Datenverarbeitung mit AWK und XSL

AWK und XSL

Christoph Stadelmann, M.Sc.

Erlug-Erlangen

4. Mai 2024

Inhaltsverzeichnis

Warum AWK und XSLT?

- Suche in Daten.
- Datenaufbereitung.
- AWK: Interpretation von ASCII-Dateien erfolgt zeilenweise.
- XSLT: Interpretation von XML-Dateien.

Hier:

- Erstellung von Daten zum Laserschneiden.

Auflisten von Dateien im Terminal.

Eingabe im Terminal

```
ls -1
```

Regel in einer Makefile

```
.PHONY: filelist_1.txt
filelist_1.txt:
	( ls -1 )
	@echo ""
	@echo ""
	( ls -1 )   >   $@
	echo "done"
```

Ausgabe

```
Ada-Programm.odg
Ada-Programm.pdf
awk
AWK-Rotation.odg
AWK-Rotation.pdf
AWK-Skallierung.odg
AWK-Skallierung.pdf
AWK-Translation.odg
AWK-Translation.pdf
filelist_1.txt
makefile
```

Filtern von Dateiliste.

Eingabe im Terminal

```
ls -1 | awk '/^AWK/{ print ( $$0 ); };'
```

Regel in einer Makefile

```
# Darstellung von Kanalisierung und Filterung von Datenströmen       (AWK)
.PHONY: filelist_2.txt
filelist_2.txt:
        ( ls -1 | awk '/^AWK/{ print ( $$0 ); };' )
        @echo ""
        @echo ""
        ( ls -1 | awk '/^AWK/{ print ( $$0 ); };' )  >  $@
```

Ausgabe

```
AWK-Rotation.odg
AWK-Rotation.pdf
AWK-Skallierung.odg
AWK-Skallierung.pdf
AWK-Translation.odg
AWK-Translation.pdf
```

Ausgabe von Dateiliste als HTML-Datei.

Eingabe im Terminal

```
ls -1 | awk '/^AWK/{ print ( $$0 ); };' | awk -f awk/AWK__print_as_html.awk
```

Regel in einer Makefile

```
.PHONY: filelist_3.html
filelist_3.html:
	( ls -1 | awk '/^AWK/{ print ( $$0 ); };' | awk -f awk/AWK__print_as_html.awk  )
	@echo ""
	@echo ""
	( ls -1 | awk '/^AWK/{ print ( $$0 ); };' | awk -f awk/AWK__print_as_html.awk)   >   $@
```

AWK-Skript (AWK__print_as_html.awk)

```
BEGIN{
   print( "<html>" );
   print( "    <body>" );
};

{
   printf( "        <p>" );
   printf( $0 );
   print( "</p>");
};

END{
   print( "    <body>" );
   print( "<html>" );
};
```

Ausgabe

```
<html>
   <body>
      <p>AWK-Rotation.odg</p>
      <p>AWK-Rotation.pdf</p>
      <p>AWK-Skallierung.odg</p>
      <p>AWK-Skallierung.pdf</p>
      <p>AWK-Translation.odg</p>
      <p>AWK-Translation.pdf</p>
   <body>
<html>
```

Ausgabe von Dateiliste als HTML-Datei.

Eingabe im Terminal

```
( echo "<html><body>"       ; \
    echo "    <p><b><u>*.pdf:</u></b></p>"       ; \
  ( ls -1 *.pdf | awk -f awk/AWK__print_as_html_body.awk  ) ; \
    echo "    <p><b><u>*.odg:</u></b></p>"       ; \
  ( ls -1 *.odg | awk -f awk/AWK__print_as_html_body.awk  ) ; \
    echo "    <p><b><u>*.html:</u></b></p>"      ; \
  ( ls -1 *.html | awk -f awk/AWK__print_as_html_body.awk  ) ; \
    echo "</html></body>")
```

Regel in einer Makefile

```
.PHONY: filelist_4.html
filelist_4.html:
        ( echo "<html><body>"       ; \
          echo "    <p><b><u>*.pdf:</u></b></p>"       ; \
        ( ls -1 *.pdf | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "    <p><b><u>*.odg:</u></b></p>"       ; \
        ( ls -1 *.odg | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "    <p><b><u>*.html:</u></b></p>"      ; \
        ( ls -1 *.html | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "</html></body>")
        @echo ""
        @echo ""
        ( echo "<html><body>"       ; \
          echo "    <p><b><u>*.pdf:</u></b></p>"       ; \
        ( ls -1 *.pdf | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "    <p><b><u>*.odg:</u></b></p>"       ; \
        ( ls -1 *.odg | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "    <p><b><u>*.html:</u></b></p>"      ; \
        ( ls -1 *.html | awk -f awk/AWK__print_as_html_body.awk  ) ; \
          echo "</html></body>")   >   $@
```

AWK-Skript (AWK__print_as_html_body.awk)

```
# /usr/bin/awk

{
   printf( "       <p>" );
   printf( $0 );
   print( "</p>");
};
```

Ausgabe

```
<html><body>
   <p><b><u>*.pdf:</u></b></p>
      <p>Ada-Programm.pdf</p>
      <p>AWK-Rotation.pdf</p>
      <p>AWK-Skallierung.pdf</p>
      <p>AWK-Translation.pdf</p>
   <p><b><u>*.odg:</u></b></p>
      <p>Ada-Programm.odg</p>
      <p>AWK-Rotation.odg</p>
      <p>AWK-Skallierung.odg</p>
      <p>AWK-Translation.odg</p>
   <p><b><u>*.html:</u></b></p>
      <p>filelist_3.html</p>
      <p>filelist_4.html</p>
</html></body>
```

Auflisten von Dateien als XML-Datensatz.

Eingabe im Terminal

```
( echo "<xml>" ;  ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ;  echo "</xml>")
```

Regel in einer Makefile

```
.PHONY: filelist_0.xml
filelist_0.xml:
	( echo "<xml>"       ; \
	( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ; \
	  echo "</xml>")
	@echo ""
	@echo ""
	( echo "<xml>"       ; \
	( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ; \
	  echo "</xml>")   >   $@
```

AWK-Skript (AWK__print_as_xml_body.awk)

```
# /usr/bin/awk

{
   printf( "<file name=\"" );
   printf( $0 );
   print( "\"/>");
};
```

Ausgabe

```
<xml>
<file name="Ada-Programm.odg"/>
<file name="Ada-Programm.pdf"/>
<file name="awk"/>
<file name="AWK-Rotation.odg"/>
<file name="AWK-Rotation.pdf"/>
<file name="AWK-Skalierung.odg"/>
<file name="AWK-Skalierung.pdf"/>
<file name="AWK-Translation.odg"/>
<file name="AWK-Translation.pdf"/>
<file name="filelist_0.xml"/>
<file name="makefile"/>
</xml>
```

Zusammenführen von XML-Datensätzen.

Eingabe im Terminal

```
echo "<xml>" ; cat descriptions.xml ;
( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ;  echo "</xml>"
```

descriptions.xml

```
<descritions>
   <descrition extention=".odg"      caption="OpenDokument Zeichnung"/>
   <descrition extention=".pdf"      caption="PDF-Dokument"/>
   <descrition extention=".xml"      caption="XML-Dokument"/>
   <descrition extention=".html"     caption="HTML-Dokument"/>
</descritions>
```

Regel in einer Makefile

```
.PHONY: filelist_1.xml
filelist_1.xml:
        ( echo "<xml>"      ; \
        cat descriptions.xml ; \
        ( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ; \
          echo "</xml>")
        @echo ""
        @echo ""
        ( echo "<xml>"      ; \
        cat descriptions.xml ; \
        ( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ; \
          echo "</xml>")   >   $@
```

Ausgabe

```
<xml>
<descritions>
   <descrition extention="odg"      caption="OpenDokument Zeichnung"/>
   <descrition extention="pdf"      caption="PDF-Dokument"/>
</descritions>
<file name="Ada-Programm.odg"/>
<file name="Ada-Programm.pdf"/>
<file name="awk"/>
<file name="AWK-Rotation.odg"/>
<file name="AWK-Rotation.pdf"/>
<file name="AWK-Skalierung.odg"/>
<file name="AWK-Skalierung.pdf"/>
<file name="AWK-Translation.odg"/>
<file name="AWK-Translation.pdf"/>
<file name="descriptions.xml"/>
<file name="filelist_0.xml"/>
<file name="filelist_1.xml"/>
<file name="makefile"/>
</xml>
```

XSLT (Verarbeiten von XML-Datensätzen).

XSLT-Skript

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text"/>

[...]

<xsl:template name="TP_add_description">
   <xsl:param name="PARAM_filename"/>
   <xsl:call-template name="TP_print_space">
      <xsl:with-param name="PARAM_string"          select="$PARAM_filename"/>
      <xsl:with-param name="PARAM_length"          select="50"/>
   </xsl:call-template>
   <xsl:for-each select="/xml/descritions/descrition">
      <xsl:if test="contains($PARAM_filename, @extention)">
      <xsl:text>[</xsl:text>
      <xsl:value-of select="@caption"/>
      <xsl:text>]</xsl:text>
      </xsl:if>
   </xsl:for-each>
</xsl:template>

<xsl:template name="TP_interpret_file">
   <xsl:text>&#10;</xsl:text>
   <xsl:value-of select="@name"/>
   <xsl:call-template name="TP_add_description">
      <xsl:with-param name="PARAM_filename"          select="@name"/>
   </xsl:call-template>
</xsl:template>

<xsl:template match="/xml">
   <xsl:for-each select="file">
      <xsl:call-template name="TP_interpret_file"/>
   </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Eingabe im Terminal

```
( echo "<xml>" ; cat descriptions.xml ; \
( ls -1 | awk -f awk/AWK__print_as_xml_body.awk  ) ; \
echo "</xml>") | xsltproc xslt/XSL__xml2txt.xsl
```

descriptions.xml

```
<descritions>
   <descrition extention=".odg"       caption="OpenDokument Zeichnung"/>
   <descrition extention=".pdf"       caption="PDF-Dokument"/>
   <descrition extention=".xml"       caption="XML-Dokument"/>
   <descrition extention=".html"      caption="HTML-Dokument"/>
</descritions>
```

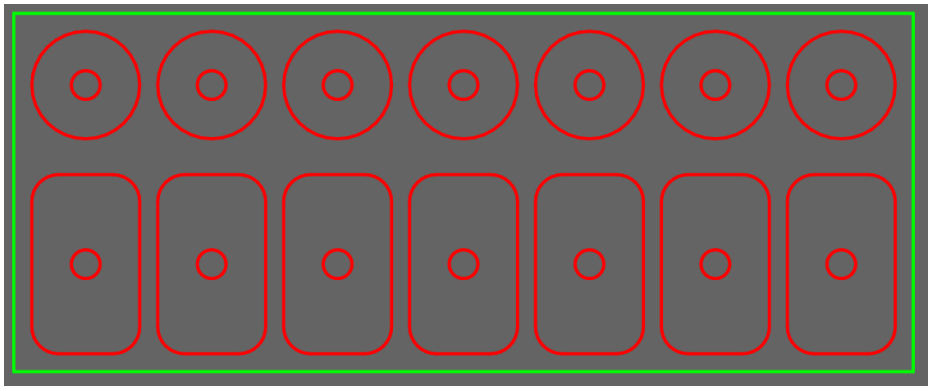
Ausgabe

```
Ada-Programm.odg                [OpenDokument Zeichnung]
Ada-Programm.pdf                [PDF-Dokument]
awk
AWK-Rotation.odg                [OpenDokument Zeichnung]
AWK-Rotation.pdf                [PDF-Dokument]
AWK-Skalierung.odg              [OpenDokument Zeichnung]
AWK-Skalierung.pdf              [PDF-Dokument]
AWK-Translation.odg             [OpenDokument Zeichnung]
AWK-Translation.pdf             [PDF-Dokument]
descriptions.xml                [XML-Dokument]
filelist_0.xml                  [XML-Dokument]
filelist_1.xml                  [XML-Dokument]
filelist_2.html                 [HTML-Dokument]
filelist_2.txt
makefile
xslt
```

Erstellung von Laser-Schnittdaten; Ada-Programnm (Distanzhülsen).

ADA-Programm

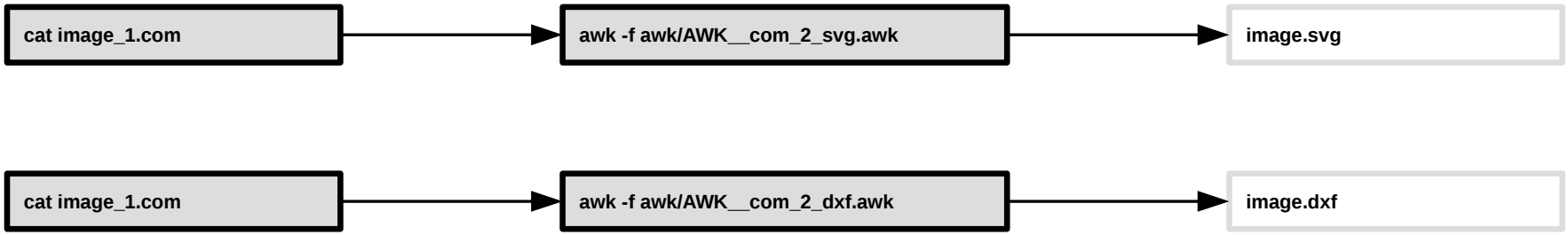
```
procedure main is
   procedure draw_huelse is
   begin
      SVG_File.Put_Circle_Cut( 0.0,    0.0,      1.6 );
      SVG_File.Put_Circle_Cut( 0.0,    0.0,      6.0 );
   end;

   procedure draw_Plate is
   begin
      SVG_File.Put_Circle_Cut( 0.0,    0.0,      1.6 );
      SVG_File.Put_Roundbox_Cut( 0.0,     0.0,    12.0,    20.0,    3.0 );
   end;

begin
   SVG_File.Open( "image.com" );
   SVG_File.Put_Box_Mark( 0.0, 0.0,   100.0,   40.0 );

   SVG_origin.Set_Origin_Level_0(      0.0,     12.0 );
   for VAR_x in -3..3 loop
      SVG_origin.Set_Origin_Level_1(     (14.0 * Float(VAR_x)),     0.0);
      draw_huelse;
   end loop;

   SVG_origin.Set_Origin_Level_0(      0.0,     -8.0 );
   for VAR_x in -3..3 loop
      SVG_origin.Set_Origin_Level_1(     (14.0 * Float(VAR_x)),     0.0);
      draw_Plate;
   end loop;
   SVG_File.Close;
end main;
```



Programmausgabe

```
#INIT
#LINE_MARK -5.00000E+01 -2.00000E+01 1.00000E+02 0.00000E+00
#LINE_MARK -5.00000E+01 2.00000E+01 1.00000E+02 0.00000E+00
#LINE_MARK -5.00000E+01 -2.00000E+01 0.00000E+00 4.00000E+01
#LINE_MARK 5.00000E+01 -2.00000E+01 0.00000E+00 4.00000E+01
#LINE_CUT -4.20000E+01 1.36000E+01 3.12145E-01 -3.07436E-02
#LINE_CUT -4.16879E+01 1.35693E+01 3.00149E-01 -9.10492E-02
#LINE_CUT -4.13877E+01 1.34782E+01 2.76619E-01 -1.47856E-01
#LINE_CUT -4.11111E+01 1.33304E+01 2.42459E-01 -1.98980E-01
#LINE_CUT -4.08686E+01 1.31314E+01 1.98981E-01 -2.42459E-01
#LINE_CUT -4.06696E+01 1.28889E+01 1.47856E-01 -2.76619E-01

[...]

#LINE_CUT 3.73333E+01 1.49441E+00 5.18661E-01 2.77230E-01
#LINE_CUT 3.78520E+01 1.77164E+00 5.62780E-01 1.70717E-01
#LINE_CUT 3.84147E+01 1.94236E+00 5.85271E-01 5.76439E-02
#EXIT
```

Erstellen von SVG- und DXF-Daten.



Auszug aus einer COM-Datei

```
#INIT
#LINE_MARK -5.00000E+01 -2.00000E+01 1.00000E+02 0.00000E+00
#LINE_MARK -5.00000E+01 2.00000E+01 1.00000E+02 0.00000E+00
#LINE_MARK -5.00000E+01 -2.00000E+01 0.00000E+00 4.00000E+01
#LINE_MARK 5.00000E+01 -2.00000E+01 0.00000E+00 4.00000E+01
#LINE_CUT -4.20000E+01 1.36000E+01 3.12145E-01 -3.07436E-02
#LINE_CUT -4.16879E+01 1.35693E+01 3.00149E-01 -9.10492E-02
#LINE_CUT -4.13877E+01 1.34782E+01 2.76619E-01 -1.47856E-01
#LINE_CUT -4.11111E+01 1.33304E+01 2.42459E-01 -1.98980E-01
#LINE_CUT -4.08686E+01 1.31314E+01 1.98981E-01 -2.42459E-01
#LINE_CUT -4.06696E+01 1.28889E+01 1.47856E-01 -2.76619E-01

[...]

#LINE_CUT 3.73333E+01 1.49441E+00 5.18661E-01 2.77230E-01
#LINE_CUT 3.78520E+01 1.77164E+00 5.62780E-01 1.70717E-01
#LINE_CUT 3.84147E+01 1.94236E+00 5.85271E-01 5.76439E-02
#EXIT
```

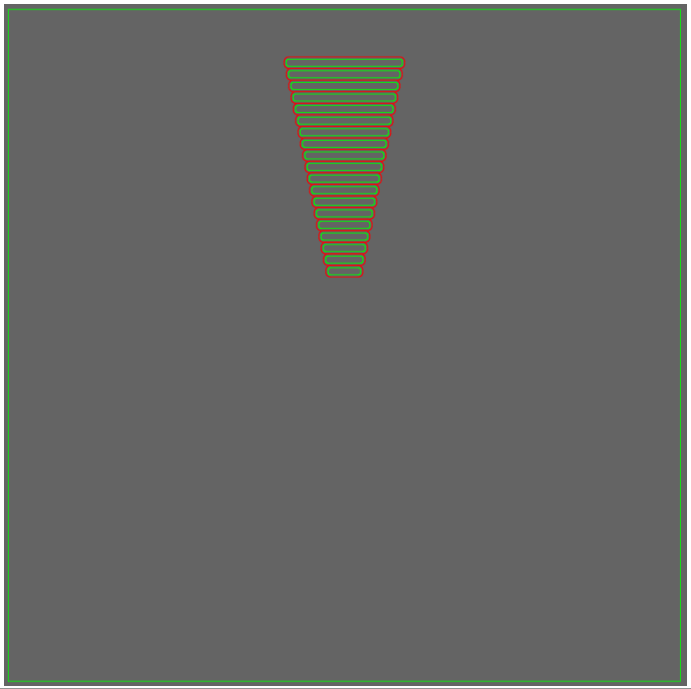
Regel in einer Makefile

```
image.svg: image_1.com
        ( cat image_1.com | awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"

image.dxf: image_1.com
        ( cat image_1.com | awk -f awk/AWK__com_2_dxf.awk )   >   $@
        echo "do done"
```

Ada-Programnm (Tortenstück).

Ada-Quelltext

```
procedure main is
   procedure draw_Item( PRAM_Width: in Float ) is
   begin
      SVG_File.Put_Roundbox_Cut( 0.0,     0.0,     PRAM_Width,          5.0,    2.0 );
      SVG_File.Put_Roundbox_Mark( 0.0,     0.0,    PRAM_Width - 2.0,    3.0,    1.0 );
   end;
begin
   SVG_File.Open( "image.com" );

   SVG_origin.Set_Origin_Level_0(      0.0,     -8.0 );
   for VAR_y in 8..26 loop
      SVG_origin.Set_Origin_Level_1(    0.0,  (5.0 * Float(VAR_y)));
      draw_Item(  2.0 * Float(VAR_y)   );
   end loop;
   SVG_File.Close;
end main;
```



Regel in einer Makefile

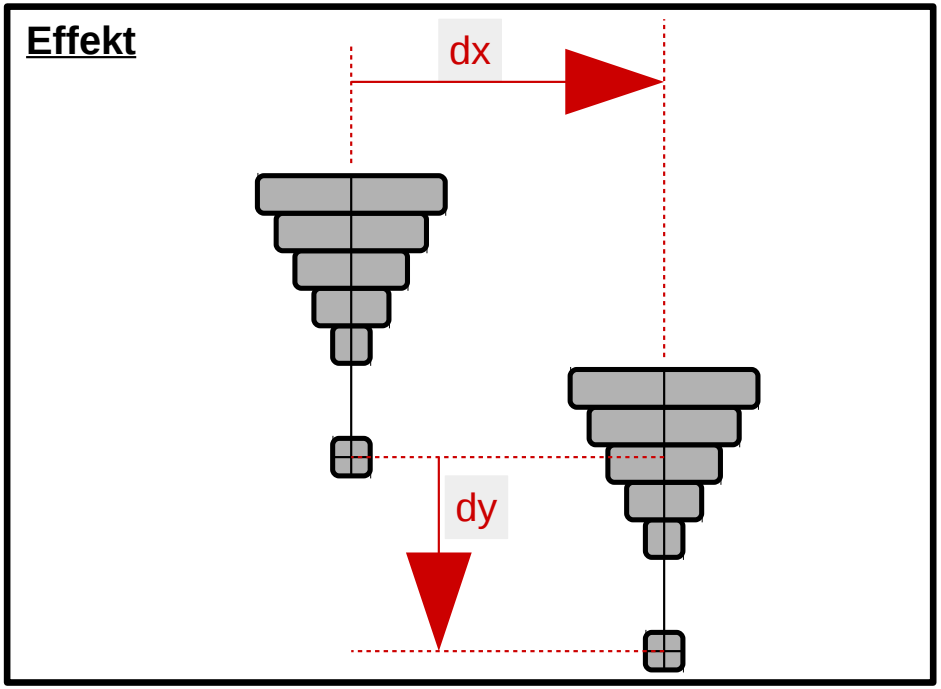
```
$(OUTPUT_NAME).svg: image.com frame.com
        ( ( cat frame.com ; cat image.com ) | awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"
```

AWK (Translation).

AWK-Quelltext

```
# /usr/bin/awk

BEGIN{
    X = PARAM_X;
    Y = PARAM_Y;
};

/^#LINE_MARK/{
    # X, Y,      dx, dy
    printf( "#LINE_MARK  " );
    printf( $2 + X );
    printf( " " );
    printf( $3 + Y );
    printf( " " );
    printf( $4 );
    printf( " " );
    print ( $5 );
};

/^#LINE_CUT/{
    # X, Y,      dx, dy
    printf( "#LINE_CUT  " );
    printf( $2 + X );
    printf( " " );
    printf( $3 + Y );
    printf( " " );
    printf( $4 );
    printf( " " );
    print ( $5 );
};

END{  };
```

Effekt



Regel in einer Makefile

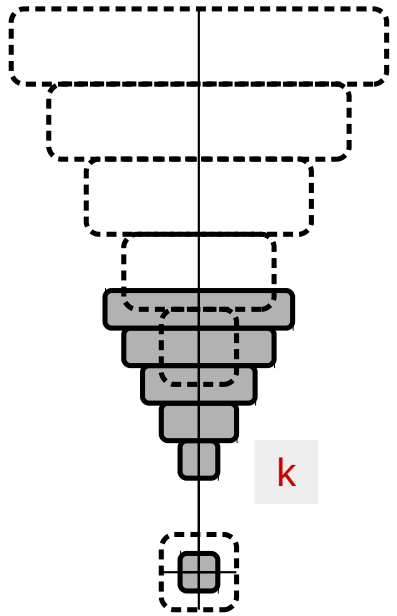
```
image_translate.svg: image.com frame.com
        ( ( cat frame.com ; ( cat image.com |  \
        awk -v PARAM_X=50 -v PARAM_Y=-50 -f awk2/AWK__com__translate_x_y.awk ) )
         | awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"
```

AWK (Skalierung).

AWK-Quelltext

```
# /usr/bin/awk

BEGIN{
    K = PARAM_K;
};

/^#LINE_MARK/{
    # X, Y,      dx, dy
    printf( "#LINE_MARK  " );
    printf( $2 * (1.0 * K) );
    printf( " " );
    printf( $3 * (1.0 * K) );
    printf( " " );
    printf( $4 * (1.0 * K) );
    printf( " " );
    print ( $5 * (1.0 * K) );
};

/^#LINE_CUT/{
    # X, Y,      dx, dy
    printf( "#LINE_CUT  " );
    printf( $2 * (1.0 * K) );
    printf( " " );
    printf( $3 * (1.0 * K) );
    printf( " " );
    printf( $4 * (1.0 * K) );
    printf( " " );
    print ( $5 * (1.0 * K) );
};

END{ };
```

Effekt



Regel in einer Makefile

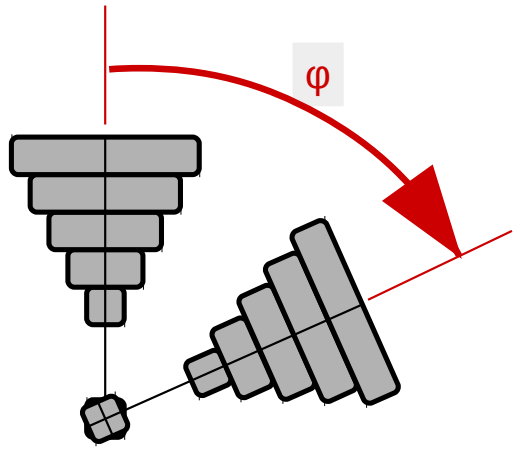
```
image_skale.svg: image.com frame.com
        ( ( cat frame.com ; ( cat image.com | \
        awk -v PARAM_K=0.5  -f awk2/AWK__com__skale_k.awk ) ) \
          | awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"
```

AWK (Rotation).

AWK-Quelltext

```
# /usr/bin/awk

BEGIN{
   SI = sin( -2.0 * 3.159 * (PARAM_P / 360) );
   CO = cos( -2.0 * 3.159 * (PARAM_P / 360) );
};

/^#LINE_MARK/{
   # X, Y,      dx, dy
   printf( "#LINE_MARK  " );
   printf( CO * $2   -   SI * $3 );
   printf( " " );
   printf( SI * $2   +   CO * $3 );
   printf( " " );
   printf( CO * $4   -   SI * $5   );
   printf( " " );
   print ( SI * $4   +   CO * $5   );
};

/^#LINE_CUT/{
   # X, Y,      dx, dy
   printf( "#LINE_CUT  " );
   printf( CO * $2   -   SI * $3 );
   printf( " " );
   printf( SI * $2   +   CO * $3 );
   printf( " " );
   printf( CO * $4   -   SI * $5   );
   printf( " " );
   print ( SI * $4   +   CO * $5   );
};

END{  };
```

Effekt



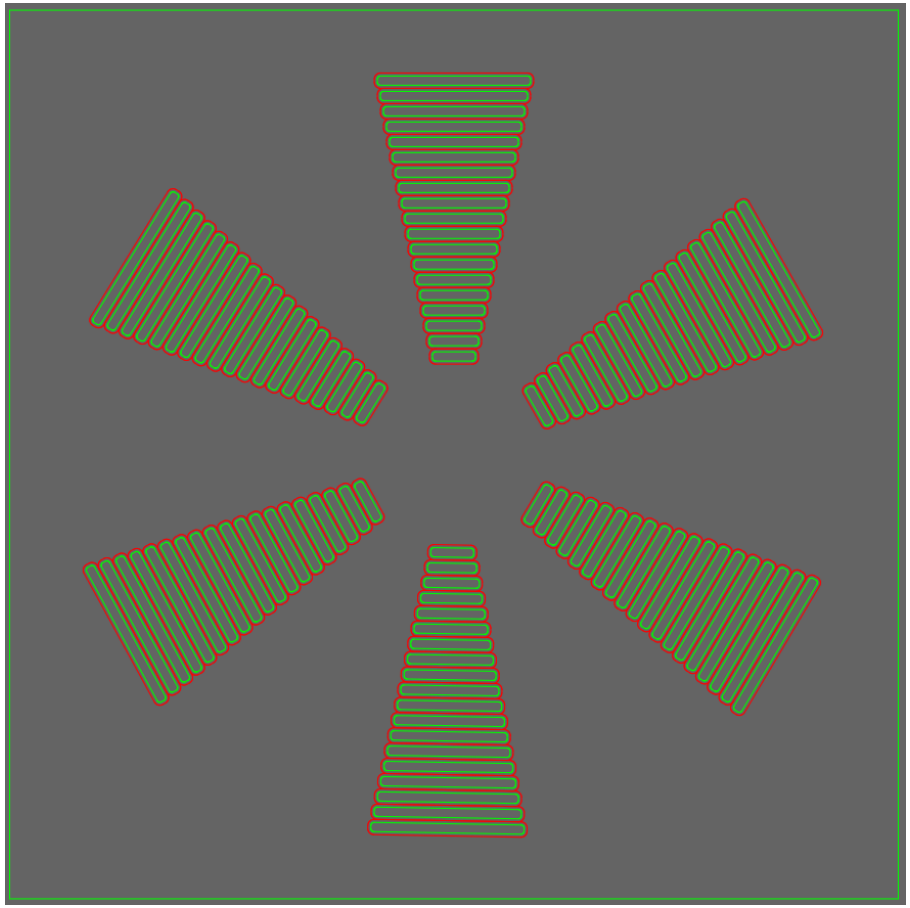
Regel in einer Makefile

```
image_rotate.svg: image.com frame.com
        ( ( cat frame.com ; ( cat image.com | \
        awk -v PARAM_P=90 -f awk2/AWK__com__rotate_p.awk ) )  | \
        awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"
```

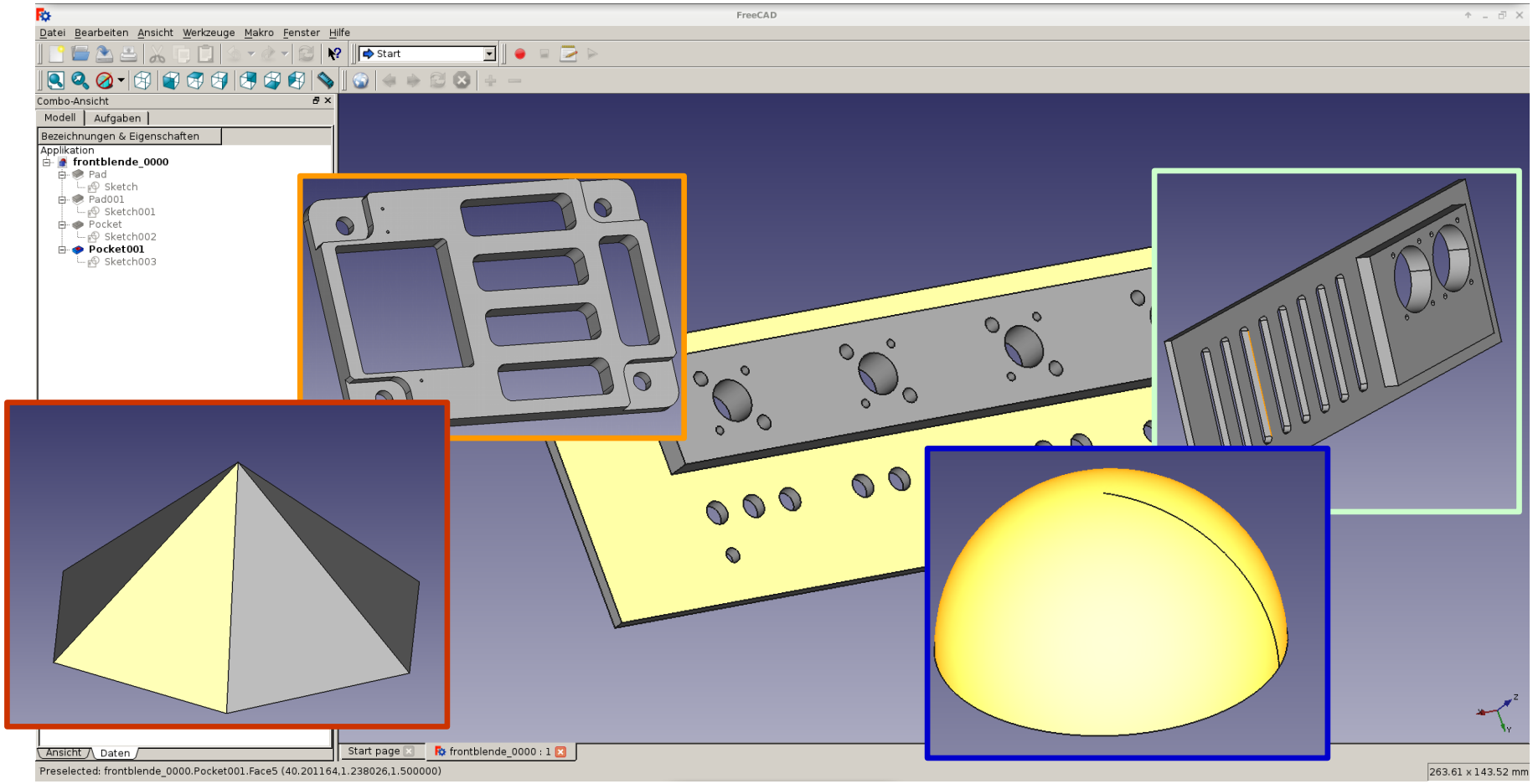
Komplexer Signalfluss.

Make-Quelltext

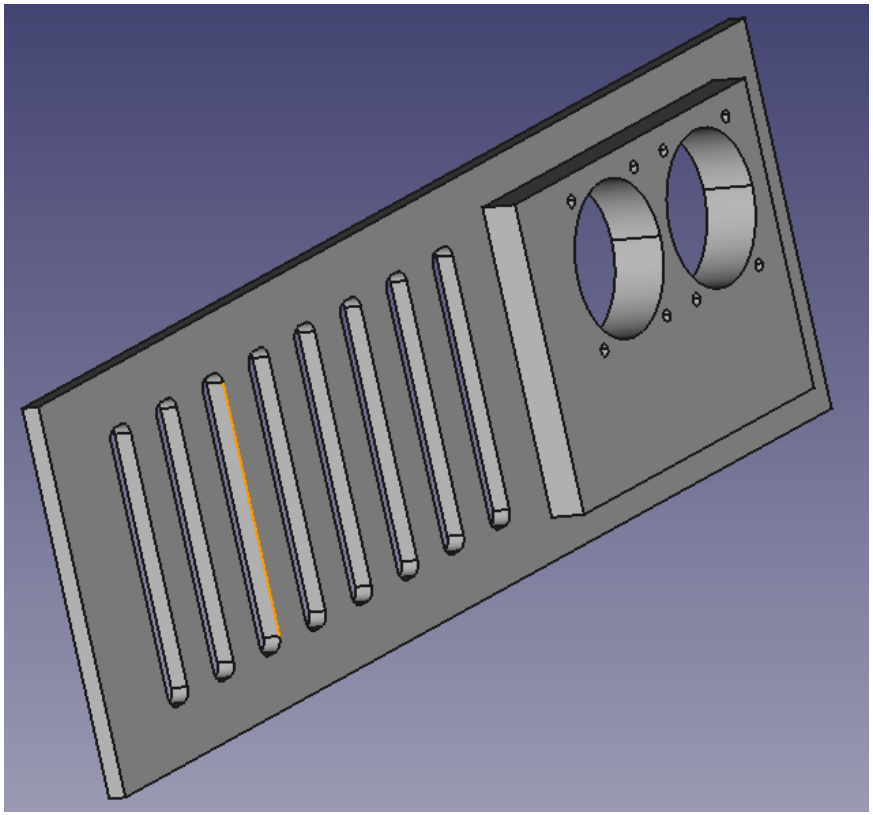
```
image_rotate_mult.svg: image.com frame.com
        ( ( cat frame.com ; \
        ( cat image.com | awk -v PARAM_P=0   -f awk2/AWK__com__rotate_p.awk ) ; \
        ( cat image.com | awk -v PARAM_P=60  -f awk2/AWK__com__rotate_p.awk ) ; \
        ( cat image.com | awk -v PARAM_P=120 -f awk2/AWK__com__rotate_p.awk ) ; \
        ( cat image.com | awk -v PARAM_P=180 -f awk2/AWK__com__rotate_p.awk ) ; \
        ( cat image.com | awk -v PARAM_P=240 -f awk2/AWK__com__rotate_p.awk ) ; \
        ( cat image.com | awk -v PARAM_P=300 -f awk2/AWK__com__rotate_p.awk ) ) \
          | awk -f awk/AWK__com_2_svg.awk )   >   $@
        echo "do done"
```



Konstruktion von 3D-Körpern mit Freecad.



Aufbau von AST-Dateien.



AST-Datei

```
solid Mesh
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex 37.550320 10.120491 -1.500000
      vertex 14.000000 20.500000 -1.500000
      vertex 37.700882 11.231961 -1.500000
    endloop
  endfacet
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex 37.700882 11.231961 -1.500000
      vertex 14.000000 20.500000 -1.500000
      vertex 37.950462 12.325460 -1.500000
    endloop
  endfacet
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex -107.500000 -31.500000 -1.500000
      vertex -89.414215 -21.914213 -1.500000
      vertex -89.000000 -22.232050 -1.500000
    endloop
  endfacet

[...]

  facet normal 0.000000 1.000000 -0.000000
    outer loop
      vertex 95.000000 25.000000 1.500000
      vertex 25.000000 25.000000 7.500000
      vertex 95.000000 25.000000 7.500000
    endloop
  endfacet
endsolid Mesh
```

Verarbeitung von AST-Dateien; Erstellung von Schnitten.

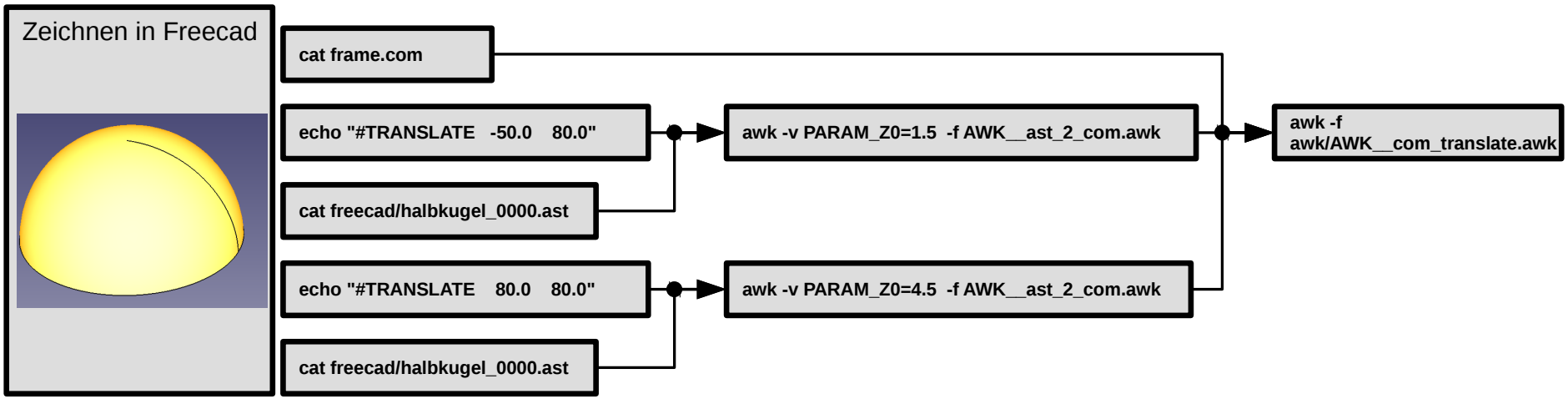
AST-Datei

```
solid Mesh
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex 37.550320 10.120491 -1.500000
      vertex 14.000000 20.500000 -1.500000
      vertex 37.700882 11.231961 -1.500000
    endloop
  endfacet
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex 37.700882 11.231961 -1.500000
      vertex 14.000000 20.500000 -1.500000
      vertex 37.950462 12.325460 -1.500000
    endloop
  endfacet
  facet normal 0.000000 0.000000 -1.000000
    outer loop
      vertex -107.500000 -31.500000 -1.500000
      vertex -89.414215 -21.914213 -1.500000
      vertex -89.000000 -22.232050 -1.500000
    endloop
  endfacet

[...]

  facet normal 0.000000 1.000000 -0.000000
    outer loop
      vertex 95.000000 25.000000 1.500000
      vertex 25.000000 25.000000 7.500000
      vertex 95.000000 25.000000 7.500000
    endloop
  endfacet
endsolid Mesh
```

AWK-Quelltext

```
# /usr/bin/awk

[ ... ]

BEGIN{
   Z = PARAM_Z0;
   FUNC_reset();
};

/vertex[ ]+/{
   if( vertex_Count == 0 ){
      X1 = $2;
      Y1 = $3;
      Z1 = $4;
   };
   if( vertex_Count == 1 ){
      X2 = $2;
      Y2 = $3;
      Z2 = $4;
   };
   if( vertex_Count == 2 ){
      X3 = $2;
      Y3 = $3;
      Z3 = $4;
   };
   vertex_Count = (vertex_Count + 1);
};

/endloop/{
   if( vertex_Count == 3 ){
      FUNC_interpret();
   };
   FUNC_reset();
};

END{ };
```

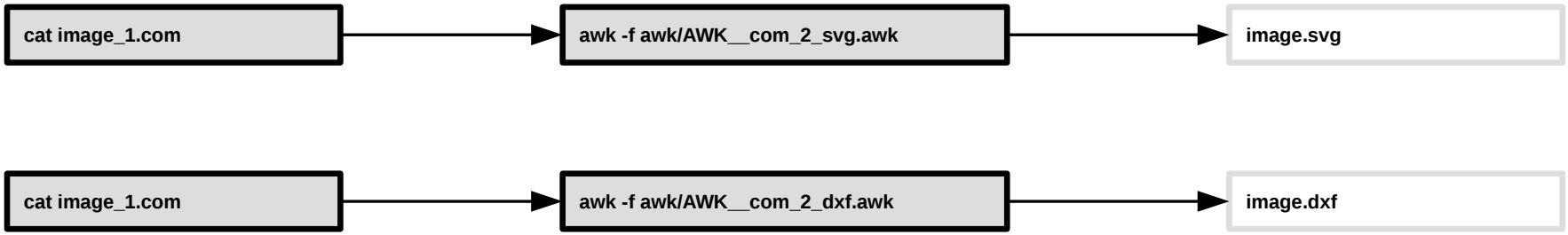
Erstellen und Platzieren von Laser-Schnitten.



Regel in einer Makefile

```
image_1.com:   makefile   freecad/halbkugel_0000.ast   awk/*.awk
        ( cat frame.com; \
        ( echo "#TRANSLATE   -50.0    80.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=1.5  -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE    80.0    80.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=4.5  -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE   -50.0   -80.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=7.5  -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE    80.0   -80.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=10.5 -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE  -105.0     0.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=49.5 -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE    -5.0     0.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=52.5 -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE    50.0     0.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=55.5 -f AWK__ast_2_com.awk ) ;  \
        ( echo "#TRANSLATE   100.0     0.0" ;    cat freecad/halbkugel_0000.ast        | awk -v PARAM_Z0=58.5 -f AWK__ast_2_com.awk ) ;  \
        echo "" ) | awk -f awk/AWK__com_translate.awk  >   $@
        echo "do done"
```

Erstellung von SVG- und DXF-Dateien.



Auszug aus einer COM-Datei

```
#LINE_MARK -145 -145 0.00000E+00 290.00000E+00
#LINE_MARK 145 -145 0.00000E+00 290.00000E+00
#LINE_MARK -145 -145 290.00000E+00 0.00000E+00
#LINE_MARK -145 145 290.00000E+00 0.00000E+00
#PARAM_Z0 = "1.5"
#LINE_CUT 9.9487 79.0205 0.0202473 0.979498
#LINE_CUT 9.9487 79.0205 -0.0607076 -0.977825
#LINE_CUT -78.7703 27.4164 1.50244 -0.773369
#LINE_CUT -80.2395 28.2512 1.46919 -0.834798
#LINE_CUT -75.9345 25.9636 -1.33342 0.67941
#LINE_CUT -66.8385 137.526 -1.60924 -0.515529
#LINE_CUT -65.2093 137.975 -1.62917 -0.448587
#LINE_CUT -69.871 136.548 1.42328 0.462454
```

Regel in einer Makefile

```
image.svg: image_1.com
        ( cat image_1.com | awk -f awk/AWK__com_2_svg.awk )  >  $@
        echo "do done"

image.dxf: image_1.com
        ( cat image_1.com | awk -f awk/AWK__com_2_dxf.awk )  >  $@
        echo "do done"
```

Zusammenfassung

Terminal

- In der Komandozeile können Programme zur Datenverarbeitung ausgeführt werden.

ADA

- Ermöglicht z.B. das Erstellen von Lasercat-Daten.

AWK

- Verarbeitung in Zeilen struktuerierter Daten.

make

- Ermöglicht die Struktuerierung von komplexen Verarbeitungssequenzen.

FreeCAD

- Ermöglicht die Konstruktion von 3D-Objekten.

XSLT

- Verarbeitung in XML struktuerierter Daten.

Literatur

Buchtitel	Autor
Repetitorium der Höheren Mathematik	Gerhard Merziger, Thomas Wirth
SED und AWK	Arnold Robbins
XSLT 2.0	Frank Bongers

Vielen Dank für Ihre Aufmerksamkeit.