```
Attaching 1 probe...
^C

@:
[11, 12)              123 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
```

上面的输出显示出错的代码全部都是 11。根据 Linux 头文件内的定义（asm-generic/errno-base.h）：

```
#define EAGAIN          11      /* Try again */
```

错误值 11 的意思是“重试”，这个错误是在运行过程正常出现的错误。

6.3.12 funccount

第 4 章中介绍了 funccount(8)，它是一个可以统计事件和函数调用频率的 BCC 工具。这个工具可以显示函数调用的频率，可以用来调查系统中软件占用 CPU 的问题。profile(8) 可能可以显示哪个函数正在占用 CPU，但是不能解释为什么[1]：是因为这个函数执行过程很慢，还是因为这个函数每秒被调用了几百万次。

在下面的例子中，我们在一个繁忙的生产系统中统计了内核中以 tcp_ 开头的所有函数，也就是 TCP 相关函数的调用频率：

```
# funccount 'tcp_*'
Tracing 316 functions for "tcp_*"... Hit Ctrl-C to end.
^C
FUNC                                    COUNT
[...]
tcp_stream_memory_free                 368048
tcp_established_options                381234
tcp_v4_md5_lookup                      402945
tcp_gro_receive                        484571
tcp_md5_do_lookup                      510322
Detaching...
```

上面的输出显示，tcp_md5_do_lookup() 函数调用最频繁，跟踪过程中共计调用了 510 322 次。

1 profile(8)不能直接回答这个问题。profile(8)这类的性能分析器是通过定时采样CPU指令指针的方式工作的，如果有对应函数的反汇编字节，那么也许可以通过对比的方式分析该函数到底是调用次数很频繁，还是正在一个循环中。但是在实际中，这样做会很难，详情参见2.12.2节。