Programmation concurrente

BARBE Victor 403715
GAUCHER Pierre-Louis 403783
Arturo Arellano Coyotl 168824
Ruben Jaramillo Gomez 168406

1) marshall.c

In marshall.c, we are trying to write in a file that will be recovered by another C file. We got four char values (c1, c2, c3, c4) and one int (i), that we will try to print in “file.bin”.

We first must create a pointer of FILE p to create our “file.bin”. We open it on write mode to modify its content.

After checking if the file has correctly opened, we can now process the writing in “file.bin”.

Because the four char values are only written on one byte, we can easily write them in “file.bin” using fputc method.

Because “i” is an integer, it is written on 32 bits – 4 bytes. It is basically 0000 0000 0000 1000 in binary. Since we are in little endian and we want to write in big endian we will be sending the bytes using the pointer in the order 3, 2 and 1.

To do so, we create a pointer on “i” that will allow us to get each byte of the variable. We know have to use fputc to write in the file the required values. We give as parameters in fputc the first four digits of “i” in binary using pointeur[index], with index going from 3 to 0.

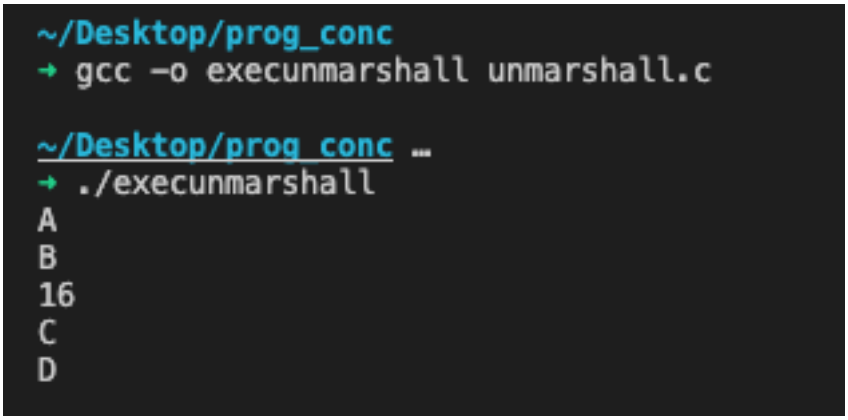
Now that the variables are written in the correct order in the binary file, we have to open it using unmarshall.c

2) unmarshall.c

In unmarshall.c, the goal is to read the “file.bin” file generated by marshall.c, store the values in variables bytes by bytes. There will be no problems for the char values because they are also char in unmarshall.c. We simply have to use fgetc() method to store it into variables.

However, “i” is now a “long” type, which means that he’s written on 64 bits – 8 bytes. We will get the previous values of “i” using fgetc again. Since the variable is now written in big endian and we want it in little endian, we are going from index 3 to index one again.

However, this time the variable i is declared as a long. Since it has 8 bytes, we have to fill them with the value 0 at the end of the program. Then we can display our variables, and we see that the value changed as expected.


~/Desktop/prog_conc
→ gcc -o execunmarshall unmarshall.c
~/Desktop/prog_conc …
→ ./execunmarshall
A
B
16
C
D