B0 - Workshop

B-INN-000

Workshop

Reversing Pokemon game save

1.0

{EPITECH.}

Workshop

binary name: none
language: python3
compilation: none

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- 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).

Goal

In february 2021 the Pokemon game will celebrate its 25th anniversary. To pay an homage to this game series we'll use our programming skills and more generaly our knowledge about computers to write a simple save game editor for the Pokemon Red and Blue games on the GameBoy

Tools needed

- For sake of simplicity we'll write our code in Python3 (it's already on your dump).
- You'll also need a Gameboy emulator and the game.

```
Terminal
~/B-INN-000> sudo dnf install visualboyadvance-m
```

- We'll also use a tool called vbindiff to help us figuring out the save data format of those games.

```
Terminal
~/B-INN-000> sudo dnf install vbindiff
```

- An hex file editor

```
Terminal
~/B-INN-000> dnf install ghex
```

A word on how saves works on the original GameBoy

The gameboy console does not have any "secondary memory" (hard drive, SDD, flash memory).
The only memory avaible to this console is:

- it's CPU registers and the RAM.
  However those are "volatile" memory, that is to say that once the console is turned off, everything stored in RAM and and the CPU's registers are gone.
- the game ROM (Read Only Memory).
  However this as it names implies, this type of memory is Read Only (it's impossible to write to it, so it's impossible to save our progression throught the game on it)

But how it is possible to save our progession for those games ?

+ The Memory Bank Switching and battery powered RAM old trick:

On certain type of game cartridge it's possible to find additionnal RAM storage connected to a battery.

Remember RAM is zeroed once the console is turned off… but what if we have some RAM connected to a battery permanently ? Yes, it allows us to write data that can be preserved even if the console is switched off.

When the player wants to save, some values from the Internal RAM are copied to the External Battery Powered RAM.
When the player wants to load, some the data stored in the External Battery Powered RAM are copied in the Internal RAM.

+ SAVE GAME EDITOR ?

To write a save game editor, we'll just need to figure out where different data structure are stored in the External RAM (Player name, amount of money, inventory, party, etc…) and change the values at those addresses

STEP 1: CHANGING THE PLAYER/RIVAL NAME

+ HOW TO FIND WHERE THE PLAYER/RIVAL NAME ARE STORED ?

When you start a new game allows us to pick a new name for the player and the rival.
The maximum lenght of those name is 7 characters.

So if the player is named "AAAAAAA" in one save file and "ZZZZZZZ" in another save file, then by comparing the 2 save files at one point we should find 7 identicals bytes in both files and the difference between those char will be 25 ('Z' - 'A' = 25)

To launch the game on 2 different saves:

```
Terminal
~/B-INN-000> mkdir saveA
```

```
Terminal
~/B-INN-000> mkdir saveB
```

```
Terminal
~/B-INN-000> cp pokered.gb saveA
```

```
Terminal
~/B-INN-000> cp pokered.gb saveB
```

```
Terminal
~/B-INN-000>
visualboyadvance-m saveA/pokered.gb
```

```
Terminal
~/B-INN-000>
visualboyadvance-m saveB/pokered.gb
```

Once you save the game you'll end up with 2 different save files "pokered.sav" in saveA and in saveB

Use the program vbindiff to find the address of where the player's name and the rival's name are stored

For example in saveA/pokered.sav you may have:

0x00 – 0x08	AE	F6	D1	07	02	00	FF	31
0x09 – 0x10	B3	DD	FF	FF	FF	00	00	00
0x11 – 0x18	FF	01	**8F**	**8F**	**8F**	**8F**	**8F**	**8F**
	8F	04	00	12	D6	FF	FF	…
…	…	.	Addr: 0x13	…	…	…	…	…
	…	…	…	…	…	…	…	…
	…	…	…	…	D3	FF	FF	01
0x38 – 0x40	0D	00	01	D4	8E	EF	FF	42

and in saveB/pokered.sav you may have:

0x00 – 0x08	AE	F6	D1	07	02	00	FF	31
0x09 – 0x10	B3	DD	FF	FF	FF	00	00	00
0x11 – 0x18	FF	01	**8D**	**8D**	**8D**	**8D**	**8D**	**8D**
	8D	04	00	12	D6	FF	FF	…
…	…	.	Addr: 0x13	…	…	…	…	…
	…	…	…	…	…	…	…	…
	…	…	…	…	D3	FF	FF	01
0x38 – 0x40	0D	00	01	D4	8E	EF	FF	42

See the difference ?

+ MODIFING

Once we have this address, Let's modify one of our save file by using ghex to change the bytes at those address
if the player is named "AAAAAAA" it should be [0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80] changing those bytes to [0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81] the player should be now named: "BBBBBB", and changing those bytes to [0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88] the player should be now named "CDEFGHI"
Try it !

Note that the game uses a different character table than the ASCII table (in ASCII the char 'A' correspond to 0x41 in hexadecimal)

+ Testing

Fire up the emulator at check it out !
Oh no, it say that the save file is corrupted.

+ Fixing the checksum

Developpers are smart…
Imagine your saving your game and the during the save process the console is turns off before the end of the writing process ? Our save will be corrupted !
To prevent you to play on a corrupted save, a checksum is made at the end of the saving process.

What is a checksum?

A checksum is a small block of data derived from another bigger block of data allowing to make error detection.
In order to use our modified save we'll have to calculate a checksum and writing it's result somewhere in the savefile

Use the checksumfix.py program to fix your save

+ Step 1 Bonus:

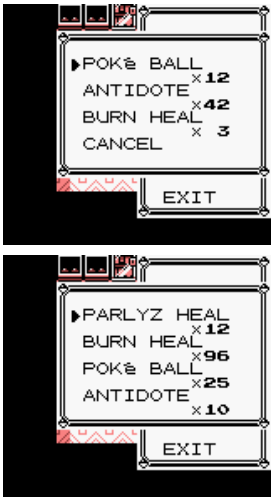
Write a program that opens a save file, modify the player names to "EPITECH", fix the checksum and write back the modified save file.
Here is a starting code

Terminal

```
~/B-INN-000> cat pokesave_editor.py
#!/bin/env python3
import sys
def fix_checksum(save_data):
    checksum = 0xFF
    for byte in save_data[0x2598:0x3523]:
    checksum -= byte
    save_data[0x3523] = checksum & 0xFF
    return save_data
with open(sys.argv[1], "rb+") as save_file:
    save_data = bytearray(save_file.read())
    ##### Your code to modify save_data HERE...
    save_data = fix_checksum(save_data)
    save_file.seek(0, 0)
    save_file.write(save_data)
```

Step 2: Changing the player's inventory.

On two different saves put some different items in your inventory (and different amount of those items).



Use vbindiff to find the difference between the two files.
Find the address where the inventory is stored

+ Spoiler

Think about it how your inventory should be stored in that file ?

- The games need to know how many different items you have in the inventory (since you cannot have more than 20 items, this value should be coded on 1 byte)
- The games need to know what kind of items you have (It should use 1 byte to store the item index number, for example the item with the index 0x01 correspond to a Master Ball, and the item with the index 0x14 correspond to a Potion) and the amount of those items you own.

  If in one save file your first item is a Potion and you have 42 of those and in the other save file your first item in the list is a Pokeball and you have 12 of those by calculating the difference between each byte in both save you should have somewhere a difference of -30 on a certain byte. This specific byte store the amount of the first item you own.

+ Step 2 Bonus:

Add to your program a function to set the quantity of the first object of your inventory to 99), do not forget the fix the checksum

Step 3: Modifying the party

Find the address where the list of pokemon in your party are stored
Once it's done, try to modify the composition of your party.

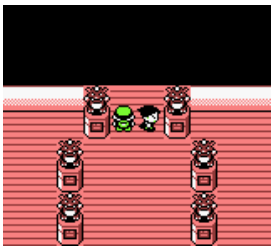
See poke_index.txt

Step 3: Bonus

Add a function to change your first pokemon in your party for any valid pokemon you could find in "poke_index.txt"

Step 4: Dead end ?

Load the save file for the step 4.
You're just before the final boss room, however your party sucks.



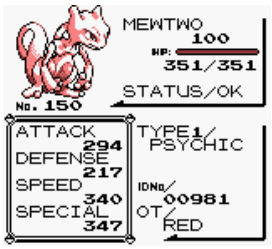
Try to modify the stats of one of your pokemon to beat the final boss (spoiler: it will be hard)

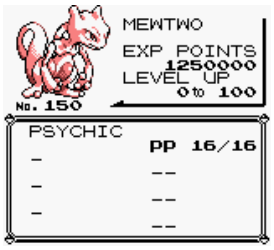
+ Tips:

Pokemon stats (HP, Attack, Defense, etc.) are defined in a 44 bytes structure

C.F. poke_stat_struct.txt, list_move_index.txt

Try to make your Pokemons look like this:




MEWTWO
EXP POINTS
1250000
LEVEL UP
0 to 100
No. 150
PSYCHIC
pp 16/16
-
--
-
--
-
--