(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102262582 A
(43)申请公布日 2011.11.30

(21)申请号 201010183985.8

(22)申请日 2010.05.25

(71)申请人 芯讯通无线科技(上海)有限公司
地址 200335 上海市长宁区金钟路633号A楼

(72)发明人 杨雄伟

(74)专利代理机构 上海智信专利代理有限公司 31002
代理人 薛琦

(51)Int.Cl.
G06F 11/36(2006.01)

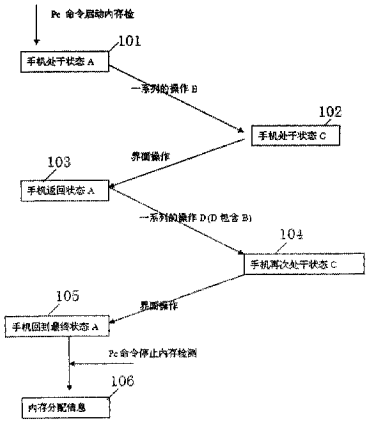
权利要求书 1 页 说明书 8 页 附图 3 页

(54)发明名称

移动终端及其内存泄露检测方法

(57)摘要

本发明公开了一种移动终端及其内存泄露检测方法，其包括：一命令解析模块，用于接收并且分析pc发送过来的数据然后执行对应的操作，对各模块进行测试；一内存记录模块，用于记录分配内存的地址、函数名和行数，及释放内存的时候删除该记录；一数据分析模块，用于打印并分析未被释放的内存信息。本发明内存泄露检测方法可以提高检测的速度，精确度。具有很强的可操作性，甚至可以作为移动终端的一个常驻模块，由测试人员来操作发现问题。采用本发明的方法，可以很全面的检测手机系统具体模块的内存泄漏，覆盖更多的测试路径，使内存泄露检查全面、快速。



CN 102262582 A

1. 一种移动终端，其特征在于，其包括：

一命令解析模块，用于接收并且分析 pc 发送过来的数据然后执行对应的操作，对各模块进行测试；

一内存记录模块，用于记录分配内存的地址、函数名和行数，及释放内存的时候删除该记录；

一数据分析模块，用于打印并分析未被释放的内存信息。

2. 如权利要求 1 所述的移动终端，其特征在于，打印信息中只出现一次的是属于被测试模块一次性分配的数据，不属于泄漏；打印信息中重复多次出现的数据，则为内存泄漏。

3. 如权利要求 1 所述的移动终端，其特征在于，该移动终端为移动通信终端或带界面操作的嵌入式设备。

4. 一种如权利要求 1 所述移动终端的内存泄露检测方法，其特征在于，其包括如下步骤：

S_1、使移动终端的被测试模块处于初始入口状态，在 PC 端发送启动内存检测命令，移动终端启动内存记录模块；S_1

S_2、手动操作进入移动终端的被检测模块，对该被检测模块进行各项功能操作，然后回到初始入口状态；

S_3、重复步骤 S_2，此时的操作包含步骤 S_2 中的操作，同时也可以进行更多的操作，再次返回初始入口状态；

S_4、在 PC 端发送停止内存检测命令，同时接收移动终端发送来的内存分配信息数据。

5. 如权利要求 4 所述的内存泄露检测方法，其特征在于，打印信息中只出现一次的是属于被测试模块一次性分配的数据，不属于泄漏；打印信息中重复多次出现的数据，则为内存泄漏。

6. 如权利要求 4 所述的内存泄露检测方法，其特征在于，步骤 S_2 中的对被测试模块的各项功能操作包括反复开启关闭该模块，使用该模块的某些功能以及并发运行其他模块。

7. 如权利要求 4 所述的内存泄露检测方法，其特征在于，在步骤 S_1-S_3 中初始入口状态保持一致，即移动终端的显示界面、开启的应用、硬件外设和工作状态均保持一致。

移动终端及其内存泄露检测方法

技术领域

[0001] 本发明涉及一种移动终端内存泄露检测方法，特别适用于手机等移动通信终端，及带界面操作的嵌入式设备。

背景技术

[0002] 现在的移动终端（如手机系统）已经非常庞大，用记录所有内存分配和释放记录来确定内存泄露将是非常艰难的，许多内存的分配本来就是一次性的，直到关机都不会被释放，通过记录所有内存的信息来定位内存泄露就显得效率低下，代价比较大，特别是有些内存泄露甚至是手机内部的交互事件引起的，传统的方法根本无法觉察到！

发明内容

[0003] 本发明要解决的技术问题是为了克服现有技术中移动终端内存泄露检测效率低下等缺陷，提供一种高效率和快速的移动终端及其内存泄露检测方法。

[0004] 本发明是通过下述技术方案来解决上述技术问题的：

[0005] 一种移动终端，其特点在于，其包括：

[0006] 一命令解析模块，用于接收并且分析 pc 发送过来的数据然后执行对应的操作，对各模块进行测试；

[0007] 一内存记录模块，用于记录分配内存的地址、函数名和行数，及释放内存的时候删除该记录；

[0008] 一数据分析模块，用于打印并分析未被释放的内存信息。

[0009] 较佳地，打印信息中只出现一次的是属于被测试模块一次性分配的数据，不属于泄漏；打印信息中重复多次出现的数据，则为内存泄漏。

[0010] 较佳地，该移动终端为移动通信终端或带界面操作的嵌入式设备。

[0011] 本发明的另一技术方案为：一种所述移动终端的内存泄露检测方法，其特点在于，其包括如下步骤：

[0012] S_1、使移动终端的被测试模块处于初始入口状态，在 PC 端发送启动内存检测命令，移动终端启动内存记录模块；

[0013] S_2、手动操作进入移动终端的被检测模块，对该被检测模块进行各项功能操作，然后回到初始入口状态；

[0014] S_3、重复步骤 S_2，此时的操作包含步骤 S_2 中的操作，同时也可以进行更多的操作，再次返回初始入口状态；

[0015] S_4、在 PC 端发送停止内存检测命令，同时接收移动终端发送来的内存分配信息数据。

[0016] 较佳地，打印信息中只出现一次的是属于被测试模块一次性分配的数据，不属于泄漏；打印信息中重复多次出现的数据，则为内存泄漏。

[0017] 较佳地，步骤 S_2 中的对被测试模块的各项功能操作包括反复开启关闭该模块，使

用该模块的某些功能以及并发运行其他模块。

[0018] 较佳地，在步骤 S_1-S_3 中初始入口状态保持一致，即移动终端的显示界面、开启的应用、硬件外设和工作状态均保持一致。

[0019] 本发明的积极进步效果在于：本发明内存泄露检测方法可以提高检测的速度，精确度。具有很强的可操作性，甚至可以作为移动终端的一个常驻模块，由测试人员来操作发现问题。采用本发明的方法，可以很全面的检测手机系统具体模块的内存泄漏，覆盖更多的测试路径，使内存泄露检查全面、快速。

附图说明

[0020] 图 1 为本发明的移动终端的模块图。

[0021] 图 2 为本发明的内存泄露检测方法的流程图。

[0022] 图 3 为本发明的手机与 pc 机数据交互流程图。

[0023] 图 4 为本发明的内存分配信息数据分析图。

具体实施方式

[0024] 下面结合附图给出本发明较佳实施例，以详细说明本发明的技术方案。

[0025] 如图 1-4 所示，本发明的移动终端，包括：一命令解析模块 1，用于接收并且分析 pc 发送过来的数据然后执行对应的操作，对各模块进行测试，一般的嵌入式设备都具备与 pc 交互数据的功能，本发明在其基础上扩展一个 pc 对手机的命令项即可；一内存记录模块 2，用于记录分配内存的地址、函数名和行数，及释放内存的时候删除该记录；一数据分析模块 3，用于打印并分析未被释放的内存信息。其中，该移动终端可以为移动通信终端或带界面操作的嵌入式设备。其中停止内存记录模块的时候，打印未被释放的内存信息。

[0026] 而本发明的移动终端的内存泄露检测方法，至少要包括如下步骤：

[0027] S_1、使移动终端的被测试模块处于初始入口状态，在 PC 端发送启动内存检测命令，移动终端启动内存记录模块；

[0028] S_2、手动操作进入移动终端的被检测模块，对该被检测模块进行各项功能操作，然后回到初始入口状态；

[0029] S_3、重复步骤 S_2，此时的操作包含步骤 S_2 中的操作，同时也可以进行更多的操作，再次返回初始入口状态；

[0030] S_4、在 PC 端发送停止内存检测命令，同时接收移动终端发送来的内存分配信息数据。

[0031] 本发明的总体思想为：首先确定手机初始状态，然后对手机进行一系列可以重复进行的操作，然后再退回到手机的初始状态，通过分析内存分配的增量信息即可以准确定位手机内存泄露！

[0032] 本实施例中，移动终端选用手机，被测试模块以蓝牙模块为例加以说明。

[0033] 如图 2，步骤 101，先使手机处于蓝牙入口界面（初始状态 A），通过数据线连接 pc 机与手机，pc 端发送启动内存记录的命令，手机端接收到命令启动内存记录，此时开始记录内存的分配状态；步骤 102，这时候可以进入蓝牙模块菜单进行操作，例如反复开启关闭蓝牙功能，搜索连接其他蓝牙设备，设置蓝牙可见状态，通过蓝牙收发文件等，可以把要测试

的模块的各种功能都运行一遍，交互事件也可以进行，例如蓝牙操作过程中拨打接听电话，播放音乐等，图中这一系列操作称为 B。步骤 103，然后返回到初始状态 A。之后步骤 104，再进入蓝牙模块进行各种操作，称作为 D，此时的操作应包含上一次的操作，也可以进行更多的操作，即数量上，D ＞＝ B。最后步骤 105，再返回蓝牙入口界面（状态 A）。步骤 106，再次通过 pc 端发送停止内存检测的命令，同时接收手机端发送过来的内存分配信息数据。上述步骤 101 ～ 106 在实际中的具体实现是本领域的现有技术，并非本发明的发明点所在。

[0034] 其中，手机初始状态与操作完成后返回的状态一致包含以下几个方面：

[0035] 1、手机显示界面处于一致；2、手机实际开启应用一致（例如，操作过程中开启了蓝牙模块，回到原界面状态的时候蓝牙模块也处于开启状态）；3、手机包含的硬件外设一致（例如开始的时候连接了蓝牙耳机，最后的状态也必须是连接蓝牙耳机）；4、工作状态也应保持一致（例如开始的时候有 mp3 背景播放，结束状态也应有 mp3 背景播放）。

[0036] 信息分析过程如图 4 所示，步骤 201，取内存分配记录；步骤 202，打印信息中只出现一次的是属于被测试模块一次性分配的数据，不属于泄漏；步骤 203，如果重复多次出现的数据，则可以定性为内存泄漏，进行重点分析。

[0037] 其中，以一种编程语言为例，内存记录模块的源代码如下：

[0038] // 初始化内存记录模块的数组

[0039] void mem_dbg init(void)；

[0040] // 分配内存的时候记录分配内存信息的函数名和行数

[0041] void mem_dbg_addr_record(void*pAddr, unsigned int bytes, char*fn, unsigned int ln)；

[0042] // 内存被释放的时候清除记录的信息

[0043] void mem_dbg_addr_clesr(void*pAddr)；

[0044] // 结束检测的是时候将信息 dump 到 pc 端

[0045] void mem_dbg_dump(void)；

[0046] // 用于 pc 端发送命令给手机侧的时候开始和结束监控

[0047] void mem_dbg_monitor(U32 bFlag)；

[0048] //----------------memory debug start-------------

[0049] typedef S8 signed char；

[0050] typedef S32 int；

[0051] typedef U8 unsigned char；

[0052] typedef U32 unsigned int；

[0053] typedef struct_MEM_DBG_TAG

[0054] {

[0055] S8 Record[72]；

[0056] U32 uMemAddr；

[0057] struct_MEM_DBG_TAG*pNext；

[0058] }T_MEM_DBG_CTRL；

[0059] #define MAX_DBG_CTRL 2400// 可根据需要调整大小

[0060] T_MEM_DBG_CTRL g_mem_dbg_ctrl[MAX_DBG_CTRL] ＝ {0}；

```
[0061]  T_MEM_DBG_CTRL*p_mem_dbg_active = NULL;
[0062]  T_MEM_DBG_CTRL*p_mem_dbg_free = NULL;
[0063]  U32 g_mem_dbg_init_flag = FALSE;
[0064]  /************************************************
[0065]   *mem_dbg_init-initial record group when start
[0066]   *DESCRIPTION:-
[0067]   *
[0068]   *Input:
[0069]   *Output:
[0070]   *Returns:
[0071]   *
[0072]   *modification history
[0073]   *--------------------
[0074]  ************************************************/
[0075]  void mem_dbg_init(void)
[0076]  {
[0077]      U32 uIndex = 0;
[0078]      for(uIndex = 0;uIndex < MAX_DBG_CTRL-1;uIndex++)
[0079]      {
[0080]  g_mem_dbg_ctrl[uIndex].pNext = &g_mem_dbg_ctrl[uIndex+1];
[0081]      }
[0082]      g_mem_dbg_ctrl[MAX_DBG_CTRL_1]·pNext = NULL;
[0083]      p_mem_dbg_free = &g_mem_dbg_ctrl[0];
[0084]      p_mem_dbg_active = NULL;
[0085]    }
[0086]  /************************************************
[0087]   *mem_dbg_addr_record-record the addr when malloc buffer
[0088]   *DESCRIPTION:-
[0089]   *
[0090]   *Input:
[0091]   *Output:
[0092]   *Returns:
[0093]   *
[0094]   *modification history
[0095]   *--------------------
[0096]  ************************************************/
[0097]  void mem_dbg_addr_record(void*pAddr,unsigned int bytes,char*fn,
unsigned int ln)
[0098]  {
```

```
[0099]      T_MEM_DBG_CTRL*pTemp ;
[0100]      if( ! g_mem_dbg_init_flag)
[0101]      {
[0102]          return ;
[0103]      }
[0104]      pTemp = p_mem_dbg_free ;
[0105]      if(pTemp == NULL)
[0106]      {
[0107]          return ;
[0108]      }
[0109]      pTemp-> uMemAddr = (unsigned int)pAddr ;
[0110]      sprintf(pTemp-> Record, " (%.*s, %d)sz :%d" ,28,fn,ln,bytes) ;
[0111]       //---to pointer next free space,and add new pointer
toactivelist------
[0112]      p_mem_dbg_free = p_mem_dbg_free-> pNext ;
[0113]      pTemp-> pNext = p_mem_dbg_active ;
[0114]      p_mem_dbg_active = pTemp ;
[0115]  }
[0116]  /************************************************
[0117]   *mem_dbg_addr_clear-to clear addr when free buf accord tothe addr
[0118]   *DESCRIPTION :-
[0119]   *
[0120]   *Input :free memory addr
[0121]   *Output :
[0122]   *Returns :
[0123]   *
[0124]   *modification history
[0125]   *--------------------
[0126]  ************************************************/
[0127]   void mem_dbg_addr_clear(void*pAddr)
[0128]   {
[0129]      T_MEM_DBG_CTRL*pActiveTemp = NULL ;
[0130]      T_MEM_DBG_CTRL*pPriorTemp = NULL ;
[0131]      U32 count = 0 ;
[0132]      if( ! g_mem_dbg_init_flag)
[0133]      {
[0134]          return ;
[0135]      }
[0136]      pActiveTemp = p_mem_dbg_active ;
```

```
[0137]      while(pActiveTemp)
[0138]      {
[0139]         if(pActiveTemp->uMemAddr == (unsigned int)pAddr)
[0140]   {//find the free pointer
[0141]      //-----del from active list-------
[0142]      if(NULL == pPriorTemp)
[0143]      {//the frist pointer
[0144]         p_mem_dbg_active = em_dbg_active->pNext;
[0145]      }
[0146]      else
[0147]      {
[0148]         pPriorTemp->pNext = pActiveTemp->pNext;
[0149]      }
[0150]      //----add to free pointer list----
[0151]      pActiveTemp->uMemAddr = 0;
[0152]      pActiveTemp->pNext = p_mem_dbg_free;
[0153]      p_mem_dbg_free = pActiveTemp;
[0154]      break;
[0155]   }
[0156]   else
[0157]   {
[0158]      pPriorTemp = pActiveTemp;
[0159]      pActiveTemp = pActiveTemp->pNext;
[0160]   }
[0161]   if(count++ > MAX_DBG_CTRL)
[0162]   {
[0163]      break;
[0164]   }
[0165]      }
[0166]      if(count > MAX_DBG_CTRL)
[0167]      {
[0168]        printf(" tp_os_mem_dbg_free:error" );
[0169]      }
[0170]   }
[0171]   /*************************************************************
[0172]    *mem_dbg_dump  -printf memory malloc info to pc
[0173]    *DESCRIPTION:-
[0174]    *
[0175]    *Input:
```

```
[0176]  *Output：
[0177]  *Returns：
[0178]  *
[0179]  *modification history
[0180]  *--------------------
[0181] ************************************************************/
[0182] void mem_dbg_dump(void)
[0183] {
[0184]   T_MEM_DBG_CTRL*pTemp = p_mem_dbg_active；
[0185]   while(pTemp)
[0186]   {
[0187]      printf(" leak info：% s" ,pTemp- > Record)；
[0188]      pTemp = pTemp- > pNext；
[0189]       }
[0190]  }
[0191]  /************************************************
[0192]   *mem_dbg_monitor-to start or stop monitor
[0193]   *DESCRIPTION:-
[0194] *to start at entry of an applicaton,stop at exit of an
[0195]   *application
[0196]   *Input：
[0197]   *Output：
[0198]   *Returns：
[0199]   *
[0200]   *modification history
[0201]   *--------------------
[0202]   ************************************************/
[0203]   void mem_dbg_monitor(U32 bFlag)
[0204]   {
[0205]       g_mem_dbg_init_flag = bFlag；
[0206]       if(g_mem_dbg_init_flag)
[0207]       {
[0208]          mem_dbg_init()；
[0209]     }
[0210]     else
[0211]     {
[0212]          mem_dbg dump()；
[0213]      }
[0214] }
```

[0215] //----------------memory debug end----------------

[0216] 虽然以上描述了本发明的具体实施方式，但是本领域的技术人员应当理解，这些仅是举例说明，本发明的保护范围是由所附权利要求书限定的。本领域的技术人员在不背离本发明的原理和实质的前提下，可以对这些实施方式做出多种变更或修改，但这些变更和修改均落入本发明的保护范围。

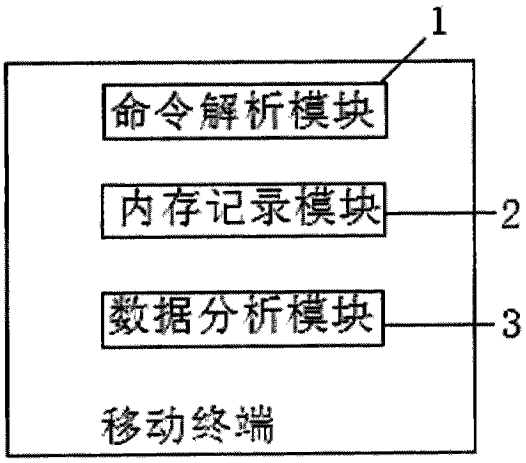


图 1

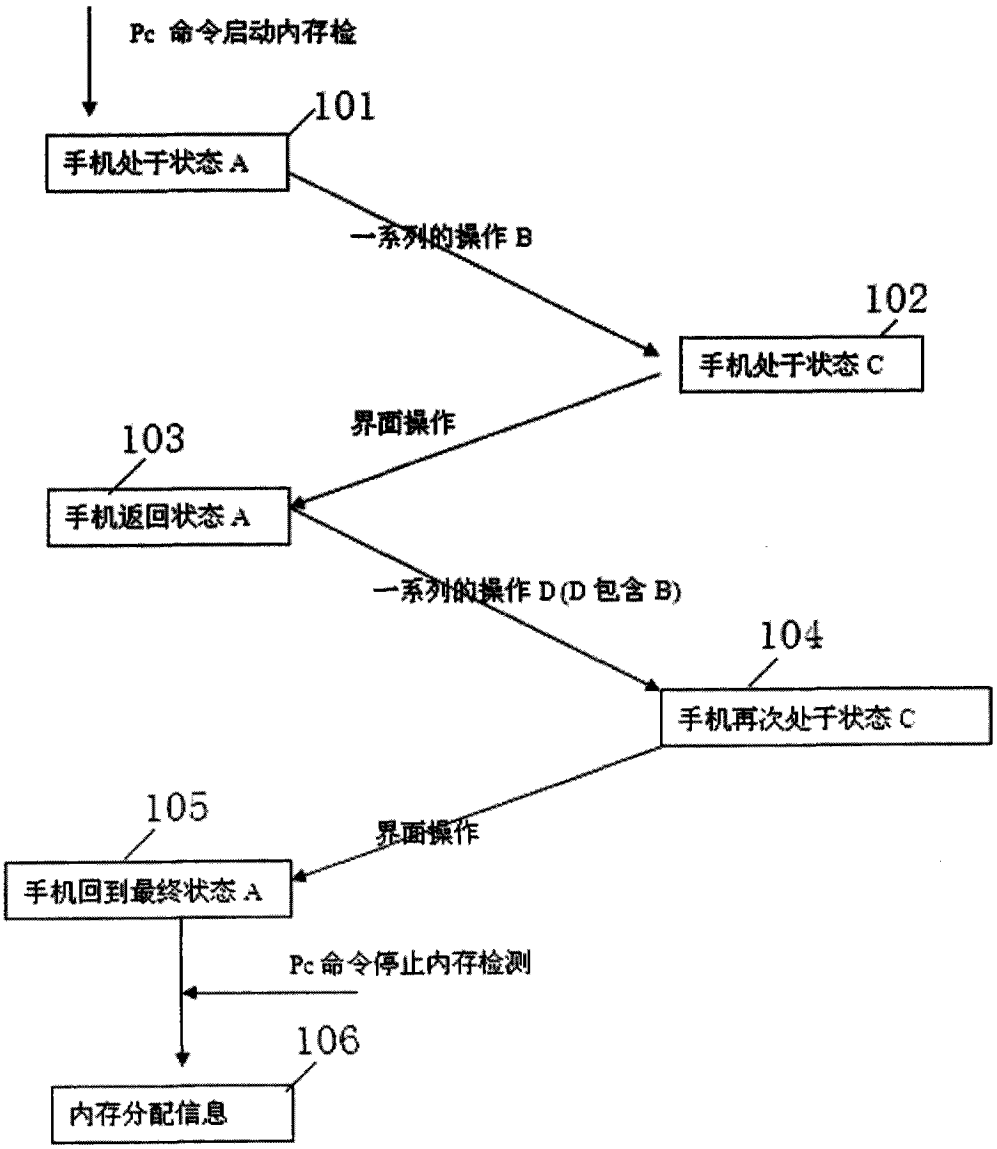

Pc 命令启动内存检
101
手机处于状态 A
一系列的操作 B
102
手机处于状态 C
界面操作
103
手机返回状态 A
一系列的操作 D(D 包含 B)
104
手机再次处于状态 C
105
界面操作
手机回到最终状态 A
Pc 命令停止内存检测
106
内存分配信息

图 2

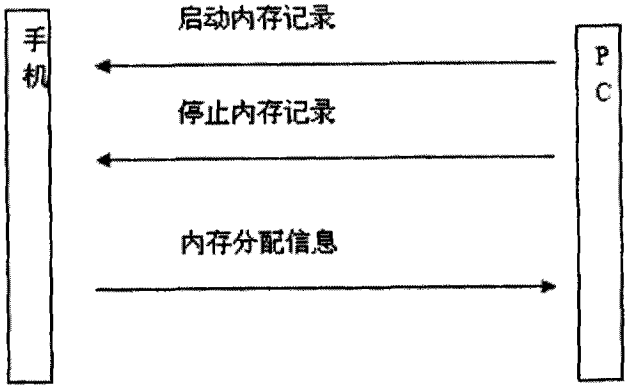

手机
PC
启动内存记录
停止内存记录
内存分配信息

图 3

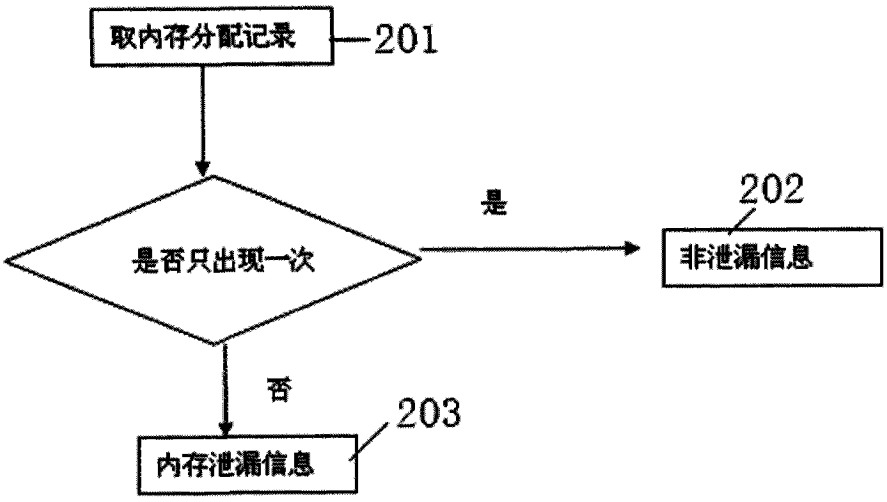


图 4