Dokumentation der Original Perl-Skripte

1. Skript 1: Web-Crawler und Datenverarbeitung

Dieses Perl-Skript dient dazu, Web-Daten von einer bestimmten URL abzurufen, diese zu verarbeiten und die Ergebnisse in HTML- und GIF-Dateien zu speichern.

Allgemeine Beschreibung

- **Umgebungsvariablen**: Setzt `GS_LIB` auf das lokale Verzeichnis `./fonts`.
- **Dateipfade**: Definiert Pfade zu HTML und Datenverzeichnissen.
- **Externe Abhängigkeiten**: Beinhaltet externe Perl-Module `cgi-lib.pl`, `form.ph`, `timelocal.pl`.
- **Initialisierung**: Definiert verschiedene Dateinamen und Variablen.

Funktionen

1. **Start**:
    - **Daten einlesen**: Ruft `ReadParse` auf, um eingehende CGI-Daten zu lesen.
    - **Überprüfung**: Prüft, ob der Parameter `name` leer ist. Falls ja, wird ein Formular angezeigt und das Skript beendet.
    - **Vorbereitung**: Setzt und verarbeitet den Suchbegriff.
2. **Datenabruf**:
    - **URL abrufen**: Wechselt in das `tmp`-Verzeichnis und führt einen `w3mir`-Befehl aus, um eine Suche bei AltaVista durchzuführen.
    - **Daten filtern**: Filtert und extrahiert URLs aus dem Ergebnis.
3. **HTML-Verarbeitung**:
    - **Bilder filtern**: Extrahiert Bild-URLs aus dem HTML-Code.
    - **GIF-Verarbeitung**: Öffnet und verarbeitet GIF-Dateien.
4. **Ergebnis speichern**:
    - **HTML**: Speichert die gefilterten Daten und bereitet eine HTML-Datei vor.
    - **GIF**: Generiert ein GIF-Bild basierend auf den HTML-Daten.
5. **Bereinigungen**:
    - **Löschen**: Löscht temporäre Dateien.

Berichterstellung

Funktion des Skripts: Das Perl-Skript automatisiert den Prozess des Abrufens und Verarbeitens von Web-Daten. Es beginnt mit der Abfrage von AltaVista, filtert die Ergebnisse, insbesondere Bilder, und bereitet diese Daten dann für die Anzeige in HTML und GIF auf. Das Skript ist darauf ausgelegt, die gefilterten Informationen in einem HTML-Dokument zu speichern und eine entsprechende Bilddatei zu generieren.

Spezifische Verarbeitungsschritte:

- **Datenabruf und Filterung**: Nutzt externe Tools (`w3mir`, `getHREF.pl`) zur Web-Abfrage und Linkextraktion.
- **Bildverarbeitung**: Identifiziert und speichert Bild-URLs, konvertiert diese in GIF-Formate und bereinigt HTML-Inhalte.
- **Ergebnisgenerierung**: Erzeugt HTML-Seiten und Bilddateien mit eingebetteten Daten, die auf der ursprünglichen Web-Abfrage basieren.

Dokumentation des PostScript-Teils

2. Skript 2: PostScript Schriftarten und allgemeine Definitionen

Allgemeine Beschreibung

- **Font-Erstellung**: Definiert Funktionen und Befehle zur Erstellung von benutzerdefinierten Schriftarten in PostScript.
- **Struktur**: Definiert Schriftarten und deren Zeichencodes, die zur Erstellung von Grafiken verwendet werden können.

Funktionen

1. **MakeKHFont**:
    - **Font-Erstellung**: Initialisiert eine neue Schriftart mit bestimmten Eigenschaften wie `ItalicAngle`, `PenWidth`, `UniqueID` und `FontName`.
    - **Zeichencodierung**: Setzt die Zeichenkodierung und die Darstellung der Schriftzeichen.
2. **BuildKHChar**:
    - **Zeichendaten**: Verarbeitet die Zeichendaten und stellt sicher, dass sie korrekt skaliert und gezeichnet werden.
3. **Zeichendefinitionen**:
    - **Symbole und Zeichen**: Definiert verschiedene Zeichen und Symbole wie `hyphen`, `bar`, `periodcentered`, etc.

Berichterstellung

Funktion des PostScript-Skripts: Das PostScript-Skript dient zur Definition und Erstellung von benutzerdefinierten Schriftarten. Diese können in Grafiken und Dokumenten verwendet werden, um spezifische Symbole und Textstile darzustellen.

Spezifische Verarbeitungsschritte:

- **Schrifterstellung**: Die Funktion `MakeKHFont` erlaubt die dynamische Erstellung von Schriftarten mit benutzerdefinierten Eigenschaften.
- **Zeichencodierung**: Definiert, wie die einzelnen Zeichen dargestellt werden, einschließlich ihrer Abmessungen und Linienführung.

- **Symbole und Zeichen**: Stellt eine Vielzahl von speziellen Zeichen zur Verfügung, die in verschiedenen Kontexten verwendet werden können.

Fazit

Zusammenfassung von "Without Addresses"

"Without Addresses" ist ein interaktives Kunstprojekt von Blank & Jeron, das die Themen Identität, Anonymität und digitale Kommunikation erforscht. Es besteht aus zwei wesentlichen Skripten: **check** und **font.pl**, die verwendet werden, um Textnachrichten anonym zu versenden und zu empfangen.

check verarbeitet die Eingaben und generiert die grafische Ausgabe im Browserfenster, während **font.pl** den Font für die Handschrift erzeugt.

Kommentarierter Quelltext von check (PDF)

Kommentarierter Quelltext von font.pl (PDF)

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

# Uncomment to set the Ghostscript library path
# $ENV{'GS_LIB'} = "./fonts";

use Socket;
require "../cgi-lib.pl";
require "./form.ph";
require "timelocal.pl";

# Initialize data and temp directories
$data = "data";
$temp = "temp";

# Get current time for unique ID
$id = time;

# Autoflush the output
$| = 1;

# Set the content type for HTTP response
print "Content-type: text/html\n\n";

# Run makefont script
system("./makefont");

# Parse form data
&ReadParse;

# Check if 'name' parameter is empty
if ($in{'name'} eq "") {
    print $form;
    # exit;  # Exit is commented out
}

# Set search pattern and name
$pat = "\"$in{'name'}\"";
$name = $in{'name'};
$pat =~ s/\ /\\\+/g;  # Escape spaces for the search query
$name = "sex";  # Hardcoded name

# Change directory to 'files'
chdir("files");

# Perform web search using w3mir and extract URLs using getHREF.pl
open(F, "../w3mir -l http://www.altavista.com/cgi-bin/query?pg=q\\&sc=on\\&hl=on\
\&q=$name\\&kl=XX\\&stype=stext | ../getHREF.pl |");
while (<F>) {
    push(@out, $_);
}
close(F);

# Get the first URL and shift the array
$where = $out[0];
shift(@out);

# Open a file to save the output
open(F, ">$id");

# Clean up the output data
map {s/\)//eg} @out;
map {s/\(//eg} @out;
map {s/
//eg} @out;
map {s/\&\#(\d{1,3})\;//eg} @out;
map {s/(\&.*?\;)//gesxi} @out;
```

```
map {s/(<SCRIPT.*?SCRIPT>)/&gather($1)/gesxi} @out;
map {s/(<.*?>)/&gather($1)/gesxi} @out;
map {s/(\{.*?\})//gesxi} @out;
map {s/(\".*?\")//gesxi} @out;

# Filter out HTML tags and empty lines
@out2 = grep(!/<.*?/, @out);
@out = grep(!/.*?>/, @out2);
shift(@out);

# Write cleaned data to the file
print F @out;
close(F);

# Convert GIF to PNM and then to PS (PostScript)
open(IN, "/usr/X11R6/bin/giftopnm $id.gif | /usr/X11R6/bin/pnmtops |");
while ($line = <IN>) {
    push(@image, $line) unless $line =~ /showpage/;
}
close(IN);

# PostScript header for drawing
$start = "
gsave
72 2.54 div dup scale
5 10 translate

/top 2 31 exp 1 sub def

0 0 .5 setrgbcolor
usertime srand
%rand top div 2 mul setlinewidth 1 setlinecap
.1 setlinewidth 1 setlinecap
1 1 15 { pop
    rand top div 15 mul
    rand top div 24 mul
    moveto 0 0 rlineto
} for
stroke

grestore

/LM 70 def
/MM 100 def
/lh {gsave newpath 0 0 moveto
    ({) true charpath flattenpath pathbbox
        4 1 roll
            pop  pop pop grestore
    } def

/nl {currentpoint
    lh 1.4 mul sub
    exch pop
    LM exch
    moveto
    } def

gsave
/Hershey-Script-Complex   findfont
  24 scalefont
    setfont
        MM 600 moveto
270 rotate
1 1 0 setrgbcolor
        (      $where) show
```

```
grestore
%/Hershey-Script-Complex-Bold   findfont
/KHRand   findfont
  24 scalefont
    setfont
        LM 700 moveto
0 0 .5 setrgbcolor
        (              $name) show  nl

/KHRand                   findfont
  20 scalefont
    setfont
%        LM 700 moveto
0 0 .5 setrgbcolor
";

# PostScript footer
$end = "
        (...) show  nl
showpage
";

# Read input file
open(F, "<$id");
@input = <F>;
close(F);

# Join input lines and replace spaces with newlines
$string = join(" ", @input);
$string =~ s/ /\n/g;

# Write formatted output to the file
open(F, ">$id");
select(F);

$~ = "OUTLINE";
write;
close(F);

select(STDOUT);

# Create PostScript lines from input file
open(T, "<$id");
while ($line = <T>) {
    chop($line);
    push(@ps, "($line) show  nl", "\n") unless $line =~ /=|\.gif| if|function|return|
applet|seed|length|\}|\{/;
}
close(T);

# Create final PostScript file
$ps = "$id.ps";
open(F, ">$ps");
print F @image;
print F $start;
print F @ps;
print F $end;
close(F);

# Generate image using Ghostscript
open(F, "<$ps");
while (<F>) {
    $code .= $_;
}
close(F);
```

```
open(GS, "|/usr/bin/gs -sDEVICE=ppmraw -sOutputFile=$id.ppm -q 2>/dev/null ");
print GS $code;
close(GS);

print "done";
exit(0);

# Convert PPM to GIF and set transparency (commented out)
# /usr/X11R6/bin/pnmcrop < $1.ppm | /usr/X11R6/bin/ppmtogif > $1.gif
# giftrans -t#ffffff $1.gif > /opt/www/htdocs/sero/wa2/files/$1.gif

system("../gs.sh $id");
print "ok";
exit;

# system("giftrans -t#ffffff $id.gif > /opt/www/htdocs/sero/wa2/files/$id.gif");

# RGB color functions
&rgb_r(100..200);
$r = sprintf("%x", $all[0]);
&rgb_g(100..200);
$g = sprintf("%x", $all[1]);
&rgb_b(60..120);
$b = sprintf("%x", $all[2]);

# Log data
open(F, ">/opt/www/htdocs/sero/wa2/files/data.db");
$string .= $time."::".$name."::".$ENV{'REMOTE_ADDR'}."\n";
print F "$string";
close(F);

# Generate HTML output
open(H, ">/opt/www/htdocs/sero/wa2/files/$id.html");
print H "
    <TITLE> $name </TITLE>
    <BODY BGCOLOR=#$r$g$b>
    <P>
    <BR>
    <UL>
    <IMG SRC=$id.gif>
";
close(H);

format OUTLINE =
~~ ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$string
.

# Main form output
print $form;
exit;

# Subroutines for gathering and processing data
sub gather {
    $line = $_[0];
    $ok = $_[0];
    if ($line =~ /<a href/) {
        push(@href, $ok);
    }
    if ($line =~ /img|IMG|Img/) {
        if ($line =~ /"/) {
            ($foo1, $gif, $foo2) = split(/"/, $line);
        } else {
            ($foo1, $foo2) = split(/=/, $line);
            ($a, $b, $c) = split(/ /, $foo1);
            if ($a = "") {
```

```
                $gif = $b;
            } else {
                $gif = $a;
            }
        }
        # push(@img, $gif, "\n");
    }
    return "";
}

sub image_pos {
    @all = @_;
    srand;
    local(@temp);
    push(@temp, splice(@all, rand(@all), 1)) while @all;
    @all = @temp;
    return @all;
}

sub rgb_r {
    @all = @_;
    srand;
    local(@temp);
    push(@temp, splice(@all, rand(@all), 1)) while @all;
    @all = @temp;
    return @all;
}

sub rgb_g {
    @all = @_;
    srand;
    local(@temp);
    push(@temp, splice(@all, rand(@all), 1)) while @all;
    @all = @temp;
    return @all;
}

sub rgb_b {
    @all = @_;
    srand;
    local(@temp);
    push(@temp, splice(@all, rand(@all), 1)) while @all;
    @all = @temp;
    return @all;
}
```

```
% Name der Schriftdatei
$font="KHRand.gsf";

% Anfang des PostScript-Skripts
$begin="
%!
% Allgemeine Definitionen für die KH-Schriften.
/top 2 31 exp 1 sub def   % Berechnet den oberen Wert für die Zufallszahlengenerierung
usertime srand            % Initialisiert den Zufallszahlengenerator mit der aktuellen
Zeit

/MakeKHFont where
 { pop 80 string
    { currentfile 1 index readline pop
       (%END) eq { exit } if
    } loop pop
 } if

userdict begin
/KHBuildDict 10 dict def  % Erstellt ein neues Wörterbuch für die Schriftartdefinition

/BuildKHChar {
        exch begin
                PenWidth setlinewidth               % Setzt die Linienstärke
                Encoding exch get                    % Holt das aktuelle Zeichen aus der
Kodierung
                dup CharData exch known not { pop /.notdef } if
                CharData exch get                   % Holt die Zeichenpfad-Daten
        end
        /cd exch def

        /llx 99 def /lly 99 def /urx -99 def /ury -99 def  % Initialisiert die
Begrenzungsrahmen

        true

        cd 2 get {                              % Verarbeitet die Pfaddaten
                exch {
                        dup 32 eq {                     % Prüft auf Leerzeichen
                                pop true
                        } {
                                82 sub cd 0 get sub           % Berechnet den X-Wert des
Pfads

                                dup llx lt { /llx exch def } if
                                dup urx gt { /urx exch def } if
                                pop false
                        } ifelse
                } {
                        neg 91 add                       % Berechnet den Y-Wert des Pfads

                        dup lly lt { /lly exch def } if
                        dup ury gt { /ury exch def } if
                        pop true
                } ifelse
        } forall pop

        cd 1 get cd 0 get sub 0 llx 1 sub lly 1 sub urx 1 add ury rand top div 24 mul add
        setcachedevice                          % Setzt die Cache-Geräteeinstellungen für
das Zeichen

        /domove true def
        true

        cd 2 get {                              % Zeichnet die Pfade des Zeichens
                exch {
```

```
                        dup 32 eq {
                                pop true
                                /domove true def
                        } {
                                82 sub cd 0 get sub
                                false
                        } ifelse
                } {
                        neg 91 add
                        domove {
                                moveto                          % Bewegt den Stift
                                /domove false def
                        } {
                                lineto                          % Zeichnet eine Linie
                        } ifelse
                        true
                } ifelse
        } forall pop

        1 setmiterlimit 1 setlinejoin 1 setlinecap  % Setzt Linienparameter
        stroke
} bind def

/MakeKHFont {
        10 dict begin
        /ItalicAngle exch def                   % Setzt den Schrägwinkel
        /PenWidth exch def                      % Setzt die Linienstärke
        /UniqueID exch def                      % Setzt die eindeutige ID der Schrift
        /FontName exch def                      % Setzt den Namen der Schrift
        /CharData exch def                      % Setzt die Zeichendaten

        /FontType 3 def                         % Setzt den Schrifttyp
        /FontMatrix
                [1 33 div
                0
                ItalicAngle neg dup sin exch cos div 33 div
                1 33 div
                0 0] def                                % Definiert die Schriftmatrix
        /Encoding StandardEncoding def          % Setzt die Standard-Kodierung
        /BuildChar {KHBuildDict begin BuildKHChar end} def  % Definiert die BuildChar-
Prozedur
        /FontBBox {-10 -10 30 30} def           % Setzt die Begrenzungsrahmen der Schrift

        FontName currentdict end definefont pop
} bind def
end             % userdict
%END

150 dict dup begin
";

% Ende des PostScript-Skripts
$end="
end
/KHRand 4295200 0.8 0 MakeKHFont
";

% Definiert die Zeichendaten
$chars="
/hyphen [-6 6 (NRVR)] def
/bar [-4 4 (RBRb)] def
/periodcentered [-5 5 (RQQRRSSRRQ)] def
/numbersign [-10 11 (SBLb YBRb LOZO KUYU)] def
/ampersand [-13 13 (\\O\\N[MZMYNXPVUTXRZP[L[JZIYHWHUISJRQNRMSKSIRGPFNGMIMKNNPQUXWZY[[[\\Z\
\Y)] def
/currency [-11 11 (IIJKKOKUJYI[ [IZKYOYUZY[[ IIKJOKUKYJ[I I[KZOYUYYZ[[)] def
```

```
/bullet [-2 2 (QPPQPSQTSTTSTQSPQP RQQRRSSRRQ)] def
/slash [-7 7 (K^YF)] def
/backslash [-7 7 (KFY^)] def
/tilde [-8 8 (LTLRMPOPUSWSXR LRMQOQUTWTXRXP)] def
/bracketleft [-7 7 (OBOb PBPb OBVB ObVb)] def
/bracketright [-7 7 (TBTb UBUb NBUB NbUb)] def
/braceleft [-7 7 (TBRCQDPFPHQJRKSMSOQQ RCQEQGRISJTLTNSPORSTTVTXSZR[Q]Q_Ra QSSUSWRYQZP\
\P^Q`RaTb)] def
/braceright [-7 7 (PBRCSDTFTHSJRKQMQOSQ RCSESGRIQJPLPNQPURQTPVPXQZR[S]S_Ra SSQUQWRYSZT\
\T^S`RaPb)] def
/less [-12 12 (ZIJRZ[)] def
/greater [-12 12 (JIZRJ[)] def
/asciitilde [-12 12 (IUISJPLONOPPTSVTXTZS[Q ISJQLPNPPQTTVUXUZT[Q[O)] def
/asciicircum [-11 11 (JTROZT JTRPZT)] def
/percent [-12 12 ([FI[ NFPHPJOLMMKMIKIIJGLFNFPGSHVHYG[F WTUUTWTYV[X[ZZ[X[VYTWT)] def
/at [-13 14 (WNVLTKQKOLNMMPMSNUPVSVUUVS QKOMNPNSOUPV WKVSVUXVZV\\T]Q]O\
\L[JYHWGTFQFNGLHJJILHOHRIUJWLYNZQ[T[WZYYZX XKWSWUXV)] def
/section [-8 8 (UITJUKVJVIUGSFQFOGNINKOMQOVR OMTPVRWTWVVXTZ
PNNPMRMTNVPXU[ NVSYU[V]V_UaSbQbOaN_N^O]P^O_)] def
/dagger [-8 8 (RFQHRJSHRF RFRb RQQTRbSTRQ LMNNPMNLLM LMXM TMVNXMVLTM)] def
/daggerdbl [-8 8 (RFQHRJSHRF RFRT RPQRSVRXQVSRRP RTRb R^Q`RbS`R^ LMNNPMNLLM LMXM
TMVNXMVLTM L[N\\P[NZL[ L[X[ T[V\\X[VZT[)] def
/A [-13 10 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ)] def
/B [-12 12 (UGTHSJQOOUNWLZJ[ THSKQSPVOXMZJ[H[GZGXHWIXHY
OLNNMOKOJNJLKJMHOGRFXFZG[I[KZMXNTORO XFYGZIZKYMXN TOWPXQYSYVXYWZU[S[RZRXSU
TOVPWQXSXVWYU[)] def
/C [-10 11 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/D [-12 11 (UGTHSJQOOUNWLZJ[ THSKQSPVOXMZJ[H[GZGXHWJWLXNZP[S[UZWXYTZOZLYIWGTFQE
THSKPUPXQZS[X[YZWWYSYKXIWHVHUJTJZKZNZP)] def
/E [-12 11 (OJJJ[ TJIZ TJNKOLRL[N] RL[S] TLZL ]L^ LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN
XOZQ[S[UZWWX)] def
/F [-11 10 (OJJJ[ TJIZ TJNKOLRL[N] RL[S] TLZL ]L^ LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN
XOZQ[S[UZWWX)] def
/G [-11 12 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[ SXYX SYYX)] def
/H [-12 13 (OJZJ[ TJ\\J TMVM ]MMM ]MZJ M[J LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN XOZQ[S[UZWWX)]
def
/I [-7 8 (OJIJ[ TI\\I TJMJ ]MLL]M\\J LMMNNOQOSNTMTKRIQGOFNEOJQMTPTXU[W[YZZYZT
RRISHPGNFOHOIQKSMTMPUYWZ)] def
/J [-10 10 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[ SXYX SYYX)] def
/K [-11 10 (KJVJQQLTLVKXIWJVJVIZIXJX JIJ[LXMXIXLM QLQMQLTMVMYK[J\\J\\JYM KJYLXMXIZIXLMX)]
def
/L [-9 9 (OHJHJ[ THIZ THNKOLRL[N] RL[S] TLZL ]L^ LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN
XOZQ[S[UZWWX)] def
/M [-13 12 (OJZJ[ TJ\\J TMVM ]MMM ]MZJ M[J LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN XOZQ[S[UZWWX)]
def
/N [-12 11 (OJZJ[ TJ\\J TMVM ]MMM ]MZJ M[J LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN XOZQ[S[UZWWX)]
def
/O [-11 12 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/P [-11 11 (UGTHSJQOOUNWLZJ[ THSKQSPVOXMZJ[H[GZGXHWIXHY
OLNNMOKOJNJLKJMHOGRFXFZG[I[KZMXNTORO XFYGZIZKYMXN TOWPXQYSYVXYWZU[S[RZRXSU
TOVPWQXSXVWYU[)] def
/Q [-12 13 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[ SXYX SYYX)] def
/R [-11 12 (UGTHSJQOOUNWLZJ[ THSKQSPVOXMZJ[H[GZGXHWJWLXNZP[S[UZWXYTZOZLYIWGTFQE
THSKPUPXQZS[X[YZWWYSYKXIWHVHUJTJZKZNZP)] def
/S [-10 10 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/T [-12 12 (KJJJ[ TJIZ TJNKOLRL[N] RL[S] TLZL ]L^ LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN
XOZQ[S[UZWWX)] def
/U [-12 12 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/V [-12 12 (KHJJJLKNNOQOUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
```

```
WFUGRJPMNQMUMXNZP[)] def
/W [-12 13 (KHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/X [-13 13 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ)] def
/Y [-12 12 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ)] def
/Z [-11 12 (OJZJ[ TJ\\J TMVM ]MMM ]MZJ M[J LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN XOZQ[S[UZWWX)]
def
/0 [-12 11 (OIHHJJKOLNNPOTNXLWIWJUJVKWLXPZQ[S[UZWXXWYVXT WRJXJWKXMXOZQ[)] def
/1 [-10 11 (J[JZHZIXL] RJHJZHZIZJ] J[JZHZIZJ^ RJHJ[HZIZI[J^ RHL\\H)] def
/2 [-11 11 (GHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/3 [-10 10 (KHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/4 [-11 10 (OHJHJ[ THIZ THNKOLRL[N] RL[S] TLZL ]L^ LMNNMOKOJNJLKJMHOGRFXFZG[I[KZMYN
XOZQ[S[UZWWX)] def
/5 [-10 11 (GHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/6 [-10 11 (GHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/7 [-10 10 (JHL\\H RJHJ[HZIZI[J^ RHL\\H RJHJ[HZIZI[J^)] def
/8 [-10 10 (KHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/9 [-10 10 (KHJJJLKNNOQQUNWMYKZIZGYFWFTGQJOMMQLULXMZP[R[UZWXXVXTWRURSSRU
WFUGRJPMNQMUMXNZP[)] def
/Agrave [-12 10 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ
^U` T`Y^W)] def
/Acircumflex [-12 10 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY
XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ ^U^\\[\\`^W)] def
/Atilde [-12 10 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[ T[TYSVRTPRNQLQKRKTLWOZR[V[XZ
^U_ `^Y_ ^V^\\)] def
/Aogonek [-13 10 (XFVHTKQPOSLWIZG[E[DZDXEWFXEY XFWJUTT[ XFU[
```