```
# 以下为 Python 实现
def parseLine(line: str, delimiter=","):
    output_list = []
    if '"' not in line:
        output_list = line.split(delimiter)
    else:
        division_list = line.split(delimiter)
        quoting = False
        cache = ""
        for item in division_list:
            if quoting:  # 处在一个引文中，把字段拼起来
                if item[-1] == '"':
                    cache += item
                    cache = cache[1:len(cache) - 1]
                    cache = cache.replace('""', '"')
                    output_list.append(cache)
                    cache = ""
                    quoting = False
                else:
                    cache += item
                    cache += delimiter
            else:  # 不是引文状态：
                if len(item) == 0:
                    output_list.append("")
                elif item[0] == '"':  # 如果第一项是引号，说明开始了一个引文
                    if item[-1] == '"':
                        item = item[1:len(item) - 1]  #去掉开头和结尾的引号
                        item = item.replace('""', '"')
                        output_list.append(item)
                    else:
                        quoting = True
                        cache += item
                        cache += delimiter
                else:
                    output_list.append(item)
    return output_list
def parseCSV(csv_text: str, delimiter: str = ","):
    csv_text.replace("\r\n", "\n")
    table: list = csv_text.split("\n")
    for i in range(len(table)):
        table[i] = parseLine(table[i], delimiter)
    return table
def fixLineLength(csv_sheet):
    max_length = 0
    for row in csv_sheet:
        if len(row) > max_length:
            max_length = len(row)
    for i in range(len(csv_sheet)):
        row: list = csv_sheet[i]
        while len(row) < max_length:
            row.append("")
            csv_sheet[i] = row
    return csv_sheet
```

```
def writeCSV(sheet, output_path="output.csv", delimiter=",", sheet_encoding="UTF-8",
line_break="\n"):
    output = open(output_path, "w", encoding=sheet_encoding)
    for row in sheet:
        for i in range(len(row)):
            col = str(row[i])
            if '"' in col:  # 判断是否引文
                output.write('"')
                output.write(col.replace('"', '""'))
                output.write('"')
            elif "," in col or "\n" in col or delimiter in col:
                output.write('"')
                output.write(col)
                output.write('"')
            else:
                output.write(col)
            if i != len(row) - 1:
                output.write(delimiter)
        output.write(line_break)
    output.close()
def fixCSV(path, output_path="output.csv", origin_delimiter=",", target_delimiter=",",
origin_encoding="UTF-8",
           target_encoding="UTF-8", target_line_break="\n", fix_length=True):
    my_file = open(path, encoding=origin_encoding)
    csv_ext = my_file.read()
    table = parseCSV(csv_ext, delimiter=origin_delimiter)
    if fix_length:
        table = fixLineLength(table)
    writeCSV(table, output_path, target_delimiter, target_encoding, target_line_break)
// 以下为 C++实现
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
#include <fstream>
std::vector<std::string> parseLine(const std::string& line, const std::string& delimiter
= ",") {
    std::vector<std::string> output_list;
    bool quoting = false;
    std::string cache;
    std::istringstream stream(line);
    std::string item;
    while (std::getline(stream, item, delimiter[0])) {
        if (quoting) {
            if (item.back() == '"') {
                cache += item.substr(0, item.size() - 1);
                size_t pos = cache.find("\"\"");
                if (pos != std::string::npos && pos < cache.size() - 1) {
                    cache.replace(pos, 2, "\"");
                }
                output_list.push_back(cache);
                cache.clear();
                quoting = false;
```

```
            } else {
                cache += item + delimiter;
            }
        } else {
            if (item.empty()) {
                output_list.push_back("");
            } else if (item.front() == '"' && item.back() == '"') {
                item = item.substr(1, item.size() - 2);
                item = item.replace(item.find("\"\""), 2, "\"");
                output_list.push_back(item);
            } else if (item.front() == '"') {
                quoting = true;
                cache = item;
            } else {
                output_list.push_back(item);
            }
        }
    }
    if (quoting) {
        output_list.push_back(cache);
    }
    return output_list;
}
std::vector<std::vector<std::string>> parseCSV(const std::string& csvText, const
std::string& delimiter = ",") {
    std::vector<std::vector<std::string>> data;
    std::istringstream stream(csvText);
    std::string line;
    while (std::getline(stream, line)) {
        data.push_back(parseLine(line, delimiter));
    }
    return data;
}
std::vector<std::vector<std::string>> fixLineLength(const
std::vector<std::vector<std::string>>& csv_sheet) {
    size_t maxLength = 0;
    for (const auto& row : csv_sheet) {
        maxLength = std::max(maxLength, row.size());
    }
    std::vector<std::vector<std::string>> fixed_csv_sheet = csv_sheet;
    for (auto& row : fixed_csv_sheet) {
        while (row.size() < maxLength) {
            row.push_back("");
        }
    }
    return fixed_csv_sheet;
}
std::string writeCSV(const std::vector<std::vector<std::string>>& table, const
std::string& delimiter = ",", const std::string& sheet_encoding = "UTF-8", const
std::string& line_break = "\n") {
    std::string csv_text;
    for (const auto& row : table) {
        for (size_t i = 0; i < row.size(); ++i) {
```

```
            const std::string& col = row[i];
            if (col.find('"') != std::string::npos || col.find(delimiter) !=
std::string::npos || col.find('\n') != std::string::npos) {
                csv_text += "\"";
                csv_text += col;
                csv_text += "\"";
            } else {
                csv_text += col;
            }
            if (i != row.size() - 1) {
                csv_text += delimiter;
            }
        }
        csv_text += line_break;
    }
    return csv_text;
}
void fixCSV(const std::string& path, const std::string& output_path = "output.csv",
            const std::string& origin_delimiter = ",", const std::string&
target_delimiter = ",",
            const std::string& origin_encoding = "UTF-8", const std::string&
target_encoding = "UTF-8",
            const std::string& target_line_break = "\n", bool fix_length = true) {
    std::ifstream my_file(path, std::ios::in | std::ios::binary);
    if (!my_file.is_open()) {
        std::cerr << "无法打开文件: " << path << std::endl;
        return;
    }
    std::stringstream buffer;
    buffer << my_file.rdbuf();
    std::string csv_ext = buffer.str();
    my_file.close();
    std::vector<std::vector<std::string>> table = parseCSV(csv_ext, origin_delimiter);
    if (fix_length) {
        table = fixLineLength(table);
    }
    std::ofstream output_file(output_path, std::ios::out | std::ios::binary);
    if (!output_file.is_open()) {
        std::cerr << "无法打开文件: " << output_path << std::endl;
        return;
    }
    std::string csv_text = writeCSV(table, target_delimiter, target_encoding,
target_line_break);
    output_file << csv_text;
    output_file.close();
}
// 以下为 JavaScript 实现
function parseLine(line, delimiter=","){
    let output_list = [];
    if (line.indexOf('"')===-1){
       output_list = line.split(delimiter);
    }else{
        let division_list = line.split(delimiter);
```

```
        let quoting = false;
        let cache = ""
        for (let item of division_list){
            if (quoting){
                if (item.slice(-1) ==='"'){
                cache += item;
                cache = cache.slice(1,- 1);
                cache = cache.replace('""', '"');
                output_list.push(cache);
                cache = ""
                quoting = false
                }else {
                cache += item
                cache += delimiter
                }
            }else {
             if (item.length === 0){
                output_list.push("")
             }else if (item[0]==='"'){
                if (item.slice(-1)==='"') {
                   item = item.slice(1,-1)
                   item = item.replace('""','"')
                   output_list.push(item)
                }else {
                   quoting = true
                   cache += item
                   cache += delimiter
                }
             }else {
                output_list.push(item)
             }
          }
        }
    }
    return output_list
}
function parseCSV(csvText,delimiter=","){
    let data = csvText.split(/\r\n|\n/);
    for (let i=0;i< data.length;i++) {
       data[i] = parseLine(data[i], delimiter);
    }
    return data
}
function fixLineLength(csv_sheet){
    let maxLength = 0;
    for (let row of csv_sheet){  //遍历每一行，确定最长的一行
       if (row.length > maxLength){
          maxLength = row.length;
       }
    }
    csv_sheet.map(function (row,index){
       while (row.length < maxLength){
          row.push("")
```

```
        }
        csv_sheet[index] = row
    })
    return csv_sheet
}
function writeCSV(table,delimiter=",", sheet_encoding="UTF-8",line_break = "\n"){
    let csv_text = ""
    for (let row of table) {
        for (let i = 0; i < row.length; i++){
            let col = row[i]
            if (col.includes('"')) {
                csv_text += '"';
                csv_text += col.replace('"','""');
                csv_text += '"';
            }else if (col.includes(",") || col.includes('\n') || col.includes(delimiter))
{
                csv_text += '"';  // 如果这一项里有逗号或是换行符，就给它加上双引号。
                csv_text += col;
                csv_text += '"';
            }else{
                csv_text += col;
            }
            if (i !== row.length -1) {
                csv_text += delimiter;
            }
        }
        csv_text += line_break; // 换行
    }
    return new Blob([csv_text], {type: `text/plain;charset=${sheet_encoding}`})
}
function fixCSV(csvFile, original_delimiter=",", target_delimiter = ",",
original_encoding="UTF-8",target_encoding="UTF-8",target_line_break="\n",fix_length=tr
ue){
    return new Promise((resolve,reject)=>{
        let reader = new FileReader();
        reader.onloadend = function(){
            let table = parseCSV(reader.result, original_delimiter);
            if (fix_length){
                table = fixLineLength(table);
            }
            let output_file = writeCSV(table,target_delimiter,original_encoding,)
            resolve(output_file);
        }
        reader.onerror = function () {
            reject("警告！无法读取指定的 CSV 文件！")
        }
        reader.readAsText(csvFile)
    })
}
```