**NAME**

preallocate, myMalloc, myFree, affichage – Preallocate, allocate, free dynamic memory and display of each bloc's size from the heap

**SYNOPSIS**

**#include <bibliotheque.h>**

**int preallocate(unsigned size);**
**void * myMalloc(unsigned size);**
**int myFree(void * ptr);**
**int affichage();**

**DESCRIPTION**

Each blocs of the heap are of 'en_tete' structure type, which is the structure of a chained list :

```
struct en_tete {
        struct en_tete * suivant;       /* a pointer to the next free bloc of the heap */
        unsigned taille;                /* size of the bloc */
}
```

**preallocate()** initializes the heap by calling sbrk() which returns a pointer to the allocated memory. Then, the static variable *tas* (of 'en_tete' structure type also) is set. The first bloc of the heap is also set. His size is equal to *size*.

**myMalloc()** allocates *size* bytes and returns a pointer to the allocated memory. The memory is not cleared. The method used to allocate a memory bloc is First-Fit. That's to say, the first free memory bloc which size is large enough is used. The rest of the space is used to create another bloc. Be careful, because, for using **myMalloc()**, a heap must ever have been created before. Thus, if the heap has not been initialized or, if the bloc is too large to be saved in the heap, this function returns the value 0, after having set the variable *errno* to ENOMEM as it is required by the Unix98 standard.

**myFree()** frees the memory space pointed to by *ptr*. If *ptr* doesn't belong to the heap or, if the heap has not been initialized, no operation is performed. First of all, we search for the previous free bloc. Then, we check if an adjcent bloc is present just before (= the previous free bloc ) or just after the bloc which has to be freed. Finally, the chained list is updated.

**affichage()** displays the size( bytes ) of each bloc of the heap separated by the character '-'.

**RETURN VALUE**

**preallocate()** returns 0 if the request is a success and -1 otherwise.

**malloc()** returns a pointer to the allocated memory, or NULL if it has failed.

**free()** returns 0 if the request is a success and -1 otherwise.

**affichage()** returns no value.

**BUGS**

Officially, there are no bugs !! But, only officially ;-)