

W UNIVERSITY of WASHINGTON L06: CPP Wrap-Up, Linking, File I/O CSE333, Summer 2020

## Poll Everywhere

[pollev.com/cse33320su](http://pollev.com/cse33320su)

- What will happen when we try to compile and run?

```
bash$ gcc -Wall -DFOO -DBAR -o condcomp condcomp.c
bash$ ./condcomp
```

A. Output "333"  
B. Output "334"  
C. Compiler message about EVEN  
D. Compiler message about BAZ  
E. We're lost...

```
#include <stdio.h>
#ifndef FOO
#define EVEN(x) !(x%2)
#endif
#ifndef DBAR
#define BAZ 333
#endif

int main(int argc, char** argv) {
    int i = EVEN(42) + BAZ;
    printf("%d\n", i);
    return 0;
}
```

10

10

W UNIVERSITY of WASHINGTON L06: CPP Wrap-Up, Linking, File I/O CSE333, Summer 2020

## Linkage Issues

STYLE TIP

- Every global (variables and functions) is `extern` by default
  - Unless you add the `static` specifier, if some other module uses the same name, you'll end up with a collision!
    - Best case: compiler (or linker) error ☺
    - Worst case: stomp all over each other
- It's good practice to:
  - Use `static` to "defend" your globals
    - Hide your private stuff!
    - This can include both private variables and private "helper" functions
  - Place external declarations in a module's header file
    - Header is the public specification

This is done in ex5, and is something you should do in the HW's

17

17

W UNIVERSITY of WASHINGTON L06: CPP Wrap-Up, Linking, File I/O CSE333, Summer 2020

## C Stream Functions (1 of 2)

STYLE TIP

- Some stream functions (complete list in `stdio.h`):
  - `FILE* fopen(filename, mode);`
    - Opens a stream to the specified file in specified file access mode
  - `int fclose(stream);`
    - Closes the specified stream (and file).
  - `int fprintf(stream, format, ...);`
    - Writes a formatted C string
    - `- printf(...);` is equivalent to `fprintf(stdout, ...);`
  - `int fscanf(stream, format, ...);`
    - Reads data and stores data matching the format string

NULL on error, check for this!

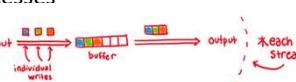
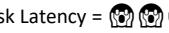
Should always close a file when done

22

22

W UNIVERSITY of WASHINGTON L06: CPP Wrap-Up, Linking, File I/O CSE333, Summer 2020

## Why Buffer?

- Performance – avoid disk accesses
  - Group many small writes into a single larger write
- Disk Latency =  (Jeff Dean from LADIS '09)
  - Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns
- Convenience – nicer API
  - We'll compare C's `fread()` with POSIX's `read()`

It takes a really long time to go all the way to disk!!!

32

32