# 提高移植性/安全性技巧之自定义类型系统

## 背景问题？

## 解决方法：

在设计实现一个软件程序时，为了提高程序在不同平台（体系结构、编译环境等，主要是编译环境）的移植性，避免程序中数据对象的类型因为平台的不同而实现不同导致的在取值范围上的不可预期的bug。

程序在实现时，应该建立自己的数据类型系统，定义专用的基本数据类型。在程序中声明或定义数据对象时，使用自定义的数据类型。这样在移植时，只需要根据具体的平台环境，重新定义这些数据类型即可，程序代码不用改变。

基本数据类型通常包括：

8/16/32位的有无符号数等；

## 示例：lwip协议栈的数据类型系统

### 目的：

为了尽可能减少对外部环境（体系结构、编译器环境等）的依赖，提高lwip的可移植性，以方便lwip在不同平台上的移植。lwip协议栈在实现时，定义了自己的数据类型系统。

### 实现：

#### 概述：

1、 **就协议栈整体**而言，有一套基本的数据类型系统，可被协议栈的所有功能模块使用；

2、**每个功能模块**在实现时，如果需要也会定义自己模块内部的数据类型，作为内部及外部的操作本模块内数据时使用。

#### 基本数据类型系统（包含在用户自定义的移植头文件cc.h中）

由系统程序员在移植lwip时，自定一个cc.h的头文件，其中包含数据类型部分的定义。

#### 各模块自定义数据类型（包含在各模块头文件中[如果有]）