Unicode Stinks

An essay by Blue-Maned_Hawk

Introduction

Unicode is the single most widely-used standard for encoding text on computer systems. It is significantly better than everything that came before it; instead of a computer needing to juggle eighty different ways to interpret a piece of text, it only needs to deal with one standard that comprehensively contains damn near every writing system in the world. This has significantly improved the computing world, and Unicode is certainly one of the best inventions in recent times.

However, Unicode is not perfect, and really kinda sucks in quite a lot of places. Many people across the internet have discussed these flaws in great detail before. However, while I will be reiterating these flaws in this essay, this will mostly be as background for the main purpose of this essay: to detail a hypothetical method by which one could construct a better replacement for Unicode.

Unicode's Flaws

Backwards Compatibility

Backwards compatibility is generally seen as a good thing, which makes sense; after all, making it so that you can continue to use the old things while still taking advantage of the new things is extremely useful, as has been proven in many, many circumstances. However, a major issue with it is that there are a lot of circumstances where the group maintaining something will want to remove something for a good reason, such as if it's going to be replaced with something new and better, or if it seemed like a good idea but never ended up getting used much, or any number of other circumstances.

With backwards compatibility as a policy, this removal is impossible. Often, this leads to things being deprecated instead, meaning that people aren't *supposed* to use them, but there's literally nothing that's stopping them from doing so. This means that, over time, something that mandates backwards compatibility will become crusty and bloated with old gunk that can't be removed. Usually, once something gets to this point, the best choice is just to start over, to create a new major release (or, in some cases [such as with OpenGL], to create a whole new thing entirely!).

However, Unicode has refused to do this. This means that all of their mistakes, all of their bad decisions, all of it is *required* to continue onward into new releases. That's how we end up with deprecated characters, how we end up with ridiculous alias situations, how we end up with the silliness surrounding the byte order mark. Yet even with all of these problems, the Unicode Consortium refuses to do anything.

This requirement of backwards compatibility also leads to disorganization. Because new character can't be inserted between other characters (since this would lead to a reshuffling of the codepoints, which the Consortium deems unacceptable), they instead need to be inserted wherever there happens to be space, usually appending them to the end. This leads to, for example, the Latin script being, instead of one block, spread out across approximately too many blocks.

The Philosophy of Unicode

Unicode's supposed goal is to make it so that everyone can write in their own language across the world's computing systems. However, in many cases it deviates from this goal, and in some cases it completely ignores this goal. In other instances, it technically achieves this goal, but it does so in a flawed way.

Consider, for instance, the existence of the Mathematical Alphanumeric Symbols block. This block encodes bold, italic, serif, sans-serif, monospace, script, and blackboard bold variants for letters, but only if they're from the Latin alphabet or the Greek alphabet. However, we know that Unicode has the ability to compress things like this—for example, instead of encoding every possible character that could have a macron over it, it just has the combining macron character. Why couldn't a similar approach be taken here? (This is made even sillier by the fact that the Unicode Consortium explicitly states that people shouldn't use these for formatting, but instead for when it's useful to differentiate these non-graphically [or some bullshit like that]—as you may have seen, the internet at large certainly disregards that request, particularly when formatting restrictions are in place.)

And on the topic of the combining diacritics, consider that there also exist precomposed characters, despite combining diacritics theoretically making them unnecessary. These are kept as separate things to facilitate round-trip conversion with other formats, which seems to go against Unicode's goal to be a single unifying standard. The most ridiculous case of this is possibly the capital a-with-a-ring, which has not two but *three* separate ways to write it, the third being the ångström sign.

The ångström sign brings up another issue: the same character being encoded in multiple places. Consider, for instance, the fact that the micro sign is considered a different character from the micro sign. This is supposedly so that they can be considered as separate characters in separate circumstances…but in other cases, the Consortium has refused to do this, such as in regards to fonts. Why are fonts considered an implementation detail, but context isn't?

A Replacement

As can be seen in these examples of its flaws, Unicode is riddled with problems, and the Consortium is unlikely to consider fixing them. So let's consider instead what a new standard would look like, a replacement for Unicode for the modern era, without the cruft and historical baggage and ridiculous little complexities that Unicode brings.

First off, this standard would be one that would use semantic versioning to indicate breakages of backwards compatibility, and it would be run by a group that would be willing to break backwards compatibility if necessary. In addition, the standard would not seek to be compatible with existing standards, even Unicode.

All characters would be encoded using 32 bits. While it is true that this would take up a potentially exceedingly large amount of space, as any particular piece of text would be using only a portion of this new standard, any good compression algorithm would be able to use this fact to its advantage for compression anyways. Besides, using a variable-length encoding would run into issues of its own—take, for instance, how UTF-8 is anglocentric.

Characters would be organized hierarchically, instead of linearly. Perhaps it could start by differentiating between control characters and graphical characters, then under graphical characters it could differentiate between combining characters and noncombining character, then differentiate between general-purpose and script-specific characters. (Of course, in a real standard, this would be much more carefully thought out. These are just examples.)

Obviously, this can't just be created and then expected to be used in everything. Most everything in the modern world's computer system is designed to use Unicode, and changing to a new standard would be *very* difficult. A new standard like this would most likely come into existence as part of a larger effort to completely overhaul the world's computer systems, induced by the fact that the current world's computer system is terrible on every level…but that's a story for another time. For now, this remains only theoretical, and we simply have to deal with Unicode.

Thank you for reading this essay. I hope it was enlightening for you.
