1. 代码说明：这一段代码是预处理第二阶段，即已经读入数据入缓冲区中，接下来检索“<”符号，确定每个<符号的类型，以及所处的位置。变量定义见第2节注释。

2. 结构体定义：

1. **typedef** **enum** XMLEventLabel\_t  //定义“<”符号代表事件
2. {
3. STAG = 1,
4. ETAG = 2,
5. PI = 3,
6. COMMENT = 4,
7. CDATA = 5
8. }XMLEventLabel;
9. **typedef** **struct** XMLEvents\_t  //定义事件相对位置及长度
10. {
11. XMLEventLabel i\_label;
12. **char** \*p\_event\_start;
13. int64\_t i\_event\_length;
14. **char** \*event\_stream;
15. int64\_t i\_event\_stream\_length;
16. }XMLEvents;
17. **typedef** **struct** XMLDataSet\_t
18. {
19. **char** \*p\_start\_data\_set;
20. int64\_t i\_data\_set\_length;
21. XMLEvents \*events[MAX\_COUNT\_EVENTS]; //事件数组
22. int64\_t i\_events;
23. **char** \*XMLstream;
24. int64\_t XMLstream\_length;
25. }XMLDataSet;
27. **typedef** **struct** XMLSTagStack\_t
28. {
29. **int** dataset\_index;
30. **int** event\_index;
31. **struct** XMLSTagStack\_t \*next;
32. }XMLSTagStack;
33. **typedef** **struct** XMLParserContext\_t  //对xml文件维护一个上下文
34. {
35. **FILE** \*XMLfilein;   //xml源文件指针
36. **FILE** \*XMLstreamout;   //xml输出文件指针
37. **char** \*XMLbuf;   //xml全文缓冲区
38. int64\_t XMLlength;   //xml长度（字节）
39. int64\_t i\_count\_data\_sets;   //总事件数
40. XMLDataSet \*pp\_data\_sets[MAX\_COUNT\_DATA\_SETS]; //解析数据块
41. //multi-thread
42. int64\_t i\_parse;
43. int64\_t i\_post;
44. //int unresolved\_stag\_num;
45. XMLSTagStack\* unresolved\_stag\_stack\_head;
46. }XMLParserContext;

3. 模块代码：

函数定义：

1. **void** divide\_event\_edge(XMLDataSet \*dataset, **char** \*pdata, int64\_t length); //事件长度小于数据块大小
2. int64\_t divide\_event(XMLDataSet \*dataset, **char** \*pdata);  //事件长度大于数据块大小
3. **static**  **void** divide\_event\_stag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i) //划分开始tag
4. **static**  **void** divide\_event\_pi(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分pi命令tag
5. **static**  **void** divide\_event\_com\_cdata(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分cdata tag以及comment tag
6. **static**  **void** divide\_event\_etag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分endtag
7. **static**  **void** divide\_event\_edge\_stag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分开始tag
8. **static**  **void** divide\_event\_edge\_pi(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分pi命令tag
9. **static**  **void** divide\_event\_edge\_com\_cdata(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分cdata tag以及comment tag
10. **static**  **void** divide\_event\_edge\_etag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)  //划分endtag

其中代码如下：

1. #include "divide\_event.h"
2. **static**  **void** divide\_event\_edge\_etag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
3. {
4. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
5. memset(event, 0, **sizeof**(XMLEvents));
6. event->i\_label = ETAG;
7. event->p\_event\_start = pdata + \*i;
8. (\*i)++;
9. dataset->events[dataset->i\_events++] = event;
10. }
12. **static**  **void** divide\_event\_edge\_com\_cdata(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
13. {
14. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
15. memset(event, 0, **sizeof**(XMLEvents));
16. **if**(pdata[\*i + 2] == '-' && pdata[\*i + 3] == '-')
17. {
18. event->i\_label = COMMENT;
19. event->p\_event\_start = pdata + \*i;
20. **while**(pdata[\*i - 2] != '-' || pdata[\*i - 1] != '-' || pdata[\*i] != '>')
21. (\*i)++;
22. (\*i)++;
23. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
24. dataset->events[dataset->i\_events++] = event;
25. }
26. **else** **if**(pdata[\*i + 2] == '[' && pdata[\*i + 3] == 'C' && pdata[\*i + 4] == 'D' && pdata[\*i + 5] == 'A'
27. && pdata[\*i + 6] == 'T' && pdata[\*i + 7] == 'A' && pdata[\*i + 8] == '[')
28. {
29. event->i\_label = CDATA;
30. event->p\_event\_start = pdata + \*i;
31. **while**(pdata[\*i - 2] != ']' || pdata[\*i - 1] != ']' || pdata[\*i] != '>')
32. (\*i)++;
33. (\*i)++;
34. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
35. dataset->events[dataset->i\_events++] = event;
36. }
37. }
39. **static**  **void** divide\_event\_edge\_pi(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
40. {
41. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
42. memset(event, 0, **sizeof**(XMLEvents));
43. event->i\_label = PI;
44. event->p\_event\_start = pdata + \*i;
45. **while**(pdata[\*i - 1] != '?' || pdata[\*i] != '>')
46. (\*i)++;
47. (\*i)++;
48. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
49. dataset->events[dataset->i\_events++] = event;
50. }
52. **static**  **void** divide\_event\_edge\_stag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
53. {
54. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
55. memset(event, 0, **sizeof**(XMLEvents));
56. event->i\_label = STAG;
57. event->p\_event\_start = pdata + \*i;
58. (\*i)++;
59. dataset->events[dataset->i\_events++] = event;
60. }

63. **static**  **void** divide\_event\_etag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
64. {
65. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
66. memset(event, 0, **sizeof**(XMLEvents));
67. event->i\_label = ETAG;
68. event->p\_event\_start = pdata + \*i;
69. (\*i)++;
70. dataset->events[dataset->i\_events++] = event;
71. }
72. **static** **void** divide\_event\_com\_cdata(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
73. {
74. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
75. memset(event, 0, **sizeof**(XMLEvents));
76. **if**((\*i + 3) < DATA\_SET\_MAX && pdata[\*i + 2] == '-' && pdata[\*i + 3] == '-')
77. {
78. event->i\_label = COMMENT;
79. event->p\_event\_start = pdata + \*i;
80. **while** (pdata[\*i - 2] != '-' || pdata[\*i - 1] != '-' || pdata[\*i] != '>')
81. {
82. (\*i)++;
83. **if** ((\*i) == DATA\_SET\_MAX)
84. {
85. printf("event size is too large\n");
86. error\_state = 1;
87. **return**;
88. }
89. }
90. **if**((\*i) == DATA\_SET\_MAX)
91. {
92. dataset->events[dataset->i\_events++] = event;
93. **return**;
94. }
95. (\*i)++;
96. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
97. dataset->events[dataset->i\_events++] = event;
98. }
99. **else** **if**((\*i + 3) >= DATA\_SET\_MAX)//not longer enough
100. {
101. event->i\_label = COMMENT;
102. event->p\_event\_start = pdata + \*i;
103. dataset->events[dataset->i\_events++] = event;
104. (\*i) += 3;
105. }
106. **else** **if**((\*i + 8) < DATA\_SET\_MAX && pdata[\*i + 2] == '[' && pdata[\*i + 3] == 'C' && pdata[\*i + 4] == 'D' && pdata[\*i + 5] == 'A'
107. && pdata[\*i + 6] == 'T' && pdata[\*i + 7] == 'A' && pdata[\*i + 8] == '[')
108. {
109. event->i\_label = CDATA;
110. event->p\_event\_start = pdata + \*i;
111. **while** (pdata[\*i - 2] != ']' || pdata[\*i - 1] != ']' || pdata[\*i] != '>')
112. {
113. (\*i)++;
114. **if** ((\*i) == DATA\_SET\_MAX)
115. {
116. printf("event size is too large\n");
117. error\_state = 1;
118. **return**;
119. }
120. }
121. **if**((\*i) == DATA\_SET\_MAX)
122. {
123. dataset->events[dataset->i\_events++] = event;
124. **return**;
125. }
126. (\*i)++;
127. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
128. dataset->events[dataset->i\_events++] = event;
129. }
130. **else** **if**((\*i + 8) >= DATA\_SET\_MAX)//not longer enough
131. {
132. event->i\_label = CDATA;
133. event->p\_event\_start = pdata + \*i;
134. dataset->events[dataset->i\_events++] = event;
135. (\*i) += 8;
136. }
137. }
138. **static**  **void** divide\_event\_pi(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
139. {
140. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
141. memset(event, 0, **sizeof**(XMLEvents));
142. event->i\_label = PI;
143. event->p\_event\_start = pdata + \*i;
144. **while** ((\*i) < DATA\_SET\_MAX && (pdata[\*i - 1] != '?' || pdata[\*i] != '>'))
145. {
146. (\*i)++;
147. **if** ((\*i) == DATA\_SET\_MAX)
148. {
149. printf("event size is too large\n");
150. error\_state = 1;
151. **return**;
152. }
153. }
154. **if**((\*i) == DATA\_SET\_MAX)
155. {
156. dataset->events[dataset->i\_events++] = event;
157. **return**;
158. }
159. (\*i)++;
160. event->i\_event\_length = pdata + \*i - event->p\_event\_start;
161. dataset->events[dataset->i\_events++] = event;
162. }
163. **static**  **void** divide\_event\_stag(XMLDataSet \*dataset, **char** \*pdata, **int** \*i)
164. {
165. XMLEvents \*event = (XMLEvents \*)malloc(**sizeof**(XMLEvents));
166. memset(event, 0, **sizeof**(XMLEvents));
167. event->i\_label = STAG;
168. event->p\_event\_start = pdata + \*i;
169. (\*i)++;
170. dataset->events[dataset->i\_events++] = event;
171. }
173. **void** divide\_event\_edge(XMLDataSet \*dataset, **char** \*pdata, int64\_t length)
174. {
175. **int** i = 0;
176. dataset->p\_start\_data\_set = pdata;
177. dataset->i\_data\_set\_length = length;
178. **while**(i < length)
179. {
180. **if** (pdata[i] == '<')
181. {
182. **switch** (pdata[i + 1])
183. {
184. **case** '/':
185. divide\_event\_edge\_etag(dataset, pdata, &i);
186. **break**;
187. **case** '!':
188. divide\_event\_edge\_com\_cdata(dataset, pdata, &i);
189. **break**;
190. **case** '?':
191. divide\_event\_edge\_pi(dataset, pdata, &i);
192. **break**;
193. **default**:
194. divide\_event\_edge\_stag(dataset, pdata, &i);
195. }
196. }
197. **else**
198. i++;
199. **if** (error\_state == 1)
200. **return**;
201. }
202. }
204. int64\_t divide\_event(XMLDataSet \*dataset, **char** \*pdata)
205. {
206. **int** i = 0;
207. dataset->p\_start\_data\_set = pdata;
208. **while**(i < DATA\_SET\_MAX)
209. {
210. **if** (pdata[i] == '<')
211. {
212. **if** (i + 1 == DATA\_SET\_MAX)
213. {
214. dataset->i\_data\_set\_length = DATA\_SET\_MAX - 1;
215. pdata += DATA\_SET\_MAX - 1;
216. **return** dataset->i\_data\_set\_length;
217. }
218. **switch** (pdata[i + 1])
219. {
220. **case** '/':
221. divide\_event\_etag(dataset, pdata, &i);
222. **break**;
223. **case** '!':
224. divide\_event\_com\_cdata(dataset, pdata, &i);
225. **break**;
226. **case** '?':
227. divide\_event\_pi(dataset, pdata, &i);
228. **break**;
229. **default**:
230. divide\_event\_stag(dataset, pdata, &i);
231. }
232. }
233. **else**
234. i++;
235. **if** (error\_state == 1)
236. **return** 0;
237. }
238. dataset->i\_data\_set\_length = dataset->events[dataset->i\_events - 1]->p\_event\_start - pdata;
239. pdata = dataset->events[dataset->i\_events - 1]->p\_event\_start;
240. free(dataset->events[dataset->i\_events - 1]);//remount to the last TAG
241. dataset->events[--dataset->i\_events] = NULL;
243. **return** dataset->i\_data\_set\_length;
244. }