From the keyboard of flee@cs.psu.edu (Felix Lee):
:And Perl is definitely awkward with data types. I haven't yet found a
:pleasant way of shoving non-trivial data types into Perl's grammar.

Yes, it's pretty aweful at that, alright. Sometimes I write perl programs
that need them, and sometimes it just takes a little creativity. But
sometimes it's not worth it. I actually wrote a C program the other day
(gasp) because I didn't want to deal with a game matrix with six links per node.

:Here's a very simple problem that's tricky to express in Perl: process
:the output of "du" to produce output that's indented to reflect the
:tree structure, and with each subtree sorted by size. Something like:
:   434 /etc
: | 344 .
: | 50 install
: | 35 uucp
: | 3 nserve
: | | 2 .
: | | 1 auth.info
: | 1 sm
: | 1 sm.bak

At first I thought I could just keep one local list around
at once, but this seems inherently recursive. Which means
I need an real recursive data structure. Maybe you could
do it with one of the %assoc arrays Larry uses in the begat
programs, but I broke down and got dirty. I think the hardest
part was matching Felix's desired output exactly. It's not
blazingly fast: I should probably inline the &childof routine,
but it *was* faster to write than I could have written the
equivalent C program.

--tom

--
"GUIs normally make it simple to accomplish simple actions and impossible
to accomplish complex actions."   --Doug Gwyn (22/Jun/91 in comp.unix.wizards)

    Tom Christiansen   tchrist@convex.com   convex!tchrist