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

# arabjoin - a simple filter to render Arabic text
# © 1998-06-18 roman@czyborra.com
# Freeware license at http://czyborra.com/
# Latest version at http://czyborra.com/unicode/
# PostScript printout at http://czyborra.com/unicode/arabjoin.ps.gz

# This filter takes Arabic text (encoded in UTF-8 using the Unicode
# characters from the U+0600 Arabic block in logical order) as input
# and performs Arabic glyph joining on it and outputs a UTF-8 octet
# stream that is no longer logically arranged but in a visual order
# which gives readable results when formatted with a simple Unicode
# renderer like Yudit that does not handle Arabic differently yet
# but simply outputs all glyphs in left-to-right order.

# This little script also demonstrates that Arabic rendering is not
# that complicated after all (it makes you wonder why some software
# companies are still asking hundreds of dollars from poor students
# who just want to print their Arabic texts) and that even Perl 4 can
# handle Unicode text in UTF-8 without any nifty new add-ons.

# Usage examples:

# echo "أهلاً بالعالم!" | arabjoin
# prints ﺃﻫﻼً ﺑﺎﻟﻌﺎﻟﻢ!
# which is the Arabic version of "Hello world!"

# | recode ISO-8859-6..UTF-8 | arabjoin | uniprint -f cyberbit.ttf
# prints an Arabic mail of charset=iso-8859-6-i on your printer

# | arabjoin | xviewer yudit
# delegates an Arabic UTF-8 message to a better viewer

# ftp://sunsite.unc.edu/pub/Linux/apps/editors/X/ has uniprint in yudit-1.0
# ftp://ftp.iro.umontreal.ca/pub/contrib/pinard/pretest/ has recode-3.4g
# http://czyborra.com/unicode/ has arabjoin
```

```
# http://czyborra.com/unix/ has xviewer
# http://www.bitstream.com/cyberbit.htm or
# ftp://ccic.ifcss.org/pub/software/fonts/unicode/ms-win/ or
# ftp://ftp.irdu.nus.sg/pub/language/bitstream/ has cyberbit.ttf

# This is how we do it: First we learn the presentation forms of each
# Arabic letter from the end of this script:

while(<DATA>)
{
   ($char, $_) = /^(\S+)\s+(\S+)/;
   ($isolated{$char},$final{$char},$medial{$char},$initial{$char}) =
         /([\xC0-\xFF][\x80-\xBF]+)/g;
}

# Then learn the (incomplete set of) transparent characters:

foreach $char (split (" ", "
ً ٌ ٍ َ ُ ِ □
□□□□□□□□□□□□□□□□□□"))
{
   $transparent{$char}=1;
}

# Finally we can process our text:

while (<>)
{
   s/\n$//; # chop off the end of the line so it won't jump upfront

   @uchar = # UTF-8 character chunks
         /([\x00-\x7F]|[\xC0-\xFF][\x80-\xBF]+)/g;

   # We walk through the line of text and do contextual analysis:

   for ($i = $[; $i <= $#uchar; $i = $j)
   {
```

```
        for ($b=$uchar[$j=$i]; $transparent{$c=$uchar[++$j]};){};

        # The following assignment is the heart of the algorithm.
        # It reduces the Arabic joining algorithm described on
        # pages 6-24 to 6-26 of the Arabic character block description
        # in the Unicode 2.0 Standard to four lines of Perl:

        $uchar[$i] =  $a && $final{$c} && $medial{$b}
        ||  $final{$c} && $initial{$b}
        ||  $a && $final{$b}
        ||  $isolated{$b}
        ||  $b;

        $a = $initial{$b} && $final{$c};
}

# Until the Unicode Consortium publishes its Unicode Technical
# Report #9 (Bidirectional Algorithm Reference Implementation)
# at http://www.unicode.org/unicode/reports/techreports.html
# let us oversimplify things a bit and reverse everything:

$_= join ('', reverse @uchar);

# The following 8 obligatory LAM+ALEF ligatures are encoded in the
# U+FE70 Arabic Presentation Forms-B block in Unicode's
# compatibility zone:

s/ﻵ/ﻷ/g;
s/ﻶ/ﻸ/g;
s/ﻷ/ﻹ/g;
s/ﻸ/ﻺ/g;
s/ﻹ/ﻻ/g;
s/ﻺ/ﻼ/g;
s/ﻵ/ﻷ/g;
s/ﻶ/ﻸ/g;

# Bitstream's Cyberbit font offers 57 of the other 466 optional
```

ligatures in the U+FB50 Arabic Presentation Forms-A block:

```
s/تم/ﰎ/g;
s/في/ﰲ/g;
s/لج/ﰿ/g;
s/لح/ﱀ/g;
s/لخ/ﱁ/g;
s/لم/ﱂ/g;
s/لى/ﱃ/g;
s/لي/ﱄ/g;
s/نم/ﱎ/g;
s/ ٌّ/ﱞ/g;
s/ ٍّ/ﱟ/g;
s/ َّ/ﱠ/g;
s/ ُّ/ﱡ/g;
s/ ِّ/ﱢ/g;
s/بر/ﱪ/g;
s/بن/ﱭ/g;
s/بي/ﱯ/g;
s/تر/ﱰ/g;
s/تن/ﱳ/g;
s/تي/ﱵ/g;
s/ني/ﲏ/g;
s/ير/ﲑ/g;
s/ين/ﲔ/g;
s/بج/ﲜ/g;
s/بح/ﲝ/g;
s/بخ/ﲞ/g;
s/بم/ﲟ/g;
s/تج/ﲡ/g;
s/تح/ﲢ/g;
s/تخ/ﲣ/g;
s/تم/ﲤ/g;
s/ثم/ﲦ/g;
s/جم/ﲨ/g;
s/حم/ﲪ/g;
s/خم/ﲬ/g;
```

```
    s/سم/ﺳﻢ/g;
    s/لج/ﻟﺞ/g;
    s/لح/ﻟﺢ/g;
    s/لخ/ﻟﺦ/g;
    s/لم/ﻟﻢ/g;
    s/له/ﻟﻪ/g;
    s/مج/ﻣﺞ/g;
    s/مح/ﻣﺢ/g;
    s/مخ/ﻣﺦ/g;
    s/مم/ﻣﻢ/g;
    s/نج/ﻧﺞ/g;
    s/نح/ﻧﺢ/g;
    s/نخ/ﻧﺦ/g;
    s/نم/ﻧﻢ/g;
    s/يج/ﻳﺞ/g;
    s/يح/ﻳﺢ/g;
    s/يخ/ﻳﺦ/g;
    s/يم/ﻳﻢ/g;
    s/لمح/ﻟﻤﺢ/g;
    s/الله/ﷲ/g;
    s/وسلم/ﻭﺳﻠﻢ/g;
    s/جل/ﺟﻞ/g;

    print "$_\n";
}

# The following table lists the presentation variants of each
# character.  Each value from the U+0600 block means that the
# necessary glyph variant has not been assigned a code in Unicode's
# U+FA00 compatibility zone.  You may want to insert your private
# glyphs or approximation glyphs for them:

__END__
ء        ﺀ
آ        ﺁﺂ
أ        ﺃﺄ
ؤ        ﺅﺆ
```

إ إإ

ئ ئئئئ

ا اا

ب ببببب

ة ةة

ت تتتت

ث ثثثث

ج ججججج

ح ححح

خ خخخخ

د دد

ذ ذذ

ر رر

ز زز

س سسسس

ش ششش

ص صصص

ض ضضض

ط طططط

ظ ظظظظ

ع ععع

غ غغغ

ـ ـــــــ

ف فففف

ق قققق

ك ككك

ل لل

م ممم

ن نن

ه هههه

و وو

ى ى // □□

ي ييي

□ □ // □

□ □□

□ □□

□ □