## Use a File-Scope Variable for the Log's Stream

**Where is the stream for the log?**

Let's put all of the log functionality in a file.

The **stream** can be **a file-scope variable**:

```
FILE* logfile = NULL;
```

Let's **initialize the stream in the first call** to **printlog**.

We should have a function to close the stream, too, but we won't write that function.

  

## First Task: Open the Log File

Now we can start to write…

```
int printlog (const char* fmt, ...)
{
  if (NULL == logfile) {
    logfile = fopen ("the_log","a");
    if (NULL == logfile) {
      return -1;
    }
  }
```

First call?

Append to end of existing file.

Nowhere to write if **fopen** fails.

Return negative value to indicate failure.

  

## Are You Ready to Rewrite **printf**?

**Now what?**

**Rewrite printf?**

**That part is left to you.**

**But there's an easier way…**

**vfprintf**

(The "v" is for variadic/variable arguments.)

  

## Declare and Initialize a Variable Arguments List Variable

```
int printlog (const char* fmt, ...);
```

**To use the arguments** after fmt,
- we must **declare and initialize**
- **a "variable argument list" variable**.
- The type is **va_list**:

```
va_list args;
va_start (args, fmt);
```

**va_start** is a preprocessor macro that starts the variable-length list after the specified argument—in this case, **fmt**.

  