18-642:
Global Variables Are Evil!

9/11/2017

Carnegie Mellon University

Global Variables Are Evil!

Edge Case Research

- **Anti-Patterns:**
  - More than a few read/write globals
  - Globals shared between tasks/threads
  - Variables have larger scope than needed

- **Global variables are visible everywhere:**
  - Use of globals indicates poor modularity
    - Globals are prone to tricky bugs and race conditions
  - Local static variables are best if you need persistence
    - File static variables can be OK if used properly
    - Don't make procedures globally visible if not needed

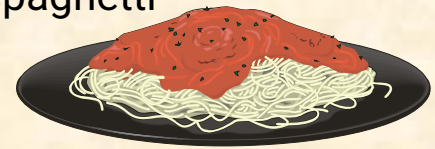
Global vs. Static Variables

Edge Case Research

- **Globals:**

```
uint32_t gVar = 0;
void gProc(…)  { … }
```

- **Global risks**
  - Written from anywhere
    - Debugging: who wrote it?
  - Read from anywhere
    - Changes break everything
  - Multithreaded race conditions
  - Increased complexity
    - Data flow “spaghetti”



- **File Static:**

```
static uint32_t fsVar = 0;
static void fsProc(…)  { … }
```

  - Only inside .c file
  - Use with small .c files
  - Like C++ “private”

- **Local Static:**

```
void gProc(…)
{ static uint32_t sVar = 0;
… }
```

  - Persistent variable value
  - Can’t be seen outside procedure

Avoiding And Removing Globals

Edge Case Research

- **Define smallest scope possible (variables and procedures)**
  - Change global to file static; file static to local static
- **Arrange .c files based on access to data**
  - Example: time of day updated by ISR
    - File static time of day variable in TimeOfDay.c
    - Put timer tick ISR in TimeOfDay.c
    - Put procedure to disable interrupts & read time of day in TimeOfDay.c

- **Configuration values & constants**
  - Use const keyword – prevents multiple writers
  - Read-only access to global configuration data structure
  - Limit visibility to need-to-know within relevant .h file

Best Practices For Avoiding Globals

Edge Case Research

- **Use smallest practical scope for variables & procedures**
  - Ideally, zero global variables
  - Use file static if you must; local static if you can
  - A good compiler will generate efficient code

- **Reorganize code to reduce scope**
  - Write anything except locking variables only in one place
  - File static variables for small groups of functions
    - More or less the idea of C++ private keyword
    - Take care of data locking when reading
- **Global Variable Pitfalls**
  - Lots of global variables is a sign of bad code