B4 - Unix System Programming

B-PSU-400

malloc

Memory ALLOCation

2.2

malloc

binary name:	libmy_malloc.so
repository name:	PSU_$ACADEMICYEAR_malloc
repository rights:	ramassage-tek
language:	C
compilation:	via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Unauthorized Functions: mmap, munmap, *alloc, free, dlopen, dlsym.

Using only **brk/sbrk** and your amazing mind, rewrite the 5 following C library functions: **malloc**, **calloc**, **realloc**, **reallocarray** and **free**.

Do not even think about using the malloc function.

Your function prototypes must be the same as the ones from the C library, and must be contained in the shared library compiled by your Makefile.

```
void *malloc(size_t size);
void free(void *ptr);
void *calloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);
void *reallocarray(void *ptr, size_t nmemb, size_t size);
```

All programs using malloc should work with your own malloc. Remember to test it with several existing programs.
Some programs use their own allocation system and memory management. Think about what kind of tests you're going to perform.

Allocation Strategy

You allocation strategy must be like follow:
- you must align your memory on a power of 2.
- the break must always be aligned on a multiple of 2 pages.
- you must implement the best fit algorithm.

Bonus

As a bonus you could:
- handle */etc/malloc.conf* file
- handle memory allocation by multy-thread process.