

Goto, Exceptions, and Assembly in C

CS 2022: Introduction to C

Instructor: Hussam Abu-Libdeh

Cornell University
(based on slides by Saikat Guha)

Fall 2009, Lecture 12

Switch Statement

- ▶ N-way `if` ($N > 2$), but equality check only
- ▶ Only integers
 - ▶ But then many things in C are glorified integers
 - ▶ Notably, enums

Switch Statement

```
enum days {Sun, Mon, Tue, Wed, Thu, Fri, Sat};  
...  
enum days day = ...;  
switch (day) {  
    case Sat:  
        ...  
        break;  
  
    case Sun:  
        ...  
        break;  
  
    case Mon:  
        printf("Sounds like someone has a case of the Mondays.\n");  
    case Wed:  
    case Fri:  
        ...  
        break;  
  
    default:  
        ...  
  
}
```

Goto

- ▶ Unstructured control flow
 - ▶ (unlike `if`, `switch`, `for` etc.)
- ▶ Evil
- ▶ Except when it's not
- ▶ Especially, when it is the cleanest

Goto

```
...  
goto foo;  
...  
  
foo:  
...
```

Goto

Extremely useful for

- ▶ breaking out of deeply nested loops
- ▶ handling errors and exceptions
 - ▶ by writing code that cleans up resources in reverse order of allocation
 - ▶ and jumping to the correct position in the list if allocation fails at some point

Exceptions (kinda)

- ▶ To break out of a deep call stack quickly
- ▶ Think goto breaking out of deep loops, but applied to function calls

Exceptions (kinda)

- ▶ To break out of a deep call stack quickly
- ▶ Think goto breaking out of deep loops, but applied to function calls
- ▶ setjmp and longjmp

Inline Assembly

- ▶ For when no C statement exists for the task
- ▶ Or when the compiler isn't generating the assembly you want

```
asm("...assembly code..."); // Basic form
```

```
asm("code" : output); // Assembly -> C
```

```
asm("code" : ... : input); // C -> C
```

For more info check out:

<http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html>