vmem.library AutoDocs(tm)

COLLABORATORS

	TITLE : vmem.library AutoDocs(tm)	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 1, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

vmem.library AutoDocs(tm)

1.1 main

TABLE OF CONTENTS

1.2 background

?vmem.library/--background--- vmem.library/-- ↩
background---

THE ** VIRTUAL MEMORY ** LIBRARY

(VERSION 1)

These are the autodocs for programming vmem.library. VMem is a very simple set of six (6) functions, and four (4) macros. All you need to do to use VMem is :

* Call
  vmAllocBlock
  (or use the macro
  vmAllocData
) to get a control handle. (This is used by VMem to keep track of what's happening to your data).

* Call
  vmLock
  (or use the macro
  vmLockData
) to get a lock on the data you wanna use at that time.

* Access your data.

* Call
  vmUnLock
  (or use the macro
  vmUnLockData
) to unlock the data again.

* Call
  vmFreeBlock
  (or use the macro
  vmFreeData
) to free the control handle you got by calling
  vmAllocBlock
  (or using
  vmAllocData
)

The macro versions of the four functions just a simple way to alloc one element of data without the extra args required by the functions. (The functions have extra options to cope with array use.

The other two (2) functions are replacements for exec.library's AllocMem() and FreeMem() functions. They provide a few more functions than exec's versions, including flushing Virtual Memory if exec's AllocMem would normally fail.

See the individual autodocs below for a more detailed explanation.

1.3 vmallocblock

```
              ?vmem.library/vmAllocBlock                              vmem.library/ ↩
                 vmAllocBlock

NAME
        vmAllocBlock

SYNOPSIS
        vmBlock *vmb = vmAllocBlock(ULONG size,ULONG nels,ULONG memflags)
                  d0                  d0          d1          d2

FUNCTION
        Allocates a vmBlock for control of data which will use virtual
    memory.

INPUTS

    size:
    -   size of each element in array
    nels:
    -   number of elements in array
    memflags:
    -   standard exec.library/AllocMem() memory attribute flags

RESULT

    vmb:
    -   Virtual memory control handle (vmBlock).This block stores
        information about your data,and is passed to the other
        virtual memory functions to let them know what state your
        data is in.

EXAMPLE

    vmBlock *vmb;
    struct Preferences *prefs_ptr;/*largest structure I could think of*/
    ULONG i;
    ...

    if(!(vmb =
             vmAllocBlock
             (sizeof(struct Preferences),10,MEMF_PUBLIC)))
    {
        /*  vmAllocBlock failed - you're either *VERY* low on memory,
            or the the path that vmem.library is using is full */

        ...

    }else
    {
        for(i=0;i<NUM_ELS;i++)
        {
            if(!(prefs_ptr =
             vmLock
             (vmb,i)))
```

```
            {
                /* data lock failed - disk error , or VERY low on mem */
                ....

            }else
            {
                /* you've got your data,do whatever you want with it */

                ....

                /*  unlock the data when you've finished with it,so vmem
                    will know when it's not in use */


             vmUnLock
             (vmb,i);

            };
        };

        /*  when you've finished with all the data, free the control
            block */


             vmFreeBlock
             (vmb);
    };
```

NOTES

BUGS

Doesn't handle saving across multiple disks/devices yet.

SEE ALSO

```
             vmFreeBlock
             ,
             vmLock
             ,
             vmUnLock
             ,
             vmAllocData
                    exec.library/AllocMem autodoc (for memory flag definitions)
```

1.4 vmallocdata

```
             ?vmem.library/vmAllocData                              vmem.library ↩
                /vmAllocData
```

NAME

```
        vmAllocData         (MACRO)

SYNOPSIS

        vmBlock *b = vmAllocData(ULONG blocksize,ULONG memflags);

FUNCTION

    This is defined in Libraries/VMem.h as


            vmAllocBlock
            (blocksize,1,memflags)

    It just Allocates a vmBlock which isn't an array,i.e. Only contains
    one element.See that function's autodoc for details.

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO


            vmAllocBlock
            ,
            vmLockData
            ,
            vmUnLockData
            ,
            vmFreeData
```

1.5 vmallocmem

```
            ?vmem.library/vmAllocMem                                vmem. ↩
               library/vmAllocMem

NAME
        vmAllocMem

SYNOPSIS
        VOID *memptr = vmAllocMem(ULONG size,ULONG attrs)
                d0                  d0          d1

FUNCTION
        Identical to exec.library/AllocMem except does extra checking
    and vmem flushing if needed.
```

```
INPUTS
        Indentical to exec.library/AllocMem. See that function's autodoc
    for more information.

RESULT
    result:
    -   Pointer to memory which was allocated,or NULL if something went
        wrong.

EXAMPLE

NOTES

BUGS

SEE ALSO

             vmFreeMem
             ,  exec.library/AllocMem.
```

1.6 vmflush

```
             ?vmem.library/vmFlush                                          vmem. ↩
                library/vmFlush

NAME
        vmFlush

SYNOPSIS
        ULONG memflushed = vmFlush(ULONG memneeded,ULONG memflags);
                d0                      d0              d1

FUNCTION
        Flushes required amount unused virtual memory from the system.

INPUTS
        memneeded:
        -   Amount of *ADDITIONAL* free memory you need in bytes.
        memflags:
        -   Type of memory to flush (standard exec.library/AllocMem
            memory attributes)

RESULT
        memflushed:
        -   Amount of memory you got (in bytes).

EXAMPLE

NOTES
        memflushed is the memory that was freed into the free memory
    list by this function - by the time it returns,another program
    might have pulled it from under you, so check results of
             vmAllocMem
                    even if this function tells you there's enough mem.
```

```
BUGS

SEE ALSO
```

1.7 vmfreeblock

```
             ?vmem.library/vmFreeBlock                                vmem.library ↩
                /vmFreeBlock

NAME
        vmFreeBlock

SYNOPSIS
        void vmFreeBlock(vmBlock *b)
                a0

FUNCTION
        Frees a virtual memory control block when you have finished with
    it.

INPUTS
        b:
        -   pointer to the virtual memory block to free.

RESULT
        Frees all memory used for this virtual memory data,deletes any
    virtual memory files which exist on disk,and does other general
    cleanup stuff.

EXAMPLE

    See
             vmAllocBlock
             ()'s example above.

NOTES

BUGS

SEE ALSO


             vmAllocBlock
             ,
             vmLock
             ,
             vmUnLock
             ,
             vmFreeData
```

1.8 vmfreedata

```
             ?vmem.library/vmFreeData                                    vmem. ↩
                library/vmFreeData

NAME

        vmFreeData          (MACRO)

SYNOPSIS

        void vmFreeData(vmBlock *);

FUNCTION

    This is defined in Libraries/VMem.h as
             vmFreeBlock
             (b).It does exactly
    the same as
             vmFreeBlock
             ,and is only provided to make the macros

             vmAllocData
             ,
             vmLockData
             ,and
             vmUnLockData
              more programmer-friendly.
    See
             vmFreeBlock
             ()'s autodoc for usage.

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO


             vmFreeBlock
             ,
             vmAllocData
             ,
             vmLockData
             ,
             vmUnLockData
```

1.9 vmfreemem

```
             ?vmem.library/vmFreeMem                                    vmem. ↩
                library/vmFreeMem

NAME
        vmFreeMem

SYNOPSIS
        VOID vmFreeMem(VOID *memptr,ULONG size)
                       d0           d1

FUNCTION
        Identical to exec.library/FreeMem except does extra checking.

INPUTS
        Indentical to exec.library/FreeMem. See that function's autodoc
    for more information.

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

             vmAllocMem
             ,    exec.library/FreeMem
```

1.10 vmlock

```
             ?vmem.library/vmLock                                       vmem. ↩
                library/vmLock

NAME
        vmLock

SYNOPSIS
        VOID *ptr = vmLock(vmBlock *b,ULONG el)
              d0                  a0       d0

FUNCTION
        Locks a particular element of virtual memory data for use.

INPUTS
        b:
        -   Virtual Memory control block which required data belongs to.

        el:
        -   Particular element of data that you want to access.

RESULT

        ptr:
```

```
        -   Actual memory address of data element which is now safe to
            access.

EXAMPLE

    See
             vmAllocBlock
             ()'s example above.

NOTES

BUGS

SEE ALSO


             vmAllocBlock
             ,
             vmUnLock
             ,
             vmFreeBlock
             ,
             vmLockData
```

1.11 vmlockdata

```
             ?vmem.library/vmLockData                                    vmem. ↩
                library/vmLockData

NAME

    vmLockData          (MACRO)

SYNOPSIS

    VOID *memptr = vmLockData(vmBlock *blk)

FUNCTION

    This is defined in Libraries/VMem.h as vmLock(blk,0).
    It just locks a block of memory for a vmBlock with one element.
    See that function's autodoc for details

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO
```

vmLock
,
vmAllocData
,
vmUnLockData
,
vmFreeData

1.12 vmunlock

?vmem.library/vmUnLock vmem. ↩
library/vmUnLock

```
NAME
        vmUnLock

SYNOPSIS
        void vmUnLock(vmBlock *b,ULONG el)
                              a0          d0

FUNCTION
        Unlocks a virtual memory data element which was locked by

             vmLock
             ().

INPUTS
    b:
    -   pointer to the virtual memory control block for the data.
    el:
    -   The number of the element to unlock.

RESULT
        Unlocks the data.When data has no more locks on it,
    and real memory gets low,the data will be saved to a vmem file,
    and the real memory is freed.

EXAMPLE

    See
             vmAllocBlock
             ()'s example, above.

NOTES
        Note that this function takes the NUMBER of the element you
    wish to unlock,and not a pointer to the data as you might expect.

BUGS

SEE ALSO


             vmAllocBlock
             ,
```

vmLock
,
vmFreeBlock
,
vmUnLockData

1.13 vmunlockdata

?vmem.library/vmUnLockData vmem.library/ ↩
vmUnLockData

NAME

vmUnLockData (MACRO)

SYNOPSIS

void vmUnLockData(vmBlock *blk);

FUNCTION

This is defined in Libraries/VMem.h as
vmUnLock
(blk,0). It just
unlocks a block of memory for a vmBlock with one element. See that
function's autodoc for details.

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

vmUnLock
,
vmAllocData
,
vmLockData
,
vmFreeData