Subject: Re: System Resources ???
Posted by marq on Sat, 24 Feb 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Jim Brown <jtb@cdc.noaa.gov> wrote
> I am a bit concerned with how IDL handles the allocation of
> system resources. Things evidently are not being freed up
> properly...

I think this question came up a lot of times - to cite from
the FAQ:

The HTML version of the IDL FAQ is available at:

ftp://fermi.jhuapl.edu/www/s1r/idl/idl_faq/idl_faq.html

Ray Sterner sterner@tesla.jhuapl.edu
The Johns Hopkins University North latitude 39.16 degrees.
Applied Physics Laboratory West longitude 76.90 degrees.
Laurel, MD 20723-6099
WWW Home page: ftp://fermi.jhuapl.edu/www/s1r/people/res/res.html

<stuff deleted>

T27. Why is memory not released back to the operating system after an
array is deleted?

By Eric Korpela of Berkeley

This is a result of IDL being written in C and using the C library functions
(malloc and free) for memory allocation. In most C libraries, memory that is
freed is NOT returned to the operating system. The C program retains this
memory and will reuse it for future calls to malloc (assuming that the new
allocation will fit in the freed block).

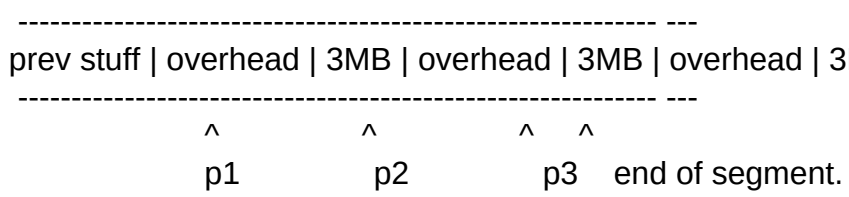
Another way of considering it is in terms of how memory allocation is done
under UNIX. New memory is allocated using brk() or sbrk() which control
the size of the data segment. These routines are called by malloc().

Suppose you allocate 3 1 MB regions of memory under C.

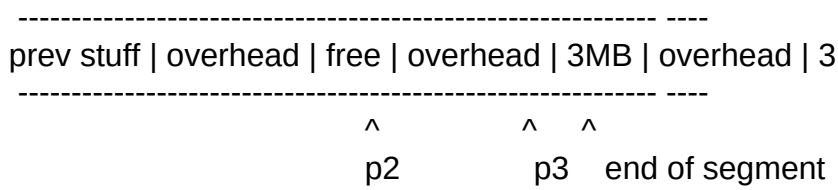
```
char *p1=(char *)malloc(3*1024*1024);
char *p2=(char *)malloc(3*1024*1024);
char *p3=(char *)malloc(3*1024*1024);
```

Here's what your data segment would look like assuming malloc had to call

sbrk().



Now we free(p1).



Notice that the free memory is still in the data segment. If free had called brk to reduce the size of the segment, the 3MB pointed to my p3 would be outside the data segment! SIGSEGV city! If free had moved the allocated memory to lower addresses so the segment size could be reduced without losing data, then p2 and p3 would point to invalid addresses, and we'd be forced to use handles rather than pointers and call GetPointerFromHandle() every time we wanted to access the memory. Ick! Just like Windows!

Hope this helps...

Regards,

Chris Marquardt (marq@strat01.met.fu-berlin.de)