<<< mmgmnew.obj >>>

High-speed Memory Control Function: beta version

September 20, 1996

Sony Computer Entertainment Inc.

[0] About previous version - mmgm.obj

High-speed Memory Control Function(beta version) was released as mmgm.obj(DTL-S 2190: Library Ver.3.3) ago. The process of mmgmnew.obj is almost same as mmgm. obj, but We recommand to use mmgmnew.obj after this. Because mmgm.obj rewrite the contents of kernel, sometimes the problems(hung-up,etc...) occur when the program is changed by LoadExec(). It does not occur by mmgmnew.obj.
If you replace mmgm.obj with mmgmnew.obj, you should modify the source code as follows.

1) Change InitHeap2() to InitHeap3()
2) Change malloc() to malloc3()
3) Change free() to free3()

[1] Overview

This object file includes the dynamic memory control server. This server is memory-resident, and adds a part of the system-call processing in order to (1) fix a bug and to (2) speed up the processing.

This object file will be extended to be embedded in the libapi in the future, and a group of memory control functions described above will be replaced completely.
However, since the memory control strategy for the current memory control functions will be substantially changed, it can be predicted that some problems will be caused by the operations on applications. Thus, this object file is distributed as a beta version reducing the functionality.

If you have any troubles or questions in the test use, please contact the technical support for BBS by mail specifying this object file name.
R&D group of SCE in Tokyo intends to develop the release version reflecting your advice and inquiries.

[2] Points of Modification

2.1
Memory control strategy
--> Until now the memory has been allocated from the bottom address toward the top address, but in this object file the memory is allocated from the top address toward the bottom address.

2.2
Non-support Function
--> The functions, calloc3() and realloc3(), will be released when the official version is completed.

2.3
Bug-fixed
--> The function, free(), has not released all the memory blocks. Consequently malloc() has sometimes failed even if there are available memory blocks. This
bug has been fixed.

2.4
Speeding-up the processing

[3] Notes in the Operation
3.1
This object file must be memory-resident. The easiest way is to link this file
to the application.

[4] Function

InitHeap3 Initializing a heap area

Syntax void InitHeap3(head, size)
void *head;
size_t• @size;

Arguments head Heap start address
size Heap size (a multiple of 4, in bytes)

Explanation This function initializes a group of memory control functions in this object file.
After this, malloc3(), etc. become usable. Since there is a overhead, all the bytes of size arent be available.
The overhead for a memory block is 8 bytes.

Return value None.

malloc3 Allocates main memory

Syntax void *malloc3(s)
size_t• @s;

Arguments s Number of bytes to be allocated

Explanation This function secures a block of s bytes from memory heap. This function is supported by InitHeap3().

Return value This function returns a pointer to the secured memory block. If it has failed to secure a block, it returns NULL.

[5] Obtaining the module top address
With the following procedure the top address of the execution module can be obtained, but prior to that, the next 2 conditions must be satisfied.

(1) Using the Psy-Q development environment.
(2) Not modifying the order of sections with lnk file.

Procedure 1:
Write the function (get_tail()) which returns the top address of BSS section
in the assembly language. This address becomes the top address of the execution module.

Procedure 2:
Call the above function in the application to obtain the top address, and then
allocate the heap area with InitHeap2(). After this, malloc() and free() becomeusable.

<< Subroutine to return the top address >>

```
        section  .bss
        xdef     get_tail
        section  .text
get_tail:
        la       v0,sectend(.bss)
        jr       ra
```

End of documents