<https://blog.csdn.net/cnclenovo/article/details/38556133>

实现思路是：遍历/proc目录下的所有进程描述文件夹，从而获取进程列表。

代码如下：

**[cpp]** [view plain](https://blog.csdn.net/cnclenovo/article/details/38556133) [copy](https://blog.csdn.net/cnclenovo/article/details/38556133)

1. #include <stdio.h>
2. #include <dirent.h>
3. #include <unistd.h>
4. #include <stdlib.h>
5. #include <string.h>
7. **typedef** **struct**{
8. pid\_t pid;
9. **char** name[256];//进程名称
10. **int** vmsize;//虚拟内存信息
11. }proc\_info\_st;//保存读取的进程信息
13. #define PROC\_NAME\_LINE 1//名称所在行
14. #define PROC\_PID\_LINE 4//pid所在行
15. #define PROC\_VMSIZE\_LINE 12//虚拟内存所在行
16. #define BUFF\_LEN 1024 //行缓冲区的长度
18. **void** read\_proc(proc\_info\_st\* info,**const** **char**\* c\_pid);//读取进程信息
19. **int** read\_line(**FILE**\* fp,**char**\* buff,**int** b\_l,**int** l);//读取一行
21. **int** main()
22. {
23. //打开目录
24. DIR \*dir;
25. **struct** dirent \*ptr;
26. **if** (!(dir = opendir("/proc")))
27. **return** 0;
28. //读取目录
29. **while** (ptr = readdir(dir))
30. {//循环读取出所有的进程文件
31. **if** (ptr->d\_name[0] > '0' && ptr->d\_name[0] <= '9')
32. {
33. //获取进程信息
34. proc\_info\_st info;
35. read\_proc(&info,ptr->d\_name);//读取信息
36. printf("pid:%d\npname:%s\nvmsize:%d\n",info.pid,info.name,info.vmsize);
37. printf("\n\n");//再空两行
38. }
39. }
40. closedir(dir);
41. }

44. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
45. \*\*说明:根据进程pid获取进程信息,存放在proc\_info\_st结构体中
46. \*\*
47. \*\*输入:
48. \*\*        /proc\_info\_st\* info  返回进程信息
49. \*\*        /char\* c\_pid  进程pid的字符串形式
50. \*\*
51. \*\*
52. \*\*
53. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
54. **void** read\_proc(proc\_info\_st\* info,**const** **char**\* c\_pid)
55. {
56. **FILE**\* fp = NULL;
57. **char** file[512] = {0};
58. **char** line\_buff[BUFF\_LEN] = {0};//读取行的缓冲区
60. sprintf(file,"/proc/%s/status",c\_pid);//读取status文件
61. **if** (!(fp = fopen(file,"r")))
62. {
63. printf("read %s file fail!\n",file);
64. **return**;
65. }
67. **char** name[32];
68. //先读取进程名称
69. **if** (read\_line(fp,line\_buff,BUFF\_LEN,PROC\_NAME\_LINE))
70. {
71. sscanf(line\_buff,"%s %s",name,(info->name));
72. }
74. fseek(fp,0,SEEK\_SET);//回到文件头部
75. //读取进程pid
76. **if** (read\_line(fp,line\_buff,BUFF\_LEN,PROC\_PID\_LINE))
77. {
78. sscanf(line\_buff,"%s %d",name,&(info->pid));
79. }
81. fseek(fp,0,SEEK\_SET);//回到文件头部
82. //读取进程vmsize
83. **if** (read\_line(fp,line\_buff,BUFF\_LEN,PROC\_VMSIZE\_LINE))
84. {
85. sscanf(line\_buff,"%s %d",name,&(info->vmsize));
86. }

89. fclose(fp);
91. }

94. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
95. \*\*说明:读取文件的一行到buff
96. \*\*
97. \*\*输入:
98. \*\*     /FILE\* fp  文件指针
99. \*\*        /char\* buff  缓冲区
100. \*\*     /int b\_l  缓冲区的长度
101. \*\*        /l  指定行
102. \*\*
103. \*\*输出:
104. \*\*     /true 读取成功
105. \*\*     /false 读取失败
106. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
107. **int** read\_line(**FILE**\* fp,**char**\* buff,**int** b\_l,**int** l)
108. {
109. **if** (!fp)
110. **return** **false**;
112. **char** line\_buff[b\_l];
113. **int** i;
114. //读取指定行的前l-1行,转到指定行
115. **for** (i = 0; i < l-1; i++)
116. {
117. **if** (!fgets (line\_buff, **sizeof**(line\_buff), fp))
118. {
119. **return** **false**;
120. }
121. }
123. //读取指定行
124. **if** (!fgets (line\_buff, **sizeof**(line\_buff), fp))
125. {
126. **return** **false**;
127. }
129. memcpy(buff,line\_buff,b\_l);
131. **return** **true**;
132. }