Subject: Re: Modifying an array while conserving memory
Posted by Randall Skelton on Fri, 24 May 2002 11:45:28 GMT
View Forum Message <> Reply to Message

On 23 May 2002, Craig Markwardt wrote:

> At one time I devised a set of functions called ARRINSERT and
> ARRDELETE, in which I tried to do insertion and deletion operations as
> efficiently as I thought possible.
>
> I don't think I was able to get around your concern of necessarily
> allocating at least as much storage as the original data. What is
> really needed at the IDL internal level, is the ability to **augment**
> the storage of an existing array efficiently. I tend to do that alot,
> say when I'm building a list element by element.

It seems this will be my 3rd feature request of the week as my long-shot attempts at using realloc on an IDL passed array have failed miserably :(

Cheers,
Randall

--

```
void IDL_CDECL am_resize(int argc, IDL_VPTR argv[], char *argk) {

    /* Local */
    float *A;
    long n, l;

    /* IDL inputs: Ensure we get a float array from IDL */
    IDL_ENSURE_ARRAY(argv[0]);

    if (argv[0]->value.arr->n_dim != 1) {
        IDL_Message(IDL_M_NAMED_GENERIC, IDL_MSG_LONGJMP,
            "Passed array must only contain one dimension");
    }

    if (argv[0]->type != IDL_TYP_FLOAT) {
        IDL_Message(IDL_M_NAMED_GENERIC, IDL_MSG_LONGJMP,
            "Passed array must be of float precision");
    }

    /* Get a local pointer to the passed array */
    A = (float *) argv[0]->value.arr->data;

    /* Get number of elements */
    n = (long) argv[0]->value.arr->n_elts;
```

```
    /* IDL inputs: Ensure we get a scalar from IDL */
    IDL_ENSURE_SCALAR(argv[1]);

    /* Handle different precisions for new length */
    switch (argv[1]->type) {
        case IDL_TYP_FLOAT:  l = (long) argv[1]->value.f;
                             break;
        case IDL_TYP_DOUBLE: l = (long) argv[1]->value.l;
                             break;
        case IDL_TYP_INT:    l = (long) argv[1]->value.i;
                             break;
        case IDL_TYP_LONG:   l = (long) argv[1]->value.l;
                             break;
        default: IDL_Message(IDL_M_NAMED_GENERIC, IDL_MSG_LONGJMP,
                    "Type of passed length not recognized");
    }

    /* printf("Got: %li elemnts.  New length: %li.", n, l); */

    /* Watch me Seg fault... */
    if (realloc(A, 0) == NULL) {
        IDL_Message(IDL_M_NAMED_GENERIC, IDL_MSG_LONGJMP,
            "Cannot allocate memory");
    }

    /* Reset the number of elements in the IDL variable structure */
    /* argv[0]->value.arr->n_elts = (IDL_MEMINT) l; */

}
```