SMART HOME DEVICE LOG CONVERSION TOOL

Due Date:

April 29, 23:55

Goal

You are tasked to implement a command-line tool to convert logs of smart home devices between CSV, Binary and XML formats. Additionally, an XSD file must be prepared to validate the XML structure. All programming must be done in C programming language and the project will be carried out in groups of ***two***.

This assignment will help you solidify your understanding of:

- File I/O operations in C
- Different file formats (CSV, Binary, XML)
- JSON configuration parsing
- Schema validation via XSD
- Endian representations

Your program must handle various CSV formats by accepting the following required arguments: -separator <1|2|3> (1=comma, 2=tab, 3=semicolon) and -opsys <1|2|3> (1=windows, 2=linux, 3=macos). It should also support the -h flag to display usage instructions.

Context

You are provided with a sample CSV file named ***smartlog.csv*** which contains logs from a smart thermostat system.

Each log entry includes:

Attribute	**Description**
device_id	Unique ID, format must be 3 uppercase letters followed by 4 digits (e.g., "THM1234"). ***REQUIRED***
timestamp	ISO 8601 timestamp (e.g., "2025-03-01T14:25:00").
temperature	Temperature in Celsius (float, range: -30.0 to 60.0).
humidity	Humidity percentage (integer, 0–100).
status	Operational status as emoji: 🟢 (ON), 🔴 (OFF), ⚠️ (ERROR).
location	Device location, max 30 characters.
alert_level	One of: "LOW", "MEDIUM", "HIGH", "CRITICAL".
battery	Battery level (integer, 0–100). ***REQUIRED***
firmware_ver	Version string in format vX.Y.Z (e.g., v1.0.2 and X,Y,Z are digits). Must follow valid versioning. Range must be defined explicitly.
event_code	Integer (0–255), used for logging internal system events.

DEUCENG
Dokuz Eylül University
Dept of Computer Engineering

Tool Requirements

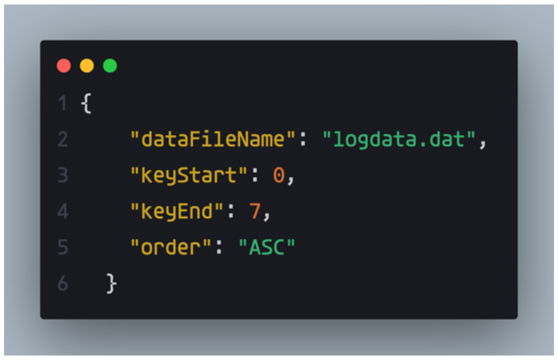
PART 1: CSV to Binary Conversion

Read the ***smartlog.csv*** file. Convert each record into a binary format and save it as 'logdata.dat'. Support variations in CSV delimiters and ensure compatibility with different operating system line endings (Windows \r\n, Linux \n, Mac \r).

PART 2: Binary to XML

Read configuration from ***setupParams.json*** using ***libjson*** library and generate an XML file using ***lib2xml*** library.

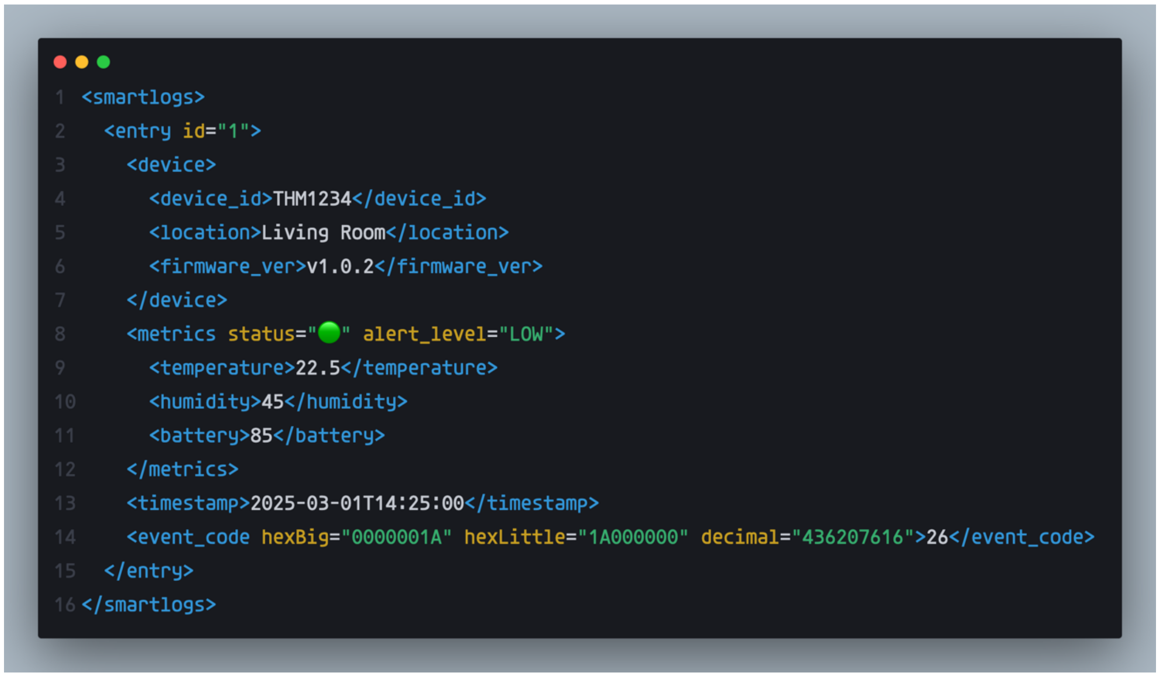
Example:



```
{
    "dataFileName": "logdata.dat",
    "keyStart": 0,
    "keyEnd": 7,
    "order": "ASC"
  }
```

- Sort records can be based on key field between the offsets [keyStart, keyEnd].
- Order can be either “ASC” or “DESC” from **setupParams.json** file and both must be supported.
- XML file ***must*** be generated from the binary file only. (Not directly from CSV)

Example output XML format:

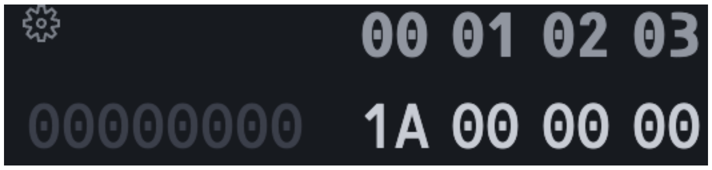


```
<smartlogs>
  <entry id="1">
    <device>
      <device_id>THM1234</device_id>
      <location>Living Room</location>
      <firmware_ver>v1.0.2</firmware_ver>
    </device>
    <metrics status="🟢" alert_level="LOW">
      <temperature>22.5</temperature>
      <humidity>45</humidity>
      <battery>85</battery>
    </metrics>
    <timestamp>2025-03-01T14:25:00</timestamp>
    <event_code hexBig="0000001A" hexLittle="1A000000" decimal="436207616">26</event_code>
  </entry>
</smartlogs>
```

Please pay attention that the root element of the output XML file is the name of the output file. For each line that was read from the file, an entry number is assigned as the "id" attribute starting from 1 and its value is increased by 1. The tags of the subelements under "entry" in the XML file are found at the header of the "***records.csv***" file.

The field "***event_code***" is an element comprised of a value and three attributes. The content of this element is a rating value. The first attribute is the corresponding hexadecimal value of the given number in the Big Endian format, second attribute is the corresponding hexadecimal value of the given number in the Little Endian format. The last attribute is the converted value of the number in the attribute *hexLittle* to its corresponding decimal value.

To clarify it, in the below scenario, hex editor is little endian. Considering the system is 64 bit, an integer value is placed in the first 4 bytes of a binary file. According to the hexadecimal value, the number is "26" in decimal format. Because, the least significant byte is read first.



```
          00 01 02 03
00000000  1A 00 00 00
```

If the above hex editor was big endian, the ordering of the reading would change and the most significant byte would be read first. In this case, the value would be read as "436207616" in decimal format.

PART 3: XSD Validation

Write an XSD schema file to validate structure and constraints, including:

- Required fields (specified at the first table)
- Patterns for specific formats (device_id, firmware_ver)
- Value restrictions (enum lists, numeric ranges)
- You will be provided with validate.c to integrate to your code

Command Line Usage:

```
# Usage:
./deviceTool <input_file> <output_file> <conversion_type> -separator <1|2|3> -opsys <1|2|3> [-h]

# Arguments:
< input_file >      = Source file to read from
< output_file >     = Target file to write to (or the XSD file for validation)
< conversion_type >:
  1 → CSV to Binary
  2 → Binary to XML (reads binary file name from setupParams.json)
  3 → XML validation with XSD
-separator <P1>     = Required field separator (1=comma, 2=tab, 3=semicolon)
-opsys <P2>         = Required line ending type (1=windows, 2=linux, 3=macos)

# Optional Flags:
-h                  Display help message and exit

# Examples:
./deviceTool smartlog.csv logdata.dat 1 -separator 1 -opsys 2
./deviceTool smartlogs.xml 2 -separator 1 -opsys 2
./deviceTool smartlogs.xml smartlogs.xsd 3 -separator 1 -opsys 2
./deviceTool -h
```

DEUCENG
Dokuz Eylül University
Dept of Computer Engineering

Submission Guidelines

- Name your C file as <your_id>.c and XSD file as <your_id>.xsd
- Ensure readable, documented, and well-structured code
- Late submissions will not be accepted
- If you fail to follow the naming conventions you will be deducted 10 pts

Honesty

Your submissions will be scanned among each other as well as the Internet repository (including ChatGPT). Any assignments that are over the similarity threshold of a system for Detecting Software Similarity will get zero. We strongly encourage you not to submit your assignment rather than a dishonest submission.

Grading Policy

Component	Weight
CSV file reading	20%
JSON parsing	10%
Binary file creation	10%
Binary file reading	10%
XML file creation	15%
XSD file preparation	15%
Little Endian - Big Endian conversion	10%
Command Line Usage	10%

For Questions

For any questions about the assignment, please post under the topic "Assignment1 Questions" in the Forum on the SAKAI platform. Check previous questions before posting. No private email questions will be answered.

Good Luck!