

Besprechung der `sister`
Tafelübung Systemprogrammierung

Andreas Schärtl <andreas.schaertl@fau.de>

WS 2020

Typischer Fehler: Stringformat

- `strcat(3)`, und `strcpy(3)` funktionieren. Aber es ist schwer zu verstehen, was da wie formatiert wird.
- Stattdessen: Besser `sprintf(3)` verwenden.

```
const char *wwwpath = ...;  
const char *relpath = ...;
```

```
char full_path[strlen(wwwpath) + 1 + strlen(relpath) + 1];  
sprintf(full_path, "%s/%s", wwwpath, relpath);
```

Typischer Fehler: Race Conditions

- Beispiel: Prüfen, ob Datei bei `pathname` eine reguläre Datei ist. Nur dann öffnen.
- Race Condition zwischen `stat(2)` und `fopen(3)`!

Besser:

- Eigentlich besser: `fstat(2)`.
- Zugegeben: Ggf. Umweg `fopen(3)` → `fileno(3)` → `fstat(2)` notwendig.

Typischer Unschönheit: Cleanup

- Innerhalb einer Funktion sammeln sich viele Ressourcen, die wieder freigegeben werden müssen.
- In jedem vorzeitigen return müssen bisher allozierte Ressourcen freigegeben werden.

```
FILE *f = fopen(...);  
File *g = fopen(...);  
void *buf = xmalloc(...);
```

```
// ...
```

```
if (error0) {  
    fclose(f);  
    fclose(g);  
    free(buf);  
}
```

```
// ...
```

```
if (error1) {  
    fclose(f);  
    fclose(g);  
    free(buf);  
}
```

- Fehleranfällig, insbesondere beim Refactoring.

Typischer Unschönheit: Cleanup

Besser:

```
FILE *f = fopen(...);
File *g = fopen(...);
void *buf = xmalloc(...);

// ...

if (error0) {
    goto fail;
}

// ...

if (error1) {
    goto fail;
}

fail:
if (f) fclose(f);
if (g) fclose(g);
if (buf) free(buf);
```

(*Disclaimer:* goto ist in neueren Sprachen auf dem absteigenden Ast. In C aber total valide.)

Typischer Fehler: Programming by Coincidence

- Am Anfang ist alles schwer, also probiert man gerne mal Dinge aus. Das ist gut, aber man sollte auch verstehen, *warum* und *wie* etwas funktioniert.
- Code wird nicht besser, wenn man überall `fflush(3)` und `errno = 0` verstreut.
- Alles andere ist *Programming by Coincidence*¹.

Besser:

- Lieber noch mal das Manual lesen, noch mal genau gucken. Noch mal überlegen: Warum mache ich das gerade?
- Erfahrung kommt mit der Zeit ☺.

¹<https://softwareengineering.stackexchange.com/questions/201972/how-to-overcome-programming-by-coincidence>

- Jetzt Besprechung einer beispielhaften Aufgabe.