计算机中C语言的应用特点分析

韩润华，丁颖
西安明德理工学院，陕西 西安 710124
DOI:10.61369/EDTR.2026080010

摘　　要： 作为计算机领域经典的结构化编程语言，C语言具备运行效率高、底层操作性强、适配范围广等诸多突出优势，始终是底层系统研发与嵌入式工程开发的核心基础语言。在信息技术高速迭代的当下，Java、Python等主流高级语言凭借便捷的开发特性与完善的生态体系，在各类上层软件开发场景中得到大范围普及。然而在硬件驱动开发、工业自动化控制、物联网终端程序设计等高性能、高实时性的专业场景中，高级语言仍存在明显的技术短板，无法完全替代C语言的核心地位。本文立足于工程开发实际场景，重点探究C语言在系统程序编写、硬件资源调度、跨平台程序研发过程中的技术特征，重点剖析其运行高效性、硬件可控性与程序开发灵活性三大核心优势。结合具体工程应用实例，客观总结该语言在现代计算机开发领域的适用价值与应用局限，可为技术人员合理选用编程工具、搭建软件开发方案提供有效的理论参考。

关 键 词： C语言；应用特点；系统开发

Analysis of the Application Characteristics of C Language in Computer Science

Han Runhua, Ding Ying
Xi'an Mingde Institute of Technology, Xi'an, Shaanxi 710124

Abstract: As a classic structured programming language in the field of computer science, C language boasts numerous outstanding advantages, including high operational efficiency, strong low-level operation capabilities, and broad adaptability, making it a core foundational language for the development of low-level systems and embedded engineering projects. In the current era of rapid technological iteration in information technology, mainstream high-level languages such as Java and Python have gained widespread popularity in various upper-layer software development scenarios due to their convenient development features and comprehensive ecosystems. However, in professional scenarios requiring high performance and real-time capabilities, such as hardware driver development, industrial automation control, and Internet of Things (IoT) terminal programming, high-level languages still exhibit significant technical limitations and cannot fully replace the central role of C language. Based on practical engineering development scenarios, this paper focuses on exploring the technical characteristics of C language in system programming, hardware resource scheduling, and cross-platform program development, with an emphasis on analyzing its three core advantages: operational efficiency, hardware controllability, and program development flexibility. By integrating specific engineering application examples, this paper objectively summarizes the applicability value and limitations of C language in modern computer development, providing effective theoretical references for technicians to reasonably select programming tools and establish software development plans.

Keywords: C language; application characteristics; system development

自1972年正式问世以来，C语言凭借精简的语法体系、高效的编译运行机制、极低的资源占用率，长期服务于计算机操作系统研发与硬件程序开发工作，是支撑底层计算机技术发展的基础性编程语言。在软件开发技术多元化发展的背景下，各类新型高级编程语言不断涌现，极大降低了应用程序的开发难度，广泛应用于网络开发、数据处理、智能算法研发等诸多前沿领域。即便高级语言的应用场景持续拓展，但在对程序实时性、稳定性、硬件适配性要求严苛的工业场景中，C语言依旧发挥着不可替代的作用，是物联网设备、工业控制终端、嵌入式系统开发的核心工具。本文结合实际编程开发经验与工程实践案例，系统探究C语言在现代计算机体系中的应用价值，深入剖析其技术优势与使用短板，明确其在当下软件开发体系中的应用边界，为底层程序开发与编程语言选型提供理论支撑。

一、C语言在系统开发中的核心地位

（一）操作系统开发的基石

作为底层硬件开发语言的首选，C语言的主要原因是它具有清晰简洁的语法和接近硬件的优势。拿Linux操作系统内核为例，该内核中近80%代码是用C语言编写的，例如在进程调度、内存管理等内核模块，都是通过C语言对硬件内存和寄存器进行直接操作的。而在硬件或设备驱动开发时，开发者也可根据自身经验通过C语言中的指针直接操作GPIO(GeneralPurposeInputOutput)口，例如树莓派在进行项目开发时，只需几行代码就可控制外接传感器高低电平信息，使其驱动LED灯的亮与灭或者获取温度信息。C语言这一特性对操作系统等对时间精度有要求的系统(RealtimeOS)来说必不可少。然而，高级语言(如Python或Java)运行于虚拟机或解释器内部，无法绕过中间环节直接对硬件进行操作，更像一把“电钻”，只有在必要的情况下，开发者才能够发挥其“螺丝刀”的作用，“拧紧”每一个系统部分[1]。

（二）硬件资源的高效管理

C语言的指针内存分配和释放（malloc和free）尽管古老、笨拙和直接，却适用于那些非常珍惜内存的嵌入式程序开发。很多实时性的工业控制器，在不考虑暂停的自动垃圾回收机时，会导致系统控制器的响应偏差（譬如，机械臂的运动控制指令，Java语言运行着不可预料的时间），而利用C语言人工控制分配和释放内存空间，可以保证诸如工业机器人运动控制等必须任务的毫秒响应。在工业机器人上利用C语言实现内存池的方式就是预先分配出一大块连续内存，进行固定大小的循环分配，而不必担心内存的碎片化。另外，C语言的位运算、结构体内存对齐功能等都可以通过指令完成对内存的管理，做到更为高效的存储。在某些单片机系统内，开发人员还可以利用位域(bit-field)将一组状态变量压缩进一个字节，节省嵌入系统的昂贵硬件。这就是指针内存分配所控制的所谓“能不花钱就不花，能少花钱就少花”的原则。

（三）开发工具链的成熟生态

C语言经过近30年的成熟，拥有了工具生态。GCC编译器能够编译x86服务器和ARM嵌入式芯片之间的底层程序，一个源文件，只需改变编译器指令编译到另一台机器上，只需更改命令行参数即可；GDB用来调试系统底层核心模块的系统崩溃问题，例如Linux内核态设备驱动模块的空指针异常现象；Valgrind用来检查应用程序内存泄露情况，并支持在代码中标识各种内存操作，以改进代码性能；SQLite数据库的核心源代码使用纯C编写，通过C语言#ifdef和#endif的支持，实现了在不同环境间(Windows、Linux和MacOS)的适配；Python语言对底层文件读写的封装也使用了C语言封装，如cPickle。这些应用工具的描述可以让我们看出C语言并非孤军奋战，而是借助丰富的生态接口和应用得以不断支持着底层的开发及维护[2]。

（四）极强的代码可移植性与平台适配能力

在现代计算机系统开发与嵌入式工程应用中，代码跨平台移植能力是衡量编程语言实用性的重要指标，而C语言具备其他高级语言难以比拟的轻量化、高适配、易移植优势。C语言程序编译不依赖专属运行环境，代码精简、冗余量小，脱离虚拟机与解释器依赖，可直接适配各类低端单片机、高端处理器、嵌入式终端、工控设备等不同层级硬件平台，适用场景覆盖民用、工业、车载、物联网等多个领域。不同于Java依赖虚拟机、Python依赖解释器才能运行的限制，标准C语言语法规范统一、编译机制简单，只需根据目标硬件架构调整底层编译参数，即可快速完成代码移植复用，大幅缩短嵌入式设备、工控系统的开发周期。目前市面上绝大多数智能单片机、STM系列芯片、车载控制单元、物联网终端设备的官方开发库、驱动源码均以C语言为标准开发语言，形成统一的开发规范。这种高度的通用性与移植性，让C语言能够适配各类差异化硬件场景，实现一套核心代码多设备复用，极大提升了底层系统开发的通用性与高效性。

二、C语言在跨平台开发中的灵活性与局限性

（一）可移植性的双刃剑特性

跨平台性的先天支持离不开C语言，那便是以ANSI标准为基础制定的统一的语法和一些基本库函数的实现，比如打开文件的库函数fopen和写文件的库函数fwrite，用这两种库函数编写的一段C语言程序，可以在Windows、Linux以及macOS等平台上编译，并且输出一致的运行结果。因此很多跨平台工具的开发，都是基于C语言来完成的，如FFmpeg多媒体库，其底层代码都是由C语言完成，并且在主流的操作系统上均能够进行编译和运行，跨平台性的优势较为明显。但是硬件级不同所造成的错误容易成为跨平台性的一个陷阱，比如关于字节序方面的问题，采用ARM架构设计的嵌入式设备的运行模式都是以小端字节序的方式来保存数据，但是目前对于网络协议的默认定义均为以大端字节序的方式来保存数据，若编程人员在程序设计的过程中，没有特意调用htonl()以及ntohl()函数的方式将字节序进行转换，结果可能会导致某温度传感器的采集数据在经过程序转换为整数形式时，出现对传感器读回温度值进行错误转换的问题。在实际的工程实践中，程序设计人员会选择使用一些预定义的宏来判断当前系统的字节序，进而手动为程序设计对应的一些逻辑实现，这时会出现代码能够移植和运行，但是程序内部的逻辑设计中缺乏代码逻辑扩展的缺点，编程人员不仅需要对C语言的语法较为熟悉，还需要对底层硬件逻辑较为熟悉，否则不能从平台的差异性的视角来判定系统的运行时的异常问题[3]。

（二）系统级接口的适配优势

对于Windows系统和Linux系统的API差异，C语言的条件编译就是让一行代码不同操作系统的API调用“各行其事”，如Windows的线程创建方法调用CreateThread()函数，Linux使用pthread_create()函数，可以在代码里写入#ifdefWIN32，#ifdefLinux两种条件分支，自动选取操作系统的相应系统接口，就类似C语言程序一样。开源数据库SQLite正是这种机制的践行者：其提供了预置的包括VxWorks、Android等多个系统的系统层适配文件，C代码中编写对应的文件锁实现、内存分配实现，并最终

由条件编译选择文件锁和内存实现，最终实现了“一行代码，多平台”。正因为有这种可移植性，所以让C语言成为了实现多平台系统工具的语言，如用于处理音视频编解码的FFmpeg核心代码也全为C语言编译，可以直接应用在iOS和Android应用上。然而代价就是需要编写针对不同的平台，编写相互隔离的代码分支，对维护多系统进行的测试代码也很高，如在Linux使用高效的I/O模型epoll就不可以直接移植到Windows上，必须在条件编译下，调换为IOCP，再调整其事件循环实现。

三、C语言在性能优化中的不可替代性

（一）执行效率的极致追求

C语言的灵魂是“低级”。相比于Python这样的解释性语言，在解释器或虚拟机的高层中，C语言将最终的源码直接编译为机器码，不存在一层层的调用开销。以快排算法为例，在Python实现中，循环内部是动态解析的循环逻辑，且内含动态类型转换，在C语言中静态分配内存地址进行处理，循环内部的指令执行效率能够获得十几倍的增长。对于一个图像实时处理的例子，在树莓派上部署深度学习的人脸检测代码，C语言基于图像本身的特征算法能够实现实时处理每秒20帧以上，而同样的程序在Python中可能出现卡顿的情况，或者帧率不足的问题。或者在编译的代码内部直接加入内联汇编，完成特定的代码段优化，例如某个音频编码库针对特定的加速代码，实现对于寄存器分配算法的调整，将四元数的傅里叶变换的计算时间压缩到极致。它就像是高楼大厦下的钢筋水泥，一切在高端应用程序需要实现的“高性能”需求下，最后仍是在C语言中的某些代码片段中得以实现——数据库中的索引构建，游戏模拟物理的场景[4]。

（二）实时系统的刚性需求

对实时性有特别要求的应用场景，如工业机器人控制、航空航天等只能选用C语言。因为对工业机械臂进行运动路径规划的计算，往往要求每毫秒计算数个坐标值以及驱动电机相应，任何不可预知的延迟都可能导致报废或者人员伤亡。典型的如汽车整车生产线上的焊接机器人，采用C语言编写的控制程序对硬串行端口的定时器进行操作，经过调整之后，让机器每隔0.5毫秒完成一次闭环控制。然而，Java、Go等语言由于其垃圾回收机制存在的不可避免的、持续在100~200微秒级别的延时却无法满足这种需求。自动驾驶则将这种优势变得更加明显，汽车周围散落在空间中的激光雷达可以提供每秒几百万个三维坐标点，C语言实现的滤波算法使用C语言本身直接在内存中连续读写和DMA（DirectMemoryAccess）绕过CPU直接将数据传输的时间降低到纳秒级别。即使被称作下一代内存安全语言的Rust，在所有者模型的设计上，也引入了额外的判断逻辑来与具体的状态关联，并且不可能说对软件层面的一些计算的不可预料的响应时间，而C语言因为直接操作硬件本身的“代码即机器”的透明性，可以被认为“总是”作为实时系统的开发金标准[5]。

（三）资源受限环境的生存法则

在每KB甚至每字节计算力的年代，C语言的内存手动管理功能是一种生存技能。在一些用8位单片机（一般只有4KB的内存）实现智能家居温控器的案例中，许多开发者都需要把温度上限、工作模式等8个状态打包存储为一个字节，而这在C语言里只需一个结构体位宽度定义即可实现，根本不用关心这个打包组合在芯片上的“单位价格”。而Rust标准库及编译后的代码太大，一个电器的控制（比如一个空调，吹热或冷、除湿或不用）、微控制器芯片、外围电路乃至整个器件都要考虑。比如一款空气净化器的控制单片机，需要考虑并行读取PM2.5传感器和风扇的PWM（脉宽调制）驱动信号，由C语言编写的代码中把传感器结果以数据缓冲区形式分配内存，并将指针指向缓冲区内存单元，然后通过指针“搬运”内存到寄存器。总计占用的芯片程序和数据内存不超过2KB。而Rust，即使编译链接的时候不链接std库，也要增加30%以上的安全检验指令，因为嵌入式内存空间少，凡是写代码都占用内存，即使是像#error这样简单的指令，也会增加一两行。在这种绝对需求控制资源的时代，C语言可以继续成为万物互联终端、可穿戴设备等嵌入式领域的主力平台。毕竟在电池容量能提供的最大电流只有毫安级的时候，程序里多余的代码行都可能是扼杀电池续航能力的“罪魁祸首”[6]。

四、结语

作为“底层的语言”，C语言基于对底层“硬件”贴近的原则，其还在更多系统开发、优化性能的应用中发挥着作用。C语言对于本就缺少“现代语言”的高级抽象设计，还起到在“需要与底层硬件交互”和“需要最高性能”时做底层的语言的作用。如何运用好C语言和其他语言结合解决问题，根据具体情况而言，对于有系统的开发者在系统层使用C语言，同时可以将“对上层业务逻辑”的开发者结合自身的业务属性进行分析，使用其他高级语言构建业务逻辑框架，这样可以达到不同的效果。

参考文献

[1]宁万龙 . 计算机中C语言的应用特点分析 [J]. 造纸装备及材料 ,2020,49(1):83.

[2]刘鑫茹 . 对计算机中C语言的应用特点的分析 [J]. 轻纺工业与技术 ,2020,49(1):145-146.

[3]于鑫 . 计算机应用C语言的特点分析 [J]. 电脑校园 ,2021(1):18.

[4]余勋 , 王捷 . 浅谈C语言编程技巧在C语言学习中的应用 [J]. 信息通信 ,2013,No.129(07):108.

[5]段煅 .C语言编程技巧在C语言学习中的应用 [J]. 电脑编程技巧与维护 ,2010,No.218(20):150-151.

[6]陈玉仙 . 关于C语言教学几个问题的探讨 [J]. 电脑知识与技术 ,2013,9(13):3111-3113.

