# ProcessTree

枚举：

Signal：

*QUIT*(3), *KILL*(9), *TERM*(15);

方法：

1. **public** **static** **boolean** isAlive(String pid)
   1. 执行"kill", "-0", pid，如果没有抛出异常，且返回值为0，则表示存活
2. **public** **static** **void** killProcess(String pid, Signal signal)
   1. 发送指定的信号"kill", "-" + signal.getValue(), pid
3. **public** **static** **void** killProcessGroup(String pgrpId, Signal signal)
   1. 发送指定信号到group："kill", "-" + signal.getValue() , "-"+pgrpId

## ProcfsBasedProcessTree

继承自ProcessTree，使用/proc/下的文件

/\*\*

\* A Proc file-system based ProcessTree. Works only on Linux.

\*/

属性：

1. **private** Map<String, ProcessInfo> processTree = **new** HashMap<String, ProcessInfo>();
2. **private** String procfsDir;（即/proc/）

方法：

1. **public** ProcfsBasedProcessTree getProcessTree()
   1. 获取procfsDir下的所有进程文件夹processList（文件名为进程号）
   2. 把当前的processTree保存到oldProcs
   3. 遍历processList，对每个String proc：
      1. 新建ProcessInfo pInfo = **new** ProcessInfo(proc);
      2. 构建*constructProcessInfo*(pInfo, procfsDir)，如果构建成功：
         1. 添加到allProcessInfo.put(proc, pInfo);
         2. 如果proc.equals(**this**.pid)，则：
            1. 设置me = pInfo;
            2. 添加到processTree.put(proc, pInfo);
   4. 如果me == **null**，则直接**return** **this**
   5. 遍历allProcessInfo，把所有进程添加到其父进程（即pInfo.getPpid()），这一步构造了完整的进程树
   6. 添加pInfoQueue.addAll(me.getChildren());
   7. 遍历pInfoQueue，构建一颗完成的树存放到processTree（即以当前节点为根的子树，这里以广度优先遍历）
   8. 使用processTree和oldProcs更新jiffies
2. **private** **static** ProcessInfo constructProcessInfo(ProcessInfo pinfo, String procfsDir)
   1. 读取进程文件下的*PROCFS\_STAT\_FILE*文件（即stat文件）
   2. 从文件读取信息：(name) (ppid) (pgrpId) (session) (utime) (stime) (vsize) (rss)，填充到ProcessInfo

### ProcessInfo

属性：

1. **private** String pid; // process-id
2. **private** String name; // command name
3. **private** Integer pgrpId; // process group-id
4. **private** String ppid; // parent process-id
5. **private** Integer sessionId; // session-id
6. **private** Long vmem; // virtual memory usage
7. **private** Long rssmemPage; // rss memory usage in # of pages
8. **private** Long utime = 0L; // # of jiffies in user mode
9. **private** **final** BigInteger MAX\_LONG = BigInteger.*valueOf*(Long.*MAX\_VALUE*);
10. **private** BigInteger stime = **new** BigInteger("0");
11. **private** **int** age;
12. **private** Long dtime = 0L;
13. **private** List<ProcessInfo> children = **new** ArrayList<ProcessInfo>();