```
#!/bin/bash
(
#
#
# New smart print filter
#
#
# determines input file magic
#
# looks in the default filter plugin (FPI) directory
# finds all *.fpi files, then finds a combination that will
# yield the desired file type, or will indicate this is impossible.
#


function filtfrom {
    echo -ne ${1%-to-*}
}

function filtto {
    echo -ne ${1#*-to-}
}

#
# given filters as input vars, find next level available given the
# first arg is the starting point
#
function nextlvl {

    local try
    local start
    local all
    local depth

#
#
# $1 is starting point, find something that will work
#
    start="$1"
    shift

    depth="$1"
    shift

    all="$@"

#
#   get out of here if too deep!
#
    if [ $depth -ge $MAX_DEPTH ]; then
        return 1
    fi
    if [ $DEBUG_TREE ]; then
       echo "Starting point = $start" >> /tmp/filter.debug
    fi

    if [ $(filtto $start) = $DESIRED_TO ]; then
      echo " DONE"
        return 0
```

```
    fi

    while [ $1 ]; do
      try=$1
      shift
        if [ $DEBUG_TREE ]; then
        echo "for $start trying $try" >> /tmp/filter.debug
      fi
      if [ $(filtfrom $try) = $(filtto $start) ]; then
          echo -n "$start.fpi:$depth:CONT "

          if [ $(filtto $try) = $DESIRED_TO ]; then
            echo -n "$try.fpi:$((depth+1)):DONE "
            return 0
          else
#           echo -n $try
            nextlvl $try $((depth+1)) $all
#           echo "|G is $G| "
              if [ $DEBUG_TREE ]; then
             echo "|rt is $?|" >> /tmp/filter.debug
            fi
            if [ "$?" = "0" ]
                then
                   if [ $DEBUG_TREE ]; then
                  echo "for $start we are done"  >> /tmp/filter.debug
                   fi
#              return 0
            else
                   if [ $DEBUG_TREE ]; then
                  echo "for $start we have failed" >> /tmp/filter.debug
                   fi
                   return 1
                fi
          fi
      fi
#     echo ""
    done
}


#
# MAIN
#
#

#
#
#   setup some global variables used by this script
#
#

#
#   FPIDIR points to where print filter plug-ins are stored
#   Normally these will be installed with a package via RPM
#
    FPIDIR=/usr/lib/rhs/rhs-printfilters/

    PATH=${FPIDIR}:${PATH}
```

```
#
#   MAX_DEPTH determines how long a string of filters will be
#   tried as a possible printing solution. How many input
#   formats will take 6 filters to output Postscript!
#   Unlikely this will need to be changed.
#
    MAX_DEPTH=6

#
#   define these to gets lots of feedback
#   output is appended on /tmp/filter.debug
#
    DEBUG_TREE=""
    DEBUG_FILTER=""

#
#   Setup variables available to all filter plug-ins
#
#


#
#   SPOOLDIR is directory which lpd is spooling from
#
    export SPOOLDIR=$(pwd)

#
#   Get queue specific information (which was written by printtool)
#
    source ${SPOOLDIR}/general.cfg

    if [ "$DEBUG_FILTER" != "" ]; then
      echo "Desired print format is $DESIRED_TO" >> /tmp/filter.debug
      echo "Paper size is $PAPERSIZE" >> /tmp/filter.debug
      echo -n "A form feed will " >> /tmp/filter.debug
      if [ "$SEND_EOF" = "" ]; then
        echo "not be sent." >> /tmp/filter.debug
      else
        echo "be sent." >> /tmp/filter.debug
      fi
    fi

    cd $FPIDIR
    fpis=$(ls *.fpi 2> /dev/null | tr '\n' ' ' | sed 's/\.fpi//g')

#
# let's see if its a compressed file first
#
#
# Figure out the magic of the input file
#
    magic=$(file -)
    $FPIDIR/rewindstdin
    magic=${magic#*: }
    if [ "$DEBUG_FILTER" != "" ]; then
      echo "Magic is |$magic|" >> /tmp/filter.debug
    fi
    case `echo $magic | tr 'A-Z' 'a-z'` in
        *packed*|*gzip*|*compress* )
                                            DECOMPRESS="gzip -dc";;
```

```
        * )
                                                DECOMPRESS="";;
    esac

#
# Figure out the magic of the input file
#
    if [ "$DECOMPRESS" = "" ]; then
       magic=$(file -)
    else
       magic=$($DECOMPRESS - | file -)
    fi
    $FPIDIR/rewindstdin
    magic=${magic#*: }
    if [ "$DEBUG_FILTER" != "" ]; then
      echo "Magic is |$magic|" >> /tmp/filter.debug
    fi
    case `echo $magic | tr 'A-Z' 'a-z'` in
        *empty* )
                                            exit;;
        "pc bitmap data"* )
                                              startpnt="INPUT-to-bmp";;
        "gif image data"* )
                                              startpnt="INPUT-to-gif";;
        "jpeg image data"* )
                                              startpnt="INPUT-to-jpeg";;
        "tiff image data"* )
                                              startpnt="INPUT-to-tiff";;
        "sun raster image data"* )
                                              startpnt="INPUT-to-rast";;
        "pgm"*|"pbm"*|"ppm"* )
                                              startpnt="INPUT-to-pnm";;
        postscript* )
                                              startpnt="INPUT-to-ps";;
        "tex dvi file"* )
                                              startpnt="INPUT-to-dvi";;
        "fig image text"* )
                                              startpnt="INPUT-to-fig";;
# troff is safe again with groff-1.11a or later we hope
        "troff or preprocessor"* )
                                              startpnt="INPUT-to-troff";;
        "rpm"* )
                                              startpnt="INPUT-to-rpm";;
        *ascii*|*text*|*english*|*script* )
                                              startpnt="INPUT-to-asc";;
        *data*|*escape* )
                                              startpnt="INPUT-to-prdata";;
        *pcl* )
                                              startpnt="INPUT-to-prdata";;
         * )
                                              startpnt="INPUT-to-unknown";;
    esac


#
# here is where we could put in hook to call user routine(s) to
# handle extra magics they've defined filters for
#
#  call_user_magic_hook()
```

```
#
    if [ "$DEBUG_FILTER" != "" ]; then
       echo "Type of file is $startpnt" >> /tmp/filter.debug
    fi

    if [ "$startpnt" = "Dont know" ]; then
       echo "Error - input file type is unknown - cannot print"
       exit 1
    fi

#
# catch some easy cases without having to figure out best path the hard way
#
    bestpath=""
    foundbest="NO"
    if [ $(filtto $startpnt) = "asc" ]; then
       if [ "$ASCII_TO_PS" = "NO" ]; then
          bestpath="$startpnt | asc-to-printer.fpi"
        foundbest="YES"
       fi
    elif [ $(filtto $startpnt) = "prdata" ]; then
          bestpath="$startpnt | cat -"
        foundbest="YES"
    elif [ $(filtto $startpnt) = $DESIRED_TO ]; then
          bestpath="$startpnt | $DESIRED_TO-to-printer.fpi"
          foundbest="YES"
    fi

    if [ "$foundbest" != "YES" ]; then
#
#   we go through and find best path
#
    G=`nextlvl "$startpnt" "0" $fpis`

    if [ "$DEBUG_FILTER" != "" ]; then
      echo "$G" >> /tmp/filter.debug
    fi

#
# now sort out the best path of all available
#
#
# if no processing required, depth will equal 'DONE'
#
    if [ "${G# *}" != "DONE" ]; then
      root=""
      bestdepth=$((MAX_DEPTH*2))
      bestpath=""
      curdepth="0"
      depth="0"
      foundbest="NO"
      for i in $G; do
          entry=${i%%:*}
        depth=${i#*:}
          depth=${depth%:*}
        if [ $depth -le $curdepth ]; then
             while [ $(($depth <= $curdepth && $curdepth >= 0)) -eq 1 ]; do
                root=${root%* | *}
                curdepth=$(($curdepth - 1))
```

```
            done
          fi
         if [ $(($curdepth < 0)) -eq 1 ]; then
            root=""
          fi
          curdepth=$depth
         if [ "$root" = "" ]; then
           root="$entry"
         else
           root="$root | $entry"
          fi
         if [ ${i##*:} = "DONE" ]; then
            if [ "$DEBUG_FILTER" != "" ]; then
              echo "$root -> depth = $depth" >> /tmp/filter.debug
            fi
            if [ $depth -lt $bestdepth ]; then
               foundbest="YES"
               bestdepth=$depth
               bestpath=$root
            fi
          fi
      done
    fi

    if [ "$foundbest" = "YES" ]; then
       bestpath="$bestpath | $DESIRED_TO-to-printer.fpi"
    fi
#
# end of doing it the hard way
#
    fi
#
#   we have to add filter to convert desired format to something the
#   printer can handle. May be as simple as 'cat'
#
#
#   ok we got here, and if input data magic is 'data' we'll let it
#   through, hoping it really will work on the printer!
#   Note we still reject lots of magics, like ELF, by doing this
#   which is what we want
#
#
#   getting bad, but trapping all "special" cases here
#
#
    if [ "$foundbest" = "NO" ]; then
       printf "No way to print this type of input file: $magic \014"
       exit 0
    else
#
#   fix up the best path so we can run it
#
       if [ "$DECOMPRESS" = "" ]; then
         bestpath="cat - ${bestpath#* }"
       else
         bestpath="$DECOMPRESS ${bestpath#* }"
       fi
    fi
```

```
#
#   any post-filter to run (like smbclient?)
#
    if [ "$PRINTER_TYPE" = "SMB" ]; then
       bestpath="$bestpath | ${FPIDIR}/smbprint ${SPOOLDIR}/acct"
    elif [ "$PRINTER_TYPE" = "NCP" ]; then
       bestpath="$bestpath | ${FPIDIR}/ncpprint ${SPOOLDIR}/acct"
    fi

    if [ "$DEBUG_FILTER" != "" ]; then
      echo "Best path of depth $bestdepth is $bestpath" >> /tmp/filter.debug
    fi

#
# run the command!
#
    eval $bestpath 2>/dev/null

#
#
#   see if we need to send a form feed to eject the page from printer
#
#    if [ "$SEND_EOF" != "" ]; then
#      printf "\014"
#    fi

    exit 0
) | /usr/bin/sockprn 192.168.0.42 333
```