



Handling stack traces

Better output

Deferred user space stack tracing

- Reads the user stack when going back to user space
- Safe to schedule and fault (read user space)
- Can get more than just user space addresses

Reading user space addresses

- Due to relocatable loading, address change every execution
- User address itself is useless
- At user entry, can read the VMA
 - Map address to the location of the executable file
 - Can use debug info (later) to get even line numbers

Reading VMAs are not cheap

- Getting the file name requires a lookup
- The file name can be very long
 - Wastes ring buffer memory
- Can use a mapping to map file to what is in the ring buffer
- One method is using inode numbers
 - But Linus hates inode numbers!

Using a “inode_cache” event

```
~# trace-cmd start -e inode_cache -O userstacktrace_delay -O userstacktrace -e sys_enter_read
~# trace-cmd show
[...]
```

sed-1262	[001]		1166.257049:	sys_read(fd: 3, buf: 0x7ffc23bbcb88, count: 0x340)
sed-1262	[001]		1166.257051:	cookie=3d100000002
sed-1262	[001]		1166.257151:	<user stack unwind>

```
cookie=3d100000002
=> <0000000000023464> : 1319244 : 254:3
=> <0000000000006450> : 1319244 : 254:3
=> <0000000000009c38> : 1319244 : 254:3
=> <0000000000003731> : 1319244 : 254:3
=> <00000000000023fe> : 1319244 : 254:3
=> <0000000000003a99> : 1319244 : 254:3
=> <000000000000204c2> : 1319244 : 254:3
=> <0000000000001d21f> : 1319244 : 254:3
=> <0000000000001e7e1> : 1319244 : 254:3
=> <0000000000001d788> : 1319244 : 254:3
      sed-1262    [001] ...1. 1166.257168: inode_cache: inode=1319244 dev=[254:3]
path=/usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
```

Discussion!

