figtex2eps
Converting *xfig*-figures with LaTeX to *eps*

Håvard Berland*

June 2003

Abstract

This document describes how to use a bash script for automating the compilation of *xfig*-figures with embedded latex commands into encapsulated postscript (eps) or into pdf.

Contents

1 Introduction

Figures in publications, theses, reports and alike with the same typographical quality as the rest of the document, often typeset by LaTeX, is not seen too often.

A natural prerequisite is to be able to use the same font for any text in the figures as you use for the rest of your document. Also, for many purposes, special symbols are needed which correspond to symbols in your

*http://www.pvv.ntnu.no/~berland

report. These should look exactly the same in both your figures and body text.

The fig-file format used by xfig, supports inserting LaTeX-commands by some neat tricks, leaving all that has to be done with LaTeX to LaTeX itself.

This script, named *figtex2eps* does nothing else than providing an easy interface to underlying commands that do the real work. The real work involves running several commands and making a tex-file for each figure you need, so there is definitely the need for some utility to automate the process.

2 Usage

The usage of the script itself is fairly simple, produce your figure using *xfig*, and save it in the vector format "fig", say in the file "`foo.fig`", and then run

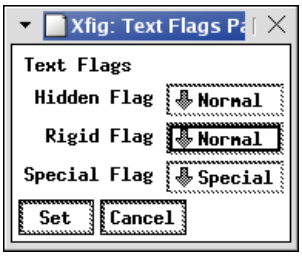
```
$ figtex2eps foo.fig
```

and after some seconds, you will have a file called "`foo.eps`" in the same directory.

If you didn't do any special things to embed latex-commands in your fig-file, this command will be pretty equivalent to an 'export to eps' in xfig. But the important thing is to include LaTeX-commands and -symbols.

2.1 Setting the *special* flag

The first thing you should do after launching xfig, is to ensure that all text you write, has the *special*-flag set.



1. Open xfig.
2. Push the big 'T' button in the toolbar at the left, and go into text mode.
3. Push the 'Text flags' button at lower (text-input-) toolbar, and set t special flag from *Normal* to *Special*, as done in the figure at right.

Now, all the text you input from now on will have this flag set, and this flag means that all treatment of text in your figure will be delegated to some other utility when exporting from the fig format. We want LaTeX to take care of everything regarding text.

2.2 Draw your figure and annotate it with text

This document will not try to learn you xfig, only how to insert LaTeX-commands. We provide a simple example of a figure with a modest use of LaTeX-features. There are some things to remember:

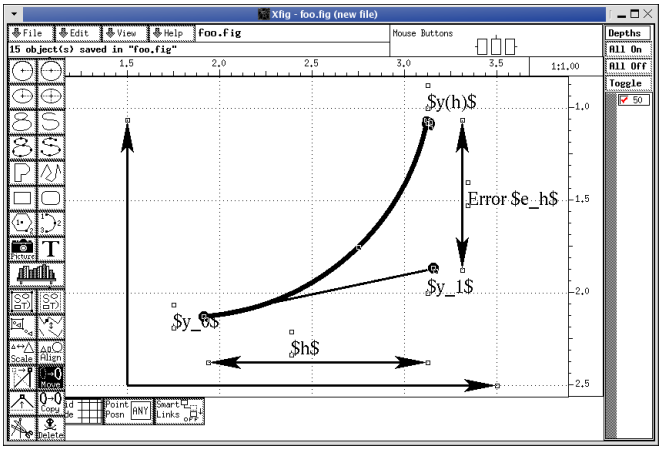


Figure 1: An example figure as shown in xfig.

- Write LaTeX as usual, that means enclosing LaTeX-constructs which require math mode with dollar signs ($), as in our example Figure 1.
- Be prepared to adjust the position of your text by trial and error. Often, the text and commands will overlap with your figure when viewed in xfig, but they won't in your final postscript.
- Be prepared for trouble if you include *very* complicated LaTeX-constructs in your figures. Debug the compilation with the verbose option (`figtex2eps -v foo.fig`). You might need to alter the preamble file, see below.
- Try to avoid resizing your figure to much when your final import into your document is done, you might get to small or to big fonts. Have in mind the final size of your figure in the document when you start to draw in xfig.

2.3 Converting to eps

When the above is done, this is the easy step. Save your figure as for example "`example.fig`" (do *not* use any of the export routines!) and run from your command line:

```
$ figtex2eps example.fig
```

and the file "`example.ps`" will be produced. During compilation, LaTeX produces a lot of output. Use the option "`-v`" (verbose) if you want to see this (could be necessary for some debugging if you write faulty LaTeX-commands). The outputted file from the figure in Figure 1 is shown in Figure 2.

Figure 2: Postscript produced by figtex2eps.

2.4 Converting to pdf

This is equally easy, by using figtex2eps' counterpart, *figtex2pdf*. This is merely a wrapper for your convenience, it calls figtex2eps on your behalf, and converts the eps-file to pdf using *epstopdf* (which must be available on your system). It also leaves the eps-file behind.

3 Modifying the LaTeX-preamble

When running figtex2eps for the first time in a directory, some output is actually produced:

```
$ figtex2eps example.fig
 (Info) Generating preamble file figtex2eps-preamble.tex
 (Info) You may edit to suit your needs if necessary and rerun
```

This tells you that a file has been generated in this directory. If you want to or have to, you may modify this file and rerun figtex2eps. If figtex2eps finds this file, it will use it, and not its own default. There could be several reasons for modifications to this:

- You have used special LaTeX-symbols in your figures, which require some more packages to be included. Then include more `usepackage`-lines in the preamble-file.
- You would like to adjust the default font size in the figures. This has been set to 12, despite that one usually uses 11 or 10 in reports. This is because one usually downscales figures when included in reports, but this may not be the case for you.

- You would like to change the font being used from Roman to something else (Palatino or Times for example).

4 How it works

The process of converting the fig-figure to eps narrows down to some simple steps:

1. Make a file with suffix `pstex`. This is a (postscript) file containing all parts of the figure that are not LaTeX-commands. This file can be viewed by `gv` if one wants to. This step is accomplished by a `fig2dev -L pstex` command.

2. Make a file with suffix `pstex_t`. This is a partial latex-document that includes latex-commands for putting the postscript picture in the `pstex` file in the background, and putting the wanted the latex-commands and other text on top of the eps-picture at the correct position. This step is accomplished by a `fig2dev -L pstex_t` command and some manual insertions of latex-code.

3. Make a tex-file which defines the font, packages to use and other necessary things, and make it include the pstex_t-file.

4. Run `latex` on our tex-file, which produces a dvi-file.

5. Make encapsulated postscript from our dvi-file with `dvips -E`

6. If `figtex2pdf` was called, convert the eps-file to pdf by the use of `epstopdf`.

Running figtex2eps with the verbose option ("`-v`") also leaves behind all the intermediate files that were created in the process.

5 Source code

The source code is available for download at the web-address
`http://www.pvv.ntnu.no/~berland/figtex2eps`,
and is also included here for reference.

```
#!/bin/bash
#
# Converts a .fig-file to a .eps-file, by using graphics from .fig as
# a backgroundpicture and latex-compiles tex-code from .fig on top
#
# Can also make pdf, by first making ps, and then running epstopdf
#
# Håvard Berland http://www.math.ntnu.no/~berland
#
# $Id: figtex2eps,v 1.7 2003/10/28 18:55:31 berland Exp $
```

```
# $Source: /home/pvv/d/berland/etc/cvsrepo/figtex2eps/prog/figtex2eps,v $
#

function usage
{
    echo ""
    echo "Usage:"
    echo " figtex2aeps <options> <figfile.fig>"
    echo ""
    echo " Options:   -v     Verbose, and don't delete temp files"
    echo "           -pdf   Generate pdf in addition via epstopdf"
    echo ""
    echo "Script written by Håvard Berland http://www.math.ntnu.no/~berland"
    echo "Documentation available at http://www.math.ntnu.no/~berland/figtex2eps"
    exit
}

function die
{
    echo "$1"
    echo "Exiting ... "
    exit
}

# A prefix used for outputting messages to the user
errorprefix=" (Error) "
infoprefix=" (Info) "

# The file where the preamble is to be written and/or found
preamblefile=figtex2eps-preamble.tex

# Option for running latex.
latexoption="-interaction=batchmode"

# Option for dvips. These are set for tex- installations based
# on teTeX, and ensures that Type1 fonts are used (via the file
# config.pdf loaded by -Ppdf), this gives correct fonts in the
# pdf-document. Set this variable to "" if this does not work,
# and try to find other ways of getting Type1 fonts.
dvipsoptions="-Ppdf -G0"

# Each of these commands must be available and in $PATH!
neededcommands="latex dvips fig2dev"

for cmd in $needecommands ; do
    which $cmd >/dev/null 2>&1 \
        || die "${errorprefix}Could not find $cmd in your path!"
done



if [ -z "$1" ]; then
    echo "${errorprefix}Mandatory input arguments not provided."
    usage
fi

# Default, throw away unwanted output from commands.
out=">/dev/null"

# Default option to run dvips in quiet mode:
dvipsout="-q"
```

```
# If verbose, alter the above set variables.
if [ "$1" = "-v" ]; then
    out=""
    dvipsout=""
    latexoptions=""
    shift
fi

# Check if user wants pdf (we might be called via wrapper script)
dopdf=""
if [ "$1" = "-pdf" ]; then
    dopdf="yes"
    shift
fi

# In case of changed order of options (this way of dealing
# with options is certainly not scalable !!!)
if [ "$1" = "-v" ]; then ## Do SAME as above..
    out=""
    dvipsout=""
    latexoptions=""
    shift
fi



# Check our mandatory input argument
figfile=$1 # may or may not include .fig as an ending

if [ ! -f $figfile ]; then
    # Test if user just dropped the ending:
    if [ -f "$figfile.fig" ]; then
        figfile="$figfile.fig"
    else
        die "${errorprefix}Could not find the xfig-file $figfile"
    fi
fi

# Check that the figfile is not empty
if [ ! -s $figfile ]; then
    die "${errorprefix}The file $figfile is empty."
fi

# If the 'file' and 'grep' command is available, we just as
# well check that it really is a FIG, not just
# some other file with the suffix .fig
format=`which >/dev/null 2>&1 file && which >/dev/null 2>&1 grep && file $figfile`
if [ ! -z "$format" ]; then
    if [ -z `echo "$format" | grep "FIG image text" >/dev/null && echo yes` ]; then
        echo "${errorprefix}According to the utility 'file', $figfile is not a proper fig-file."
        die "${errorprefix}'file' says: \"$format\""
    fi
fi

base="${figfile%.fig}"
outfile=$base.eps

# The user is also allowed to provide an output file if really necessary:
if [ ! -z "$2" ]; then
    outfile=$2
fi
```

```
## Make a preamble file if it does not exist
if [ ! -s "$preamblefile" ] ; then
    echo "${infoprefix}Generating preamble file $preamblefile"
    echo "${infoprefix}You may edit to suit your needs if necessary and rerun"

    # CVS tip if the directory CVS exists.
    if [ -d "CVS" ] ; then
        echo "${infoprefix}You might want to do a 'cvs add $preamblefile' as well"
    fi
    touch $preamblefile \
        || die "${errorprefix}Could not write to $preamblefile, check your permissions"

    ## This could have been redone with a HERE document..
    echo "%" > $preamblefile
    echo "% This is a preamble file for 'figtex2eps'. You may edit things" \
        >> $preamblefile
    echo "% here if necessary. Typically you might want to change the font" \
        >> $preamblefile
    echo "% size, the font or add some more packages for your latex commands" \
        >> $preamblefile
    echo "% in your figures. " >> $preamblefile
    echo "%" >> $preamblefile
    echo "% If you make errors in here, rerun figtex2eps with '-v' (verbose) and" \
        >> $preamblefile
    echo "% check the error messages from latex, and then fix here." >> $preamblefile
    echo "% You may also just delete this file if you are in trouble" >> $preamblefile
    echo "% and a new default one will be generated" >> $preamblefile
    echo "%" >> $preamblefile
    echo "\documentclass[12pt]{article}" >> $preamblefile
    echo "" >> $preamblefile
    echo "% Packages for most mathematical latex commands:" >> $preamblefile
    echo "\usepackage{amsfonts}" >> $preamblefile
    echo "\usepackage{amsmath}" >> $preamblefile
    echo "\usepackage{amssymb}" >> $preamblefile
    echo "" >> $preamblefile
    echo "\usepackage{ae}       % This is in case you also want to make pdf afterwards" \
        >> $preamblefile
    echo "" >> $preamblefile
    echo "% You might want the palatino font instead, then uncomment the following" \
        >> $preamblefile
    echo "% two lines, and do not use the ae package above" >> $preamblefile
    echo "%\usepackage{palatino}" >> $preamblefile
    echo "%\usepackage{palatcm} % Palatino math fonts " >> $preamblefile
    echo "" >> $preamblefile
    echo "\usepackage[dvips]{color}" >> $preamblefile
    echo "\usepackage{epsfig}" >> $preamblefile
fi

if [ ! -z "$out" ] ; then
        # Not verbose:
        fig2dev >/dev/null -L pstex $figfile $base.pstex
        fig2dev >/dev/null -L pstex_t $figfile $base.pstex_t_2
else
        # verbose
        fig2dev -L pstex  $figfile  $base.pstex
        fig2dev -L pstex_t  $figfile  $base.pstex_t_2
fi

echo "\begin{picture}(0,0)%" > $base.pstex_t
echo "\epsfig{file=$base.pstex}%" >> $base.pstex_t
echo "\end{picture}%" >> $base.pstex_t
```

```
cat $base.pstex_t_2 >> $base.pstex_t
cat $preamblefile > $base.tex
echo "\setlength{\textwidth}{100cm}" >> $base.tex
echo "\setlength{\textheight}{100cm}" >> $base.tex
echo "\begin{document}" >> $base.tex
echo "\pagestyle{empty}" >> $base.tex
echo "\input{$base.pstex_t}" >> $base.tex
echo "\end{document}" >> $base.tex
if [ ! -z "$out" ] ; then
    # Not verbose:
    latex $latexoptions $base.tex >/dev/null \
        || die "${errorprefix}Latex failed, rerun with '-v' (verbose)"
else
    # Verbose:
    latex $base.tex || die "${errorprefix}Latex failed"
fi

dvips -E $dvipsoptions $base.dvi $dvipsout -o $outfile \
    || die "${errorprefix}dvips failed, rerun with '-v' (verbose)"


# Make pdf if user wants to.
if [ ! -z "$dopdf" ]; then
    cmd="epstopdf"
    which $cmd >/dev/null 2>&1 \
        || die "${errorprefix}Could not find $cmd in your path, only eps generated!"
    $cmd $base.eps --outfile=$base.pdf
fi

# delete if not verbose
if [ ! -z "$out" ]; then
    rm -f $base.pstex $base.pstex_t $base.pstex_t_2  \
        $base.tex $base.aux $base.log $base.dvi
fi
```