

进程切换次数观测实验

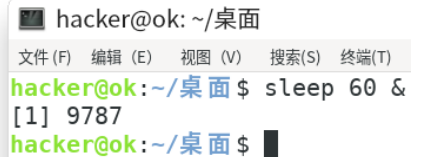
一、代码截图和运行结果截图

```
1 #include <linux/sched.h>
2
3 BEGIN
4 {
5     printf("使用: bpftrace ./switchcnt.bt [PID]\n");
6     printf("统计某进程的切换次数, 按 Ctrl-C 显示结果\n\n");
7     @time = nsecs;
8 }
9
10 kprobe:dequeue_task_fair
11 {
12     $target_pid = $1;
13     if (pid == $target_pid) {
14         printf("进程 %d 被调用\n", pid);
15         @count++;
16     }
17 }
18
19 END
20 {
21     printf("\n进程共被调用 %d 次\n", @count);
22     @time = (nsecs - @time) / 1000000000; // 纳秒转秒
23     printf("监控时间共 %d 秒\n", @time);
24 }
```

使用: bpftrace ./switchcnt.bt [PID]
统计某进程的切换次数, 按 Ctrl-C 显示结果

进程 9787 被调用
^C
进程共被调用 1 次
监控时间共 26 秒

@count: 1
@time: 26



```
hacker@ok: ~/桌面
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T)
hacker@ok:~/桌面$ sleep 60 &
[1] 9787
hacker@ok:~/桌面$
```

二、ebpf 程序执行的时机

1.BEGIN 块:

时机: 程序开始执行时。

作用: 输出提示信息并初始化 @time 变量为当前时间 (以纳秒为单位)。

2.kprobe

时机: 每次内核函数 dequeue_task_fair 被调用时。dequeue_task_fair 是一个与进程调度相关的内核函数。

作用: 检查当前被调度的进程是否是目标进程, 并记录其被调度的次数。

3.END 块

时机: 程序结束执行时 (通常是用户按下 Ctrl-C 中断程序时)。

作用: 输出目标进程的总被调度次数以及监控的总时间。

三、关键语句的含义

1.BEGIN 块: 在 eBPF 程序开始时执行。

(1) printf: 输出提示信息, 告知用户如何使用该程序。

(2) @time = nsecs: 获取当前时间的纳秒数, 并存储在 @time 变量中, 用于计算监控的总时间。

2.kprobe:dequeue_task_fair: 在内核函数 dequeue_task_fair 被调用时执行。

(1) \$target_pid = \$1: 将传入的 PID 参数存储在 \$target_pid 变量中。

(2) if (pid == \$target_pid): 检查当前被调度的进程 ID 是否与目标进程 ID 相同。

(3) printf: 如果匹配, 输出该进程 ID 被调度的信息。

(4) @count++: 计数器 @count 增加 1, 记录目标进程被调度的次数。

3.END 块: 在 eBPF 程序结束时执行。

- (1) printf: 输出目标进程被调度的总次数。
- (2) @time = (nsecs - @time) / 1000000000: 计算程序运行的总时间 (秒), 将纳秒转换为秒。
- (3) printf: 输出监控的总时间。