The Ten Commandments for C Programmers (Annotated Edition)

Henry Spencer

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

This is still wise counsel, although many modern compilers search out many of the same sins, and there are often problems with *lint* being aged and infirm, or unavailable in strange lands. There are other tools, such as Saber C, useful to similar ends.

''Frequently'' means thou shouldst draw thy daily guidance from it, rather than hoping thy code will achieve *lint*'s blessing by a sudden act of repentance at the last minute. De-linting a program which has never been *lint*ed before is often a cleaning of the stables such as thou wouldst not wish on thy worst enemies. Some observe, also, that careful heed to the words of *lint* can be quite helpful in debugging.

''Study'' doth not mean mindless zeal to eradicate every byte of *lint* output—if for no other reason, because thou just canst not shut it up about some things—but that thou should know the cause of its unhappiness and understand what worrisome sign it tries to speak of.

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

Clearly the holy scriptures were mis-transcribed here, as the words should have been ''null pointer'', to minimize confusion between the concept of null pointers and the macro NULL (of which more anon). Otherwise, the meaning is plain. A null pointer points to regions filled with dragons, demons, core dumps, and numberless other foul creatures, all of which delight in frolicing in thy program if thou disturb their sleep. A null pointer doth *not* point to a 0 of any type, despite some blasphemous old code which impiously assumes this.

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.

A programmer should understand the type structure of his language, lest great misfortune befall him.

Contrary to the heresies espoused by some of the dwellers on the Western Shore, 'int' and 'long' are not the same type. The moment of their equivalence in size and representation is short, and the agony that awaits believers in their interchangeability shall last forever and ever once 64-bit machines become common.

Also, contrary to the beliefs common among the more backward inhabitants of the Polluted Eastern Marshes, 'NULL' does not have a pointer type, and must be cast to the correct type whenever it is used as a function argument.

(The words of the prophet Ansi, which permit NULL to be defined as having the type 'void *', are oft taken out of context and misunderstood. The prophet was granting a special dispensation for use in cases of great hardship in wild lands. Verily, a righteous program must make its own way through the Thicket Of Types without lazily relying on this rarely-available dispensation to solve all its problems. In any event, the great deity Dmr who created C hath wisely endowed it with many types of pointers, not just one, and thus it would still be necessary to convert the prophet's NULL to the desired type.)

It may be thought that the radical new blessing of ''prototypes'' might eliminate the need for caution about argument types. Not so, brethren. Firstly, when confronted with the twisted strangeness of variable numbers of arguments, the problem returns... and he who has not kept his faith strong by repeated practice shall surely fall to this subtle trap. Secondly, the wise men have observed that reliance on prototypes doth open many doors to strange errors, and some indeed had hoped that prototypes would be decreed for purposes of error checking but would not cause implicit conversions. Lastly, reliance on prototypes causeth great difficulty in the Real World today, when many cling to the old ways and the old compilers out of desire or necessity, and no man knoweth what machine his code may be asked to run on tomorrow.

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.

The prophet Ansi, in her wisdom, hath added that thou shouldst also scourge thy Suppliers, and demand on pain of excommunication that they produce header files that declare their library functions. For truly, only they know the precise form of the incantation appropriate to invoking their magic in the optimal way.

The prophet hath also commented that it is unwise, and leads one into the pits of damnation and subtle bugs, to attempt to declare such functions thyself when thy header files do the job right.

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

As demonstrated by the deeds of the Great Worm, a consequence of this commandment is that robust production software should never make use of *gets()*, for it is truly a tool of the Devil. Thy interfaces should always inform thy servants of the bounds of thy arrays, and servants who spurn such advice or quietly fail to follow it should be dispatched forthwith to the Land Of Rm, where they can do no further harm to thee.

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.

All true believers doth wish for a better error-handling mechanism, for explicit checks of return codes are tiresome in the extreme and the temptation to omit them is great. But until the far-off day of deliverance cometh, one must walk the long and winding road with patience and care, for thy Vendor, thy Machine, and thy Software delight in surprises and think nothing of producing subtly meaningless results on the day before thy Thesis Oral or thy Big Pitch To The Client.

Occasionally, as with the *ferror()* feature of *stdio*, it is possible to defer error checking until the end when a cumulative result can be tested, and this often produceth code which is shorter and clearer. Also, even the most zealous believer should exercise some judgement when dealing with functions whose failure is totally uninteresting... but beware, for the cast to void is a two-edged sword that sheddeth thine own blood without remorse.

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.

Numberless are the unwashed heathen who scorn their libraries on various silly and spurious grounds, such as blind worship of the Little Tin God (also known as "Efficiency"). While it is true that some features of the C libraries were ill-advised, by and large it is better and cheaper to use the works of others than to persist in re-inventing the square wheel. But thou should take the greatest of care to understand what thy libraries promise, and what they do not, lest thou rely on facilities that may vanish from under thy feet in future.

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.

These words, alas, have caused some uncertainty among the novices and the converts, who knoweth not the ancient wisdoms. The One True Brace Style referred to is that demonstrated in the writings of the First Prophets, Kernighan and Ritchie. Often and again it is criticized by the ignorant as hard to use, when in truth it is merely somewhat difficult to learn, and thereafter is wonderfully clear and obvious, if perhaps a bit sensitive to mistakes.

While thou might think that thine own ideas of brace style lead to clearer programs, thy successors will not thank thee for it, but rather shall revile thy works and curse thy name, and word of this might get to thy next employer. Many customs in this life persist because they ease friction and promote productivity as a result of universal agreement, and whether they are precisely the optimal choices is much less important. So it is with brace style.

As a lamentable side issue, there has been some unrest from the fanatics of the Pronoun Gestapo over the use of the word "man" in this Commandment, for they believe that great efforts and loud shouting devoted to the ritual purification of the

> language will somehow redound to the benefit of the downtrodden (whose real and grievous woes tendeth to get lost amidst all that thunder and fury). When preaching the gospel to the narrow of mind and short of temper, the word ‘‘creature’’ may be substituted as a suitable pseudoBiblical term free of the taint of Political Incorrectness.

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.

> Though some hasty zealots cry ‘‘not so; the Millenium is come, and this saying is obsolete and no longer need be supported’’, verily there be many, many ancient systems in the world, and it is the decree of the dreaded god Murphy that thy next employment just might be on one. While thou sleepest, he plotteth against thee. Awake and take care.
>
> It is, note carefully, not necessary that thy identifiers be limited to a *length* of six characters. The only requirement that the holy words place upon thee is uniqueness within the *first* six. This often is not so hard as the belittlers claimeth.

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.

> This particular heresy bids fair to be replaced by ‘‘All the world’s a Sun’’ or ‘‘All the world’s a 386’’ (this latter being a particularly revolting invention of Satan), but the words apply to all such without limitation. Beware, in particular, of the subtle and terrible ‘‘All the world’s a 32-bit machine’’, which is almost true today but shall cease to be so before thy resume grows too much longer.