Unifont

Paul Hardy
This manual describes Unifont, a bitmap-based font covering the Unicode Basic Multilingual Plane and beyond, along with Unifont’s utility programs.

Copyright © 2023 Paul Hardy.

hex2otf section Copyright © 2023 何志翔 (He Zhixiang)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts and no Back-Cover Texts.
# Table of Contents

1 Introduction .......................................................... 1

2 Tutorial ................................................................. 2
   2.1 Unicode ............................................................ 2
   2.2 Unifont Structure ................................................ 2
   2.3 Hex File Format ................................................... 2
   2.4 Using Graphical Tools ............................................ 4
   2.5 Using Hexdraw ..................................................... 6
   2.6 Checking Coverage ............................................... 7
   2.7 Custom Builds ..................................................... 8
   2.8 OpenType .......................................................... 9
   2.9 One Glyph per File ............................................... 10
   2.10 Viewing a Unifont File Interactively ......................... 10
   2.11 Seeing the Big Picture (Literally!) ......................... 10
   2.12 Combining Circles ............................................... 11
   2.13 Installing Fonts on GNU/Linux .............................. 12
   2.14 Creating a Brand New Font .................................... 12
   2.15 Updates to Unicode ............................................ 13

3 Reference .............................................................. 14
   3.1 bdfimplode ....................................................... 14
       3.1.1 bdfimplode NAME ........................................ 14
       3.1.2 bdfimplode SYNOPSIS ................................ 14
       3.1.3 bdfimplode DESCRIPTION .............................. 14
       3.1.4 bdfimplode FILES ...................................... 14
       3.1.5 bdfimplode SEE ALSO .................................. 14
       3.1.6 bdfimplode AUTHOR .................................... 14
       3.1.7 bdfimplode LICENSE .................................... 14
       3.1.8 bdfimplode BUGS ....................................... 14
   3.2 hex2bdf ............................................................ 14
       3.2.1 hex2bdf NAME ............................................ 14
       3.2.2 hex2bdf SYNOPSIS ................................... 14
       3.2.3 hex2bdf DESCRIPTION ................................. 15
       3.2.4 hex2bdf OPTIONS ...................................... 15
       3.2.5 hex2bdf EXAMPLE .................................... 15
       3.2.6 hex2bdf FILES ......................................... 15
       3.2.7 hex2bdf SEE ALSO ................................... 15
       3.2.8 hex2bdf AUTHOR ....................................... 15
       3.2.9 hex2bdf LICENSE ...................................... 15
       3.2.10 hex2bdf BUGS ........................................ 15
   3.3 hex2otf ............................................................. 16
       3.3.1 hex2otf NAME ........................................... 16
3.12.6 unibmp2hex FILES ........................................... 28
3.12.7 unibmp2hex SEE ALSO ...................................... 28
3.12.8 unibmp2hex AUTHOR ......................................... 28
3.12.9 unibmp2hex LICENSE ......................................... 28
3.12.10 unibmp2hex BUGS ........................................... 28
3.13 unibmpbump ......................................................... 29
  3.13.1 unibmpbump NAME ........................................... 29
  3.13.2 unibmpbump SYNOPSIS ....................................... 29
  3.13.3 unibmpbump DESCRIPTION ..................................... 29
  3.13.4 unibmpbump OPTIONS ......................................... 29
  3.13.5 unibmpbump FILES ........................................... 30
  3.13.6 unibmpbump SEE ALSO ....................................... 30
  3.13.7 unibmpbump AUTHOR ......................................... 30
  3.13.8 unibmpbump LICENSE ......................................... 30
  3.13.9 unibmpbump BUGS ........................................... 30
3.14 unicoverage ....................................................... 30
  3.14.1 unicoverage NAME ........................................... 30
  3.14.2 unicoverage SYNOPSIS ....................................... 30
  3.14.3 unicoverage DESCRIPTION ..................................... 30
  3.14.4 unicoverage OPTIONS ......................................... 30
  3.14.5 unicoverage EXAMPLE ......................................... 31
  3.14.6 unicoverage FILES ........................................... 31
  3.14.7 unicoverage SEE ALSO ....................................... 31
  3.14.8 unicoverage AUTHOR ......................................... 31
  3.14.9 unicoverage LICENSE ......................................... 31
  3.14.10 unicoverage BUGS .......................................... 31
3.15 unidup ............................................................. 31
  3.15.1 unidup NAME ................................................. 31
  3.15.2 unidup SYNOPSIS ............................................. 31
  3.15.3 unidup DESCRIPTION .......................................... 31
  3.15.4 unidup FILES ................................................ 32
  3.15.5 unidup SEE ALSO ............................................. 32
  3.15.6 unidup AUTHOR .............................................. 32
  3.15.7 unidup LICENSE .............................................. 32
  3.15.8 unidup BUGS ................................................ 32
3.16 unifont-viewer .................................................. 32
  3.16.1 unifont-viewer NAME ......................................... 32
  3.16.2 unifont-viewer SYNOPSIS ..................................... 32
  3.16.3 unifont-viewer DESCRIPTION .................................. 32
  3.16.4 unifont-viewer FILES .......................................... 32
  3.16.5 unifont-viewer SEE ALSO ..................................... 33
  3.16.6 unifont-viewer AUTHOR ....................................... 33
  3.16.7 unifont-viewer LICENSE ....................................... 33
  3.16.8 unifont-viewer BUGS .......................................... 33
3.17 unifont1per ...................................................... 33
  3.17.1 unifont1per NAME ........................................... 33
  3.17.2 unifont1per SYNOPSIS ....................................... 33
3.22  unigencircles ................................................. 40
     3.22.1  unigencircles NAME ............................... 40
     3.22.2  unigencircles SYNOPSIS ......................... 40
     3.22.3  unigencircles DESCRIPTION .................... 40
     3.22.4  unigencircles EXAMPLE ......................... 41
     3.22.5  unigencircles FILES ............................. 41
     3.22.6  unigencircles SEE ALSO ......................... 41
     3.22.7  unigencircles AUTHOR .......................... 41
     3.22.8  unigencircles LICENSE ......................... 41
     3.22.9  unigencircles BUGS ............................. 41
3.23  unigenwidth ............................................... 41
     3.23.1  unigenwidth NAME .............................. 41
     3.23.2  unigenwidth SYNOPSIS ......................... 41
     3.23.3  unigenwidth DESCRIPTION .................... 41
     3.23.4  unigenwidth EXAMPLE ......................... 42
     3.23.5  unigenwidth FILES ............................. 42
     3.23.6  unigenwidth SEE ALSO ......................... 42
     3.23.7  unigenwidth AUTHOR .......................... 42
     3.23.8  unigenwidth LICENSE ......................... 42
     3.23.9  unigenwidth BUGS ............................. 42
3.24  unihex2bmp ................................................ 42
     3.24.1  unihex2bmp NAME .............................. 42
     3.24.2  unihex2bmp SYNOPSIS ......................... 42
     3.24.3  unihex2bmp DESCRIPTION .................... 42
     3.24.4  unihex2bmp OPTIONS ......................... 43
     3.24.5  unihex2bmp EXAMPLE ......................... 43
     3.24.6  unihex2bmp FILES ............................. 43
     3.24.7  unihex2bmp SEE ALSO ......................... 43
     3.24.8  unihex2bmp AUTHOR .......................... 43
     3.24.9  unihex2bmp LICENSE ......................... 43
     3.24.10 unihex2bmp BUGS ................................. 43
3.25  unihex2png ................................................ 43
     3.25.1  unihex2png NAME .............................. 43
     3.25.2  unihex2png SYNOPSIS ......................... 44
     3.25.3  unihex2png DESCRIPTION .................... 44
     3.25.4  unihex2png OPTIONS ......................... 44
     3.25.5  unihex2png EXAMPLE ......................... 44
     3.25.6  unihex2png FILES ............................. 44
     3.25.7  unihex2png SEE ALSO ......................... 44
     3.25.8  unihex2png AUTHOR .......................... 44
     3.25.9  unihex2png LICENSE ......................... 45
     3.25.10 unihex2png BUGS ................................. 45
3.26  unihexfill ................................................ 45
     3.26.1  unihexfill NAME .............................. 45
     3.26.2  unihexfill SYNOPSIS ......................... 45
     3.26.3  unihexfill DESCRIPTION .................... 45
     3.26.4  unihexfill OPTIONS ......................... 45
### 3.26.5 `unihexfill` FILES

- `files` section starts...  

### 3.26.6 `unihexfill` EXAMPLE

- `example` section starts...

### 3.26.7 `unihexfill` SEE ALSO

- `see also` section starts...

### 3.26.8 `unihexfill` AUTHOR

- `author` section starts...

### 3.26.9 `unihexfill` LICENSE

- `license` section starts...

### 3.26.10 `unihexfill` BUGS

- `bugs` section starts...

### 3.27 `unihexgen`

- `name` section starts...

### 3.27.1 `unihexgen` NAME

- `name` section starts...

### 3.27.2 `unihexgen` SYNOPSIS

- `synopsis` section starts...

### 3.27.3 `unihexgen` DESCRIPTION

- `description` section starts...

### 3.27.4 `unihexgen` OPTIONS

- `options` section starts...

### 3.27.5 `unihexgen` FILES

- `files` section starts...

### 3.27.6 `unihexgen` EXAMPLE

- `example` section starts...

### 3.27.7 `unihexgen` SEE ALSO

- `see also` section starts...

### 3.27.8 `unihexgen` AUTHOR

- `author` section starts...

### 3.27.9 `unihexgen` LICENSE

- `license` section starts...

### 3.27.10 `unihexgen` BUGS

- `bugs` section starts...

### 3.28 `unihexpose`

- `name` section starts...

### 3.28.1 `unihexpose` NAME

- `name` section starts...

### 3.28.2 `unihexpose` SYNOPSIS

- `synopsis` section starts...

### 3.28.3 `unihexpose` DESCRIPTION

- `description` section starts...

### 3.28.4 `unihexpose` FILES

- `files` section starts...

### 3.28.5 `unihexpose` SEE ALSO

- `see also` section starts...

### 3.28.6 `unihexpose` AUTHOR

- `author` section starts...

### 3.28.7 `unihexpose` LICENSE

- `license` section starts...

### 3.28.8 `unihexpose` BUGS

- `bugs` section starts...

### 3.29 `unihexrotate`

- `name` section starts...

### 3.29.1 `unihexrotate` NAME

- `name` section starts...

### 3.29.2 `unihexrotate` SYNOPSIS

- `synopsis` section starts...

### 3.29.3 `unihexrotate` DESCRIPTION

- `description` section starts...

### 3.29.4 `unihexrotate` OPTIONS

- `options` section starts...

### 3.29.5 `unihexrotate` EXAMPLES

- `examples` section starts...

### 3.29.6 `unihexrotate` FILES

- `files` section starts...

### 3.29.7 `unihexrotate` SEE ALSO

- `see also` section starts...

### 3.29.8 `unihexrotate` AUTHOR

- `author` section starts...

### 3.29.9 `unihexrotate` LICENSE

- `license` section starts...

### 3.29.10 `unihexrotate` BUGS

- `bugs` section starts...

### 3.30 `unijohab2html`

- `name` section starts...

### 3.30.1 `unijohab2html` NAME

- `name` section starts...

### 3.30.2 `unijohab2html` SYNOPSIS

- `synopsis` section starts...

### 3.30.3 `unijohab2html` DESCRIPTION

- `description` section starts...

### 3.30.4 `unijohab2html` OPTIONS

- `options` section starts...

### 3.30.5 `unijohab2html` EXAMPLES

- `examples` section starts...

### 3.30.6 `unijohab2html` FILES

- `files` section starts...

### 3.30.7 `unijohab2html` SEE ALSO

- `see also` section starts...

### 3.30.8 `unijohab2html` AUTHOR

- `author` section starts...

### 3.30.9 `unijohab2html` LICENSE

- `license` section starts...
1 Introduction

This document describes the process of using the GNU Unifont utilities to create a font. The steps described in the “Using Graphical Tools” section in the “Tutorial” chapter are more or less the steps that I (Paul Hardy) followed to add thousands of glyphs to GNU Unifont, except that I didn’t have the luxury of just typing make to make a new font while adding those glyphs in the beginning.

I streamlined the font build process after I was done drawing the Unicode 5.1 glyphs.


DISCLAIMER: Donald Knuth warned in his Metafont book that if someone started designing type, they would never again be able to look at a page of text normally and just read its content. There is a point of no return beyond which a serious font designer begins looking at how individual letters in a font on a page are drawn, and how they might be improved. Be warned!

— Paul Hardy (unifoundry@unifoundry.com) 2008, 2013
2 Tutorial

This chapter provides a step-by-step tutorial on using the Unifont utility programs to modify a font in the GNU Unifont format.

2.1 Unicode

Unicode is an international standard to encode all the world’s scripts with one universal scheme. Unicode is the default encoding for web pages and is gaining popularity in many other applications. To learn more about Unicode, look at code charts, and see the latest developments, check out http://unicode.org

Unifont follows the Unicode encoding scheme. Unicode defines the numeric value of a character, but does not define one particular font. There can be (and are) many fonts that support a subset of Unicode characters.

In 1998, Roman Czyborra observed that there was still no font, free or commercial, with complete Unicode coverage. He envisioned a low-quality bitmapped font as an easy way to produce a font that covered much of the Unicode standard.

2.2 Unifont Structure

GNU Unifont is a bitmapped pixel font, which is also converted to an outline TrueType font. Roman Czyborra began this font in 1998 with a goal of having one glyph rendered for each visible character in the Unicode Basic Multilingual Plane (Plane 0, the first 65,536 characters). His original writing on this is at http://czyborra.com/unifont/.

(Note that the term “character” is used very loosely here for simplicity; the Unicode Standard has a stricter definition of what constitutes a character.)

The font is dual-width. Each character is 16 pixels tall, and either 8 or 16 pixels wide. The characters are stored in a unique .hex file format invented by Roman Czyborra as a convenient way of giving each character exactly a one line specification. Conversion between this .hex format and Bitmap Distribution format (BDF) font format is trivial.

Glyphs can actually be 8, 16, 24, or 32 pixels wide, but this extended capability should be considered experimental. Prior to the release of Unicode 10.0.0, hooks were put in place in some of the Unifont utilities to allow for the possibility of future fonts to represent complex glyphs such as Cuneiform. Unicode 10.0.0 created a need for quadruple-width glyphs (31 pixels wide, encoded as 32 pixels wide) for some historic Chinese ideographs assigned to the Plane 0, also referred to as the Basic Multilingual Plane (BMP); that currently is the only place glyphs other than single- or double-width are used. As a result, those Unifont utilities that support quadruple-width glyphs have updated descriptions in the reference section of this document.

2.3 Hex File Format

By convention, files containing the Unifont native font format have the extension “.hex”. Their format is extremely simple, consisting of two fields separated with a colon (“:”) and ending with a newline.
The first field is the Unicode code point, in hexadecimal. For all Plane 0 code points, this is a four digit hexadecimal number. Hexadecimal digits are (in order) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The Unicode Standard uses a hexadecimal number to assign each character a location. These locations are called “code points” and their range is 0 through 10FFFF, inclusive.

The range 0000 through FFFF, inclusive, is called the Basic Multilingual Plane (BMP), or Plane 0. This plane contains glyphs for most of the world’s modern writing scripts.

Unifont utilities support glyphs across the entire Unicode range. The current distribution includes glyphs for Unicode’s Plane 0, Plane 1 (the Supplementary Multilingual Plane, or SMP), and beyond. Coverage of the SMP is only partial.

The first field in a .hex file should be either four digits long for the Basic Multilingual Plane, or six digits long for higher Unicode planes, following the convention of the Unicode Standard.

The second field is a string of hexadecimal digits. There are 32 digits for a character that is 8 pixels wide, and 64 digits for a character that is 16 pixels wide.

The good news is you don’t have to worry about these long digit strings. Roman Czyborra wrote a utility, hexdraw, to convert .hex fonts to a form that can be edited with a plain text editor, then converted back into .hex format.

Paul Hardy wrote two utilities to do the same thing except with bitmapped graphics images for editing with a graphics editor: unihex2bmp converts a block of 256 characters into a graphics file, and unibmp2hex converts such a graphics file back into .hex format. These bitmaps display the 256 characters in a block arranged in a 16 by 16 character grid. The graphics editor must maintain the image as a monochrome (black and white) file, with one bit per pixel. After conversion from a .bmp file back to a .hex file, the next step is conversion to a BDF font file. A BDF file can only encode a pixel being on or off (i.e., black or white only with no intermediate shades of gray).

Andrew Miller later converted unihex2bmp and unibmp2hex to Perl, transforming them into unihex2png and unipng2hex, respectively. These programs convert Unifont .hex files to and from Portable Network Graphics files.

The unihex2png, unipng2hex, and hexdraw programs handle the full Unicode code point range of 0x00000000 through 0x10FFFF. The unihex2bmp and unibmp2hex programs support the full 32-bit unsigned integer range of 0x00000000 through 0xFFFFFFFF, but have not been tested extensively beyond the Unicode upper limit of 0x10FFFF. The range of the C programs might be truncated in the future to only cover to 0x10FFFF, the limit of the Unicode code point space.

Rebecca Bettencourt has created two free software programs on her KreativeKorp.com website that are well-suited for Unifont editing: Bits’n’Picas and PowerPaint. The Bits’n’Picas program can read, directly graphically edit, and write native Unifont .hex format files. The PowerPaint program can also edit bitmaps, and is compatible with Unifont’s unibmp2hex and unipng2hex utilities. For more information, visit Rebecca’s software page on her website at https://www.kreativekorp.com/software/. Kreative License: Bits’n’Picas and PowerPaint are dual-licensed under the Mozilla Public License version 1.1 and the GNU Lesser General Public License version 3.
2.4 Using Graphical Tools

Let’s look at an example. Suppose you want to modify the Coptic letters in the range U+2C80..U+2CFF (“U+” is Unicode shorthand). These letters are in the upper half of the block U+2C00..U+2CFF. The Unicode utilities in this package refer to this as “page” 2C. “Page” is not a Unicode term — it is just a term unique to this package to refer to a block of 256 code points/characters. The equivalent Unicode term is “row”; however, that term could cause confusion with a row being 16 code points in a code chart; hopefully the term “page” is less ambiguous.

The steps to follow will be:

1. Convert the .hex version of the page 2C range as a 16 by 16 bitmapped grid.
2. Modify the bitmap in any graphics editor, being careful to re-save it as a Windows Bitmap (.bmp) or Wireless Bitmap file when finished.
3. Convert the modified bitmap back into a .hex font file.
4. Merge the results with the original unifont.hex file (or whatever its name might be).
5. Run unidup on the resulting sorted file to guard against duplicate character definitions.
6. Create the new bitmapped version of the font.
7. Check the compiled font for duplicates.
8. If there are duplicates, remove them and go back to Step 5.
9. Create the new TrueType version or other versions of the font.

Note that Rebecca Bettencourt’s Bits’n’Picas program (mentioned in the previous section) can cover Steps 1 through 3.

If the script has combining characters (such as accent glyphs), also add their code points to the proper *combining.txt file in the directory for the corresponding Unicode plane. That way, when the font is converted to TrueType those glyphs will be correctly positioned. For a script with combining characters, all glyphs that can appear with combining characters must have the same width so that the combining characters will be properly positioned.

**Step 1:** Convert the .hex range into a bitmap grid. Assuming our font file is named unifont.hex, type

```
unihex2bmp -p2C < unifont.hex > uni2C.bmp
```

**Step 2:** Modify uni2C.bmp with your favorite graphics editor. Note that whatever graphics editor you use should ideally preserve the file as a black and white bitmap (monochrome), with one bit per pixel. However, unibmp2hex will convert a 24- or 32-bit RGB (Red, Green, and Blue) image back into a .hex file.

During editing, you can draw guidelines outside the actual 16x16 font pixel area; they will be ignored when converting back into .hex format. You can also erase the grid borders between code points on purpose or by accident, and it will have no effect on the generated .hex file. Just don’t erase the code point numbers on the outer edges of the grid. The conversion from .bmp back to .hex only looks at the “U+0000” in the upper left-hand corner of the bitmap graphic and other code point numbers, and at each code point’s 16x16 pixel area inside its 32x32 pixel grid area. Every other artifact in the final graphics file outside these areas is ignored.
If a new version of Unicode adds glyphs to a page that were previously unassigned, be sure to remove the newly-assigned code points from the corresponding unassigned.hex file because the code point is no longer unassigned.

**Step 3:** Convert the edited .bmp file back into .hex format:

```
unibmp2hex < uni2C.bmp > uni2C.hex
```

Note that the conversion from a bitmap image to a .hex file cannot distinguish between a legitimate single- or double-width space character and a code point that does not have an assigned value. Therefore, space glyphs are separately contained in the spaces.hex file.

**Step 4:** Merge the results with the original unifont.hex file. This requires several sub-steps:

- Edit the original unifont.hex file and delete the lines that begin with “2C”.
- Insert the uni2C.hex file into unifont.hex, either with a text editor such as emacs or vi, or with a GNU/Linux command such as:

```
sort uni2C.hex unifont.hex > new-unifont.hex
```

This second option (using sort) is preferred, because unidup (in Step 5) might miss duplicate code points if your final result isn’t in proper order.

**Step 5:** Make sure there are no duplicates with unidup:

```
unidup < unifont.hex
```

or

```
unidup < new-unifont.hex
```

depending on the name of your final font file. If there is no output, your modified font contains no duplicates.

This editing is best done on an input .hex file, such as unifont-base.hex.

**Step 6:** Create the new bitmapped version of the font. In the font/ directory, type

```
make hex
```

**Step 7:** Check the compiled font for duplicates. Change to the font/compiled/ directory and run

```
unidup < mynewfontfile.hex
```

for whatever font file you created.

**Step 8:** If there are duplicates, remove them in the font/ directory and go back to Step 5.

**Step 9:** Create the new TrueType version of the font and all other bitmapped versions. From the font/ directory, type

```
make distclean && make
```

Then be prepared to wait a long time unless you are using a computer with plenty of RAM and CPU horsepower. Your computer should have at least 256 Megabytes of virtual memory (RAM), and at least 500 Megabytes of free disk space.

To only create a BDF font, in the font/ directory just type

```
make bdf
```

To only create a BDF and PCF font, in the font/ directory type

```
make pcf
```

Creating a BDF font is the first step in creating a PCF font (not counting generating the compiled master “.hex” input file). BDF fonts can be created just with the tools in this
Chapter 2: Tutorial

package. PCF fonts are created by running `bdftopcf` on the BDF font. TrueType fonts require FontForge.

The Unifont package also includes two programs for working with Portable Network Graphics (PNG) files instead of BMP files. These utilities are `unihex2png` and `unipng2hex`. They work in a similar manner to the corresponding programs `unihex2bmp` and `unibmp2hex`, respectively.

To use `unihex2png` instead of `unihex2bmp`, continuing the example of the Coptic script in the U+2Cxx range, type:

```
unihex2png -p 2C -i unifont.hex -o uni2C.png
```

Note that with `unihex2bmp` specifying input and output files is optional, while with `unihex2png` at least the PNG filename must be given explicitly. More specifically, `unihex2png` will read a .hex file format input from STDIN if no “-i” argument is specified, but the name of the binary PNG file must always be specified with the “-o” option.

Then edit the resulting PNG file to your heart’s content. When done, convert the file back into a `unifont.hex` format file. In this example, type:

```
unipng2hex -i uni2C.png -o uni2C.hex
```

Similar to `unihex2png`, the binary PNG file must be specified with “-i” but the .hex format file will be written to STDOUT if the “-o” option is omitted.

Finally, merge your changes in with your main .hex font file as described previously in this section.

### 2.5 Using Hexdraw

Roman Czyborra’s original utility to edit glyphs is the `hexdraw` Perl script. Using the same script as in the previous chapter, Coptic, here are the steps for modifying `unifont.hex` using `hexdraw`.

First, realize that Unifont has tens of thousands of glyphs (characters, using the term character loosely). In this example, out of the tens of thousands of glyphs, we want to modify the range U+2C80..U+2CFF (only 128 glyphs).

The range U+2C80..U+2CFF could be extracted from `unifont.hex` by using the `egrep` utility to look for lines beginning with “2C8” through “2CF”, or that range could be isolated by copying `unifont.hex` into another file, and deleting all lines except the desired range.

The following steps will probably minimize typographical errors, but they aren’t the only way.

1. “Grep” the desired block of 256 glyphs (using the `grep` utility) and convert this into a text representation for editing.
2. Edit the block with a text editor, such as `emacs` or `vi`.
3. Convert the edited text file back into .hex format.
4. Delete the edited block range from the original font file.
5. Merge the two .hex files into one file.
6. Check for duplicates with `unidup`.
7. Generate new fonts as described in the “Using Graphical Tools” section above.
Step 1: Extract the desired block with grep:
```
grep `'^2C'' unifont.hex | hexdraw > uni2C.txt
```
Step 2: Edit uni2C.txt with a text editor.
Step 3: Convert the text file back into .hex format:
```
hexdraw < uni2C.txt > uni2C.hex
```
Step 4: Delete the lines in the original unifont.hex file that begin with “2C”.
Step 5: Merge the two files:
```
sort unifont.hex uni2C.hex > new-unifont.hex
```
or use Roman’s hexmerge utility:
```
hexmerge unifont.hex uni2C.hex > new-unifont.hex
```
Step 6: Check for duplicates:
```
unidup < new-unifont.hex
```
Of course, remove any reported duplicates.
Step 7: Build the font as in the “Using Graphical Tools” section above. This can be as simple as typing
```
make
```
in the font/ directory.

I (Paul Hardy) had only used hexdraw in the very beginning of my work on Unifont. Once I got my graphics programs working, I ignored it for months. Then I wanted to go through all of the Yi Syllables and Yi Radicals — over 1000 glyphs — to fine-tune their horizontal alignment after I drew them. hexdraw turned out to be the perfect tool for this. By printing hyphens (“-”) as place holders where a pixel is off, it allowed me to verify space to the left and right of each character. I later used hexdraw for similar fine-tuning of spacing on Hangul and other glyphs. It is ideal for the task.

2.6 Checking Coverage

There should never be duplicates in a .hex file. If there are, remove them before the .hex font is turned into a BDF or other font file. The recommendations above include making liberal use of unidup to avoid such a situation.

The unipagecount program will print a hexadecimal number of code points that have coverage within each 256 code point block. The hexadecimal number will therefore range from 0 (no coverage) to 100 (= 256 decimal; full coverage). If a number is ever more than 100 hexadecimal (i.e., 256 decimal), there must be an extra character (glyph) for that page of 256 code points.

To further look at the coverage within just one 256 code point page (using page 2C, containing Coptic, as our example) use
```
unipagecount -p2C < unifont.hex
```
Note that the “page number” can use upper- or lower-case letters: -p2C or -p2c will both work. That will print a 16 x 16 grid of U+2C00..U+2CFF. Of course, without placeholder glyphs for the unassigned code points from unassigned.hex in the U+2C00..U+2CFF range, unipagecount can give a lower number than the true coverage.
Using the -l flag with unipagecount will produce an HTML table with links to corresponding graphics images. You can get an idea of how this works in the font/compiled/directory after running make; the index.html file in that directory will have a table with links to the 256 glyph maps in the font/compiled/bmp/subdirectory. With unipagecount, the background color of the cells will range from red (for 0% complete in that 256 code point block) to orange (a little coverage) to yellow (more coverage) to green (complete coverage). If a cell looks light red or pink, the corresponding code page probably has duplicate characters. Verify that with

sort unifont.hex | unidup

(substituting the name of your .hex file for unifont.hex).

To see the coverage of each Unicode script, copy the file font/coverage.dat into the same directory as the unifont.hex file you’re examining. Then run

unicoverage < unifont.hex > coverage.out

This will give you all the scripts within the Unicode Basic Multilingual Plane, in order, with a percentage (0.0% through 100.0%) of each script’s coverage. Note that to get the true coverage of assigned code points, you will have to merge unassigned.hex with the rest of unifont.hex if not done by default in your setup.

Using the .hex files in font/plane00/, you can create a font with all available glyphs with

sort font/plane00/*.hex > unifont-whole.hex

and run unicoverage using the resulting unifont-whole.hex file.

### 2.7 Custom Builds

The font can be built from within the font/ directory by simply typing

make

From the top-level directory (one level above the font/ directory), typing

make BUILDFONT=1

will also build the font. The font is not built by default by typing make from the top-level directory, because a pre-built version already exists in the font/precompiled/ directory. Font files are architecture-independent, so the only reason to build the font is if you modify its composition.

By default, source glyphs are read from the font/plane00/ directory. Glyphs for unassigned code points are built into the font by default, but glyphs for Private Use Area code points are not built into the font.

All of the .hex file names can be replaced selectively on the make command line to override their default values. Their locations are relative to the font/ directory. The list of component hex file variables is:

- **UNIFONTBASE**: The bulk of Unifont scripts
- **CJK**: Most of the CJK Ideographs
- **COPYLEFT**: The Copyleft glyph
CUSTOM00  Hand-drawn quadruple-length Plane 0 glyphs
HANGUL    Hangul Syllables block
JISHEX    Plane 0 and Plane 2 glyphs for a Japanese Unifont version
SPACES    Space glyphs, single- and double-width
UNASSIGNED Glyphs for unassigned code points
PUA       Glyphs for the Private Use Area

So, for example, to build a font that includes four-digit hexadecimal code point glyphs (as white digits on a black background) for the Private Use Area, type

    make PUA="plane00/pua.hex"

because those glyphs reside in the font/plane00/pua.hex file.

To build a font that includes your own special PUA glyphs, type

    make PUA="my-cool-PUA.hex"
or whatever the name is of your PUA glyph .hex file.

To create a build that includes the supplied PUA glyphs but not the unassigned code point glyphs, type

    make PUA="plane00/pua.hex" UNASSIGNED=""

If you create a custom font build of your own in one gigantic file, you can build with just this file by declaring all the ordinary files to be null:

    make UNIFONTBASE="mycustomfont.hex" \
    CJK="" HANGUL="" UNASSIGNED="" PUA=""

Note that this command did not include an override for the glyphs for spaces contained in the font/plane00/spaces.hex file; that is, the variable SPACES was not redefined on the command line. You could also pass the argument SPACES="", but just be aware that those spaces glyphs are in a separate file for a reason. When graphical (".bmp") glyph files are converted back into hex string (".hex") format, the unibmp2hex utility doesn’t know if a blank glyph area is a space glyph or not, so it doesn’t encode anything. The font/plane00/spaces.hex file contains glyphs that are strings of 0s, with length depending on whether the space is nominally a single- or double-width space. (Unifont does not distinguish between finer spacing used in traditional typesetting, such as a thin space, en space, em space, or quad space; all spaces are either 8 pixels wide or 16 pixels wide.)

2.8 OpenType

Unifont version 14.0.03 introduced the hex2otf program, which allows direct conversion of a Unifont .hex file into an OpenType font.

hex2otf reads a file in Unifont .hex file format and optionally reads a combining-characters file to handle combining character glyphs. It then creates an output font file containing a TrueType or OpenType font.
The combining marks are not needed on glyphs that an operating system's rasterizer already supports. However, combining mark support is useful to support new Unicode additions that do not yet have widespread support. Combining mark support is also necessary for Private Use Area glyphs, because a rasterizer will not know which PUA glyphs are combining marks, or how to correctly position such marks.

`hex2otf` accepts TrueType and OpenType numeric IDs. For example, ID 1 designates the font family name. To create a subset of Unifont with name "Unifont Subset", type the following command with files in the appropriate locations:

```
hex2otf hex=unifont-base.hex pos=plane00-combining.txt \ 
1="Unifont Subset" format=cff,bitmap,gpos out=unifont-subset.otf
```

For more information on `hex2otf`, see the man page in the Reference section of this document.

### 2.9 One Glyph per File

The `unifont1per` program will read a Unifont “.hex” file from standard input and create bitmap output files, one file per Unicode code point in the “.hex” file. This can be useful if you want to include single glyphs as graphic objects for illustration in a document or web page. Output files have names of the form “U+<codepoint>.bmp”, where “<codepoint>” is the Unicode code point of the glyph.

Note that if you feed `unifont1per` a huge file of glyph hexadecimal strings, you will get a directory loaded with individual “.bmp” files, so use your judgment!

### 2.10 Viewing a Unifont File Interactively

The `unifont-viewer` Perl script uses the `wxWidgets` Perl library to present a dynamic graphical representation of a Unifont hex file. This is a convenient way to scan quickly through a complete Unifont hex file. It is ideal for scanning through a Unifont hex source file after you have made changes.

Use `unifont-viewer` to open any Unifont hex file, whether in the Basic Multilingual Plane or up to the maximum Unicode code point of U+10FFFF. The font is displayed graphically in sections of 16-by-16 glyph grids (256 glyphs—a “page” in Unifont lingo). The page numbers (the upper portion of the hexadecimal code point range) appear in a list along the left-hand side. Only page ranges that are present in the Unifont hex file are listed.

When `unifont-viewer` loads a hex file, it begins by displaying the first “page” range in that file.

### 2.11 Seeing the Big Picture (Literally!)

The GNU Unifont 6.3 release introduced a new program, `unifontpic`. This produces a chart of all the Basic Multilingual Plane glyphs in Unifont. By default the chart is arranged as a 256-by-256 glyph table. Specifying the `-l` option produces a chart that is 16 glyphs wide by 4,096 glyphs long. See `unifontpic(1)` for more information.

The “long” version, created with `unifontpic -l`, only produces 16 glyphs per row. This is more useful for scrolling through on a computer screen.
GIMP, the GNU Image Manipulation Program, will properly display the resulting long .bmp file (at least under GNOME), but not all graphics utilities can. The output file is over 65,536 pixel rows tall, which causes problems with some graphics programs.

To generate a chart with all your glyphs in one giant `unifont.hex` file, type the command

```
unifontpic < unifont.hex > unifont.bmp
```

The output is a monochrome Bitmap Graphics Format (.bmp) file. If you prefer PNG files, use your favorite graphics program or conversion program to convert the BMP file to a PNG file.

This utility is especially useful if you are customizing the font, for example if adding your own Private Use Area glyphs.

Single-width and double-width glyphs are reproduced in this bitmap file at the correct aspect ratios. Triple-width and quadruple-width glyphs will be compressed 50% in the horizontal dimension to fit within a 16-by-16 pixel glyph cell.

The default 256-by-256 code point chart will render satisfactorily on a sheet of paper approximately 3 feet by 3 feet (1 meter by 1 meter) at 120 dots per inch. Thus the square format is suitable for printing.

### 2.12 Combining Circles

The earliest versions of Unifont (before the 5.1 release) used glyphs that showed dashed circles for combining characters. This is how the glyphs appear in The Unicode Standard code charts. In version 5.1, Paul Hardy was able to get combining characters to appear superimposed correctly in the TrueType version of the font. (There are no plans to try to get combining characters to work in a BDF or PCF version owing to the limitations of those bitmapped font formats.)

With combining characters working in the TrueType font, Paul Hardy created variations of Unifont with and without combining circles, the idea being that the version without combining circles would be used to create the TrueType font. The variation with combining circles was kept for reference.

Unifont Version 6.2 simplified the build to produce only one font variation, without combining circles.

Unifont Version 6.3 introduced a new utility, `unigencircles`, to superimpose combining circles over glyphs with code points in a `combining.txt` file.

The latest Unifont release contains a parallel set of font files named `unifont_sample.*`. These “Unifont Sample” font files contain glyphs with combining circles where appropriate. The “Unifont Sample” font is therefore not intended for general-purpose writing, but only for illustrating each individual glyph as represented in The Unicode Standard.

To run `unigencircles`, start with the file `font/ttfsrc/combining.txt` and type a command of this form:

```
unigencircles combining.txt < unifont.hex > unifont-circles.hex
```

where `unifont.hex` is a single file containing all the glyphs you wish to render.

Because the output is another .hex file, you can use all Unifont utilities with this resulting file just as you would with the .hex files in `font/plane00/`. 
If you want to build a font from this generated `unifont.hex` file, you could type

```
make UNIFONTBASE="unifont-circles.hex" CJK="" HANGUL="" \\
UNASSIGNED="" PUA=""
```

as discussed above. In this case, if you included `font/plane00/spaces.hex` in the `unifont.hex` input file, you should also set `SPACES=""` on the command line so that you only read in your final custom `unifont-circles.hex` file.

### 2.13 Installing Fonts on GNU/Linux

The original standard font format of Unifont was a BDF font. The newer PCF font format loads much faster when a program begins, and so is preferable for installations using the X Window System and similar environments.

Compress PCF fonts using

```
gzip -9 fontname.pcf
```

Copy the resulting `fontname.pcf.gz` file to `/usr/share/fonts/X11/misc/` or place in a local font directory if your windowing software supports that (for example, `~/fonts/`).

Copy TrueType fonts to `/usr/share/fonts/truetype/` uncompressed or place in your local font directory. Note: on some versions of Unix, such as Solaris, the name of the TrueType directory might be TrueType and might be under `/usr/share/fonts/X11/` — examine the system fonts directories, then modify the font `make install` rule accordingly.

On most flavors of GNU/Linux with the latest `xset` utility (including the latest Debian and Red Hat releases), the following command should allow you to start using the font immediately:

```
xset fp rehash
```

The safest way to make sure the system knows about the new fonts will be to restart the X Window System or even reboot.

### 2.14 Creating a Brand New Font

The original tools will only produce glyphs that are 16 pixels tall, and either 8 or 16 pixels wide. The utilities `unihex2png`, `unipng2hex`, `hexdraw`, and `hex2bdf` now also support glyph heights of 24 and 32 pixels, and glyph widths of 8, 16, 24, and 32 pixels, but this is not fully tested. PNG glyphs that are 32 pixels wide are only supported if the glyphs are also 32 pixels tall. These dimensions are available for experimental use. See the respective sections for each of these programs in the Reference chapter of this document, or their respective man pages.

To create a brand new font (or even to add a new script to Unifont in the future), plan out the basic dimensions of the characters:

- How far above the lowest pixel will the baseline appear (in other words, how many rows are necessary for descenders such as in the glyphs for ‘g’, ‘q’, and ‘y’)?
- How many pixels must be empty on top and bottom for accents (in other words, what will the height of capital letters be)?
- Must glyphs be centered, left-aligned, or right-aligned?
• For a Latin font, what will the “x-height” be (the height of a lower-case “x” and related letters, such as “n” and “r”)?

Consistent capital heights, x-heights, descender depths, and centering will produce a better looking font. Look over the entire script and plan out a template grid that is consistent for the entire script. Then use that template for each glyph you draw for the script.

Unifont characters for the most part leave the left-most or right-most column of pixels blank if possible (consistent within each script) for left-to-right scripts. Centering is done around the fourth pixel (if a glyph is eight pixels wide) or around the eighth pixel (if a glyph is 16 pixels wide).

Experimenting and (above all) having fun with these utilities is the best way to learn.

2.15 Updates to Unicode

If a currently unassigned code point is assigned to a character in the future, the font can be updated as follows:

1. Delete the code point’s entry from the corresponding font/plane00/unassigned.hex file, as that code point will no longer be unassigned.

2. Determine which existing .hex file should contain the newly defined character (examine the files to see which one contains other glyphs in the script. For Plane 0:
   • unifont-base.hex contains most scripts
   • wqy.hex contains most CJK ideographs; its name pays homage to the Wen Quan Yi font, the source of almost all of its glyphs
   • hangul-syllables.hex contains the Hangul Syllables block (U+AC00..U+D7A3); this should never have new code points added as it covers the fixed range of the Unicode Hangul Syllables block
   • spaces.hex contains the list of single- and double-width spaces

   If in doubt (for example, if a new script is added to Unicode and you are not sure which .hex file to augment), add the new glyphs to unifont-base.hex for the Plane 0 range.

3. Update the appropriate .hex file.

4. Consider if font/coverage.dat has to be updated. Follow its existing format to insert a new glyph range, being sure to change any previously unassigned ranges before or after the newly added range.

5. Make a new .hex version of the font, and verify that you did not introduce any duplicates.

6. Run unipagecount and/or unicoverage as described previously to verify that you have not mistakenly deleted any existing characters.

Enjoy!
3 Reference

3.1 bdfimplode

3.1.1 bdfimplode NAME
bdfimplode – Convert a BDF font into GNU Unifont .hex format

3.1.2 bdfimplode SYNOPSIS
bdfimplode < input-font.bdf > output-font.hex

3.1.3 bdfimplode DESCRIPTION
bdfimplode reads a BDF font from STDOUT and writes GNU Unifont .hex conversion of
the font to STDOUT.

3.1.4 bdfimplode FILES
*.bdf font files

3.1.5 bdfimplode SEE ALSO
hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1),
johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicableage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unhex2bmp(1), unhex2png(1), unhexfill(1), unhexgen(1), unhexpose(1), unhexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.1.6 bdfimplode AUTHOR
bdfimplode was written by Roman Czyborra.

3.1.7 bdfimplode LICENSE
bdfimplode is Copyright © 1998 Roman Czyborra.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.1.8 bdfimplode BUGS
bdfimplode was written to read a BDF file created by the hex2bdf script. It will not properly
handle other BDF files with differing bounding boxes.

3.2 hex2bdf

3.2.1 hex2bdf NAME
hex2bdf – Convert a GNU Unifont .hex file into a BDF font

3.2.2 hex2bdf SYNOPSIS
hex2bdf [options] < input-font.hex > output-font.bdf
3.2.3 hex2bdf DESCRIPTION

**hex2bdf** reads a sorted GNU Unifont .hex file (sorted with the Unix **sort** utility) from STDIN and writes a BDF version of the font to STDOUT.

3.2.4 hex2bdf OPTIONS

- **-f "font-name"**
  Specify the target font name. If omitted, the default font name "Unifont" is assigned.

- **-v "font-version"**
  Specify the target font version. If omitted, the default version "1.0" is assigned.

- **-c "font-copyright"**
  Specify the target font copyright information. The default is the null string.

- **-r pixel-rows**
  Specify how many pixel rows tall a glyph is. The default is the traditional Unifont 16 rows of pixels. This is an addition to support **unihex2png**(1) and **unipng2hex**(1), which allow designing glyphs that are 16, 24, or 32 pixel rows tall.

3.2.5 hex2bdf EXAMPLE

Sample usage:

```
hex2bdf -f "Unifont" -c "(C) 2013..." unifont.hex > unifont.bdf
```

3.2.6 hex2bdf FILES

*.hex GNU Unifont font files

3.2.7 hex2bdf SEE ALSO

**bdfimplode**(1), **hex2otf**(1), **hex2sfd**(1), **hexbraille**(1), **hexdraw**(1), **hexkinya**(1), **hexmerge**(1), **johab2syllables**(1), **johab2ucs2**(1), **unibdf2hex**(1), **unibmp2hex**(1), **unibmpnbump**(1), **unicoverage**(1), **unidup**(1), **unifont**(5), **unifont-viewer**(1), **unifont1per**(1), **unifontchojung**(1), **unifontksx**(1), **unifontpic**(1), **unigen-hangul**(1), **unigencircles**(1), **unigencircle**(1), **unihex2bmp**(1), **unihex2png**(1), **unihexfill**(1), **unihexgen**(1), **unihexpose**(1), **unihexrotate**(1), **unijohab2html**(1), **unipagecount**(1), **unipng2hex**(1)

3.2.8 hex2bdf AUTHOR

**hex2bdf** was written by Roman Czyborra.

3.2.9 hex2bdf LICENSE

**hex2bdf** is Copyright © 1998 Roman Czyborra.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.2.10 hex2bdf BUGS

No known bugs exist. Support for glyph heights other than 16 pixels is brand new and has not been extensively tested. "\n" t
3.3 hex2otf

3.3.1 hex2otf NAME
hex2otf – Convert a GNU Unifont .hex file to an OpenType font

3.3.2 hex2otf SYNOPSIS
hex2otf [−−help] [−−version]
hex2otf operand . . .

3.3.3 hex2otf DESCRIPTION
The hex2otf utility generates an OpenType font from a GNU Unifont .hex file. hex2otf reads a file in Unifont .hex file format, as specified with the hex=filename.hex operand, and creates an OpenType file as specified with the out=filename.otf operand. A combining character offset file may optionally be specified with the pos=combining-file operand. TrueType and OpenType ID fields are specified with operands of the form id=string. A dummy GPOS table can be generated with the gpos option. The following subsection describes why this is beneficial.

3.3.3.1 GPOS RATIONALE
A dummy GPOS table is useful for positioning combining characters consistently. For example, HarfBuzz will perform fallback mark positioning on certain conditions, trying to put combining marks closer to their base characters. But it does not work well with the blank outlines generated by this utility, because blank outlines do not have relevant glyph extent values. The dummy GPOS table contains no useful position information, but its presence is enough for HarfBuzz to skip the fallback positioning. However, both HarfBuzz and Uniscribe will still precompose characters before rendering. For example, “a\u0301” is rendered exactly as the glyph of U+00E1 (LATIN SMALL LETTER A WITH ACUTE), not the composite of two glyphs.

3.3.3.2 OUTPUT FORMAT CONSIDERATIONS
The font format should be carefully chosen. In general, using the cff and gpos operands together will produce good results on GNU/Linux, *nix, Windows, and macOS systems. The current status of different formats is as follows:

CFF outline
May cause rendering issues in the Windows Command Prompt program.

CFF 2 outline
Not recognized by Windows.

TrueType outline
On Windows, this appears to be blurred in 16 pixels size due to the ClearType anti-alias mechanism; this can be fixed by embedding bitmaps.
Bitmap-only
Contrary to the OpenType specification, a bitmap-only font has to be accompanied by blank outlines to be recognized.

3.3.3.3 FUTURE DIRECTIONS
Some font parameters are hard coded into the source code for convenience. Other approaches may be preferable.
The generated fonts may produce incorrect result for vertical layout. More font parameters might be necessary if such a need arises.
Font size may be reduced by using e.g. CFF subroutines and TrueType font programs.

3.3.4 hex2otf OPTIONS

- ‒help Print a brief help message and exit.
- ‒version Print program version information and exit.

3.3.5 hex2otf OPERANDS

hex=file Specify the input hex file pathname.
pos=file Optional. Specify the combining position file pathname.
out=file Specify the output pathname.
format=word[,word ...]
Specify the font format. Each word shall be one of the following:

cff Generate CFF outlines.
cff2 Generate CFF 2 outlines.
truetype Generate TrueType outlines.
blank Generate blank outlines. Generating a font with only blank outlines is not allowed.
bitmap Generate embedded bitmap. Required if no outlines are generated.
gpos Generate dummy GPOS table. See the “GPOS RATIONALE” section above.
gsub Generate generic GSUB table with two entries: “DFLT” and “thai”. “DFLT” improves glyph positioning in HarfBuzz, especially as relates to glyph reordering. “thai” directs HarfBuzz not to obtain Thai glyphs from the Unicode Private Use Area.

id=text Specify font attributes. id shall be an decimal integer from 0 to 255, corresponding to “Name ID” as defined in the OpenType specification <https://docs.microsoft.com/en-us/typography/opentype/spec/name#name-ids>. text shall be value of the attribute in English (United States). Characters
beyond the ASCII range shall be encoded as UTF-8. Currently defined attributes relevant to Unifont are briefly listed below. Refer to the specification for full details.
### 3.3.6 hex2otf FONTFORGE NOTE

Default FontForge package installations can incorrectly ignore the Unicode plane of glyphs beyond Plane 0 in CID-keyed fonts (such as are generated with the `format=cff` option). This is caused by the presence of the file `Adobe-Identity-0.cidmap`, which FontForge interprets incorrectly to determine Unicode code point glyph mappings if present. This file is located in `/usr/share/fontforge/cidmap` or a similar directory on GNU/Linux systems. Removing or renaming `Adobe-Identity-0.cidmap` will cause FontForge to fall back on OpenType cmap entries in the font. FontForge will then correctly display all Unicode code points.

### 3.3.7 hex2otf EXAMPLE

Sample usage:

```
hex2otf hex=unifont.hex pos=combing.txt \ 
format=cff,gpos,gsub out=unifont.otf
```
3.3.8 hex2otf EXIT STATUS

Status values are defined in `<stdlib.h>`. The program exits with status EXIT_SUCCESS upon successful font generation, or EXIT_FAILURE if an error occurred. If an error condition is encountered, hex2otf writes a brief diagnostic message to STDERR; in this event, the state of the output font file will be undefined.

3.3.9 hex2otf SEE ALSO

unifont(5), hex2bdf(1)

3.3.10 hex2otf AUTHOR

hex2otf was written by 何志翔 (He Zhixiang).

3.3.11 hex2otf LICENSE

hex2otf is Copyright © 2022 何志翔 (He Zhixiang).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.4 hex2sfd

3.4.1 hex2sfd NAME

hex2sfd − Convert a GNU Unifont .hex file into a FontForge .sfd format

3.4.2 hex2sfd SYNOPSIS

hex2sfd < input-font.hex > output-font.sfd

3.4.3 hex2sfd DESCRIPTION

hex2sfd reads a GNU Unifont .hex file from STDIN and writes an outline font representation in FontForge Spline Font Database (.sfd) format. The resulting .sfd file can then be converted by FontForge into a TrueType .ttf font.

3.4.4 hex2sfd FILES

GNU Unifont .hex font files for input, FontForge .sfd font files for output

3.4.5 hex2sfd SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1),
unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1),
unifontchojung(1), unifontksx(1), unifontpic(1), unigencircles(1), unigen-hangul(1),
unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1),
unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.4.6 hex2sfd AUTHOR

hex2sfd was written by Luis Alejandro González Miranda in 2005, with modifications by Paul Hardy in 2008.
3.4.7 hex2sfd LICENSE

hex2sfd is Copyright © 2005 Luis Alejandro González Miranda, © 2008 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.4.8 hex2sfd BUGS
No known bugs exist.

3.5 hexbraille

3.5.1 hexbraille NAME
hexbraille – Algorithmically generate the Unicode Braille range (U+28xx)

3.5.2 hexbraille SYNOPSIS
hexbraille > output-font.hex

3.5.3 hexbraille DESCRIPTION
hexbraille generates a GNU Unifont .hex format file (written on stdout) containing the
Braille dot patterns in the Unicode range U+2800..U+28FF.

3.5.4 hexbraille FILES
braille.hex output font files

3.5.5 hexbraille SEE ALSO
bdflipplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexdraw(1), hexkinya(1), hexmerge(1),
johnab2syllables(1), johnab2ucs2(1), unibdf(1), unibmp(1), unibmpbump(1),unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.5.6 hexbraille AUTHOR
hexbraille was written by Roman Czyborra, who named this script "braille.pl". In 2003,
Roman incorporated a bug fix from Dominique at unruh.de.

3.5.7 hexbraille LICENSE
hexbraille is Copyright © 1998 Roman Czyborra.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.5.8 hexbraille BUGS
No known bugs exist.
3.6 hexdraw

3.6.1 hexdraw NAME
hexdraw – Convert a GNU Unifont .hex file to and from an ASCII text file

3.6.2 hexdraw SYNOPSIS
hexdraw [−−height height] < input-font.hex > output-font.txt
hexdraw [−−height height] < input-font.txt > output-font.hex

3.6.3 hexdraw DESCRIPTION
hexdraw reads a sorted GNU Unifont .hex file from STDIN and writes a two dimensional ASCII art rendering for each glyph to STDOUT. The output file can be edited with any text editor, then converted back into a .hex file. Unifont ASCII hexadecimal glyphs are 16 pixels high and can be 8, 16, 24, or 32 pixels wide.

3.6.4 hexdraw OPTIONS
−−height
By default, hexdraw expects glyphs to be 16 rows high. Alternatively, a different height of 24, 32, or 64 can be specified, for glyphs that are up to 128 columns wide by 64 rows high.

3.6.5 hexdraw FILES
*.hex GNU Unifont font files

3.6.6 hexdraw SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.6.7 hexdraw AUTHOR
hexdraw was written by Roman Czyborra.

3.6.8 hexdraw LICENSE
hexdraw is Copyright © 1998 Roman Czyborra.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.6.9 hexdraw BUGS
No known bugs exist.
3.7 hexkinya

3.7.1 hexkinya NAME
hexkinya – Create the Private Use Area Kinya syllables

3.7.2 hexkinya SYNOPSIS
hexkinya > plane15.hex

3.7.3 hexkinya DESCRIPTION
hexkinya contains pre-defined initial, middle, and final Kinya alphabet glyphs to form syllables. The output is the Kinya Syllables Private Use Area block of the ConScript Unicode Registry (CSUR). The output range is U+0F0000 through U+0FDE6F, inclusive.

Note that Plane 15 also contains Pikto glyphs, which follow after Kinya.

3.7.4 hexkinya FILES
None.

3.7.5 hexkinya SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihexbmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.7.6 hexkinya AUTHOR
hexkinya was written by Andrew Miller.

3.7.7 hexkinya LICENSE
hexkinya is Copyright © 2014 by Andrew Miller.
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.7.8 hexkinya BUGS
No known bugs exist.

3.8 hexmerge

3.8.1 hexmerge NAME
hexmerge – Merge two or more GNU Unifont .hex font files into one

3.8.2 hexmerge SYNOPSIS
hexmerge input-font1.hex input-font2.hex > output-font.hex
3.8.3 hexmerge DESCRIPTION

hexmerge reads two or more GNU Unifont .hex files, sorts them, and writes the combined font to stdout.

3.8.4 hexmerge FILES

*.hex GNU Unifont font files

3.8.5 hexmerge SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexkinya(1), hexdraw(1),
johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unhexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.8.6 hexmerge AUTHOR

hexmerge was written by Roman Czyborra.

3.8.7 hexmerge LICENSE

hexmerge is Copyright © 1998 Roman Czyborra.
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.8.8 hexmerge BUGS

No known bugs exist.

3.9 johab2syllables

3.9.1 johab2syllables NAME

johab2syllables — Convert Unifont Johab 6/3/1 file to Hangul Syllables

3.9.2 johab2syllables SYNOPSIS

johab2syllables -i hangul-base.hex -o hangul-syllables.hex

3.9.3 johab2syllables DESCRIPTION

johab2syllables generates the modern Hangul Syllables Unicode range, U+AC00..U+D7A3, from an input Unifont .hex file encoded in Johab 6/3/1 format. This program demonstrates how to use functions in the unihangul-support.c file to map a Hangul Syllables code point to the chooseong, jungseong, and jongseong Hangul Jamo that comprise each syllable.

This program is specifically designed to generate the Unicode Hangul Syllables range only, and does not shift final consonants (jongseong) if the vowel contains two long vertical strokes. For a more general program that uses the entire Hangul Jamo modern, ancient, and extended ancient ranges of U+1100..U+11FF, U+A960..U+A97F, and U+D7B0..D7FF,
and which does shift jongseong right one column for vowels with two long vertical strokes on the right, see unigen-hangul(1). That program is used to generate the distributed Unifont Hangul Syllables range.

Hangul

- 1100-1112  Modern initial consonants (choseong)
- 1161-1175  Modern medial vowels and diphthongs (jungseong)
- 11A8-11C2  Modern final consonants (jongseong).

3.9.4 johab2syllables OPTIONS

Function

- h, --help
  Print a help message and exit.

- i input_file
  Unifont hangul-base.hex formatted input file.

- o output_file
  Unifont .hex format output file.

3.9.5 johab2syllables EXAMPLES

johab2syllables -i hangul-base.hex -o hangul-syllables.hex
johab2syllables < hangul-base.hex > hangul-syllables.hex

3.9.6 johab2syllables FILES

Unifont .hex files in Johab 6/3/1 encoding. See unifont-johab631(5) for a description of the input file structure. This program uses functions contained in the file unihangul-support.c.

3.9.7 johab2syllables SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-johab631(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.9.8 johab2syllables AUTHOR

johab2syllables was written by Paul Hardy.

3.9.9 johab2syllables LICENSE

johab2syllables is Copyright © 2023 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
3.10 johab2ucs2

3.10.1 johab2ucs2 NAME
johab2ucs2 — Convert a Johab BDF font into GNU Unifont Hangul Syllables

3.10.2 johab2ucs2 SYNOPSIS
johab2ucs2 < input-font.bdf > output-font.hex

3.10.3 johab2ucs2 DESCRIPTION
johab2ucs2 reads a Johab encoded BDF font (as used in Hanterm) from STDIN and writes
a GNU Unifont .hex file containing the Unicode Hangul Syllables to STDOUT.

3.10.4 johab2ucs2 FILES
*.bdf font files

3.10.5 johab2ucs2 SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicycle(1),
unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.10.6 johab2ucs2 AUTHOR
johab2ucs2 was written in 1998 by Jungshik Shin and given to Roman Czyborra. Paul
Hardy made some modifications and bug fixes in 2008.

3.10.7 johab2ucs2 LICENSE
johab2ucs2 is Copyright © 1998 Jungshik Shin and Roman Czyborra, © 2008 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.10.8 johab2ucs2 BUGS
No known bugs exist.

3.11 unibdf2hex

3.11.1 unibdf2hex NAME
unibdf2hex — Convert BDF font glyphs into Unifont .hex glyphs
3.11.2 unibdf2hex SYNOPSIS
unibdf2hex < input-font.bdf > output-font.hex

3.11.3 unibdf2hex DESCRIPTION
unibdf2hex reads a BDF font file, extracting selected code points and printing them on stdout in Unifont .hex format. This program was used to extract CJK ideographs from the 16x16 version of Wen Quan Yi’s BDF font. Currently the program is hard-coded to only extract those code points that comprise the "wqy.hex" source font file used to build "unifont.hex" but this source code could easily be modified.

3.11.4 unibdf2hex FILES
*.hex GNU Unifont font files

3.11.5 unibdf2hex SEE ALSO
bdimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), undup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unirotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.11.6 unibdf2hex AUTHOR
unibdf2hex was written by Paul Hardy.

3.11.7 unibdf2hex LICENSE
unibdf2hex is Copyright © 2008 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.11.8 unibdf2hex BUGS
No known bugs exist.

3.12 unibmp2hex

3.12.1 unibmp2hex NAME
unibmp2hex – Bitmap graphics file to GNU Unifont .hex file converter

3.12.2 unibmp2hex SYNOPSIS

3.12.3 unibmp2hex DESCRIPTION
unibmp2hex reads a bitmap produced by unihex2bmp before or after editing, and converts it back into a Unifont .hex format file. The graphics file contains a block of 256 Unicode code points arranged in a 16 by 16 grid. Each code point appears in a 32 by 32 pixel grid. Characters are 16 rows high, and 8, 16, or 31 (treated as 32) columns wide.
3.12.4 unibmp2hex OPTIONS

−p Specify that the code points will be assigned to the 256 block space pagenum in the .hex file. If not specified, unibmp2hex will determine the appropriate block by reading the row and column headers. Note that "page" is not a standard Unicode term. It refers to an output bitmap graphics page of 16 by 16 code points. If pagenum is greater than FF, the block resides above the Unicode Basic Multilingual Plane. In that event, the .hex file will contain eight digit hexadecimal code points rather than the Unifont standard of four hexadecimal code points.

−i Specify the input file. The default is STDIN.

−o Specify the output file. The default is STDOUT.

−w Force all output .hex glyphs to be 16 pixels wide rather than dual width (8 or 16 pixels).

3.12.5 unibmp2hex EXAMPLE

Sample usage:

unibmp2hex −imy_input_file.bmp −omy_output_file.hex

3.12.6 unibmp2hex FILES

*.bmp or *.wbmp graphics files

3.12.7 unibmp2hex SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), uniheXrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.12.8 unibmp2hex AUTHOR

unibmp2hex was written by Paul Hardy.

3.12.9 unibmp2hex LICENSE

unibmp2hex is Copyright © 2007, 2008 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.12.10 unibmp2hex BUGS

No known real bugs exist, except that this software does not perform extensive error checking on its input files. If they’re not in the format of the original bitmapped output from unihex2bmp, all bets are off.

If the output file is for a "page" containing space code points and the bitmap file squares for those code points are not empty, unibmp2hex preserves the graphics as they are drawn.
3.13 unibmpbump

3.13.1 unibmpbump NAME
unibmpbump – adjust a .bmp image for unibmp2hex processing

3.13.2 unibmpbump SYNOPSIS
unibmpbump [−i input-file.bmp] [−o output-file.bmp] [−v | --verbose] [−V | --version]

3.13.3 unibmpbump DESCRIPTION
unibmpbump reads a Microsoft Bitmap Graphics (".bmp") graphics image file and, if it appears to be in a known format, converts it for processing by unibmp2hex(1).
unibmpbump supports the following Device Independent Bitmap (DIB) header byte lengths for input image files:

<table>
<thead>
<tr>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>The BITMAPCOREHEADER format. This was the original Microsoft Windows header format. It has not been encountered, and was added only as a subset of the formats that appear below.</td>
</tr>
<tr>
<td>40</td>
<td>The BITMAPINFOHEADER format. This is the format that most graphics creation programs support, and is the header format that unibmp2hex expects.</td>
</tr>
<tr>
<td>108</td>
<td>The BITMAPV4HEADER format. This is the format that the Image Magick convert program creates when converting a PNG image to a BMP image using the &quot;--monochrome&quot; option.</td>
</tr>
<tr>
<td>124</td>
<td>The BITMAPV5HEADER format. This is the format that the Image Magick convert program creates when converting a PNG image to a BMP image without the &quot;--monochrome&quot; option; convert will encode the image as a &quot;grayscale&quot; image, but with only two colors: black and white.</td>
</tr>
</tbody>
</table>

unibmpbump is able to read files created by unihex2png that were subsequently saved as BMP files, for example by the convert(1) program, which is part of the Image Magick package. Images created by unihex2png(1) are 544 rows high by 560 columns wide. Images created by unihex2bmp(1) are 544 rows high by 576 columns wide. Thus, if the input Bitmap Graphics file is 544 by 560 pixels, unibmpbump assumes that the file was originally a Portable Network Graphics (".png") file created by unihex2png and realigns the glyphs so they are positioned as unibmp2hex expects.

3.13.4 unibmpbump OPTIONS

−i Specify an input Bitmaps Graphics (".bmp") file. If no input file is specified, input is taken from stdin. The input file must be either 544 rows by 560 columns (for a file created by unihex2png originally) or 544 rows by 576 columns (for a file created by unihex2bmp originally).

−o Specify an output Bitmaps Graphics (".bmp") file. If no output file is specified, output is sent to stdout. The output file will be 544 rows high by 576 columns wide with a 40 byte Device Independent Bitmap (DIB) header, which is the format that unibmp2hex expects.
−v, --verbose
Verbise output. Print information about the input file on stderr.

−V, --version
Print the version of unibmpbump and exit.

3.13.5 unibmpbump FILES
Bitmap Graphics (".bmp") input and output files.

3.13.6 unibmpbump SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unicoverage(1),
unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), uni-
hex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.13.7 unibmpbump AUTHOR
unibmpbump was written by Paul Hardy.

3.13.8 unibmpbump LICENSE
unibmpbump is Copyright © 2019 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.13.9 unibmpbump BUGS
No known bugs exist.

3.14 unicoverage

3.14.1 unicoverage NAME
unicoverage – Print coverage of each Unicode Script

3.14.2 unicoverage SYNOPSIS
unicoverage [−i input-file] [−o output-file] [−n]

3.14.3 unicoverage DESCRIPTION
unicoverage reads a GNU Unifont .hex font and uses data in coverage.dat (which must
reside in the current directory). The output is the percent coverage of each script listed in
the coverage.dat file. The Unicode code points in the input .hex file must be in ascending
order.

3.14.4 unicoverage OPTIONS
−i Specify the input file. The default is STDIN.
−n       Print the number of glyphs in each range instead of a percentage count.
−o       Specify the output file. The default is STDOUT.

3.14.5 unicoverage EXAMPLE
Sample usage:
   unicoverage < unifont.hex > coverage.txt

3.14.6 unicoverage FILES
coverage.dat, *.hex GNU Unifont files.

3.14.7 unicoverage SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1),
unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.14.8 unicoverage AUTHOR
unicoverage was written by Paul Hardy.

3.14.9 unicoverage LICENSE
unicoverage is Copyright © 2007, 2008 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms of
the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.14.10 unicoverage BUGS
No known real bugs exist, except that this software does not perform extensive error check-
ing on its input files.

3.15 unidup

3.15.1 unidup NAME
unidup – Scan through a sorted .hex file and report duplicate code points

3.15.2 unidup SYNOPSIS
unidup [input-font.hex]

3.15.3 unidup DESCRIPTION
unidup reads a sorted GNU Unifont .hex file (sorted with the Unix sort utility) and prints
notification of any duplicate code points on stdout. The input file can be specified on the
command line. If no file is specified, input will be read from STDIN until end of file.
3.15.4 unidup FILES
*.hex GNU Unifont font files

3.15.5 unidup SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unicoverage(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), uni-
hex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.15.6 unidup AUTHOR
unidup was written by Paul Hardy.

3.15.7 unidup LICENSE
unidup is Copyright © 2007, 2013 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.15.8 unidup BUGS
No known bugs exist.

3.16 unifont-viewer

3.16.1 unifont-viewer NAME
unifont—viewer — View a .hex font file with a graphical user interface

3.16.2 unifont-viewer SYNOPSIS
unifont—viewer &

3.16.3 unifont-viewer DESCRIPTION
unifont—viewer reads a GNU Unifont .hex file and displays glyphs in blocks of 16 by 16,
256 glyphs per screen with a graphical user interface. The .hex file is selected by opening
it with the File > Open menu item or by typing Control-O. After loading a file, a list of
"pages" of 256 glyphs will appear in the left-hand column of the window. The first such
page of glyphs in the file will appear in the main window.
unifont—viewer supports glyph heights of 16, 24, and 32 pixels. The height of a .hex font
file is selected under the File menu. The default selection is Glyph Height 16.
Exit unifont—viewer by selecting File > Exit or by typing Control-X.

3.16.4 unifont-viewer FILES
*.hex GNU Unifont font files
3.16.5 unifont-viewer SEE ALSO
bdffimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.16.6 unifont-viewer AUTHOR
unifont-viewer was written by Andrew Miller.

3.16.7 unifont-viewer LICENSE
unifont-viewer is Copyright © 2014 Andrew Miller.
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.16.8 unifont-viewer BUGS
No known bugs exist.

3.17 unifont1per

3.17.1 unifont1per NAME
unifont1per − Create BMP glyph files from GNU Unifont .hex file

3.17.2 unifont1per SYNOPSIS
unifont1per < input_file.hex

3.17.3 unifont1per DESCRIPTION
unifont1per reads a GNU Unifont .hex file and for each code point in the file, creates an output Bitmapped Graphics (".bmp") file in the current directory. The filename for each output file is "U+codepoint.bmp", where codepoint is the Unicode codepoint for the glyph. Each glyph entry in a Unifont .hex file is expected to be 16 rows tall, and can be 8, 16, 24, or 32 pixels wide. The glyph sizes in output files are determined on a glyph-by-glyph basis. This program can be of use in extracting individual glyphs to display in a document or on a web page.

WARNING! This program generates one output file per code point that appears in the input stream. If you feed this program a file with 1000 code points, it will generate 1000 files in your current directory.

3.17.4 unifont1per OPTIONS
None.

3.17.5 unifont1per FILES
*.hex GNU Unifont font files, read from standard input
3.17.6 unifont1per SEE ALSO
bdfinplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), uni-
hex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.17.7 unifont1per AUTHOR
unifont1per was written by Paul Hardy.

3.17.8 unifont1per LICENSE
unifont1per is Copyright © 2017 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.17.9 unifont1per BUGS
No known real bugs exist, except that this software does not perform extensive error check-
ing on its input files. If they’re not in the format of the original GNU Unifont .hex file, all
bets are off. Lines can be terminated either with line feed, or carriage return plus line feed.

3.18 unifontchojung

3.18.1 unifontchojung NAME
unifontchojung – Extract Hangul syllables that have no final consonant

3.18.2 unifontchojung SYNOPSIS
unifontchojung < input-font.hex > output-font.hex

3.18.3 unifontchojung DESCRIPTION
unifontchojung reads a Unifont-format .hex font file from STDIN and writes a Unifont .hex
file containing a subset to STDOUT. The subset that is output only contains syllables
that contain an initial consonant (chojung) plus middle vowel (jungseong), with no final
consonant (jongseong). This lets a font designer focus on this subset during font creation.

3.18.4 unifontchojung FILES
*.bdf font files

3.18.5 unifontchojung SEE ALSO
bdfinplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), uni-
hex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)
3.18.6 unifontchojung AUTHOR

unifontchojung was written by Paul Hardy.

3.18.7 unifontchojung LICENSE

unifontchojung is Copyright © 2012 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.18.8 unifontchojung BUGS

No known bugs exist.

3.19 unifontksx

3.19.1 unifontksx NAME

unifontksx – Extract Hangul syllables that comprise KS X 1001:1992

3.19.2 unifontksx SYNOPSIS

unifontksx < input-font.hex > output-font.hex

3.19.3 unifontksx DESCRIPTION

unifontksx reads a Unifont-format .hex font file from STDIN and writes a Unifont .hex file containing a subset to STDOUT. The subset that is output only contains the 2,350 syllables that comprise Korean Standard KS X 1001:1992. These are extracted from the Unicode Hangul Syllables block, U+AC00..U+D7A3. This lets a font designer focus on those syllables that are most common in modern Hangul usage during font development.

3.19.4 unifontksx FILES

*.bdf font files

3.19.5 unifontksx SEE ALSO

bdflmplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.19.6 unifontksx AUTHOR

unifontksx was written by Paul Hardy.

3.19.7 unifontksx LICENSE

unifontksx is Copyright © 2012 Paul Hardy.
Chapter 3: Reference

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.19.8 unifontksx BUGS
No known bugs exist.

3.20 unifontpic

3.20.1 unifontpic NAME
unifontpic – Convert GNU Unifont .hex input to a bitmap image of the whole font

3.20.2 unifontpic SYNOPSIS
unifontpic [-d nnn] [-l] [-t] [-P plane] < input-font.hex > output-font.bmp

3.20.3 unifontpic DESCRIPTION
unifontpic reads a GNU Unifont .hex file from STDIN and writes a two dimensional rendering for each glyph to STDOUT. The output displays an entire Unicode plane of 65,536 glyphs as one large graphic image, showing a grid of 256 glyphs by 256 glyphs as the default, or (at your option) a long version of 16 glyphs across by 4,096 glyphs down.

Input can be one or more files in Unifont .hex format. They don’t have to be sorted, as unifontpic will populate the entire glyph array of 65,536 code points before writing its output. The input glyph code points should all be unique, as feeding in duplicate code points will produce unpredictable results. Use unidup (1) on a sorted input of .hex files to guarantee no code point duplication.

3.20.4 unifontpic OPTIONS

- d Specify a Dots per Inch (DPI) resolution of nnn. For example, specifying –d120 will encode the bitmap graphics file output as having a resolution of 120 DPI.

- l Produce a long chart, 16 glyphs wide by 4,096 glyphs tall. The default is a wide chart, 256 glyphs wide by 256 glyphs tall.

- t Use tiny numbers for the row and column code point labels. Tiny numbers are on a 4 by 5 pixel grid. Only tiny code point labels appear on the long chart variant; this option only has effect for wide charts (the default, of 256 by 256 glyphs). If this option is not specified for the default 256-by-256 grid, row and column code point labels are taken from Unifont’s glyphs for ‘0’ to ‘9’ and ‘A’ to ‘F’.

- P Print a chart for Unicode plane number plane. The default is Plane 0, the Unicode Basic Multilingual Plane (BMP). The range of Unicode plane range is 0 through 17, inclusive. The plane number is printed on the chart title line.

3.20.5 unifontpic EXAMPLES
Sample usage:
cat *.hex | unifontpic –d120 > unifontpic.bmp
To generate a bitmap that shows combining circles, from the font/ subdirectory:

```
sort plane00/*.hex | unigencircles ttfsrc/combining.txt | unifontpic –d120
>unifontpic.bmp
```

### 3.20.6 unifontpic FILES

*.hex GNU Unifont font files

### 3.20.7 unifontpic SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoveryage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

### 3.20.8 unifontpic AUTHOR

unifontpic was written by Paul Hardy.

### 3.20.9 unifontpic LICENSE

unifontpic is Copyright © 2013 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

### 3.20.10 unifontpic BUGS

No known bugs exist.

### 3.21 unigen-hangul

#### 3.21.1 unigen-hangul NAME

unigen-hangul — Generate Hangul syllables from a Johab 6/3/1 Unifont hex file

#### 3.21.2 unigen-hangul SYNOPSIS

```
unigen-hangul –i hangul-base.hex –o hangul-syllables.hex
```

#### 3.21.3 unigen-hangul DESCRIPTION

unigen-hangul generates Hangul syllables from an input Unifont .hex file encoded in Johab 6/3/1 format. By default, the output is the Unicode Hangul Syllables range, U+AC00..U+D7A3. Options allow the user to specify a starting code point for the output Unifont .hex file, and ranges in hexadecimal of the starting and ending Hangul Jamo code points:

```
Hangul

1100−115E
```

Hangul Jamo initial consonants (choseong)
A960–A97C
Hangul Jamo Extended-A initial consonants (choseong)

1161–11A7
Hangul Jamo medial vowels and diphthongs (jungseong)

D7B0–D7C6
Hangul Jamo Extended-B medial vowels and diphthongs (jungseong)

11A8–11FF
Hangul Jamo final consonants (jongseong).

D7CB–D7FB
Hangul Jamo Extended-B final consonants (jongseong).

A single code point or 0 to omit can be specified instead of a range. A starting code point of one position before a valid starting range for a Hangul jamo series (choseong, jungseong, and/or jongseong) will first use a blank glyph for that jamo, and then cycle through remaining valid code points for the respective choseong, jungseong, or jongseong. A range can span modern and ancient, and even Hangul Jamo Extended-A and Hangul Jamo Extended-B ranges.

For example,

-j3 11A7–D7FB
Will first use no jongseong (because U+11A7 is one before the start of Hangul Jamo jongseong code points), then loop through jongseong in the Hangul Jamo range of U+11A8 through U+11FF, and then loop through jongseong in the Hangul Jamo Extended-B range of U+D7CB through U+D7FB.

3.21.4 unigen-hangul OPTIONS

Function

-h, --help
Print a help message and exit.

-all
Generate all Hangul syllables, using all modern and ancient Hangul in the Unicode range U+1100..U+11FF, assigned code points in the Extended-A range of U+A960..U+A97C, and assigned code points in the Extended-B range of U+D7B0..U+D7FF. WARNING: this will generate over 1,600,000 syllables in a 115 megabyte Unifont .hex format file. The default is to only output the 11,172 modern Hangul syllables.

-c code_point
Starting code point in hexadecimal for output file.

-j1 start-end
Choseong (jamo 1) start-end range in hexadecimal.

-j2 start-end
Jongseong (jamo 2) start-end range in hexadecimal.
Chapter 3: Reference

3.21.5 unigen-hangul EXAMPLES

unigen-hangul -c 1 -j3 11AB-11AB \\
  -i hangul-base.hex -o nieun-only.hex

This command generates Hangul syllables using all modern choseong and jungseong, and only the jongseong nieun (Unicode code point U+11AB). The output Unifont .hex file will contain code points starting at 1. Instead of specifying "-j3 11AB-11AB", simply using "-j3 11AB" will also suffice.

This next example is a series of syllable sets suggested by Ho-Seok Ee for preliminary syllable alignment checking of modern Hangul.

- The first command generates all modern syllables containing no jongseong (final consonant), starting at Unifont hexadecimal glyph location 0x1000; selecting a jongseong value that is out of range (U+1160 in this case) will use a blank filler in place of the jongseong.
- The second command generates all modern syllables containing jongseong Kiyeok (U+11AB), which has a horizontal line extending across the lower portion of a syllable, starting at Unifont hexadecimal glyph location 0x2000.
- The third command generates all modern Hangul syllables containing jongseong Rieul (U+11AF), starting at Unifont hexadecimal glyph location 0x3000.
- The fourth command generates all modern Hangul syllables containing choseong (initial consonant) Rieul (U+1105), starting at Unifont hexadecimal glyph location 0x4000.

Here is the command sequence:

unigen-hangul -c 1000 -j1 1100-1112 -j2 1161-1175 -j3 1160 \\
  -i hangul-base.hex > hangul-prep.hex
unigen-hangul -c 2000 -j1 1100-1112 -j2 1161-1175 -j3 11AB \\
  -i hangul-base.hex >> hangul-prep.hex
unigen-hangul -c 3000 -j1 1100-1112 -j2 1161-1175 -j3 11AF \\
  -i hangul-base.hex >> hangul-prep.hex
unigen-hangul -c 4000 -j1 1105 -j2 1161-1175 -j3 11A8-11C2 \\
  -i hangul-base.hex >> hangul-prep.hex

The resulting .hex file can then be examined with hexdraw, unihex2bmp, etc.
3.21.6 unigen-hangul FILES

Unifont .hex files in Johab 6/3/1 encoding. See unifont-johab631(5) for a description of the input file structure. This program uses functions contained in the file unihangul-support.c.

3.21.7 unigen-hangul SEE ALSO

bdftmplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmp-bump(1), unicoverage(1), unidup(1), unifont(5), unifont-johab631(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigencircles(1), unigen-width(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), uni-hexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.21.8 unigen-hangul AUTHOR

unigen-hangul was written by Paul Hardy.

3.21.9 unigen-hangul LICENSE

unigen-hangul is Copyright © 2023 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.21.10 unigen-hangul BUGS

No known bugs exist.

3.22 unigencircles

3.22.1 unigencircles NAME

unigencircles – Add dashed combining circles to a unifont.hex file

3.22.2 unigencircles SYNOPSIS

unigencircles combining.txt nonprinting.hex < unifont.hex > unifont-circles.hex

3.22.3 unigencircles DESCRIPTION

unigencircles reads a unifont.hex file from STDIN, adds dashed combining circles to the hex strings for code points listed in "combining.txt" but not listed in "nonprinting.hex", and writes the revised set of glyphs in unifont.hex format to STDOUT. The resulting combining character glyphs show the dashed combining circles that appear in The Unicode Standard code charts.

For each code point listed in the "combining.txt" file but not listed in the "nonprinting.hex" file, unigencircles will superimpose a single-width dashed circle in glyphs that are single-width (i.e., their hex glyph strings are 32 characters long) and will superimpose a double-width dashed circle in glyphs that are double-width (i.e., their hex glyph strings are 64 characters long).
3.22.4 unigencircles EXAMPLE

unigencircles combining.txt nonprinting.hex < unifont.hex > unifont−circles.hex

3.22.5 unigencircles FILES

*.hex files for Unifont glyph data

font/ttfsrc/combining.txt for combining code points

font/plane00/nonprinting.hex for non-printing code points

3.22.6 unigencircles SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigenwidth(1), unhex2bmp(1), unhex2png(1), unhexfill(1), unhexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.22.7 unigencircles AUTHOR

unigencircles was written by Paul Hardy.

3.22.8 unigencircles LICENSE

unigencircles is Copyright © 2013 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.22.9 unigencircles BUGS

No known real bugs exist, except that this software does not perform extensive error checking on its input files.

3.23 unigenwidth

3.23.1 unigenwidth NAME

unigenwidth – Generate C code for POSIX wcwidth and wcswidth functions

3.23.2 unigenwidth SYNOPSIS

unigenwidth unifont.hex combining.txt > wccode.c

3.23.3 unigenwidth DESCRIPTION

unigenwidth reads a collection of glyphs in Unifont’s .hex format, then reads a list of combining characters as a hexadecimal list. From these two files, it produces C code to implement the POSIX wcwidth(3) and wcswidth(3) functions. The format of these definitions is based upon POSIX 1003.1-2008 System Interfaces, pages 2251 and 2241, respectively.
3.23.4 unigenwidth EXAMPLE
Sample usage:

```
unigenwidth unifont.hex combining.txt > wccode.c
```

3.23.5 unigenwidth FILES
*.hex files for Unifont glyph data; combining.txt for combining code points.

3.23.6 unigenwidth SEE ALSO
bdfimplode(1), hex2bdf(1), hex2otf(1), hex2pdf(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1),
unibmpbump(1), unisegment(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1),
unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1),
unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.23.7 unigenwidth AUTHOR
unigenwidth was written by Paul Hardy.

3.23.8 unigenwidth LICENSE
unigenwidth is Copyright © 2013 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.23.9 unigenwidth BUGS
No known real bugs exist, except that this software does not perform extensive error checking
on its input files.

3.24 unihex2bmp

3.24.1 unihex2bmp NAME
unihex2bmp – GNU Unifont .hex file to bitmap graphics file converter

3.24.2 unihex2bmp SYNOPSIS

```
unihex2bmp [−p hexpage] [−i input_file.hex] [−o output_file.bmp] [−f] [−w]
```

3.24.3 unihex2bmp DESCRIPTION
unihex2bmp reads a GNU Unifont .hex file Unicode page of 256 code points and converts
the page into a Microsoft Windows Bitmap (.bmp) or Wireless Bitmap (.wbmp) file. The
bitmap file displays the glyphs of a Unicode block of 256 code points in a 32 by 32 pixel
grid. The glyphs themselves must be 16 rows high, and 8, 16, 24, or 31 (encoded as 32)
columns wide. The default page is 0; that is, the range U+0000 through U+00FF.
The bitmap can be printed. It can also be edited with a bitmap editor. An edited bitmap
can then be re-converted into a GNU Unifont .hex file with the unibmp2hex command.
3.24.4 unihex2bmp OPTIONS

−p  Extract page hexpage from the .hex file. The default is Page 0 (Unicode range U+0000 through U+00FF). Note that "page" is not a standard Unicode term. It refers to an output bitmap graphics page of 16 by 16 code points.

−i  Specify the input file. The default is STDIN.

−o  Specify the output file. The default is STDOUT.

−f  "Flip" (transpose) the grid, swapping rows and columns from the Unicode standard orientation.

−w  Produce a Wireless Bitmap format file instead of a Microsoft Windows Bitmap file.

3.24.5 unihex2bmp EXAMPLE

Sample usage:
   unihex2bmp −i my_input_file.hex −o my_output_file.bmp

3.24.6 unihex2bmp FILES

*.hex GNU Unifont font files

3.24.7 unihex2bmp SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-width(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.24.8 unihex2bmp AUTHOR

unihex2bmp was written by Paul Hardy.

3.24.9 unihex2bmp LICENSE

unihex2bmp is Copyright © 2007 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.24.10 unihex2bmp BUGS

No known real bugs exist, except that this software does not perform extensive error checking on its input files. If they’re not in the format of the original GNU Unifont .hex file, all bets are off. Lines can be terminated either with line feed, or carriage return plus line feed.

3.25 unihex2png

3.25.1 unihex2png NAME

unihex2png − GNU Unifont .hex file to Portable Network Graphics converter
3.25.2 unihex2png SYNOPSIS

unihex2png [-i input_file.hex] -o output_file.png [-p pagenum] [-r rows]

3.25.3 unihex2png DESCRIPTION

unihex2png reads a page of 256 Unicode code points from a GNU Unifont .hex file and converts the page into a Portable Network Graphics (PNG) file. The graphics file displays the glyphs of a Unicode block of 256 code points in a 32 by 32 pixel grid, or in a 40 by 40 pixel grid if "-r 32" is specified. The glyphs themselves can be either 16, 24, or 32 pixels tall depending on the "-r" parameter. They can be 8, 16, 24, or 32 pixels wide (widths of 32 are only supported if "-r 32" is specified). The default page is 0; that is, the range U+0000 through U+00FF, inclusive.

The PNG file can be printed. It can also be edited with a graphics editor. An edited PNG file can then be re-converted into a GNU Unifont .hex file with the unipng2hex command.

3.25.4 unihex2png OPTIONS

- Specify the input file. If not omitted, a Unifont .hex file is read from STDIN.
- Specify the output file.
- Extract page pagenum from the .hex file. The default is Page 0 (Unicode range U+0000 through U+00FF). Note that "page" is not a standard Unicode term. It refers to an output bitmap graphics page of 16 by 16 code points.
- Specify the rows of pixels in the output glyphs. Valid values are 16, 24, and 32. The default is 16 pixel rows tall.
- Print a help message of options and exit.

3.25.5 unihex2png EXAMPLE

Sample usage:

unihex2png -i my_input_file.hex -o my_output_file.png

3.25.6 unihex2png FILES

*.hex GNU Unifont font files

3.25.7 unihex2png SEE ALSO

bdftonlode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmp-bump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-width(1), unihex2bmp(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.25.8 unihex2png AUTHOR

unihex2png was written by Andrew Miller, starting by converting Paul Hardy's unihex2bmp C program to Perl.
3.25.9 unihex2png LICENSE

unihex2png is Copyright © 2007 Paul Hardy, © 2013 Andrew Miller.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.25.10 unihex2png BUGS

No known real bugs exist, but the optional pixel rows parameter is not yet supported by all
other Unifont utilities. Use of glyphs taller than the default of 16 pixels is considered ex-
perimental. Currently unihex2png, unipng2hex, hexdraw, and hex2bdf tentatively support
glyphs that are 16, 24, and 32 pixels tall.
Also, there is no extensive error checking on input files. If they’re not in the format of the
original GNU Unifont .hex file, all bets are off. Lines can be terminated either with line
feed, or carriage return plus line feed.

3.26 unihexfill

3.26.1 unihexfill NAME

unihexfill – Generate range of Unifont 4- or 6-digit hexadecimal glyphs

3.26.2 unihexfill SYNOPSIS

unihexfill < unassigned-ranges.txt > filler-glyphs.hex

3.26.3 unihexfill DESCRIPTION

unihexfill is a shell script that reads a list of code point ranges from STDIN and produces
filler glyphs of 4- or 6-digit code points on STDOUT. The format of the input file is a
combination of comment lines, single code points on a line, and start/stop pairs of code
points on a line separated by a space. Comment lines start with a semicolon (‘;’) by
convention. Start and stop code points are strings of hexadecimal digits, by convention
either four or six digits. unihexfill invokes unihexgen for each non-comment line in its input
file. If a codepoint is less than or equal to "FFFF" (i.e., 0xFFFF), a four-digit hexadecimal
number is encoded within the corresponding Unifont glyph as two digits on each of two
rows. Otherwise, a six-digit hexadecimal number is encoded as three digits on each of two
rows.

3.26.4 unihexfill OPTIONS

There are no options.

3.26.5 unihexfill FILES

*.txt as input; *.hex as output.

3.26.6 unihexfill EXAMPLE

In the Unifont source package, the file font/plane01/Makefile generates Unicode Plane 1
hexadecimal filler glyphs of unassigned code points within assigned scripts with this single-
line command:
3.26.7 unihexfill SEE ALSO
bdfileplode(1), hex2bdfl(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigenwidth(1),
unihex2bmap(1), unihex2png(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1), unipng2hex(1)

3.26.8 unihexfill AUTHOR
unihexfill was written by Paul Hardy.

3.26.9 unihexfill LICENSE
unihexfill is Copyright © 2014 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.26.10 unihexfill BUGS
No known real bugs exist, except that this software does not perform extensive error checking
on its input file. Any line that appears to begin with a hexadecimal digit is fed to
unihexgen.

3.27 unihexgen

3.27.1 unihexgen NAME
unihexgen – Generate Unifont 4- or 6-digit hexadecimal glyphs

3.27.2 unihexgen SYNOPSIS
unihexgen startpoint endpoint

3.27.3 unihexgen DESCRIPTION
unihexgen produces unifont.hex entries in the Unicode code point range startpoint to endpoint
(specified as the two command-line arguments). Output is to STDOUT. If a codepoint
is less than or equal to "FFFF" (i.e., 0xFFFF), a four-digit hexadecimal number is encoded
within the corresponding Unifont glyph as two digits on each of two rows. Otherwise, a
six-digit hexadecimal number is encoded as three digits on each of two rows.

3.27.4 unihexgen OPTIONS
There are no options.

3.27.5 unihexgen FILES
*.hex as output.
3.27.6 unihexgen EXAMPLE

To generate the Private Use Area glyphs in the Unicode range U+E000..U+F8FF, invoke unihexgen with these arguments:

```
unihexgen e000 f8ff
```

3.27.7 unihexgen SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-width(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.27.8 unihexgen AUTHOR

unihexgen was written by Paul Hardy.

3.27.9 unihexgen LICENSE

unihexgen is Copyright © 2013 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.27.10 unihexgen BUGS

No known real bugs exist, except that this software does not perform extensive error checking on its command-line arguments. If they’re not in the format of the original bitmapped output from unihexgen, all bets are off.

3.28 unihexpose

3.28.1 unihexpose NAME

unihexpose − Convert a GNU Unifont glyph into column-major format

3.28.2 unihexpose SYNOPSIS

```
unihexpose < input-font.hex > transposed-font.hex
```

3.28.3 unihexpose DESCRIPTION

unihexpose reads a sorted GNU Unifont .hex file from STDIN and writes a transposed Unifont .hex file to STDOUT. The output .hex file contains glyphs encoded in column major format, with 8 rows per output column. This will simplify output of Unifont glyphs on graphics displays with controller chips designed to autoincrement across a line of 8 pixel rows, one column at a time.

The output .hex file contains a Unifont glyph in two halves: the first half is the first 8 pixels rows, and the second half the second 8 pixel rows. The glyphs are 8 or 16 columns wide.
3.28.4 unihexpose FILES
* .hex GNU Unifont font files

3.28.5 unihexpose SEE ALSO
bdfstream(1), hex2bdff(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmp-bump(1), unicoverage(1), unidup(1), unifont(5), unifont-johab631(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigen-circles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.28.6 unihexpose AUTHOR
unihexpose was written by Paul Hardy.

3.28.7 unihexpose LICENSE
hexdraw is Copyright © 2023 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.28.8 unihexpose BUGS
No known bugs exist.

3.29 unihexrotate

3.29.1 unihexrotate NAME
unihexrotate – rotate Unifont hex glyphs in quarter turn increments

3.29.2 unihexrotate SYNOPSIS
unihexrotate [−n quarter-turns] [input-font.hex] > output-font.hex

3.29.3 unihexrotate DESCRIPTION
unihexrotate reads a GNU Unifont .hex format file named on the command line, or from stdin if no filename is given. If a number of quarter turns is specified, it rotates each glyph clockwise by that number of quarter turns, or counterclockwise if the number is negative. The resulting modified .hex file is written to stdout. The format of a .hex file is described in the unifont(5) man page.

If the number of quarter turns is not specified, unihexrotate will rotate each glyph by a default value of one quarter turn (i.e., 90 degrees) clockwise.

unihexrotate only supports Unifont .hex files with glyphs that are 8 columns and 16 columns wide. If an 8-column glyph is rotated, it is first centered within a 16-column glyph so the rotation will be performed on a 16-by-16 pixel glyph. One additional use of this program is that a glyph can be converted from 8\~ columns wide to 16\~ columns wide by specifying a rotation of 0\~ quarter turns.
3.29.4 unihexrotate OPTIONS

−n Specify an integer number of quarter turn clockwise rotations to perform, typically as "−n\^0" through "−n\^3" (alternatively as "−n=0" through "−n=3") for rotations of 0, 90, 180, or 270 degrees, respectively. Negative values will perform counterclockwise rotations by the specified number of quarter turns.

3.29.5 unihexrotate EXAMPLES

This example extracts the Mongolian glyphs in the "unifont-base.hex" file (located in the "font/plane00" directory) and pipes them to unihexrotate to rotate by the default amount of one quarter turn clockwise (i.e., by 90 degrees). The basic Mongolian glyphs are in the Unicode range U+1800 through U+18AF. The original glyphs in "unifont-base.hex" are drawn for horizontal rendering (as per the Unicode Standard). The output rotated glyphs are suitable for rendering Mongolian in its traditional vertical form. Because no filename is specified to unihexrotate in this example, its input is taken from stdin.

grep "^18[0-9A-Z]" unifont-base.hex | unihexrotate > vertical.hex

The next example reverses the rotation performed above, by one quarter turn counterclockwise. In this example, the input filename "vertical.hex" is specified.

unihexrotate −n −1 vertical.hex > horizontal.hex

3.29.6 unihexrotate FILES

Unifont .hex format input and output files.

3.29.7 unihexrotate SEE ALSO

bdfimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1), unicoverage(1), umidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-width(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.29.8 unihexrotate AUTHOR

unihexrotate was written by David Corbett.

3.29.9 unihexrotate LICENSE

unihexrotate is Copyright © 2019 David Corbett.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.29.10 unihexrotate BUGS

No known bugs exist.

3.30 unijohab2html
3.30.1 unijohab2html NAME
unijohab2html – Show combined Hangul syllable jamo juxtaposition as a web page

3.30.2 unijohab2html SYNOPSIS
unijohab2html -i hangul-base.hex -o hangul-syllables.hex

3.30.3 unijohab2html DESCRIPTION
unijohab2html reads glyphs in a Unifont .hex file organized in a Johab 6/3/1 format (6 choseong variations, 3 jungseong variations, and 1 jongseung variation). It then produces an HTML output file showing all Hangul letters in a color-coded output in 16-by-16 cell grids: blue cells show where choseong glyphs appear; green for jungseong; and red for jongseong.

Any overlaps are shown with a combination of the colors. For example, where one or more choseong (blue) letters overlap jungseong (green), the associated cell will be blue + green = cyan. Likewise, jungseong (green) and jongseong (red) overlaps will appear as green + red = yellow, etc. There is a color key at the beginning of the HTML table.

If any overlaps of choseong and jungseong exist, these will then be output in the HTML file identifying the overlapping letters in a combined Unifont .txt file format. From these overlaps, a Hangul font creator can decide whether the overlap is acceptable or should be modified.

If an overlap is with an ancient vowel or diphthong (i.e., U+1176, HANGUL JUNGSEONG A-O through U+11A7, HANGUL JUNGSEONG O-YAE), it is marked as an "Ancient Vowel" in this ending list of overlaps. If a choseong is ancient (i.e., U+11C3, HANGUL JONGSEONG KIYEOK-RIEUL, through U+11FF, HANGUL JONGSEONG SSANGNIEUN), it is marked as "Ancient Choseong" in the output. Overlapping choseong and jungseong in the Hangul Jamo Extended-A and Extended-B ranges are listed as "Extended Ancient". This identification of ancient and extended ancient versus modern Hangul will allow a font designer to optimize modern Hangul when weighing design tradeoffs.

3.30.4 unijohab2html OPTIONS

Function

-h, --help
Print a help message and exit.

-i input_file
Unifont hangul-base.hex formatted input file.

-m, --modern
Only examine modern Hangul letters.

-o output_file
HTML output file, viewable in a web browser.

3.30.5 unijohab2html EXAMPLES
unijohab2html -i hangul-base.hex -o hangul-overlaps.html
unijohab2html < hangul-base.hex > hangul-overlaps.html
Chapter 3: Reference 51

3.30.6 unijohab2html FILES

Unifont .hex files in Johab 6/3/1 encoding. See unifont-johab631(5) for a description of the input file structure. This program uses functions contained in the file unihangul-support.c.

3.30.7 unijohab2html SEE ALSO

bdflimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1), hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmp-bump(1), unicoverage(1), unidup(1), unifont(1), unifont-johab631(5), unifont-viewer(1), unifont1per(1), unifontchojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigen-hangul(1), unigencircles(1), unigenwidth(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1), unijohab2html(1), unipagecount(1), unipng2hex(1)

3.30.8 unijohab2html AUTHOR

unijohab2html was written by Paul Hardy.

3.30.9 unijohab2html LICENSE

unijohab2html is Copyright © 2023 Paul Hardy.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.30.10 unijohab2html BUGS

No known bugs exist.

3.31 unipagecount

3.31.1 unipagecount NAME

unipagecount – Count the assigned code points in a GNU Unifont .hex file

3.31.2 unipagecount SYNOPSIS

unipagecount [-Pplane] [-ppagenum] [-h1 -l]

3.31.3 unipagecount DESCRIPTION

unipagecount reads a GNU Unifont .hex file from STDIN and prints a 16 by 16 grid of the number of defined code points in each 256 character block within a Unicode plane to STDOUT. Code points proceed from left to right, then top to bottom. In all planes, code points U+*FFFE and U+*FFFF are not expected in the input hex file; they are reserved and always counted as being present in a plane.

3.31.4 unipagecount OPTIONS

-P Select a Unicode plane, from 0 through 16, inclusive. If not specified, unipagecount defaults to Plane 0 (the Basic Multilingual Plane).
Just print information on one 256 code point "page" rather than the entire
Basic Multilingual Plane. This prints a 16 by 16 table with an asterisk in every
code point that has an assigned glyph.

Print an HTML table with color-coded cell background colors instead of a plain
text table.

[The letter "l"]: Print hyperlinks to font bitmaps in the HTML table. To
create the bitmaps themselves, use the unihex2bmp program. The bitmaps are
assumed to be in the directory "bmp/".

3.31.5 unipagecount FILES
*.hex GNU Unifont font files

3.31.6 unipagecount SEE ALSO
bdflimplode(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmp-
bump(1), unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifont-
chojung(1), unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-
width(1), unihex2bmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihex-
rotate(1), unijohab2html(1), unipng2hex(1)

3.31.7 unipagecount AUTHOR
unipagecount was written by Paul Hardy.

3.31.8 unipagecount LICENSE
unipagecount is Copyright © 2007, 2014 Paul Hardy.
This program is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

3.31.9 unipagecount BUGS
No known real bugs exist, except that this software does not perform extensive error check-
ing on its input files. If they're not in the format of the original GNU Unifont .hex file, all
bets are off.

3.32 unipng2hex

3.32.1 unipng2hex NAME
unipng2hex – Portable Network Graphics to GNU Unifont .hex file converter

3.32.2 unipng2hex SYNOPSIS
unipng2hex –i input_file.png [–o output_file.hex] [–w width]

3.32.3 unipng2hex DESCRIPTION
unipng2hex reads a PNG file produced by unihex2png before or after editing, and converts
it back into a Unifont .hex format file. The PNG file contains a block of 256 Unicode code
points arranged in a 16 by 16 grid. Each code point appears in a 32 by 32 or a 40 by 40 pixel grid. Characters are either 16, 24 or 32 pixel rows high, depending on the "−r" parameter specified by \texttt{unihex2png}. They can be 8, 16, 24, or 32 pixel columns wide (widths of 32 are only supported for 32 pixel row tall characters).

3.32.4 \texttt{unipng2hex} OPTIONS

\begin{itemize}
  \item \texttt{−i} Specify the input file.
  \item \texttt{−o} Specify the output file. If omitted, a file in the Unifont .hex format is written to STDOUT.
  \item \texttt{−w} Specify the minimum width of the output glyphs. Valid values are 16, 24, and 32. The default is no minimum width.
  \item \texttt{−h} Print a help message of options and exit.
\end{itemize}

3.32.5 \texttt{unipng2hex} EXAMPLE

Sample usage:

\begin{verbatim}
unipng2hex −i my_input_file.png −o my_output_file.hex
\end{verbatim}

3.32.6 \texttt{unipng2hex} FILES

\begin{verbatim}
*.png graphics files
\end{verbatim}

3.32.7 \texttt{unipng2hex} SEE ALSO

\begin{verbatim}
bdfind(1), hex2bdf(1), hex2otf(1), hex2sfd(1), hexbraille(1), hexdraw(1), hexkinya(1),
hexmerge(1), johab2syllables(1), johab2ucs2(1), unibdf2hex(1), unibmp2hex(1), unibmpbump(1),
unicoverage(1), unidup(1), unifont(5), unifont-viewer(1), unifont1per(1), unifontchojung(1),
unifontksx(1), unifontpic(1), unigen-hangul(1), unigencircles(1), unigen-width(1),
unihexbmp(1), unihex2png(1), unihexfill(1), unihexgen(1), unihexpose(1), unihexrotate(1),
unijohab2html(1), unipagecount(1)
\end{verbatim}

3.32.8 \texttt{unipng2hex} AUTHOR

\texttt{unipng2hex} was written by Andrew Miller, starting by converting Paul Hardy’s \texttt{unibmp2hex} C program to Perl.

3.32.9 \texttt{unipng2hex} LICENSE

\texttt{unipng2hex} is Copyright © 2007, 2008 Paul Hardy, © 2013 Andrew Miller.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

3.32.10 \texttt{unipng2hex} BUGS

No known real bugs exist, but the optional pixel rows parameter is not yet supported by all other Unifont utilities. Use of glyphs taller than the default of 16 pixels is considered experimental. Currently \texttt{unihex2png}, \texttt{unipng2hex}, \texttt{hexdraw}, and \texttt{hex2bdf} tentatively support glyphs that are 16, 24, and 32 pixels tall.
Also, this software does not perform extensive error checking on its input files. If they’re not in the format of the original PNG output from \texttt{unihex2png}, all bets are off.

If the output file is for a "page" containing space code points and the PNG file squares for those code points are not empty, \texttt{unipng2hex} preserves the graphics as they are drawn. \texttt{unipng2hex} is designed to work with black and white pixels; do not use other colors.