filename: c-cleanup-5pp-20251222.txt
https://twdev.blog/2025/12/ccleanup/

golang's defer in C (kind of)
2025-12-13

Quite often I find myself landing on gcc's attributes documentation page. Not sure how that happens but this is always a gold mine. I've recently discovered __attribute__((cleanup(func))), which allows attaching functions to variables. These functions are executed when the variable goes out of scope. Sounds familiar?

Missing piece of a puzzle
__attribute__((cleanup(func))) is the building block that C is missing. It's perfect for simple resource management and an answer to implement things akin to [12]Boost.ScopeExit are equivalent of C++'s RAII. I'm gonna cut straight to the chase.

```
void cleanup_int(void *p) {
    const int *i = (const int *)p;
    printf("Cleaning %d\n", *i);
}

void foo() {
    __attribute__((cleanup(cleanup_int))) int j = 123;
    __attribute__((cleanup(cleanup_int))) int k = 456;
}

/*
 * Produces:
 *
 * Cleaning 456
 * Cleaning 123
 */
```

The next obvious step is to dress this up with some helpful macros:

```
static void cleanup_free(void *p) {
    void *mem = *(void **)p;
    free(mem);
}

static void cleanup_close(void *p) {
    int fd = *(int *)p;
    close(fd);
}

static void cleanup_fclose(void *p) {
    FILE *fh = *(FILE **)p;
    fclose(fh);
}

#define CLEANUP(func) __attribute__((cleanup(func)))

#define AUTOFREE CLEANUP(cleanup_free)
#define AUTOCLOSE CLEANUP(cleanup_close)
#define AUTOFCLOSE CLEANUP(cleanup_fclose)


int main(int argc, const char *argv[]) {
    AUTOFREE void *p = malloc(123);
    AUTOCLOSE int fd = open("/etc/fstab", O_RDONLY);
    AUTOFCLOSE FILE *fh = fopen("/etc/fstab", "r");
    return 0;
}
```

Of course, this is not portable but it's good to be aware such feature exist.
