Subject: Re: New packages announcement
Posted by cbpporter on Mon, 15 Jun 2009 12:29:43 GMT
View Forum Message <> Reply to Message

If I extend C I will break working code. If I extend C++ I will break working code again. This is not an issue, it is a design choice. As I said C is not a good practical language and making it backward compatible would "doom" it from the start. And I don't really want to extend it, rather than replace all unsafe features with safe ones, basically creating a new systems programming language which is as safe and clear Java and has the performance of C, but with slightly more memory consumption (arrays have sizes). The memory consumption of 8 bytes per array is not

the length along the pointer so you can optimize by structure size. Also, our build machines have

And believe me, I have tried every single programming language that has a fair user share and about 20 GUI toolkits before I have stumbled upon U++. U++ is the perfect GUI toolkit for me, and C++ is the perfect language (but with extremely archaic technology behind it). When I started using U++ I was doing desktop software, but I'm not doing that anymore.

But one cannot choose his programming language. In the domain I work (industrial printers) there are only two choices: C and C++. I'm researching my language proposal to see if one can merge the ease of use of C++ RAII semantics for easy resource management with a classic non OOP based way of working like in C, but in the meantime making C extremely safe.

And NULL terminated strings are not fast. You need to check the whole string to determine the length. I understand that a lot of operations on strings are serial by nature and this disadvantage disappears, but there are also a lot of operations which are not serial. Take for example strings which have a known length which is padded with zeros up to the platforms integer size (32/64 bits). You can write an optimized strcpy which will use 32 bit moves and will never have to compare against zero vs. 4 times as many 8bit moves and comparing to zero. Which do you think is faster? Or 64 bit moves vs. 8 times as many 8 bit moves with null checking? Actually U++ does this for short strings. Or string concatenation, reverse find, binary search, etc. They all are more efficient when the length of the string is known, and since a string is an array, these algorithms will work on all arrays.