Another goodie that’s in this release is the proc document-style option. It produces double-column conference proceedings format on 8½ x 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.

© 1984, 1985 Bruce Leban.
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% 
  \edef\first{#1}\edef\rest{#2}\edef\gd{1}% 
  \edef\gd{2}\edef\gd{3}\edef\solve123% 
  \solve123 
  \def\move#1#2{% 
    \message{Move from \first to \second:} \hfill 
    \line{Move from \first to \second:} \hfill 
    \message{Move from \first to \second.} 
    \first\ append{.}\rest\ gstore\first \rest\ gstore\first \first\ gstore\second 
    \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} 'x' is now (first b)
% \store{x} 'x' is now (first b)
% \rest{c} 'x' is now (rest c)
% \append{x}{y} '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#3<{
  \hbox to \hsize{%
    \hskip \towerwide{%
      \hfill (#1) \hfill%
    \hfll}
  }
}\def \showdisk#1#3{%
  \hbox to \hsize{%
    \hskip \towerwide{%
      \vbox \hfill {\hrule height \diskhigh width \#1\diskwide} \hfill%
    \hfll}
  \vskip \diskvskip)
}\def \showtower#1/#2#3{%
  \medskip
  \value(#1)
  \expandafter \showtower ./A(0pt)%
  \nointerlineskip
  \nobreak \vskip -\towerhigh 
  \value (#2)
  \expandafter \showtower ./B\{1.05\towerwide\%
  \nointerlineskip
  \nobreak \vskip -\towerhigh 
  \value (#3)
  \expandafter \showtower ./C\{2.1\towerwide\%
  \bigskip \goodbreak)
}\baselineskip=Opt
\newdimen \diskwide 
\diskwide=9pt
\newdimen \diskhigh 
\diskhigh=5pt
\newdimen \diskvskip 
\diskvskip=3pt
\newdimen \towerwide 
\towerwide=5\diskwide % Vertical spacing between disks.
\newdimen \towerhigh 
\towerhigh=5\diskhigh % 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\}