知らなきゃヤバイ

基礎ほど奥深〜い！ Cプログラミングの当たり前!?

第2回 エラーが出なくて見つけづらい！ プリプロセッサが原因のバグ

邑中 雅樹

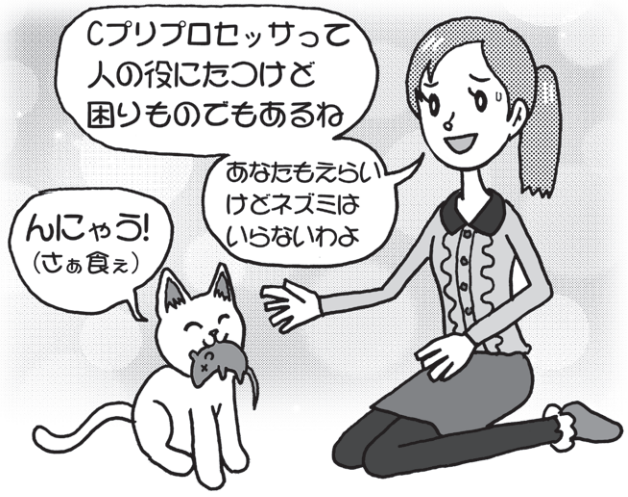

図1 Cプリプロセッサはなくてはならない存在である

Cプリプロセッサは，C言語を使う上でなくてはならない存在です（**図1**）．最初に学ぶHello Worldでさえ，#include <stdio.h>を行頭で宣言しなければ警告を受けます．この#から始まるinclude文は，プリプロセッサが提供する機能の一つです．プリプロセッサは人間の可読性を高めるためだけにある記述を，コンパイラなどが処理しやすい記述に置き換えます．

身近なものほど深い落とし穴が潜んでいるのは，プログラミングでの世界です．今回は，Cプリプロセッサが原因で発生する見付けづらいバグとその対策を紹介します．

復習：プリプロセッサって，なぜあるの？

● プログラミング言語はCPUと人間との間を取り持つ

まず，当たり前のところから入りましょう．すべてのCPUはバイナリ（機械語）を解釈します．しかし，人間はバイナリを解釈できません．すべてのプログラミング言語は，この差を埋めるために存在します．

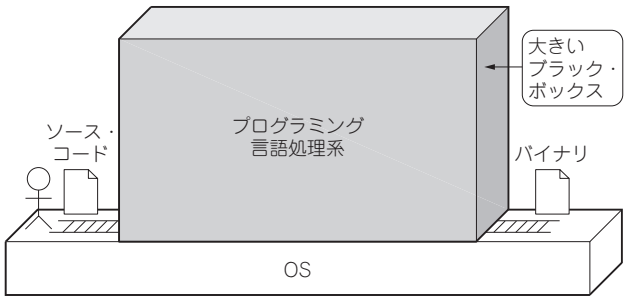

図2 ソース・コードをバイナリに変換するプログラムは巨大なブラック・ボックス

● プログラミング言語処理系は巨大なのでいくつかに分割されている

プログラミング言語処理系は，人間がかろうじて読める表記であるソース・コードを，CPUが解釈しやすいバイナリに変換します．C言語の処理系も同様です．

人間とCPUとの溝を埋める作業をするプログラミング言語処理系は，OSと並んで巨大で複雑なプログラムです（**図2**）．巨大で複雑ということは，いくつかの深刻な問題を引き起こします．まず，プログラミング言語処理系そのものの保守性が低下します．また，プログラミング言語の処理系が動作する環境を越えたリソースを要求するようになります．

後者のリソース制約は，ハードウェアの進歩により緩和され続けています．しかし，C言語が生まれた頃を想像してみてください．Unixが動作するミニコン（動作周波数：数MHz，メモリ64Kバイト程度）ですら，現在の組み込み向けCPUよりも動作クロックもRAMも少なかったのです．

そこで，プログラミング言語処理系の内部を何回かのパス（pass）に分けて，保守性や省リソースを確保するという方策がとられるようになりました（**図3**）．逐次解釈するインタプリタではなく，事前にバイナリを生成するコンパイラが好まれた理由の一つには，上記のような背景があります．また，コンパイラも，ソース・コードからアセンブリ