varargs

Mentioned elsewhere, how Apple and Linux handle variadic functions are quite different. In this chapter we will explore how variadic functions work by looking at an example in C++. Then, we'll review the differences between Apple and Linux.

How Variadic Functions Determine Number of Parameters

The TL;DR is that something among the *initial* arguments tells the function how many other arguments to expect. Commonly, it is the very first argument that contains this information but it is possible to encode this information in a parameter other than the first as long as no optional parameter precedes it.

For example, take our friend and yours: `printf`. The first argument is a template or formatting string that may contain zero or more instances of placeholders. For example:

Sample	Expected Arguments
`printf("Foo");`	None
`printf("%%d");`	None - notice the double %
`printf("%d %ld");`	An int and a long
`printf("%p %f");`	A pointer and a double

The `printf` function figures out how many arguments to expect and what types to expect from the first parameter.

Writing Variadic Functions in C or C++

You'll need to include `cstdarg` in C++ or `stdarg.h` in C++ or C.

Then, you'll use the **macros** `va_start()` and `va_end()` plus a variable of type `va_list`.

- `va_start()` is used to initialize the `va_list` variable with the address of the last required argument. This enables the variadic support functions to find the variadic parameters on the stack or in registers.
- `va_arg()`, another macro fetches the next variadic argument. The type of the argument is given as a parameter to `va_arg()`.
- `va_end()` is used to clean up the internal data structure allocated for you, behind your back.

Take a look at `Simple()` in this source code.

For a more sophisticated example, look at `LessSimple()` in the same file for something closer to `printf()`.

The above referenced source code contains ample commenting to explain what it is doing.

Differences Between Apple and Linux

Apple and Linux differ in how they implement parameter passing to variadic functions in assembly language. This is explained in more detail in the chapter on Apple Silicon.

Linux

Linux uses the first 8 scratch registers as normal. If you need to specify more than 8 arguments, the ninth and beyond go on the stack. It is up to you to determine how many arguments to expect (which you must do anyway) to determine where to find them.

Apple

Apple puts the first argument in `x0` as usual but all remaining arguments go on the stack in right to left order. There are some other restrictions - it is best to refer to this cryptically written document.