```
        e); // Ignore the return value; the event is modified in place
    }
    return events;
  }

  @Override
  public void close() {
    // No op
  }

  public static class CounterInterceptorBuilder
    implements Interceptor.Builder {

    private Context ctx;

    @Override
    public Interceptor build() {
      return new CounterInterceptor(ctx);
    }

    @Override
    public void configure(Context context) {
      this.ctx = context;
    }
  }
}
```

CounterInterceptor 类的拦截方法是线程安全的，因为实例访问的唯一变量要么是 final 类型（基础配置初始化的所有变量），要么是使用线程安全的类（作为计数器使用的 AtomicLong 实例）。处理事件列表的 intercept 方法循环调用处理单个事件的 intercept 方法。建议所有的定制拦截器都遵循这种模式。在这种情况下，由于事件恰好是在变换过程中，所以不创建新列表，仅仅返回包含修改后事件的原始列表。另外，也可以创建一个新的列表，如果需要，则添加新的事件到这个列表。事件可以丢弃，可以通过从返回的原始列表中删除，或者通过不添加事件到返回的新列表中实现。

150

单个拦截器可以返回多少事件？

拦截器不允许返回比原先传递给它的更多的事件，但它可以返回较少的事件。这背后的逻辑是，拦截器添加更多的事件可能会导致写入 Channel 的事件超过吞吐量，即使 Avro Sink 发送数据到 Avro Source，每个批次发送的事件也比吞吐量小。如果拦截器丢弃传递给它的所有事件，拦截器仍必须返回一个列表，如果所有的事件都将被丢弃，该列表可能是空的。