 (type specifiers)

### 4 **Change:** Banning implicit int

In C++ a *decl-specifier-seq* must contain a *type-specifier*. In the following example, the left-hand column presents valid C; the right-hand column presents equivalent C++:

<code>void f(const parm);</code>	<code>void f(const int parm);</code>
<code>const n = 3;</code>	<code>const int n = 3;</code>
<code>main()</code>	<code>int main()</code>
<code>/* ... */</code>	<code>/* ... */</code>

**Rationale:** In C++, implicit int creates several opportunities for ambiguity between expressions involving function-like casts and declarations. Explicit declaration is increasingly considered to be proper style. Liaison with WG14 (C) indicated support for (at least) deprecating implicit int in the next revision of C.

**Effect on original feature:** Deletion of semantically well-defined feature.

**Difficulty of converting:** Syntactic transformation. Could be automated.

**How widely used:** Common.

Reference: C89 draft and C++ ISO/IEC 14882 <http://port70.net/~nsz/c>



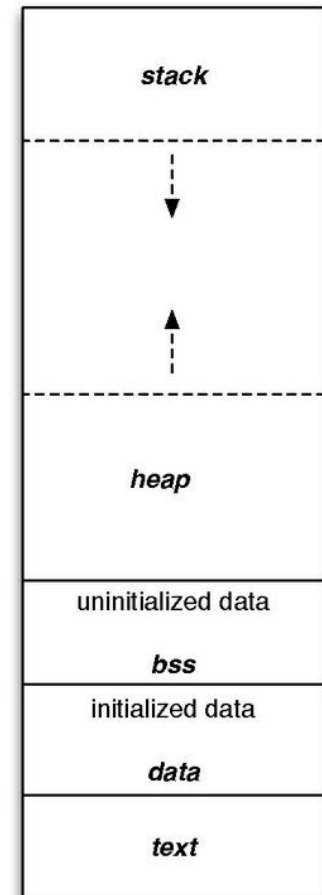
# Refine user-adjustable parameters

- In assignment 1, certain parameters are user-adjustable and passed as function arguments:
  - `canny (image, rows, cols, ...)`
- Refine following configurable parameters to hard-coded constants:
  - `rows = 240`
  - `cols = 320`
  - `sigma = 0.6`
  - `tlow = 0.3`
  - `thigh = 0.8`
- Remove command-line parsing functionality and hard-code all command-line arguments e.g.
  - `infilename = "golfcart.pgm"`

# Memory allocation in C/C++

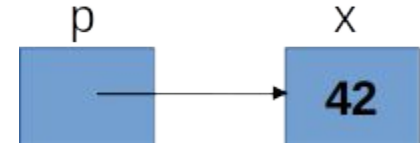
- The C/C++ programs supports following memory allocations:
  - Static
  - Automatic
  - Dynamic
- For **static-duration** and **automatic-duration** variables, the size of the allocation must be **compile-time** constant
- If the required size of data is not known until **run-time**, memory can be explicitly allocated from heap during program execution

```
void *malloc(size_t size);  
void free(void *ptr);
```



# Recap on pointers (optional)

- Pointers are variables whose values are addresses
  - The “address-of” operator (&) returns a pointer.



- Pointer Definition
  - The unary \* operator indicates a pointer type in a definition
- Pointer initialization or assignment
  - A pointer may be set to the “address-of” another variable
  - A pointer may be set to 0 (points to no object)
  - A pointer may be set to NULL (points to “NULL” object)
- Pointer Dereferencing
  - The unary \* operator dereferences a pointer to the value it points to (“content-of” operator)
  - The -> operator dereferences a pointer to a structure to the content of a structure member
- Pointer arithmetic

# Remove dynamic memory allocation

- Now, since all the user-adjustable parameters are hard-coded, we can use arrays with fixed sizes
- For example, replace `image` pointer declaration with a fixed array declaration:
  - `unsigned char *image;`
  - `-> unsigned char image[320*240];`
- And remove its related `malloc()` and `free()` codes:
  - `if((( *image) = (unsigned char *)`  
`malloc(( *rows) * ( *cols))) == NULL)`
  - ...

# Hints

- Beware of **segmentation faults** (core dumped)!
  - Seg faults are caused by accessing memory that “does not belong to you”:
    - dereference an uninitialized pointer
    - dereference a NULL pointer
    - dereference a pointer that has already been freed
    - access an out-of-bounds array index
    - ...
  - Inspect your code for suspicious memory accesses
  - Use **gdb** to debug nasty seg faults
- Befriend yourself with IEEE 1666!
  - Standard SystemC Language Reference Manual (LRM) is the ultimate resource for any ambiguity in SystemC code constructs

# Homework Submission

- Goto the **parent** directory of hw2
- Submit
  - **canny.cpp**
  - **canny.txt**
- To submit, type:
  - `~ecps203/bin/turnin.sh` (tilde key)
- To verify your submission, type:
  - `~ecps203/bin/listfiles.py`



# Questions?