Variable Length Argument Lists

- Functions that take a ***variable number*** of arguments

  `int sum(int x, ...) { … }`

  there must be at least one named formal parameter

- **`stdarg.h`** defines types, macros, and functions to deal with variable arguments

  `va_list` the type of a “pointer to a variable argument list”

  `va_start(va_list ap,` ***last*****`)`**
  initializes **`ap`** for the list that follows the formal parameter ***last***
  `va_start` must be called before any calls to **`va_arg`**

  type **`va_arg(ap,`** ***type*****`)`**
  returns the next argument, which must have datatype ***type***
  adjusts **`ap`** to point the next variable argument

  `void va_end(va_list ap)`
  called after all of the arguments have been processed

- ***No way*** to determine the ***number*** of actual arguments

  the necessary information must be encoded in the values of the arguments

  e.g., **`printf`** format codes

Example: Macros for Variable Length Argument Lists

- **`int sum(int x, ...)`** returns the sum of its non-zero integer arguments

`sum(0)`	**0**
`sum(1, 2, 3, 0)`	**6**
`sum(-1, 2, 3, -4, 0)`	**0**

- Implementation

```
#include <stdarg.h>

int sum(int x, ...) {
    va_list ap;
    int n = 0;

    va_start(ap, x);
    while (x) {
        n += x;
        x = va_arg(ap, int);
    }
    va_end(ap);
    return n;
}
```

printf-Style Functions

- Use **`stdarg.h`** to implement special-purpose **`printf`**-style functions

```
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>

void error(char *fmt, ...) {
    va_list ap;

    va_start(ap, fmt);
    fprintf(stderr, "error: ");
    for ( ; *fmt; fmt++)
        if (*fmt == '%')
            switch (*++fmt) {
            case 'd': fprintf(stderr, "%d", va_arg(ap, int)); break;
            case 's': fprintf(stderr, "%s", va_arg(ap, char *)); break;
            ...
            case 'N': fprintf(stderr, "at line %d", lineno); break;
            default:  fputc(*fmt, stderr); break;
            }
        else
            fputc(*fmt, stderr);
    va_end(ap);
    fprintf(stderr, "\n");
    exit(EXIT_FAILURE);
}
```


Library printf-Style Functions

- Some standard library functions take **`va_list`** values

```
#include <assert.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void error(char *fmt, ...) {
    va_list ap;
    va_start(ap, fmt);
    fprintf(stderr, "error: ");
    vfprintf(stderr, fmt, ap);
    va_end(ap);
    fprintf(stderr, "\n");
    exit(EXIT_FAILURE);
}

char *stringf(char *fmt, ...) {
    va_list ap;
    char *s, buf[512];
    va_start(ap, fmt);
    vsprintf(buf, fmt, ap);
    va_end(ap);
    s = malloc(strlen(buf) + 1);
    assert(s);
    return strcpy(s, buf);
}
```