Subject: Re: Need ideas for a data structure
Posted by steinhh on Thu, 03 Nov 1994 11:04:42 GMT
View Forum Message <> Reply to Message

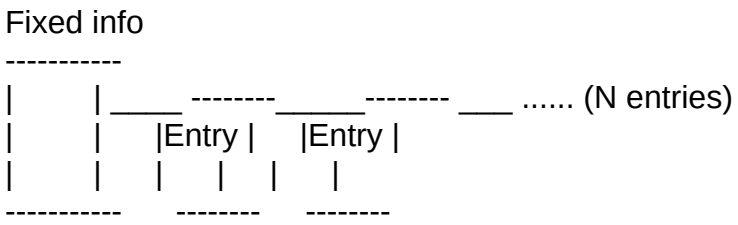
In article <3992q9$2pp@aplcomm.jhuapl.edu>, CroucAR1@subtech1.spacenet.jhuapl.edu writes:
|>
|> I'm trying to devise a way of dealing with a bunch of data items which will
|> consist of a fixed number of header fields and a widely varying number of data
|> entries. For one such item, an anonymous structure seems appropriate, but I
|> don't see how I can generalize this to an array.
|>
|> If I define an array of structures based on the largest array (which I won't
|> know ahead of time anyway), I'll waste a huge amount of memory.
|>
|> I thought of building an array of pointers to anonymous structures, but the
|> structures seem too anonymous for this to work.
|>
[..]

|> PS: I'm bi - it needs to work with both IDL and PV-Wave

I was just about to propose IDL handles, but then I saw your PS, which
makes it slightly more difficult.

But despair not! The solution is to do your own memory management --
I know it's not pretty, nor very efficient, but it works!

I gather that you have a data structure that could be represented
by the following "conglomerate":



and you'd like to store an array of such data conglomerate.

The fixed info is easily put into an array, with two tags, "start"
and "length", in addition to those representing fixed data.

Assuming that each "Entry" is of the same data type -- this is crucial --
create a global (common block) array of, say, a thousand entries, and
associate with it a global counter Nused (initialized to zero), keeping
track of how much of the array that's in use.

Now, each time you need a new conglomerate data sturcture, create the

Fixed info part. You ought to know by this time how many entries are needed, so initialize the "start" tag in the fixed info with the current value of "Nused", and increase Nused with the number of entriest that are reserved for the new data conglomerate.

If the entries are also variable, you need to go one step further down into the data structure, and implement the same idea there...

Stein Vidar
