ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

FAME

Pratical Work #2 : Memory management

C. BARÈS

1 – Memory segments

Using the "%p" field of `printf` to display the address of a variable, write a program that exposes the localization of the following memory segments :

Data Initialized global data (stored in the file)
BBS Uninitialized global data (stored in the file)
Ro-data Global constant (stored ine the file)
Heap Uninitialized global data
Stack Limited scope data stored in the execution stack
Main Function Code Memory Zone (.text)
LibC Functions Shared Libary Memory Zone
Mmap Memory area allocated by `"mmap"`

Make sure that your program uses the `"pmap -X PID"` command to display the memory map of your process, and thus be able to check the addresses of the different allocated segments. On MacOsX, have a look to the `vmmap` command.

2 – Projection of file in memory

To speed up access to large files, it is generally preferred to map them to memory rather than access them via standard read/write. The mapped file can be accessed as a memory area accessible by a pointer (an array). This functionality is made possible by the virtual memory mechanism.

To map a file, we use the `"mmap"` function (and `"munmap"` to close the memory mapping).

1. Create a test.txt file containing some text.
2. Open this file (via `"open"`), get its size back (via `"fstat"`).
3. Map the entire file to memory.
4. Reverse the bytes of the file (the start bytes of the file are at the end).
5. End memory mapping.
6. Check that the text in the file has been reversed (`"cat test.txt"`).

3 – Chained lists

1. Create a chained list containing the first n integers in ascending order.
2. Create a function that returns the length of a list.
3. Create a function that returns the average of a list.
4. Write a function that returns the list of squares from another list passed as a parameter.
5. Remove the first item from a list.
6. Remove the last item from a list.
7. Add an item at the end of a list.
8. Add an item at the beginning of a list.
9. Write a function that concatenates two lists.
10. Turn your list into a double-chained list
11. Create a double circular chained list (adapt the previous functions)
12. Display the first n first integers in descending order.