```
%!
% This program formats and displays text in fixed pitch font
% Created by Charles Rosenberg 3/25/91

% general use procedures
/inch { 72 mul } def            % convert inches to points
/cm   { 72 mul 2.54 div} def    % convert centimeters to points

% only define the value if not already defined
% caller should push the key, then the value
/cdef {
        exch dup where
        {pop pop pop}   % get rid of the arguments and dictionary
        {exch def}      % else define the value
        ifelse
} def


% check if a font exists
% takes in a font name and returns true or false depending on whether or no
t
% the font exists
/knownfont {

        /checkfontname exch def % the desired font

        /fontcnt 0 def          % the number of fonts
        /namestr 128 string def % to store file names

        % get the list of resident fonts onto the stack
        FontDirectory
        {
                pop             % remove the font dictionary
                /fontcnt fontcnt 1 add def
        } forall

        % get the list of disk fonts onto the stack
        /filenameforall where
        {
                pop             % get rid of the dict

                (fonts/*) {
                        dup length      % first chop off filename beginning
                        6 sub 6 exch getinterval
                        cvn             % convert to name object
                        /fontcnt fontcnt 1 add def
                } namestr filenameforall
        } if

        % now check for a match
        /foundflag false def
        fontcnt
        {
                checkfontname eq {/foundflag true def} if
        } repeat

        foundflag       % return the results
} def
```

```
% define the font
/pointsz 12 cdef                % MAY COME FROM SCRIPT
/fontstyle /Courier cdef        % MAY COME FROM SCRIPT


% page parameters
/pagemode 2 cdef        % page mode is 1, 2, or 4 leaves per page
                        % mode 3 is two long vertical columns
                        % MAY COME FROM SCRIPT

/leavesperpage          % set the number of leaves per page
        pagemode 3 eq {2} {pagemode} ifelse
        def

/pagewd 8.5 inch cdef   % page width is 8.5 inches, MAY COME FROM SCRIPT
/pageht 11 inch cdef    % page height is 11 inches, MAY COME FROM SCRIPT

/lmargin 0.50 inch cdef  % left margin is .5 inches, MAY COME FROM SCRIPT
/rmargin 0.50 inch cdef  % right margin is .5 inches, MAY COME FROM SCRIPT
/tmargin 0.50 inch cdef  % top margin is .5 inches, MAY COME FROM SCRIPT
/bmargin 0.50 inch cdef  % bottom margin is .5 inches, MAY COME FROM SCRIPT

/printwd pagewd rmargin sub lmargin sub def  % printable pg width
/printht pageht tmargin sub bmargin sub def  % prntable pg height


% calculate the point size and printable area size adjustment
pagemode 2 eq {

        % two leaves per page

        % rescale point size
        /sfact1 printht printwd div def
        /sfact2 printwd printht 2 div div def
        sfact1 sfact2 gt
                {sfact1} {sfact2} ifelse
        /sfact exch def

        % rescale printwd, printht and pointsz
        /pointsz pointsz sfact div def
        /printwd printwd sfact div def
        /printht printht sfact div def

        } if

pagemode 3 eq {
        % two columns per page

        % rescale point size
        /pointsz pointsz 2 div def

        % rescale printwd and printht
        /printwd printwd 2 div def

        } if

pagemode 4 eq {
        % four leaves per page

        % rescale point size
```

```
        /pointsz pointsz 2 div def

        % rescale printwd and printht
        /printwd printwd 2 div def
        /printht printht 2 div def

        } if


% use letter gothic if available, otherwise Courier
fontstyle knownfont
        {fontstyle}
        {/Courier} ifelse
findfont pointsz scalefont setfont


% first get font character width info
(g) stringwidth
pop
/dx exch def
/dy pointsz def

/charsperline printwd dx div floor cvi def  % characters per line
/linesperleaf printht dy div floor cvi def  % lines per leaf

% re-round printwd and printht
/printwd charsperline dx mul def
/printht linesperleaf dy mul def


% make an array of startx and starty for the leaves
% and do rotation and translation for 2 page mode

pagemode 1 eq {

        % one leaf per page
        % startx's
        /startx
                pagewd printwd sub 2 div                % leaf 0
                1 array astore def
        % starty's
        /starty
                pageht dup printht sub 2 div sub        % leaf 0
                1 array astore def

        % define procedure to draw page dividing lines
        /drawboxes {} def
} if

pagemode 2 eq {

        % two leaves per page

        % do rotation and translation
        90 rotate
        0 pagewd neg translate

        % startx's
        /startx
                pageht 2 div printwd sub 2 div          % leaf 0
                dup pageht 2 div add                    % leaf 1
                2 array astore def
        % starty's
        /starty
                pagewd dup printht sub 2 div sub        % leaf 0
                dup                                     % leaf 1
                2 array astore def

        % define procedure to draw page dividing lines
        /drawboxes {
                newpath
                0.5 setgray
                2 setlinewidth
                pageht 2 div pagewd moveto
                pageht 2 div 0 lineto
                stroke
                0 setgray} bind def
} if

pagemode 3 eq {

        % two columns per page

        % startx's
        /startx
                pagewd 2 div printwd sub 2 div          % leaf 0
                dup pagewd 2 div add                    % leaf 1
                2 array astore def
        % starty's
        /starty
                pageht dup printht sub 2 div sub        % leaf 0
                dup                                     % leaf 1
                2 array astore def

        % define procedure to draw page dividing lines
        /drawboxes {
                newpath
                0.5 setgray
                1 setlinewidth
                pagewd 2 div pageht moveto
                pagewd 2 div 0 lineto
                stroke
                0 setgray} bind def

} if


pagemode 4 eq {

        % four leaves per page

        % startx's
        /startx
                pagewd 2 div printwd sub 2 div          % leaf 0
                dup pagewd 2 div add                    % leaf 1
                2 copy                                  % leaf 2 & 3
                4 array astore def
        % starty's
        /starty
                pageht dup 2 div printht sub 2 div sub  % leaf 0
```

```
                dup                                     % leaf 1
                dup pageht 2 div sub                    % leaf 2
                dup                                     % leaf 3
                4 array astore def

        % define procedure to draw page dividing lines
        /drawboxes {
                newpath
                0.5 setgray
                1 setlinewidth
                pagewd 2 div pageht moveto
                pagewd 2 div 0 lineto
                stroke
                0 pageht 2 div moveto
                pagewd pageht 2 div lineto
                stroke
                0 setgray} bind def

} if



% procedure to break lines
/linebreak
{
        pop                     % don't want the next character

        % see if the current character is a special character
        dup /c exch def 9 eq {          % is it a tab?
           % if so, move to a column divisible by 8
           7 charcnt 8 mod sub dup              % get # of spaces
           charcnt add /charcnt exch def        % advance the character cou
nt
           {( ) show} repeat                    % print the spaces
        }
        { c 12 eq                       % is it a form feed?
           % if so, start a new leaf
           {nextleaf} if
        } ifelse

        /charcnt charcnt 1 add def  % advance the char count

        charcnt charsperline ge     % end of a line?
        {nextline} if
} bind def


% procedure to move onto the next line
/nextline
{
                /ypos ypos dy sub def
                startx leafcnt get ypos moveto
                /linecnt linecnt 1 add def
                linecnt linesperleaf ge     % end of a leaf?
                {nextleaf} if
                /charcnt 0 def
} bind def

% procedure to move onto the next leaf
```

```
/leaftop
{
        startx leafcnt get starty leafcnt get moveto
        /ypos starty leafcnt get def
        /linecnt 0 def
        /charcnt 0 def
} bind def

% procedure to display and move onto the next leaf
/nextleaf
{
        /leaftot leaftot 1 add def
        /leafcnt leafcnt 1 add def
        leafcnt leavesperpage ge        % end of the page?
        {nextpage} if
        leaftop
} bind def

% procedure to display and move onto the next page
/nextpage
{
        copypage erasepage
        drawboxes               % draw bounding boxes
        /leafcnt 0 def
        leaftop
} bind def


% procedure to display the text
/showtext
{

  /leafcnt 0 def        % starting at leaf 0
  /leaftot 0 def        % starting at leaf total 0
  drawboxes             % draw bounding boxes
  leaftop               % move to the top of the leaf

  {
        currentfile textbuf readline

         % a valid string so display it
        {
        {linebreak} exch kshow
        nextline
        }

        % EOF
        {
                exit
        } ifelse
  } loop

        linecnt 0 gt {showpage} if      % display the end of the last leaf
} bind def

/textbuf 32768 string def  % to hold text data

showtext
```