```
#!/bin/sh

# $Id: $
#
# This is a PDF to PostScript filter for CUPS
#
# (C) 2003 Robert Sander <robert.sander@epigenomics.com>
#
# Released under GPL
#
# NO WARRANTY AT ALL
#

set -e

PDF2PS=/usr/bin/pdftops
PDF2PS_OPTIONS=""

# Check whether pdftops supports the "-origpagesizes" option
HAVE_ORIGPAGESIZES=no
if `$PDF2PS -h 2>&1 | grep -q -- -origpagesizes 2>/dev/null`; then
    HAVE_ORIGPAGESIZES=yes
fi

echo "DEBUG: pdftops argv[$#] = $@" >&2
echo "DEBUG: PPD: $PPD" >&2
echo "DEBUG: $PDF2PS supports '-origpagesizes': $HAVE_ORIGPAGESIZES" >&2

if [ $# -lt 5 -o $# -gt 6 ]; then
    echo "ERROR: $0 job-id user title copies options [file]" >&2
    exit 1
fi

# Read from given file.
if [ -n "$6" ]; then
    exec <"$6"
fi

# Apply PPD settings.

pslevel=2
if test -e "$PPD"; then
    eval "$(sed -nre 's/^\*LanguageLevel:\s*\"?([0-9]+)\"?.*/pslevel="${pslevel:-\1}"/p' "$PPD")"
fi
echo "DEBUG: PostScript Level: $pslevel" >&2

resolution=
eval "$(printf "%s" "$5" | sed -nre 's/.*(^|\s)Resolution=([0-9.]+(x[0-9.]+)?).*/
resolution="${resolution:-\2}"/p')"
if test -e "$PPD"; then
    eval "$(sed -nre 's/^\*DefaultResolution:\s*([0-9.]+(x[0-9.]+)?).*/resolution="${resolution:-\1}"/p'
"$PPD")"
fi
echo "DEBUG: Resolution: $resolution" >&2

if test -n "$resolution"; then
    # If the resolution is not symmetric, use the higher of the two,
    # Ghostscript ps2write device does not work with asymmetric resolutions.
    xres=
    yres=
    eval "$(printf "%s" "$resolution" | sed -nre 's/.*(^|\s)([0-9]+)x([0-9]+).*/xres="\2"; yres="\3"/p')"
    if test -n "$xres" && test -n "$yres"; then
        if [ "$xres" -gt "$yres" ]; then
            resolution=$xres
        else
            resolution=$yres
        fi
```

```
    fi
fi

width=
height=
bl_x=
bl_y=
tr_x=
tr_y=
margin_l=
margin_b=
margin_r=
margin_t=
pagesize=
unit=
customw=
customh=
eval "$(printf "%s" "$5" | sed -nre 's/.*(^|\s)(media|PageSize)=(\S+).*/pagesize="${pagesize:-\3}"/p')"
if test -e "$PPD"; then
    eval "$(sed -nre 's/^\*DefaultPageSize:\s*(\S+).*/pagesize="${pagesize:-\1}"/p' "$PPD")"
fi
echo "DEBUG: Page size: $pagesize" >&2

eval "$(printf "%s" "$pagesize" | sed -nre 's/^Custom\.([0-9\.]+)x([0-9\.]+)(\S*)$/customw="\1";
customh="\2"; unit="\3"/p')"

if test -n "$customw" && test -n "$customh"; then
    echo "DEBUG: Custom page size: $customw x $customh $unit" >&2

    if test "$unit" = "in"; then
        width="$(printf "scale=0; (%s)*(72.0)/(1.00)\n" "$customw" | bc)"
        height="$(printf "scale=0; (%s)*(72.0)/(1.00)\n" "$customh" | bc)"
    elif test "$unit" = "cm"; then
        width="$(printf "scale=0; (%s)*(72.0)/(2.54)\n" "$customw" | bc)"
        height="$(printf "scale=0; (%s)*(72.0)/(2.54)\n" "$customh" | bc)"
    elif test "$unit" = "mm"; then
        width="$(printf "scale=0; (%s)*(72.0)/(25.4)\n" "$customw" | bc)"
        height="$(printf "scale=0; (%s)*(72.0)/(25.4)\n" "$customh" | bc)"
    else
        width="$(printf "scale=0; (%s)/(1.00)\n" "$customw" | bc)"
        height="$(printf "scale=0; (%s)/(1.00)\n" "$customh" | bc)"
    fi

    if test -e "$PPD"; then
        eval "$(sed -nre 's|^\*HWMargins:\s*(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s*|bl_x="\1"; bl_y="\2";
tr_x="\3"; tr_y="\4"|p' "$PPD")"

        if test -n "$tr_x"; then
            tr_x="$(printf "scale=8; (%s)-(%s)\n" "$width" "$tr_x" | bc)"
        fi
        if test -n "$tr_y"; then
            tr_y="$(printf "scale=8; (%s)-(%s)\n" "$height" "$tr_y" | bc)"
        fi
    fi
elif test -n "$pagesize" && test -e "$PPD"; then
    eval "$(sed -nre 's|^\*PaperDimension\s+'"$pagesize"'(/[^:]+\|):\s*"(\S+)\s+(\S+)".*|width="\2";
height="\3"|p' "$PPD")"

    eval "$(sed -nre 's|^\*ImageableArea\s+'"$pagesize"'(/[^:]+\|):\s*"(\S+)\s+(\S+)\s+(\S+)\s+(\S+)".*|
bl_x="\2"; bl_y="\3"; tr_x="\4"; tr_y="\5"|p' "$PPD")"
fi

test -n "$bl_x" || bl_x=0
test -n "$bl_y" || bl_y=0
test -n "$tr_x" || tr_x=$width
test -n "$tr_y" || tr_y=$height
```

```
echo "DEBUG: Width: $width, height: $height, absolute margins: $bl_x, $bl_y, $tr_x, $tr_y" >&2

if test -n "$width" && test -n "$height" && \
   test -n "$bl_x"  && test -n "$bl_y" && \
   test -n "$tr_x"  && test -n "$tr_y"; then
    margin_l="$bl_x"
    margin_b="$bl_y"
    margin_r="$(printf "scale=8; (%s)-(%s)\n" "$width" "$tr_x" | bc)"
    margin_t="$(printf "scale=8; (%s)-(%s)\n" "$height" "$tr_y" | bc)"
fi
echo "DEBUG: Relative margins: $margin_l, $margin_b, $margin_r, $margin_t" >&2

if test -n "$margin_l" && test -n "$margin_b" && \
   test -n "$margin_r" && test -n "$margin_t"; then
    inject_ps="<</.HWMargins[$margin_l $margin_b $margin_r $margin_t] /Margins[0 0]>>setpagedevice"
fi

fitplot=
eval "$(printf "%s" "$5" | sed -nre 's/.*(^|\s)(fitplot|fit-to-page)($|\s).*/fitplot=1/p')"

ppd_opts=
if test -n "$pslevel"; then
    ppd_opts="${ppd_opts:+$ppd_opts }-level$pslevel"
    if [ "$pslevel" -ne "3" ]; then
        ppd_opts="${ppd_opts:+$ppd_opts }-noembtt"
    fi
fi
#if test -n "$resolution"; then
#  ppd_opts="${ppd_opts:+$ppd_opts }-r$resolution"
#fi
if test -n "$fitplot" || [ "$HAVE_ORIGPAGESIZES" = "no" ]; then
    if test -n "$width"; then
        ppd_opts="${ppd_opts:+$ppd_opts }-paperw $width"
    fi
    if test -n "$height"; then
        ppd_opts="${ppd_opts:+$ppd_opts }-paperh $height"
    fi
fi
if test -z "$fitplot" && [ "$HAVE_ORIGPAGESIZES" = "yes" ]; then
    ppd_opts="${ppd_opts:+$ppd_opts }-origpagesizes"
fi
if test -n "$fitplot"; then
    ppd_opts="${ppd_opts:+$ppd_opts }-expand"
fi
echo "DEBUG: PPD options: $ppd_opts" >&2

# We do not supply the margins to the ps2pdf process, as this breaks
# full-bleed printing and also disturbs the printing if PPDs have too
# conservative margin definitions.
inject_ps=

# Injection
echo "DEBUG: PostScript to be injected: $inject_ps" >&2
if test -n "$inject_ps"; then
    echo "DEBUG: Injecting PostScript: $inject_ps" >&2

    orig_infile="$infile"

    infile=$(mktemp -t pdftops.XXXXXX)
    tempfiles="$tempfiles $infile"

    perl -p -e 'if (! $did) { s:(^%!.*)$:\1\n'"$inject_ps"': && $did++; }' "$orig_infile" > "$infile"
fi

# We read the data into a temporary file as Ghostscript needs this to be
# able to work with PDF input
tempfiles=
```

```
trap 'rm -f $tempfiles' 0 1 2 13 15

ifile=$(mktemp -t pdftops.XXXXXX)
tempfiles="$tempfiles $ifile"

# Allow rendering of PDF files which have non-PDF stuff before and after the
# real PDF part, like for example PJL commands. Remove everything before
# "%PDF-X.Y" and after "%%EOF"
perl -e '$f = join("", <>); $f =~ s/^.*(\%PDF-\d+\.\d+)/$1/s; $f =~ s/(\%\%EOF\r?\n?).*$/$1/s; print $f'
> "$ifile"

echo "DEBUG: Running $PDF2PS $PDF2PS_OPTIONS $ppd_opts $ifile -" >&2
$PDF2PS $PDF2PS_OPTIONS $ppd_opts "$ifile" -
```