ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 19

Vince Weaver

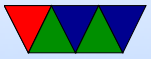
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

13 March 2026

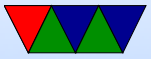
Announcements

- Still working on HW#5
- Stop by the lab after class and I'll have an Atari 2600 set up



struct padding

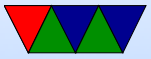
- We talked about this last time
- Most processors like "aligned" memory accesses, so if you load a 32-bit value it should be at an even 4-byte address in memory, etc
- If your struct has a mix of sizes the C compiler might pad things with empty space for better performance



forcing “packed” structs

- If you want to force structs to not have padding you can declare like this:

```
struct  __attribute__((packed)) pad4_t {
    int a;
    int b;
    char c;
    short e;
    int d;
}  pad4;
```



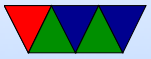
Viewing struct padding with pahole

- First compile your code with debug support (-g)
- Install pahole
- You can then view the padding on your structs

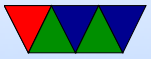
```
pahole -C pad1_t ./padding
```

- Show exampleg

```
pahole -C pad3_t ./padding
struct pad3_t {
  int         a;    /*     0     4 */
  int         b;    /*     4     4 */
  char        c;    /*     8     1 */
```



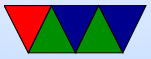
```
  /* XXX 1 byte hole, try to pack */

  short int   e;   /*    10     2 */
  int         d;   /*    12     4 */

  /* size: 16, cachelines: 1, members: 5 */
  /* sum members: 15, holes: 1, sum holes: 1 */
  /* last cacheline: 16 bytes */
};
```



goto

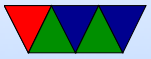
- Jump directly to another location in code
- Have a label, followed by a colon
- Can directly jump to that location

```
  /* this isn't a great example */
  fd=open();

  if (x>3) goto early_out;

  other_stuff();

early_out:
   close(fd);
```



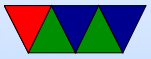
Structured Programming vs Spaghetti Code

- Structured Programming is when your code flows through regular blocks in an orderly fashion
- The alternative is having control flow jump all over the place (this is sometimes called "Spaghetti Code" because the code flow has threads all over the place like spaghetti noodles on a plate)



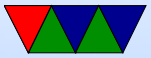
Structured Program Contents

- **Sequence** (statements in ordered sequence, inside of { } in C
- **Selection** (things like if/then/else) each flow ideally have one entry and one exit point
- **Iteration** loops. again ideally each would have one entry and one exit



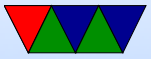
Structured Programming violations

- Most programming languages violate these things
- Early Return – as with C, can return from many points in a function. This is sometimes frowned on because you might miss cleanup that happens at the end of a function
- Early exit from block – `break` or `continue` in C
- Exception handling – (not in C)
- Multiple entry – (not really in C). Co-routines?



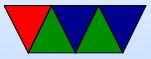
goto considered harmful

- 1968 letter by Edsger Dijkstra: *Go to statement considered harmful* Communications of the ACM 11, 3 (March 1968). 147-148.
- Became a "considered harmful" meme in CS
- " 'GOTO Considered Harmful' Considered Harmful" letter in Communications of the ACM (CACM) in March 1987
- famously BASIC full of spaghetti code, FORTRAN77 pretty bad too



Ways to Avoid

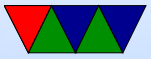
- `continue`
- `break`
- `return` from middle
- Some would argue these are essentially just GOTOs in another form



Where GOTOs used – cleanup

- Linux at least uses them at cleanup, to free resources in the reverse order they were allocated.
- If an error happens or you need to exit early you can goto the right place to only free the resources necessary
- This centralizes the cleanup in one place so if you add more or change the ordering you only have to do it in one location, rather than tracking down every place the function is returned from

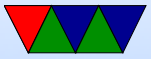
```
fd=open();
if (fd<0) goto error1;  // if fail to open
```



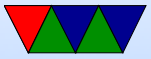
```
  ptr=malloc();
  if (ptr==NULL) goto error2; // if fail to alloc
  // ...
  free(ptr);
error2:
  close(fd);
error1:
  return 0
}
```

- Alternative is having early exit each time, but you have to keep in sync what resources need to be freed and it's easy to miss one

```
  fd=open();
  if (fd<0) {
```



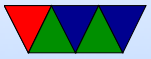
```
    return 0;
  }

  ptr=malloc();

  if (ptr==NULL) {
      close(fd);
      return 0;
  }

  free(ptr);
  close(fd);

  return 0
}
```



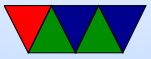
Where GOTOs used – emergency exit

- This is the only one my textbook allowed, exiting from deep nested loops

```
while(x) {
  while(y) {
    while(z) {
      while(a) {
          if (some_emergency) goto emergency_exit;
          // note, alternative is setting a flag and havi
          // at every level checking this
      }
    }
  }
}
```

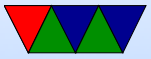


```
emergency_exit:
```



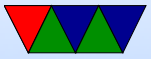
IOCCC

- Even the obfuscated C code competition discourages GOTOs



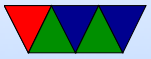
C goto rules

- Target must be in same function or block
  (Why? Big problem is if you skip stack cleanup on exit of block then the stack gets out of sync)
-



What if you really want to jump anywhere

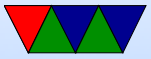
- assembly language
- or maybe BASIC



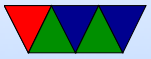
Advanced – computed goto

- This is a gcc extension, not in standard C
- Can build a jump table like in assembly
- Allows having an array of labels and jumping to one with array indexing
- Can be faster than a switch statement

```
void *label_ptr;

foo:        // label

label_ptr = &&foo;
```

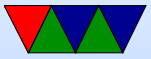


```
goto *label_ptr;
```



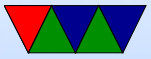
GOTO Example

- Show some BASIC of FORTRAN77 code



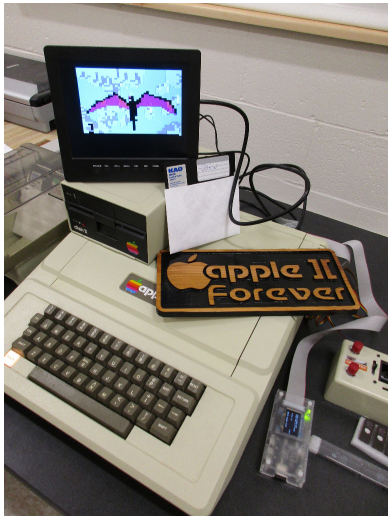
Advanced: GOTO jokes

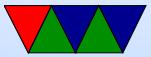
- The whole structured programming debate was a *huge* deal back in the day
- People proposed as a joke instead having `COMEFROM` statements rather than `GOTO`
- INTERCAL language implemented this?



Apple II Computer

- Originally from 1977
- Believe it or not there are C compilers for it





Apple II Examples

- Showed off a simple BASIC program with GOTO
- Showed off the source code to a "Portal" joke demake written in BASIC, with lots of spaghetti code `http://www.deater.net/weave/vmwprod/portal/`
- Showed off some of the AppleII bot entries