Description

Create a utility that converts a file of recorded big endian messages into an equivalent little endian representation. Optionally display the contents in CSV format. Optionally display a count of each message.

For example, an operator returns from a shift and gives you an SD card with a file containing the data from a field test of company's new product. A systems engineer wants to analyze the data to see if the new product is working as well as promised. That engineer has a PC-based analysis application that she wrote and that she has refined over the years. Unfortunately, the new product uses a Motorola processor so the data is stored in big endian. And we know that most PCs are little endian. So you need to write a utility to convert the file. And you'll have to give the converted file to the system engineer.

So, once you have the utility written, you will run it like this:

```
[pond@srvr1 ~]java Assign01 beFile leFile
```

The `beFile` is the file from the new product so it is in big endian format. The `leFile` is the output file in little endian format.

Your utility also provides a command line parameter that causes your utility to dump the contents of the file in a comma-separated value (CSV) format instead of little endian. For example,

```
[pond@srvr1 ~]java Assign01 -csv beFile beFile.csv
```

Finaly, you also provide a command line parameter that causes your utility to display counts of each message type. For example,

```
[pond@srvr1 ~]java Assign01 -count beFile leFile
Message counts: Setup 02, Data 3523, Status 122
```

Messages

Setup Message

The Setup Message describes the data collection session. The messages that follow the Setup message belong to the specified session. Each new Setup message indicates the start of a new session.

- id=100 (byte)
- spare (byte)
- length (16-bit)
- seq_num (32-bit)
- start_time (64-bit)
- latitude (float)
- longitude (float)
- operator (32-element byte array)

Data Message

The Data message captures each measurement made by the device during the session.

- id=112 (byte)
- spare (byte)
- length (16-bit)
- seq_num (32-bit)
- start_time (64-bit)
- speed (float)
- samples (32-bit)

Status Message

The Status message records hardware status during the session.

- id=120 (byte)
- spare (byte)
- length (16-bit)
- seq_num (32-bit)
- start_time (64-bit)
- error_code (16-bit)

- component (16-bit)
- version (32-bit)

Endian Swaps

16-bit:`0xba23 => 0x23ba`

32-bit:`0xba2304ac => 0xac0423ba`

64-bit:`0xba2304ac7538e196 => 0x096e13875ac0423ba`

Byte values do not get swapped and neither do strings or character arrays

CSV File Format

CSV is a very simple file format that Microsoft Excel, among other programs, can read and interpret. It is essentially an ASCII spreadsheet. A CSV file for your project looks like this:

```
100,0,56,1,123324145,35.1332,-74.13413,Rory

112,0,24,2,123324201,64.234,4

120,0,24,3,123324483,5432,2342,1010101
```

Restrictions

You may not use an endian swap routine from a library or from the Java language itself. You must implement your own swap method, either with byte manipulations or arithmetically.

Notes

The input file is in binary so it cannot be edited in an ordinary text editor. You will be supplied a program that generates one message of each type so you can make your own test file.

Quit if a message is not recognized or otherwise erroneous.

The file will be well-formed meaning each message will have a four-byte header with a valid length field. However, the last message may be truncated.

Extra credit: skip unrecognized message and keep processing.

gl appears to be a big endian machine.