

Proposal for a GNU Unicode Font

The [Unicode](#) Standard was first published in 1991. Seven years later there is still no complete Unicode font and Unicode text often shows up unreadable with empty boxes or question marks for missing characters. This can lead to misunderstandings and is quite frustrating.

I believe that we can get closer to a first complete Unicode font if we (a) lower our expectations about the [font quality](#) to a reasonable degree and (b) organize a distributed [GNU](#) community effort to complete and optimize that font.

The result will be a basic one-size-fits-all freeware font that you and all of your communication partners can use to make Unicode text legible on a screen or a printer.

What do you mean by "complete Unicode font"?

If we want Unicode to become a standard for average users then we cannot expect them all to go hunting for nice font combinations that may or may not cover the characters that they received by mail or HTTP. And if we want the original Unicode idea of supporting all living languages to come true, then we must not artificially limit the character repertoire to eurocentric market-power subsets like the CEN Minimum European Subset [MES](#), Microsoft's Windows Glyph List 4 [WGL4](#) or Markus Kuhn's Simple European Character Set [SECS](#).

Some people say that there will never be a Unicode font because a Unicode font is a contradiction in itself. Officially, Unicode (ISO 10646 UCS) does not encode fonts (collections of graphical shapes called "glyphs") but abstract characters (which are allowed or even required to change shape depending on the context they're in). You are supposed to use a complex rendering engine to translate Unicode strings into graphics. The renderer should use a more fine-grained indexing than a Unicode value to get at the appropriate glyph.

But is there a standard solution to universal font indexing? Not that I know of. The glyph registration standard ISO 10036 put the Association for Font Information Interchange ([AFII](#)) in charge of numbering glyphs long ago but the AFII glyph registry is far less complete and global in scope than Unicode character set. You cannot rely on AFII-indexed fonts to implement complete Unicode coverage and it is unclear if and how fast the AFII registry will catch up now that AFII has joined unicode.org.

On the other hand, you can very well pretend to forget about the subtle difference between characters and glyphs and abuse Unicode as a one-to-one glyph numbering scheme. All Unicode characters have exactly one reference glyph in the Unicode book. Pasting these glyphs next to each other on horizontal lines just like you did with [ASCII](#) and [ISO-8859-1](#) works for many languages, including most European languages, Ethiopic, Chinese, Japanese and Korean (CJK). You will get to see the text in its bare Unicode representation, in some sort of "View Control Codes" mode which may also be interesting for debugging purposes. If you don't mind an occasional accent appearing after its base character instead of attached to it or characters appearing in an unusual visual order, you'll be fine. Simple rendering preprocessors like my [arabjoin](#) script can already reduce a number of these problems within the unicoded glyph model through symmetric reordering and mapping to the precomposed presentation forms that exist in Unicode for compatibility with older charsets. By using Unicode

as a mere font encoding you can cover a great deal of the world's languages and get some quite readable results. The native languages from in and around India are currently the odd man out as nobody has publically numbered their many ligature glyphs yet. They will appear far from perfect with a bare Unicode font.

Don't let poor local rendering capabilities (bitmapped teletypewriter emulators with noncombining character cells) stand in the way of already using a high-quality encoding (Unicode). That way we are at least able to send proper instructions to high-quality typesetters. The real-life existence of Unicode text on the Internet will eventually also spur the development of universally better formatting support, be it through something like Apple's [QuickDraw GX](#), or [FreeType](#) (TrueType technology), [Ω \(TeX & MetaFont](#) for which we have [GNU font utilities](#)), [CTL](#) (X11 complex text layout), or [Berlin](#) (a next-generation windowing system).

A complete Unicode font means to me: first one graphic for each meaningful Unicode character even if it is an artificial control character graphic so that you no longer have to keep staring at black boxes.

Didn't others try this earlier?

The idea of using Unicode as a glyph encoding is by no means new. There has already been quite some activity in that area:

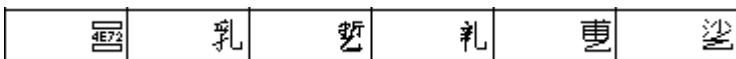
Lucida Sans Unicode

The pioneering example was Charles Bigelow <bigelow@cs.stanford.edu> & Kris Holmes' Lucida Sans Unicode font. In their 1993 article on "The design of a Unicode font" (in: Electronic Publishing, ISSN 0894-3982, volume 6.3, pages 289..385) they describe how and why they designed a first harmonized modern font of fine-crafted glyphs covering 1'700 Latin, Greek, Cyrillic (LGC), Hebrew, phonetic and mathematical Unicode 1.01 characters.

Microsoft shipped this font as l_10646.ttf with their Unicode-based Windows NT. It also set the standard for adding a Unicode encoding vector to TrueType fonts. AT&T's Unicode-based operating system [Plan9](#) also adopted the fixed-pitch Lucida Console as base font; you can download their bitmaps from the [9term](#) home page as [libXg.utf.fonts.tar.gz](#). You can also buy the Lucida font as a set of 12 Unicode blocks of hinted PostScript Type1 subfonts from [Y&Y](#) for 100 U.S. dollars. B&H's updated Unicode 2.0 version of Lucida added Vietnamese, Thai and Arabic but has not been licensed by Microsoft. You get the point: the Lucida font is beautiful but not free, and its designers made a conscious decision to design all glyphs themselves to keep a consistent design even if that costs a lot of time.

Unihan

A technique criticized by Bigelow & Holmes for their poor typographic quality is the simple assembly of a Unicode font set from various existing fonts for various scripts. One such example is the misc.unihan.16.bdf compiled in 1993 by Ross Paterson <rap@doc.ic.ac.uk> from CCLIB16.FS for GB 2312, HKU's chinese.16 for BIG-5, Jming16 for JIS X 0208, hanglm16 for KSC 5601 and surrogate bitmaps containing the hexadecimal Unicodes for the missing 20'902 - 16'411 = 4'491 Han ideographs. Wei-Lun Chao <wchao@hrz.uni-bielefeld.de> added some more fonts' glyphs in the alphabetic section in 1996 and called his new creation uni16m.bdf.



Another font in that line was the 6 MB uni24.bdf that came from Ms Ho Yean Fee's Multilingual Application Support Service ([MASS](#)) group at the Institute of Systems Science of the National University of Singapore who have now formed their own company [Star + Globe Technologies](#). uni24.bdf appears to be based on Sony's large 12x24 font with some smaller characters added from other fonts, see [uxterm.gif](#). uni24.bdf used

the [XLFD](#) name -issuni-song-medium-r-normal--26-240-100-100-p-130-unicode-1.1 and contained no copyright notice.

You will find these fonts in [HBF](#) (Hanzi Bitmap Format) on the [server](#) of the Chinese Computing Information Centre ([CCIC](#)) and its many mirrors. Man-Chi Pong <mcpong@cs.ust.hk> et al. had invented HBF in 1993 as an optimized alternative to X11's BDF and PCF font formats to reduce the outrageous file sizes and access times for fixed-width fonts covering the several thousand Han ideographs, see pages 113..123 of the April 1994 volume 10 of the X Resource, ISSN 1058-5591. Their HBF patch has not yet made it into the standard X11 fontlib, but Ross Paterson's HBF package also contains a hbftobdf converter.

uni16m and uni24 suffer from the [Hangul mess](#) problem that they still do not contain the the full set of 11'172 precomposed modern Hangul syllables in the nicely sorted [U+AC00..U+D7A3] block as specified by Unicode-2.0:1996 and [ISO-10646-1:1993-Amendment-5](#):1998 but only the KS C 5601-1987's original 2'350 precomposed Hangul syllables in the undersized Unicode-1.0:1991/ISO-10646:1993 code range [U+3400..U+4DFF] that is now being reused for the CJK Unified Ideographs Extension A.

Everson Mono

In 1995, the Irish Unicode activist [Michael Everson](#) started to draw a monospaced TrueType font (with Fontographer on his Mac) which he called [Everson Mono 10646](#). Latin, Greek, Cyrillic and Georgian are already covered and he said he intended to eventually cover all Unicode characters except for the Han ideographs. But due to system limitations his font does not exist as one handy file emono.ttf yet. The latest update appears to have been in 1996. And a license to use the font costs you 26 Irish pounds.

ETL Unicode

The developers of the [MULE](#) multilingual extension for [GNU Emacs](#) at the Japanese Electrotechnical Laboratories (ETL), most prominently Takahashi Naoto <ntakahas@etl.go.jp>, had crafted a nice line of public-domain ISO 8859 fonts in 7x14, 8x16, and 12x24 pixels height that are now included in the GNU intlfonts-1.0.tar.gz package.

In 1996, [Primoz Peterlin](#) took the initiative to rearrange their bitmaps into three monolithic Unicode fonts etl{14,16,24}-unicode.bdf, thereby reducing the redundant repetition of certain glyphs in fonts for each charset. He also used Mark Leisher's xmbdfed BDF editor to complete etl16-unicode.bdf to full coverage of the Minimal European Subset (MES) and far beyond (Extended Latin, Extended Cyrillic, Extended Greek, Armenian, Georgian). His result was placed in <ftp://ftp.x.org/contrib/fonts/> in early 1997.

After a discussion on comp.software.international,comp.std.internat,comp.windows.x,comp.fonts Primoz had chosen the [XLFD](#) (X11 Logical Font Description) label "-iso10646-1" for his Unicode fonts. This label has only now in 1998 been officially blessed into the [X11 registry](#) through [X11R6.4 public fix #02](#) by [Kaleb Keithley](#) upon [Markus Kuhn](#)'s and my humble initiative.

I find the name "etl-unicode" a misnomer because etl-unicode was not produced at ETL and is only based on the etl-fonts which were themselves based on older typefaces from kagotani@cs.titech.ac.jp and the ancient Lisp Machine's 20fg.ast. I also miss a commitment to extending and maintaining the font which is why I want to turn it into this "GNU Unifont" now.

Bitstream Cyberbit

The most impressive Unicode font so far is [Bitstream's cyberbit](#).ttf, a 13 MB serifed TrueType font covering all 20'902 Han ideographs besides basic Latin (no Vietnamese or IPA), Greek, Cyrillic, Arabic, Hebrew, Thai, and 1'153 of the Hangul syllables. It was released in 1997 for free download in the form of MS-DOS *.EXE archives (you can use the unzip utility on Unix to unpack the MS-DOS *.EXE) from <ftp.bitstream.com>. You

had to sign a license agreement restricting its use to a single copy. The Cyberbit [web page](#) was first [moved](#) and then removed and the [FTP directory](#) is now empty. The Cyberbit 1.1 font can still be found at mirror sites. Bitstream stopped giving away Cyberbit for free or as a retail product to single users in 1998. The extended Cyberbit 2.0 is only [sold](#) to big manufacturers who want to bundle the font with their software and pay royalties.

Lucida: √ξЭγ:ξ≈γ
uni16m:丙

uni24: ξ ж 丙

EversonMono
etl14:⌘ξ etl16:⌘ξ

etl24:⌘ξ

Cyberbit: ξж丙

Arial: ξж

Times: ξж

Courier: ξж

Monotype: ξж

ClearlyU: √ξЭγ:ξ≈γ

6x13: √ξЭγ:ξ≈γ

Microsoft's Fontpack

In 1998, just in time for the Euro, Microsoft offered a number of unencoded TrueType fonts (Times, Courier, Monotype, Arial, Verdana, etc.) for [free download](#) that cover little more than basic Latin, Greek, Cyrillic (more precisely OpenType's [WGL4](#) subset) but offer a pleasant variety of styles (serifed, monospaced, bold, italic).

ClearlyU Proportional

Probably because he finds a nice proportional font prettier or less limited than the ETL charcell fonts, [Mark Leisher](#) <mleisher@crl.nmsu.edu>, the author of the [xmbdfed](#) BDF font editor and very creative supporter of [Unicode on Unix platforms](#), crafted up his [cu12.bdf](#), a 17 pixel tall (12 point on 100 dpi) bitmap font that has a slightly wider coverage than the ETL font, for instance 250 basic

[Arabic presentation forms](#) since 1998-09-09. There is no webpage for this font, but it was announced in mailinglists and newsgroups and Mark encourages us to send him corrections.

6x13 fixed

Just while I was writing on this Unifont proposal, Markus Kuhn started a [6x13 font initiative](#) on 1998-09-23 to extend the default X11 font named -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1 also known under its alias names "6x13" and "fixed" into a partial -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1 and bring it up to WGL4 or SECS coverage. 6x13 was probably chosen the default font because $80 \times 6 = 480$, a line of 80 characters plus a scrollbar fit into a 500 pixel wide screen half so you can run two xterms next to one another. Markus encourages you to use xmbdfed to draw additional bitmaps and send back diff -u patches. He surprised himself: "I didn't expect that I can draw up to 100 glyphs per day and I initially completely overestimated the complexity of the task.... 100 glyphs per hour seems to be a reasonable speed for drawing low-res fonts. The font has now reached 1593 characters." That was after four days, a month later it had grown to 2581 characters. There is also a nice [demo screenshot](#).

Why is there no complete Unicode font yet?

There are several explanations why we don't have a complete Unicode font yet. The prime reason is of course the huge size of the Unicode repertoire with its 40'000+ defined characters. That number makes it a task of several man-months of fulltime work with a font editor to manually draw up all the glyphs from scratch. More intelligent techniques to derive some glyphs algorithmically from decomposition tables or mappings from existing fonts may speed things up a bit but will not be available for a large portion of the repertoire. [Bigelow & Holmes](#) quote Hippocrates' saying "Art is long, life is short."

Other huge East Asian character set standards came with reference implementations in the form of at least a bitmap font prepared by the national standardization body. The Unicode standard came as a book (ISBN [0-201-48345-9](#)) printed on paper with reference graphics for each character besides lots of descriptive text. These graphics were not provided in electronic form because the reference graphics are copyrighted by various private font designers each of whom provided theirs for the publication of the standard only. Perhaps there were also fears that Unicode could too easily be mistaken as a font had they been provided.

Another issue besides development time and the principal questions about "Unicode fonts" is their resource consumption. Even if we had a complete Unicode font, there are doubts whether people would want to waste space on their harddisks and processing time to load such a big baby containing mostly support for languages they don't even speak. I believe that a slim organization can reduce this problem and that it will disappear completely with growing hardware capacities. Nowadays, nobody is complaining that ISO-8859-1 doubled the font sizes and burdened British computers with silly German letters.

What do you mean by distributed effort?

Unicode is obviously too big and tiring for one or two persons to design a whole font for. Besides that, nobody is an expert in all the world's scripts so that it seems very natural to have people from all over the world to work on the parts they need in everyday use and they are most familiar with and merge their results. Until recently, no Unicode font had come with detailed instructions how to get updates and how to help updating it. In my opinion, this do-it-yourself must be made as simple as possible and refrain from overly complex procedures. I want to make it super-easy for anyone with a decent [GNU](#) system to add support for missing characters.

What do you mean by low quality?

I want to restrict the effort to one character cell bitmap font. Bitmap fonts may be considered an anachronistic technique as they do not scale well to random sizes and resolutions and have a coarse and stogy look. The same goes for character cell fonts that have to squeeze and stretch glyphs to fit into the character cell. But they still have some advantages: character cell bitmaps are the common denominator that can be used on almost all of today's computer platforms if fed the right way. They are also much easier to create. With bitmaps, you neither need expensive tools to draw up glyph outlines with splines or B zier curves in a 1000x1000 grid and hint them for rasterization on low resolutions nor the time to learn and apply these perfectionist tools. A bitmap is a mere matrix of white or black pixels. There is only a small degree of freedom how to draw an 'm' in a grid of 8x16 pixels so that you will not be dwelling on it for hours and progress much faster. Besides that, even with modern rasterizers like [FreeType](#) or [T1lib](#), hand-crafted bitmaps still often give better screen legibility and - unless replicated for various sizes and encodings - they are also a lot smaller in storage than the scalable outline descriptions. By going character cell, we retain the comfort of a checkerboard screen where you can do most of your layout in mail- and news-friendly plain text lines of less than 80 character cells. We can also do without complex size and adjustment information for each individual glyph and reduce the bitmaps to handy one-liners.

Do you intend to squeeze my favorite ideograph into an 8x16 pixel cell?

I did not say constant width! If we wanted the character cell to be big enough to also draw any ideograph with more than 4 vertical strokes in it, we would end up with cells at least 14 or 16 pixel wide. Such a huge cell would be way oversized for Latin letters and the usual row of 80 cells would no longer fit in a 640 pixel screen width.

Instead, I would like to follow the example of traditional CJK terminals like kterm or cterm and distinguish halfwidth (ASCII) characters occupying one cell each from fullwidth (CJK) characters occupying exactly two consecutive cells. This makes the font somehow proportional (more complex characters get more space) and fixed-pitch (the cell grid is respected) at the same time. Markus Kuhn calls this a "bi-width" font and suggests to register a SPACING "b" label for this.

The question whether a given character must be halfwidth or fullwidth becomes a bit harder to decide than it was with the [EUC](#) codes where each character encoded in two bytes also occupied two cells even if that looked crummy. The [Draft Unicode Technical Report 11](#) on East Asian Character Width does not do much more than to acknowledge the ambiguity for many characters.

8x16 appears to be the best choice for the character cell size since 8 and 16 are round numbers in the binary world and we can resort to existing 8x16 halfwidth and 16x16 fullwidth fonts, height 16 is tall enough to allow even stacked diacritics, and width 8 is just small enough to fit 80 halfwidth characters in one 640 pixel screen width.

Do we really need yet another font format?

I think that this little unifont project will benefit from a simple non-standard font description. My criticism of the standard BDF (Adobe's famous [Bitmap Distribution Format](#)) is that it is neither as human-readable and -editable as it said it would be (BDF fonts simply contain too many cryptic numbers which prompted the design of BDF editors like xfed, xfedor, xmbdfed in the first place) nor as fast and compressed as the Hanzi Bitmap Format HBF nor so handy that you can easily merge glyphs from various sources. I do not care to fetch mails through my modem containing 5 megabytes of unifont.bdf every time anybody has updated a couple of glyphs, nor to have `diff -c patches` fail because the context has changed in the meantime. HBF is not suited either because it swaps its bitmaps into external binary files that cannot be split, joined or edited with a plain text editor, and it requires special software installations.

My unifont file merely consists of lines each of which defines a glyph in the most trivial way: A hexadecimal number announces the Unicode character to be covered. And a colon separates it from the joined hexadecimal representation of the glyph's bitmap's horizontal scan lines:

```
0040:000000001C224A565252524E201E0000
0041:0000000018242442427E424242420000
4E21:000000007FFC010001003FF8210829282928292829282FE82828200820180000
```

Note that the final fullwidth glyph is easily distinguished from the halfwidth glyphs as its bit pattern is twice as long (64 hexdigits).

I can now pipe this compact hexadecimal representation through my [hexdraw](#) script:

```
#!/usr/local/bin/perl -p

sub unpack {
    local ($_) = @_;
    $_ = unpack ("B*", pack ("H*", $_));
    $width=length($_)/16;
    s/({$width})/\t$1\n/g;
    y/01/-#/;
    $_;
};

sub pack {
    local ($_) = @_;
    y/-#/01/;
    $_= unpack ("H*", pack ("B*", $_));
    y/a-f/A-F/;
    $_;
}

s/([0-9A-F]{64})/&unpack($1)/ie ||
s/([0-9A-F]{32})/&unpack($1)/ie ||
s/\t([-#]+)\n/&pack($1)/e;
```

which gives a more human-readable representation:

```
0040:  -----
      -----
      -----
```



```

-----
---###--
--#---#-
-#--#-#-
-#-#-##-
-#-#--#-
-#-#--#-
-#-#--#-
-#-#--#-
--#-###-
--#-----
---####-
-----
-----

```

```

0041:  -----
      -----
      -----
      -----
      ---##---
      --#---#-
      --#---#-
      -#-----#-
      -#-----#-
      -#####-
      -#-----#-
      -#-----#-
      -#-----#-
      -#-----#-
      -----
      -----

```

```

4E21:  -----
      -----
      -#####-
      -----#-----
      -----#-----
      -#####-
      -#-----#-----#-
      -#-#-#-#-#-#-
      -#-#-#-#-#-#-
      -#-#-#-#-#-#-
      -#-#-#-#-#-#-
      -#-#####-#-
      -#-#-----#-
      -#-----#-
      -#-----##-
      -----

```

This representation can then easily be edited with your favorite editor. No need to install any new software! Sure, [xmbdfed](#) is a nice font editor but I wouldn't want to dictate people to use it. I want to stay independent of xmbdfed. I feel much more in control of the data with my [GNU Emacs](#). I prefer to use keyboard functions instead of heavy mouse manoeuvring.

The result is then again piped through the [hexdraw](#) filter listed above to get back at the compact hexadecimal representation.

Older and newer hexadecimal representations can be merged with

```
cat new >> old
```

or with [hexmerge](#):

```
#!/usr/local/bin/perl
while (<>) { $G{$1}=$2 if /(....):(.+)\n/; }
for (sort keys %G) { print "$_: $G{$_}\n"; }
```

The hexadecimal representation can be converted to BDF with the following [hex2bdf](#) tool:

```
#!/usr/local/bin/perl

while (<>) { $glyph{$1} = $2 if /(....):(.+)\n/; }
@chars = sort keys %glyph; $[ = 1;
dbmopen (%charname, "/usr/share/unicode/unidata/charname.db", 0);

print "STARTFONT 2.1
FONT -gnu-unifont-medium-r-normal--16-160-75-75-c-80-iso10646-1
SIZE 16 75 75
FONTBOUNDINGBOX 16 16 0 -2
STARTPROPERTIES 2
FONT_ASCENT 14
FONT_DESCENT 2
ENDPROPERTIES
CHARS $#chars\n";

foreach $character (@chars)
{
    $encoding = hex($character); $glyph = $glyph{$character};
    $width = length ($glyph) > 32 ? 2 : 1;
    $dwidth = $width * 8; $swidth= $width * 500;
    $glyph =~ s/((.){$width})/\n$1/g;
    $character = "$character $charname"
        if $charname = $charname{pack("n", hex($character))};

    print "STARTCHAR U+$character
ENCODING $encoding
SWIDTH $swidth 0
DWIDTH $dwidth 0
BBX $dwidth 16 0 -2
BITMAP $glyph
ENDCHAR\n";
}

print "ENDFONT\n";
```

You can now install the resulting unifont.bdf in your X11 font directory, run `mkfontdir` and `xset fp rehash` and use it from within [yudit](#) with the following `~/.yuditrc` entry:

```
[Font Map Unifont]
Font Count=1
Font1=10646:gnu-unifont-iso10646-1
```

It is advisable to first convert the result into portable compiled format with

```
bdftopcf unifont.bdf > unifont.pcf
```

which reduces the font size from 4 to 2 MB and the initial `XLoadFont()` time from 30 seconds to 5 on my Pentium notebook with 100 MHz and 16 MB RAM. Doesn't anybody want to patch the [X11 server](#) to [mmap](#) such big font files into the memory and demand-page the needed glyphs?

I have not yet programmed a script to spit out the unifont in other formats: as a set of demand-loadable fontlets for use with 9term, as a Windows font, consolefont, or poor man's PostScript font but that ought to be achievable. The PostScript font would be a Type 3 bitmap font using the `imagemask` operator that could hopefully be plugged into [a2ps](#) or [enscript](#) to give some poor man's output resembling the good old dot matrix printer, but readable at least.

The simple one-liner textual format has a lot of merits as it allows us to use all the standard Unix text utilities to sort, join, compare or edit the font data.

You can easily `grep ^20AC unifont | hexdraw` to look at the euro sign, `grep ^26 unifont` for the miscellaneous symbols block, or `grep ^[0-2] unifont` to blank out the CJK stuff, `grep ':\.{64\$}' unifont` will filter the halfwidth characters away.

You can send around short additions or diffs:

To: unifont@czyborra.com
Subject: Dutch IJ

Hey Rome!

I think that the the current Dutch ligature IJ

```
0132:000000000772222222222222A2A760000
0133:000000022220066222222222222720418
```

looks more pleasant like this:

```
0132:0000000004242424242420202423C0000
0133:0000222200002222222222221A02221C
```

Piping this mail to [hexdraw](#) will decipher the graphical meaning. Try it as an exercise!

To extract the hexadecimal representation back out of any cell-padded BDF file I use a simple [bdfimplode](#):

```
#!/usr/local/bin/perl -n

if (/^ENCODING\s+(\d+)/) { printf ("%04X:", $1); }
elsif (/^BITMAP/) { $BITMAP=1; }
elsif (/^ENDCHAR/) { $BITMAP=0; print "\n"; }
elsif ($BITMAP) { y/a-f/A-F/; s/\n$//; print; }
```

That means that you can also use the standard BDF format to store and edit your copy of the unifont.

What characters are in there already?

In order to have a good basis to start with, I took the [etl16-unicode.bdf](#) and the GNU intlfonts-1.0 built in 1997 and merged them into the unifont.

etl16-unicode.bdf contained the mulefont's ISO 8859 glyphs by ntakahas@etl.go.jp, Armenian and Georgian by mleisher@crl.nmsu.edu and the full Extended Latin, Greek and Cyrillic completion by primoz.peterlin@biofiz.mf.uni-lj.si.

I added Thai from [thai-16.bdf](#) (typeface by Manop Wongsaisuwan), Lao from [mule-lao-16.bdf](#) (typeface by sihattha@jaist.ac.jp), halfwidth Katakana (Japanese syllabic) from Sony's [8x16rk.bdf](#), and Ethiopic in fullwidth from [ethiomx16f-uni.bdf](#) by Admas fisseha@cig.mot.com.

The CJK blocks were filled from [k16-1990.bdf](#) (typeface by kagotani@cs.titech.ac.jp), [gb16st.bdf](#) (Institute of Software, Academia Sinica), [hanglm16.bdf](#) (typeface by Daewoo), [cns-2-16.bdf](#) (Hong Kong University) as well as [jisksp16.bdf](#) and [taipei16.bdf](#) on 1998-09-16. I used the mapping tables from ftp.unicode.org to map them to Unicode positions.

I selected the Japanese font as the favourite-choice font because it seemed to have a consistent design with thin strokes and sufficient blank space at the cell margin and because the Japanese Kanji shapes enjoy a degree of

simplification that might be considered a compromise between the simplified Chinese and traditional Taiwanese extremes and because many Japanese have voiced fears that Unicode with Chinese hanzi will make their texts unreadable.

As I was particularly disturbed by the empty boxes for those missing characters, I went to draw glyphs for [modifier letters](#) and [combining characters](#) to complete the [U+0000..U+04FF] range, as well as control pictures and symbols to complete the [U+2000..U+26FF] range on 1998-09-24. Now I can type phonetics, mathematics, APL, box drawings, weather reports, chess and bridge boards in [Yudit](#) using the unifont.

[Jungshik Shin](#) immediately provided a [Perl script](#) to generate all the needed [11'172](#) precomposed [Hangul bitmaps](#) from the Johab-encoded [hanterm](#) font [jyagi16.bdf](#) on 1998-09-29. We are still thinking about thinning the bold typeface a bit to harmonize it with the rest of the unifont.

I added the [Basic Arabic presentation forms](#) on 1998-10-30. Arabic had high priority because Arabic is spoken in so many countries all over North Africa and the Middle East and I want the unifont to be usable with my [arabjoin](#) filter. I had learned that Arabic can also be represented in a character cell font from [AraMosaic's](#) authors Franck Portaneri and Fethi Amara in their article [Arabization of User Interfaces](#) in ISBN [ISBN 0-471-14965-9](#) and used their [8bit-encoded](#) charcell font naskhiRf12 as an example. While I was at it, I also added parts of the neighborhood: the four additional Persian letters, dotted Hebrew and Latin ligatures. I also threw in the Braille patterns generated by a quickly-hacked Perl [script](#).

The unifont's current coverage is:

128	U+0000..U+007F:Basic Latin
128	U+0080..U+00FF:Latin-1 Supplement
128	U+0100..U+017F:Latin Extended-A
156	U+0180..U+024F:Latin Extended-B
89	U+0250..U+02AF:IPA Extensions
57	U+02B0..U+02FF:Spacing Modifier Letters
72	U+0300..U+036F:Combining Diacritical Marks
105	U+0370..U+03FF:Greek
230	U+0400..U+04FF:Cyrillic
85	U+0530..U+058F:Armenian
82	U+0590..U+05FF:Hebrew
62	U+0600..U+06FF:Arabic
87	U+0E00..U+0E7F:Thai
65	U+0E80..U+0EFF:Lao
40	U+10A0..U+10FF:Georgian
348	U+1200..U+137F:Ethiopic
246	U+1E00..U+1EFF:Latin Extended Additional
233	U+1F00..U+1FFF:Greek Extended
77	U+2000..U+206F:General Punctuation
28	U+2070..U+209F:Superscripts and Subscripts
14	U+20A0..U+20CF:Currency Symbols
20	U+20D0..U+20FF:Combining Marks for Symbols
57	U+2100..U+214F:Letterlike Symbols
48	U+2150..U+218F:Number Forms
91	U+2190..U+21FF:Arrows
242	U+2200..U+22FF:Mathematical Operators
123	U+2300..U+23FF:Miscellaneous Technical
37	U+2400..U+243F:Control Pictures
11	U+2440..U+245F:Optical Character Recognition
139	U+2460..U+24FF:Enclosed Alphanumerics
128	U+2500..U+257F:Box Drawing
22	U+2580..U+259F:Block Elements
80	U+25A0..U+25FF:Geometric Shapes
106	U+2600..U+26FF:Miscellaneous Symbols
73	U+2700..U+27BF:Dingbats
256	U+2800..U+28FF:Braille Pattern Symbols
35	U+3000..U+303F:CJK Symbols and Punctuation

87	U+3040..U+309F:Hiragana
90	U+30A0..U+30FF:Katakana
37	U+3100..U+312F:Bopomofo
94	U+3130..U+318F:Hangul Compatibility Jamo
69	U+3200..U+32FF:Enclosed CJK Letters and Months
84	U+3300..U+33FF:CJK Compatibility
18174	U+4E00..U+9FFF:CJK Unified Ideographs
11172	U+AC00..U+D7A3:Hangul Syllables
270	U+F900..U+FAFF:CJK Compatibility Ideographs
57	U+FB00..U+FB4F:Alphabetic Presentation Forms
12	U+FB50..U+FDFF:Arabic Presentation Forms-A
4	U+FE20..U+FE2F:Combining Half Marks
28	U+FE30..U+FE4F:CJK Compatibility Forms
26	U+FE50..U+FE6F:Small Form Variants
140	U+FE70..U+FEFF:Arabic Presentation Forms-B
171	U+FF00..U+FFEF:Halfwidth and Fullwidth Forms
2	U+FFF0..U+FFFF:Specials

What characters are missing?

I stopped my drawing activities in the middle of the U+27xx Dingbats block because the attempt to distinguish more than 30 different star shapes in an 8x16 bitmap seemed hopeless, but they're probably not that necessary for now.

The more important and originally missing Hangul syllables and basic Arabic and Hebrew presentation forms were quickly added in October 1998. However, support for these scripts isn't complete yet: extended Arabic letters and ligatures, Hangul jamos and halfwidth letters still have to be added.

The entire CJK range could use some proofreading, optimizing and completing. I am waiting for an opinion from [Bitstream Inc.](#) whether we may complete the Unified Han ideograph range with [2728 miniature bitmaps](#) derived from [Cyberbit](#) 1.1 using [ttf2bdf](#) and a primitive [padcell](#) postprocessor.

I also think it would be helpful to have character glyphs for Devanagari and other Brahmi scripts even if the acceptability of simple rendering with character glyphs is extremely questionable.

Some newly added scripts like [Ethiopic](#) or [Braille](#) and characters like the [€](#) and [&quad](#); are already in the unifont but it would be megacool to support more of the [recently added](#) stuff like the Canadian Aboriginal scripts.

The GNU unifont supports no characters [beyond U+FFFF](#) like Etruscan, Klingon, or the musical symbols yet but I have started a [plane+01.hex.gz](#). The unifont converters are not prepared to accept character identifiers with five instead of four hexadecimal digits names like "10200" for U-00010200 ETRUSCAN LETTER and produce separate fonts for the additional planes yet.

Will there be a bold/italic/large/small/Taiwanese version?

I don't think so. I only want one maximally complete font to begin with. An attempt to maintain the same degree of completeness in parallel lines of 14, 16, and 24 pixels height would more than triple development time and storage costs and distract forces. (Primoz Peterlin offered to try to bring the 14- and 24-point sizes in sync with the 16-point size, though.) Attempts to add parallel bold and italic versions would also double the costs while their highlighting functions can much cheaper be simulated through the use of colors or inversion. Or bold can generated through shifted overstriking (doubling each black pixel to the right) like xterm does it.

I don't want to forbid you to use your favorite size font, I just want to have an option to occasionally switch into a font that is reliable to display all characters. I suggest to first make one complete and legible unisex font as a variation of the "First make it correct, then make it fast" principle. This is not going to be a high-quality

typographic font. Let us use compromise glyphs that make the intended meaning clear rather than be perfect for your particular language. But I welcome everybody to improve them.

If there are CJK-unified ideographs that absolutely have to be offered in different shapes for Chinese or Taiwanese legibility, I would appreciate the creation of separate lists of them so that interested parties can easily merge localized glyphs into their unifont.

How can I download the unifont?

You can download the entire [unifont.hex.gz](http://czyborra.com/unifont/unifont.hex.gz) from below and play with it. Or you can use the [unifont.cgi](http://czyborra.com/unifont/unifont.cgi) to download parts of the font or convert them into your favorite format. Add a regular expression between slashes to the unifont.cgi URL to specify the subset you want to grep for, for example use <http://czyborra.com/unifont/unifont.cgi/^20A/> to get at the Currency symbols block. Append "[draw](#)" if you want a visual impression or "[bdf](#)" if you want BDF format. Append ".[gz](#)" if your browser can handle [gzip](#) compression.

How can I receive the latest updates?

Most updates to the GNU unifont shall be small and you probably won't want to download the entire font over and over again whenever a mere couple of glyphs have been added or changed. That's why I want to distribute small **incremental updates**.

I'll try to keep track of the **change history** in the [coverage section](#) of this page and offer links to download the updated parts but it may turn out too cumbersome to mention every smaller change. There is now a new [updates directory](#) containing small files with the latest new glyphs in chronological order. I do not have the disk space to place my revision control (RCS) file unifont.hex,v online but feel free to start your own at home.

There is no public mailinglist unifont@gnu.org yet. You can send a message to unifont-request@czyborra.com to announce your interest. For the time being, you can also use the [URL-minder](#) to get alarmed of changes.

How can I add characters to the GNU unifont?

While using the GNU unifont to display Unicode text you may encounter characters that are not yet covered by the font or whose glyphs look worse than necessary. Accept this as a calling to solve the problem yourself:

Decide whether you want to add or change just one glyph or a whole group of related glyphs in consistent design: how much spare time do you have?

Decide what outside reference glyphs may help you draw acceptable glyph shapes for your Unicode characters: try looking at existing fonts covering your characters if any exist, as well as the ISO-10646 standard and the Unicode book or the code chart GIFs from charts.unicode.org.

Decide whether your glyphs fit into halfwidth cells of 8x16 or whether they require fullwidth squares: will your glyphs have to distinguish more than 4 vertical strokes?

Decide (unless you can generate your glyphs algorithmically) whether you want to use a text editor or a graphical bitmap font editor: investing some time into installing and learning xmbdfed pays off through more comfortable editing functions in bigger projects.

Decide if it would help to join in the official character names from the Unicode character database so you don't have to switch your attention between various tables and code charts indexed by non-mnemonic character

numbers and your editor but can simply fill in templates like

2555:00000000000000008000000000000000:BOX DRAWINGS DOWN SINGLE AND LEFT DOUBLE

Decide what unifont glyphs you want to use as examples for style and metrics: at which columns and rows did others put their top, bottom, sides, center strokes and diacritics? The ETL Latin typeface generally leaves the bottom two rows for hooks and diacritics extending below the baseline and four rows above a capital letter for diacritics. The one-pixel margin to the left and the right is more easily given up on the right than on the left.

Decide which parts of the unifont you want to load into your editor: do you have enough free RAM to load the entire halfwidth or the entire fullwidth subset into xmbdfed or do you only want to extend a small subset with minimal resource consumption?

Finally, draw your glyphs, save them, extract their [hexadecimal representation](#) and mail it to unifont@czyborra.com.




















Besides drawing glyphs, you can help by writing programs to make further use of the unifont or by constructively criticizing this web page.

Thanks

Thanks go out to all the persons mentioned above for their fine works and to Mark Crispin and Professor Biedl (my thesis advisor) for their words of encouragement.

[Roman Czyborra](#) <roman@czyborra.com>

1998-09-29 ... \$Date: 1998/11/07 16:02:43 \$

Name	Last modified	Size	Description
 Parent Directory		-	
 HEADER.html	1998-11-29 12:15	44K	
 autoinfix.hs.html	2014-07-07 16:47	26K	
 autoinfix.hs.html.pdf	2014-07-07 16:53	710K	
 b.annuss.pdf	2014-09-09 06:08	32K	
 banner.gif	1998-12-11 15:06	2.5K	
 banner.txt	1998-10-03 17:02	474	
 bdf2hex	1998-09-15 16:59	204	
 bdfimplode	1998-09-15 16:59	204	
 braille.pl	2003-02-15 20:52	681	
 chessboard.txt	1998-11-07 23:18	579	
 eurobsdcon2016-utf8.pdf	2016-09-27 10:08	3.1M	
 fullwidth.pl	1998-11-06 03:57	1.3K	
 gif2bdf	1998-11-09 09:15	873	
 hex2bdf	1998-09-29 10:23	860	
 hex2sfd	2005-10-27 18:30	1.9K	
 hexdraw	1998-09-15 16:50	362	
 hexmerge	1998-09-16 13:28	236	
 iyagi16.bdf.gz	1994-03-07 12:11	6.7K	

 johab.pl	1998-12-22 03:47	5.6K
 johabthin.hex.gz	1998-12-22 04:23	2.4K
 mirrored	1998-11-08 11:11	850
 missinghan.hex.gz	1998-10-07 20:12	72K
 padcell	1998-10-28 14:19	1.0K
 plane+0E.hex.gz	1998-11-08 11:50	1.0K
 plane+01.hex.gz	1998-11-06 08:33	378
 typefaces.gif	1998-10-03 20:00	2.8K
 uni16m.gif	1998-09-27 19:52	545
 unifont-7.0.06/	2014-11-10 13:13	-
 unifont.cgi	1998-10-02 01:14	527
 unifont.hex.gz	2000-01-22 21:27	589K
 unifont.pdf	2011-12-17 13:43	89K
 unifont.png	2007-09-27 13:35	606K
 unifont.png.mbox	2007-09-27 22:46	45K
 unifont.ttf	2014-07-07 16:57	12M
 updates/	2014-07-07 16:58	-
 xmbdfed.gif	1998-11-23 18:00	10K

Apache Server at czyborra.com Port 80