Another goodie that's in this release is the `proc` document-style option. It produces double-column conference proceedings format on $8\,{}^1\!/_2 \times 11$ paper. (Instead of sending in your camera-ready copy on those large sheets that they reduce by 25%, you can produce it on a high-quality output device and send it to them at its actual size.)

It has come to my attention that some installers have modified the standard document styles. **THIS IS STRICTLY FORBIDDEN.** The only changes to these styles that should be made are those necessitated by the use of different fonts. If you don't have a font that's called for in the standard style, do the best you can. If this produces noticeably different results, mention the difference in the *Local Guide.* Users expect the standard styles to produce the same output at different sites. If you must create local styles, give them different names and describe them in the *Local Guide.* The new manual describes what happens when `SAMPLE.TEX` is run with some modifications. Users will be unhappy if changes to the document style produce different results than is claimed in the book.

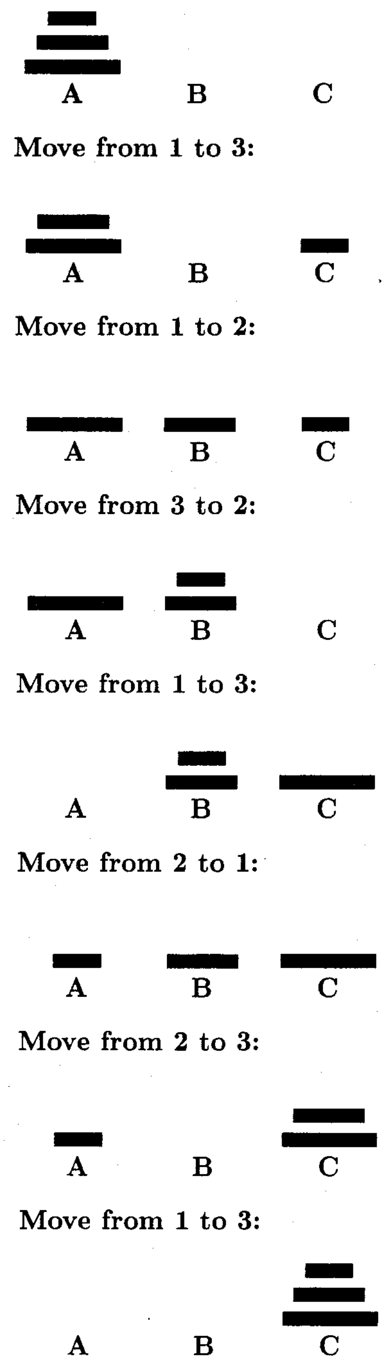
Speaking of document styles... before creating a document style for anyone else to use, talk to a typographic designer. People with no training in design who do their own formatting invariably do a rotten job. This is discussed in the new manual.

Enjoy.

A Solution to the Tower of Hanoi Problem Using TeX

Bruce Leban

Here is a solution to the classic Tower of Hanoi problem using TeX. This solution actually produces a printed solution to the problem illustrating the states of the stacks at each stage. Examination of this program may be instructive in understanding the operations of TeX's macro packages.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% \hanoi
%
% The basic macro that solves the Tower of Hanoi problem is called \hanoi.
% The first argument is the number of disks and the second is a list of disks.
% Each disk is identified by a single digit from 2 to 9 denoting its size.

\def\hanoi#1#2{%
   \numdisks=#1%
   \gdef\1{#2}\gdef\2{}\gdef\3{}%
   \showtowers123%
   \solve123%
   \vfill\eject
   }

\newcount\numdisks

% \solve#1#2#3 ::  move from #1 to #3 using #2

\def\solve#1#2#3{%
   \ifnum \numdisks=1 %
       \move#1#3%
   \else
       {\advance\numdisks by -1 %
        \solve#1#3#2}%
       \move#1#3%
       {\advance\numdisks by -1 %
        \solve#2#1#3}%
    \fi}

% \move #1#2 ::  Move from #1 to #2

\def\move#1#2{%
   \line{\bf Move from #1 to #2:  \hfill}
   \message{Move from #1 to #2.  }
   \first{#1} \append{.}{#2} \gstore{#2}
   \rest{#1} \gstore{#1}
   \showtowers123%
   }
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Lisp like functions for TeX.
%
% In order to implement the tower of hanoi, we implement a small list
% processing system in TeX.  Lists are implemented as strings of characters
% (tokens) stored in a macro.    Each variable is stored in a macro of the
% corresponding name.    For example, the variable `x' is stored in the macro
% `\x'.     Since it is convenient to pass around values directly, each function
% puts its result into the special variable `.'  (i,e., `\.').  For example, the
% Lisp code:
%     (setq a (append (first b) (rest c))
% would be coded as:
%     \first{b}          `.'  is now (first b)
%     \store{x}          `x' is now (first b)
%     \rest{c}           `.'  is now (rest c)
%     \append{x}{.}      `.'  is now (append (first b) (rest c))
%     \store{a}          `a' is now (append (first b) (rest c))
% The functions only support single-level lists (of tokens) and the function
% \first which produces the first element really produces the list of the first
% element, since these have the same representation.

% \value x  ::  \let\.=\x
% \Value x  ::  \let\:=\x  % We can use this to avoid clobbering \.
% \store x  ::  \let\x=\.
% \gstore x ::  \global\let\x=\.

\def\value #1{\expandafter\xvalue\csname#1\endcsname}
\def\xvalue{\let\.=}
\def\Value #1{\expandafter\xValue\csname#1\endcsname}
\def\xValue{\let\:=}

\def\gstore #1{\expandafter\xgstore\csname#1\endcsname=\.}
\def\xgstore{\global\let}
\def\store #1{\expandafter\let\csname#1\endcsname=\.}

% \append #1#2 ::  \.  <== (append #1 #2)

\def\append #1#2{\Value{#1}
                 \value{#2}
                 \edef\.{\:\.}}

% \first #1 ::  \.  <== (list (first #1))

\def\first #1{\value{#1}\expandafter\xfirst\.?!}
\def\xfirst #1#2!{\edef\.{#1}}

% \rest #1 ::  \.  <== (rest #1)

\def\rest #1{\value{#1}\expandafter\xrest\.??????????????????!}
\def\xrest #1#2?#3!{\edef\.{#2}}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% These functions are what actually display the towers.

\def\showname#1#2{
   \hbox to \hsize{%
      \hskip #2%
      \hbox to \towerwide{%
          \hfill {\bf #1}\hfill}%
      \hfill}}

\def\showdisk#1#2{%
   \hbox to \hsize{%
      \hskip #2%
      \hbox to \towerwide{%
          \hfill
          \vbox {\hrule height \diskhigh width #1\diskwide}%
          \hfill}%
      \hfill}%
   \vskip\diskvskip}

\def\showdisks#1#2.#3{%
   \if #1/
   \else \showdisk#1{#3} \showdisks#2.{#3}\fi}

\def\showtower#1/#2#3{%
   {\vbox to \towerhigh{%
      \vfill
      \showdisks#1/.{#3}
      \showname{#2}{#3}}}}

\def\showtowers#1#2#3{%
   \medskip
   \value{#1}
   \expandafter\showtower\./A{0pt}%
   \nointerlineskip
   \nobreak\vskip -1\towerhigh
   \value{#2}
   \expandafter\showtower\./B{1.05\towerwide}%
   \nointerlineskip
   \nobreak\vskip -1\towerhigh
   \value{#3}
   \expandafter\showtower\./C{2.1\towerwide}%
   \bigskip\goodbreak}

\baselineskip=0pt
\newdimen\diskwide\diskwide=9pt
\newdimen\diskhigh\diskhigh=5pt
\newdimen\diskvskip\diskvskip=3pt         % Vertical spacing between disks.
\newdimen\towerwide\towerwide=5\diskwide  % This is >= largest disk number.
\newdimen\towerhigh\towerhigh=5\diskhigh  % This is > number of disks.
           \advance\towerhigh 5\diskvskip

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% And now prove it all actually works.

\hanoi3{234}
```