

```
1 # 以下为 Python 实现
2 def parseLine(line: str, delimiter=","):
3     output_list = []
4     if '"' not in line:
5         output_list = line.split(delimiter)
6     else:
7         division_list = line.split(delimiter)
8         quoting = False
9         cache = ""
10        for item in division_list:
11            if quoting: # 处在一个引文中, 把字段拼起来
12                if item[-1] == '"':
13                    cache += item
14                    cache = cache[1:len(cache) - 1]
15                    cache = cache.replace('""', '"')
16                    output_list.append(cache)
17                    cache = ""
18                    quoting = False
19                else:
20                    cache += item
21                    cache += delimiter
22            else: # 不是引文状态:
23                if len(item) == 0:
24                    output_list.append("")
25                elif item[0] == '"': # 如果第一项是引号, 说明开始了一个引文
26                    if item[-1] == '"':
27                        item = item[1:len(item) - 1] # 去掉开头和结尾的引号
28                        item = item.replace('""', '"')
29                        output_list.append(item)
30                    else:
31                        quoting = True
32                        cache += item
33                        cache += delimiter
34                else:
35                    output_list.append(item)
36        return output_list
37 def parseCSV(csv_text: str, delimiter: str = ","):
38     csv_text.replace("\r\n", "\n")
39     table: list = csv_text.split("\n")
40     for i in range(len(table)):
41         table[i] = parseLine(table[i], delimiter)
42     return table
43 def fixLineLength(csv_sheet):
44     max_length = 0
45     for row in csv_sheet:
46         if len(row) > max_length:
47             max_length = len(row)
48     for i in range(len(csv_sheet)):
49         row: list = csv_sheet[i]
50         while len(row) < max_length:
51             row.append("")
52         csv_sheet[i] = row
53     return csv_sheet
```

```
1 def writeCSV(sheet, output_path="output.csv", delimiter=",", sheet_encoding="UTF-8",
2 line_break="\n"):
3     output = open(output_path, "w", encoding=sheet_encoding)
4     for row in sheet:
5         for i in range(len(row)):
6             col = str(row[i])
7             if '"' in col: # 判断是否引文
8                 output.write('"')
9                 output.write(col.replace('"', '""'))
10                output.write('"')
11            elif "," in col or "\n" in col or delimiter in col:
12                output.write('"')
13                output.write(col)
14                output.write('"')
15            else:
16                output.write(col)
17                if i != len(row) - 1:
18                    output.write(delimiter)
19            output.write(line_break)
20    output.close()
21 def fixCSV(path, output_path="output.csv", origin_delimiter=",", target_delimiter=",",
22 origin_encoding="UTF-8",
23 target_encoding="UTF-8", target_line_break="\n", fix_length=True):
24     my_file = open(path, encoding=origin_encoding)
25     csv_ext = my_file.read()
26     table = parseCSV(csv_ext, delimiter=origin_delimiter)
27     if fix_length:
28         table = fixLineLength(table)
29     writeCSV(table, output_path, target_delimiter, target_encoding, target_line_break)
30 // 以下为 C++ 实现
31 #include <iostream>
32 #include <sstream>
33 #include <vector>
34 #include <string>
35 #include <fstream>
36 std::vector<std::string> parseLine(const std::string& line, const std::string& delimiter
37 = ",") {
38     std::vector<std::string> output_list;
39     bool quoting = false;
40     std::string cache;
41     std::istringstream stream(line);
42     std::string item;
43     while (std::getline(stream, item, delimiter[0])) {
44         if (quoting) {
45             if (item.back() == '"') {
46                 cache += item.substr(0, item.size() - 1);
47                 size_t pos = cache.find("\\\"\\");
48                 if (pos != std::string::npos && pos < cache.size() - 1) {
49                     cache.replace(pos, 2, "\\");
50                 }
51                 output_list.push_back(cache);
52                 cache.clear();
53                 quoting = false;
```

```
1         } else {
2             cache += item + delimiter;
3         }
4     } else {
5         if (item.empty()) {
6             output_list.push_back("");
7         } else if (item.front() == '"' && item.back() == '"') {
8             item = item.substr(1, item.size() - 2);
9             item = item.replace(item.find("\\\\"), 2, "\\");
10            output_list.push_back(item);
11        } else if (item.front() == '"') {
12            quoting = true;
13            cache = item;
14        } else {
15            output_list.push_back(item);
16        }
17    }
18 }
19 if (quoting) {
20     output_list.push_back(cache);
21 }
22 return output_list;
23 }
24 std::vector<std::vector<std::string>> parseCSV(const std::string& csvText, const
25 std::string& delimiter = ",") {
26     std::vector<std::vector<std::string>> data;
27     std::istringstream stream(csvText);
28     std::string line;
29     while (std::getline(stream, line)) {
30         data.push_back(parseLine(line, delimiter));
31     }
32     return data;
33 }
34 std::vector<std::vector<std::string>> fixLineLength(const
35 std::vector<std::vector<std::string>>& csv_sheet) {
36     size_t maxLength = 0;
37     for (const auto& row : csv_sheet) {
38         maxLength = std::max(maxLength, row.size());
39     }
40     std::vector<std::vector<std::string>> fixed_csv_sheet = csv_sheet;
41     for (auto& row : fixed_csv_sheet) {
42         while (row.size() < maxLength) {
43             row.push_back("");
44         }
45     }
46     return fixed_csv_sheet;
47 }
48 std::string writeCSV(const std::vector<std::vector<std::string>>& table, const
49 std::string& delimiter = ",", const std::string& sheet_encoding = "UTF-8", const
50 std::string& line_break = "\n") {
51     std::string csv_text;
52     for (const auto& row : table) {
53         for (size_t i = 0; i < row.size(); ++i) {
```

```

1      const std::string& col = row[i];
2      if (col.find('"') != std::string::npos || col.find(delimiter) !=
3  std::string::npos || col.find('\n') != std::string::npos) {
4          csv_text += "\"";
5          csv_text += col;
6          csv_text += "\"";
7      } else {
8          csv_text += col;
9      }
10     if (i != row.size() - 1) {
11         csv_text += delimiter;
12     }
13 }
14 csv_text += line_break;
15 }
16 return csv_text;
17 }
18 void fixCSV(const std::string& path, const std::string& output_path = "output.csv",
19     const std::string& origin_delimiter = ",", const std::string&
20 target_delimiter = ",",
21     const std::string& origin_encoding = "UTF-8", const std::string&
22 target_encoding = "UTF-8",
23     const std::string& target_line_break = "\n", bool fix_length = true) {
24     std::ifstream my_file(path, std::ios::in | std::ios::binary);
25     if (!my_file.is_open()) {
26         std::cerr << "无法打开文件: " << path << std::endl;
27         return;
28     }
29     std::stringstream buffer;
30     buffer << my_file.rdbuf();
31     std::string csv_ext = buffer.str();
32     my_file.close();
33     std::vector<std::vector<std::string>> table = parseCSV(csv_ext, origin_delimiter);
34     if (fix_length) {
35         table = fixLineLength(table);
36     }
37     std::ofstream output_file(output_path, std::ios::out | std::ios::binary);
38     if (!output_file.is_open()) {
39         std::cerr << "无法打开文件: " << output_path << std::endl;
40         return;
41     }
42     std::string csv_text = writeCSV(table, target_delimiter, target_encoding,
43 target_line_break);
44     output_file << csv_text;
45     output_file.close();
46 }
47 // 以下为 JavaScript 实现
48 function parseLine(line, delimiter=",") {
49     let output_list = [];
50     if (line.indexOf('"')===-1) {
51         output_list = line.split(delimiter);
52     }else{
53         let division_list = line.split(delimiter);

```

```
1      let quoting = false;
2      let cache = ""
3      for (let item of division_list){
4          if (quoting){
5              if (item.slice(-1) === '"'){
6                  cache += item;
7                  cache = cache.slice(1, -1);
8                  cache = cache.replace('""', '');
9                  output_list.push(cache);
10                 cache = ""
11                 quoting = false
12             }else {
13                 cache += item
14                 cache += delimiter
15             }
16         }else {
17             if (item.length === 0){
18                 output_list.push("")
19             }else if (item[0] === '"'){
20                 if (item.slice(-1) === '"') {
21                     item = item.slice(1, -1)
22                     item = item.replace('""', '')
23                     output_list.push(item)
24                 }else {
25                     quoting = true
26                     cache += item
27                     cache += delimiter
28                 }
29             }else {
30                 output_list.push(item)
31             }
32         }
33     }
34 }
35 return output_list
36 }
37 function parseCSV(csvText, delimiter=",") {
38     let data = csvText.split(/\r\n|\n/);
39     for (let i=0; i< data.length; i++) {
40         data[i] = parseLine(data[i], delimiter);
41     }
42     return data
43 }
44 function fixLineLength(csv_sheet) {
45     let maxLength = 0;
46     for (let row of csv_sheet){ //遍历每一行，确定最长的一行
47         if (row.length > maxLength){
48             maxLength = row.length;
49         }
50     }
51     csv_sheet.map(function (row, index){
52         while (row.length < maxLength){
53             row.push("")
```

```
1      }
2      csv_sheet[index] = row
3  })
4  return csv_sheet
5  }
6  function writeCSV(table, delimiter=",", sheet_encoding="UTF-8", line_break = "\n") {
7      let csv_text = ""
8      for (let row of table) {
9          for (let i = 0; i < row.length; i++) {
10             let col = row[i]
11             if (col.includes('"')) {
12                 csv_text += '"';
13                 csv_text += col.replace('"', '""');
14                 csv_text += '"';
15             } else if (col.includes(",") || col.includes('\n') || col.includes(delimiter))
16 {
17                 csv_text += '"'; // 如果这一项里有逗号或是换行符，就给它加上双引号。
18                 csv_text += col;
19                 csv_text += '"';
20             } else {
21                 csv_text += col;
22             }
23             if (i !== row.length - 1) {
24                 csv_text += delimiter;
25             }
26         }
27         csv_text += line_break; // 换行
28     }
29     return new Blob([csv_text], {type: `text/plain;charset=${sheet_encoding}`})
30 }
31 function fixCSV(csvFile, original_delimiter=",", target_delimiter = ",",
32 original_encoding="UTF-8", target_encoding="UTF-8", target_line_break="\n", fix_length=true) {
33     return new Promise((resolve, reject) => {
34         let reader = new FileReader();
35         reader.onloadend = function() {
36             let table = parseCSV(reader.result, original_delimiter);
37             if (fix_length) {
38                 table = fixLineLength(table);
39             }
40             let output_file = writeCSV(table, target_delimiter, original_encoding, )
41             resolve(output_file);
42         }
43         reader.onerror = function () {
44             reject("警告！无法读取指定的 CSV 文件！")
45         }
46         reader.readAsText(csvFile)
47     })
48 }
49 }
```