A Programmer's Checklist :: 7 Rules
kaiwanTECH

*** VERY IMPORTANT ***
DID YOU REMEMBER TO

- ✔ Rule #1 : Check all APIs for their failure case
- ✔ Rule #2 : Compile with warnings on (`-Wall -Wextra`) and eliminate all warnings as far as is possible
- ✔ Rule #3 : Never trust [user] input; validate it
- ✔ Rule #4 : Use assertions in your code
- ✔ Rule #5 : Eliminate unused (or dead) code from the codebase immediately (Careful! Ensure *all* not-required code's eliminated; f.e. see **https://bit.ly/41GrKNW**)
- ✔ Rule #6 : Test thoroughly; 100% code coverage is the objective. Take the time and trouble to learn to use powerful tools: memory checkers (Valgrind, the sanitizer toolset), static and dynamic analyzers, code coverage tools, security checkers (checksec,...), fuzzers (AFL, syzbot), etc
- ✔ Rule #7 : Do NOT Assume Anything ("ASSUME : makes an ASS out of U and ME" :-))

Related article: Low-level Software Design

Video:

how NASA writes space-proof code
Low Level Learning

The 10 Rules - paraphrased from a comment by @rafaelbiasi:

1. 00:44 #01 Simple Control Flow
2. 01:08 #02 Limit All Loops
3. 01:40 #03 Don't use the Heap
4. 02:15 #04 Limit Function Size
5. 03:01 #05 Practice Data Hiding
6. 03:27 #06 Check Return Values
7. 04:17 #07 Limit the Preprocessor
8. 04:57 #08 Restrict Pointers Use
9. 05:33 #09 Be Pedantic
10. 05:47 #10 test test test

The Ten Commandments for C Programmers

Henry Spencer

1 Thou shalt run lint frequently and study its pronouncements with care, for verily its perception and judgment oft exceed thine.

2 Thou shalt not follow the NULL pointer, for chaos and madness await thee at its end.

3 Thou shalt cast all function arguments to the expected type if they are not of that type already, even when thou art convinced that this is unnecessary, lest they take cruel vengeance upon thee when thou least expect it.

4 If thy header files fail to declare the return types of thy library functions, thou shalt declare them thyself with the most meticulous care, lest grievous harm befall thy program.

5 Thou shalt check the array bounds of all strings (indeed, all arrays), for surely where thou typest ``foo" someone someday shall type “supercalifragilisticexpialidocious".

6 If a function be advertised to return an error code in the event of difficulties, thou shalt check for that code, yea, even though the checks triple the size of thy code and produce aches in thy typing fingers, for if thou thinkest ``it cannot happen to me", the gods shall surely punish thee for thy arrogance.

7 Thou shalt study thy libraries and strive not to re-invent them without cause, that thy code may be short and readable and thy days pleasant and productive.

8 Thou shalt make thy program's purpose and structure clear to thy fellow man by using the One True Brace Style, even if thou likest it not, for thy creativity is better used in solving problems than in creating beautiful new impediments to understanding.

9 Thy external identifiers shall be unique in the first six characters, though this harsh discipline be irksome and the years of its necessity stretch before thee seemingly without end, lest thou tear thy hair out and go mad on that fateful day when thou desirest to make thy program run on an old system.

10 Thou shalt foreswear, renounce, and abjure the vile heresy which claimeth that ``All the world's a VAX", and have no commerce with the benighted heathens who cling to this barbarous belief, that the days of thy program may be long even though the days of thy current machine be short.

A Python Easter Egg (see this article for more Linux Easter eggs!)

$ python3

Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

>>>
