# Prog1

The first thing we have done has been to compile the program and to debug it, to know how the variables are located in memory (although the memory space that is assigned varies, these are always ordered in a certain way, according to their order of appearance).

So we can say to illustrate this explanation that the memory space allocated for each variable is:

char buffer[8] --> 8 bytes in memory location 0x4028

int val1 --> 4 bytes at memory location 0x404030

int val2 --> 4 bytes at memory location 0x404034

Knowing that C does not have a certain endianess, but adapts to the characteristics of the machine on which it is executed, it has been verified that on the machine on which the program has been executed, the endianess is little endian. Therefore, buffer will be the variable written first to memory, then val1, val2 being the last.

Explanation of the code:

Texto

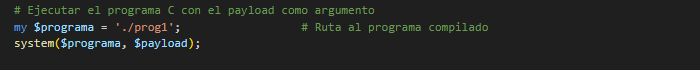
Descripción generada automáticamente

We enable strict and warnings to warn us of possible errors when executing the script.

Texto

Descripción generada automáticamente

We assign the values of val1 and val2 from the first iteration. Then the payload for the buffer overflow is constructed. A string consisting of 8 characters 'A' is created to fill the buffer. Then the values of val1 and val2 are added in little endian format using Perl's pack function.



We execute the C program prog1 with $payload as argument with the system function. In such a way that the payload is received as argument in the program and the buffer overflow occurs.

Texto

Descripción generada automáticamente

The rest of the code is to repeat the previously commented code, only that the values of val1 and val2 will be changed.

# Prog2