Kernel debugging
"*tricks*" *wasn't my idea*

panic("Why am I talking?");

Problems?

panic("Why am I talking?");

What is the problem with this panic message?

Problem 1

panic("Why am I talking?");

Who am I?

Kernel debugging "*tricks*" *wasn't my idea*

Bjoern A. Zeeb <bz@FreeBSD.org>

Problem 1

panic("Why am I talking?");

Who is 'I'?

Problem 2

panic("Why am I talking?");

Where am I talking?

Kernel debugging "*tricks*" *wasn't my idea*

BSDCan 2012, FreeBSD Developer Summit Track

Bjoern A. Zeeb <bz@FreeBSD.org>

Problem 3

panic("Why am I talking?");

What am I talking about?
(What went wrong that I panicked?)

panic("Why am I talking?");

- To make my life easier.
- To make your life easier.
- Maybe finding who'd be interested to fix some things?

A real problem?

```
> grep panic tcp_timer.c
    panic("bad timer_type");
    panic("bad timer_type");
```

If you have a backtrace it will tell you the function.

Fix

If you have the same panic message multiple times, add the function name.

So what about this?

```
> grep 'hdr not' ip6_output.c
    panic("assumption failed: hdr not split"); \
    panic("assumption failed: hdr not split");
```

So what about this?

```
#define MAKE_CHAIN(m, mp, p, i)\
    do {\
        if (m) {\
            if (!hdrsplit) \
                panic("assumption failed: hdr not split"); \
            *mtod((m), u_char *) = *(p);\
....

        if (exthdrs.ip6e_dest2) {
            if (!hdrsplit)
                panic("assumption failed: hdr not split");
            exthdrs.ip6e_dest2->m_next = m->m_next;
            m->m_next = exthdrs.ip6e_dest2;
            *mtod(exthdrs.ip6e_dest2, u_char *) = ip6->ip6_nxt;
            ip6->ip6_nxt = IPPROTO_DSTOPTS;
        }

        /* ....
         */
        MAKE_CHAIN(exthdrs.ip6e_hbh, mprev, nexthdrp, IPPROTO_HOPOPTS);
         MAKE_CHAIN(exthdrs.ip6e_dest1, mprev, nexthdrp,
                IPPROTO_DSTOPTS);
         MAKE_CHAIN(exthdrs.ip6e_rthdr, mprev, nexthdrp,
                IPPROTO_ROUTING);
```

Its all ip6_output( ) but which one there? You need the offset.

Fix

If you have the same panic message multiple times, add the function name **and the line number.**

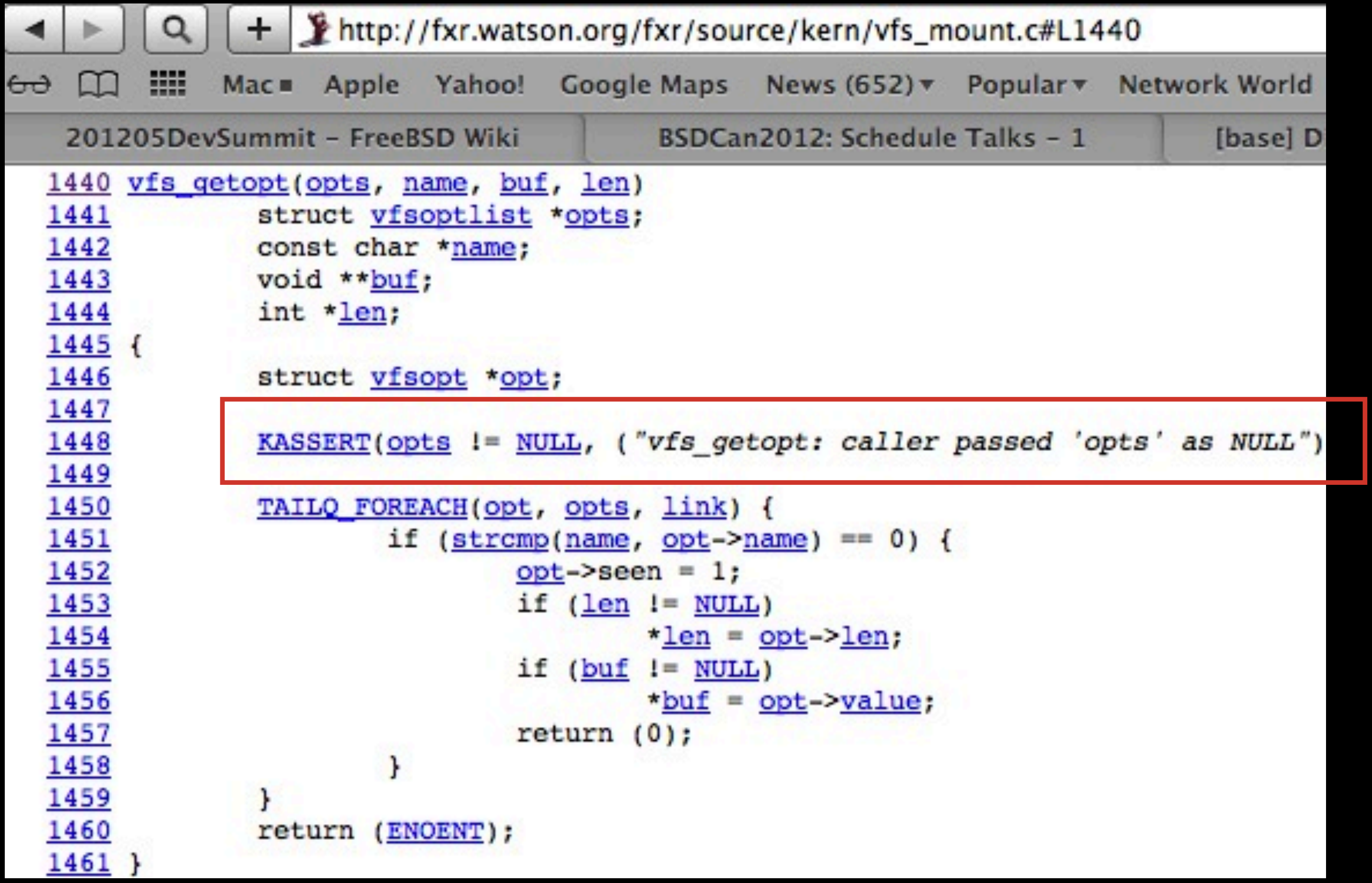
Really better?

```
KASSERT(opts != NULL,
("vfs_getopt: caller passed
'opts' as NULL"));
```

Really better?

Function name, only one panic, life is

KASSERT(opts != NULL,
("**vfs_getopt**: caller passed 'opts' as NULL"));



http://fxr.watson.org/fxr/source/kern/vfs_mount.c#L1440
Mac
Apple
Yahoo!
Google Maps
News (652)
Popular
Network World
201205DevSummit – FreeBSD Wiki
BSDCan2012: Schedule Talks – 1
[base] D

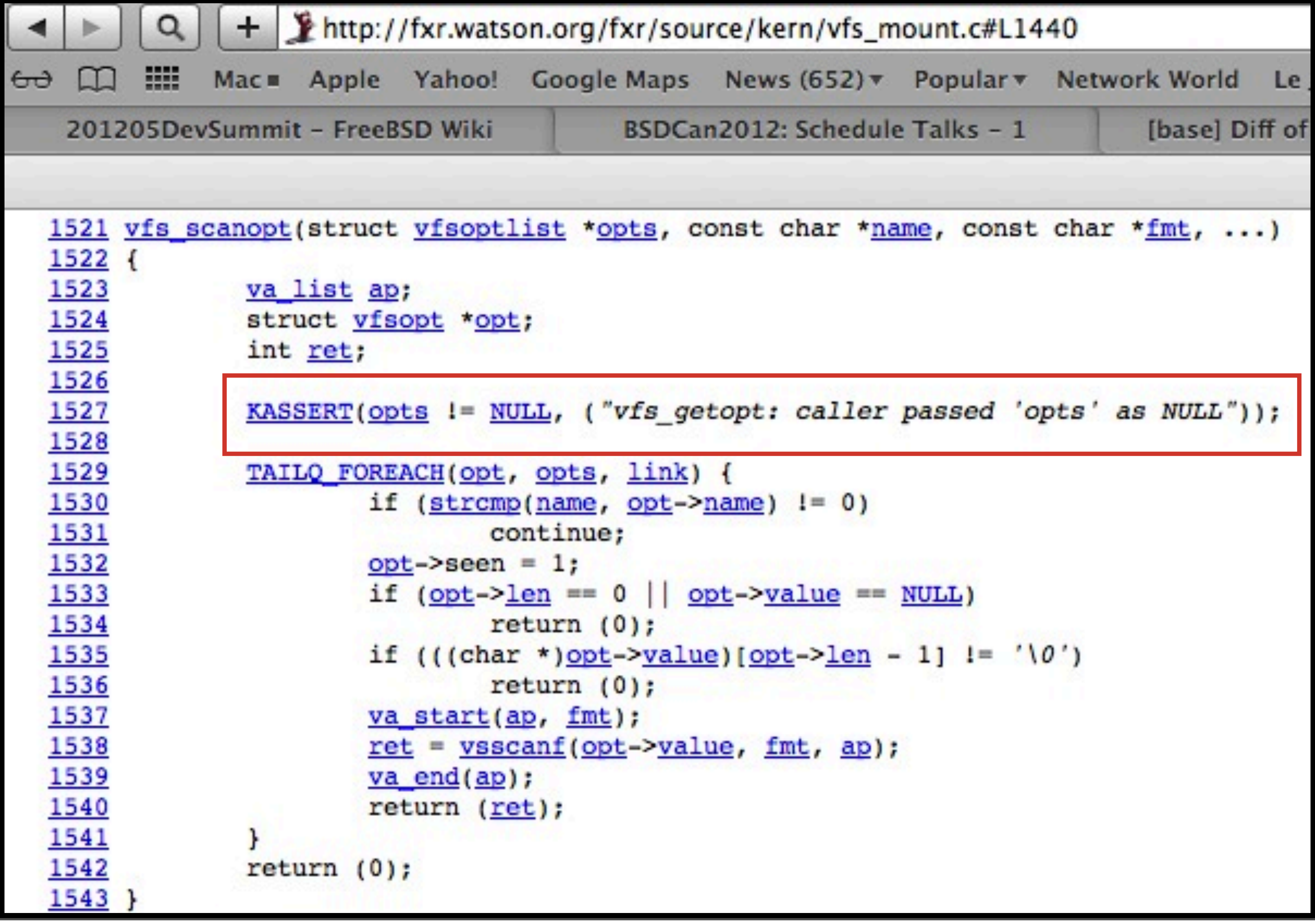
```
vfs_getopt(opts, name, buf, len)
        struct vfsoptlist *opts;
        const char *name;
        void **buf;
        int *len;
{
        struct vfsopt *opt;

        KASSERT(opts != NULL, ("vfs_getopt: caller passed 'opts' as NULL")

        TAILQ_FOREACH(opt, opts, link) {
                if (strcmp(name, opt->name) == 0) {
                        opt->seen = 1;
                        if (len != NULL)
                                *len = opt->len;
                        if (buf != NULL)
                                *buf = opt->value;
                        return (0);
                }
        }
        return (ENOENT);
}
```

Really better?

KASSERT(opts != NULL,
("**vfs_getopt**: caller passed 'opts' as NULL"));


http://fxr.watson.org/fxr/source/kern/vfs_mount.c#L1440
Mac
Apple
Yahoo!
Google Maps
News (652)
Popular
Network World
201205DevSummit – FreeBSD Wiki
BSDCan2012: Schedule Talks – 1
[base] Diff of

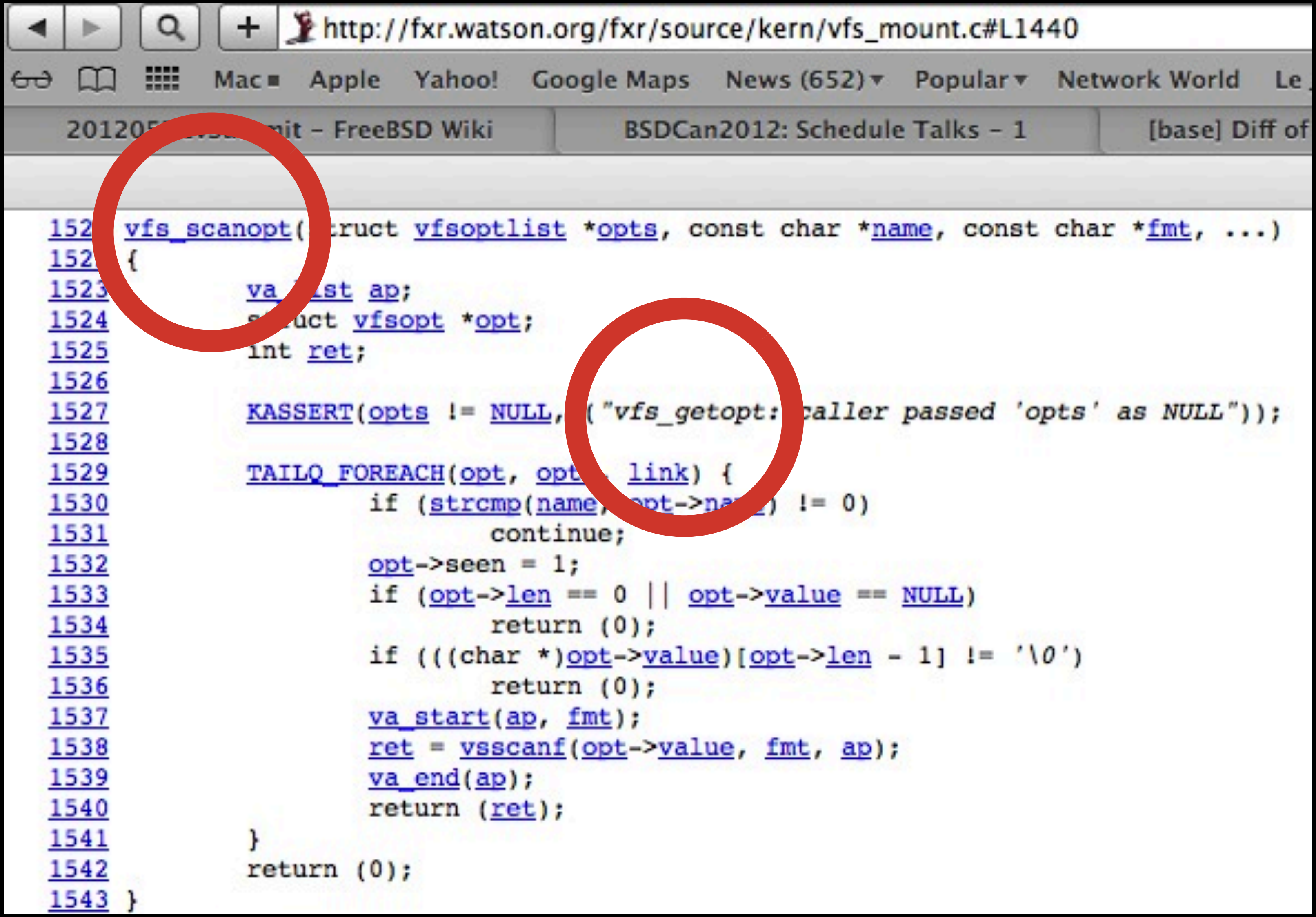
```
vfs_scanopt(struct vfsoptlist *opts, const char *name, const char *fmt, ...)
{
        va_list ap;
        struct vfsopt *opt;
        int ret;

        KASSERT(opts != NULL, ("vfs_getopt: caller passed 'opts' as NULL"));

        TAILQ_FOREACH(opt, opts, link) {
                if (strcmp(name, opt->name) != 0)
                        continue;
                opt->seen = 1;
                if (opt->len == 0 || opt->value == NULL)
                        return (0);
                if (((char *)opt->value)[opt->len - 1] != '\0')
                        return (0);
                va_start(ap, fmt);
                ret = vsscanf(opt->value, fmt, ap);
                va_end(ap);
                return (ret);
        }
        return (0);
}
```

Really better?

KASSERT(opts != NULL,
("**vfs_getopt**: caller passed 'opts' as NULL"));



```
vfs_scanopt(struct vfsoptlist *opts, const char *name, const char *fmt, ...)
{
        va_list ap;
        struct vfsopt *opt;
        int ret;

        KASSERT(opts != NULL, ("vfs_getopt: caller passed 'opts' as NULL"));

        TAILQ_FOREACH(opt, opts, link) {
                if (strcmp(name, opt->name) != 0)
                        continue;
                opt->seen = 1;
                if (opt->len == 0 || opt->value == NULL)
                        return (0);
                if (((char *)opt->value)[opt->len - 1] != '\0')
                        return (0);
                va_start(ap, fmt);
                ret = vsscanf(opt->value, fmt, ap);
                va_end(ap);
                return (ret);
        }
        return (0);
}
```

Fix

Use:

"%s", __func__

"%s:%d", __func__, __LINE__

(people who want to grep will still find the function by name and can identify the panic, but will not end up in the wrong place)

Some oddities

```
if (apic_id > MAX_APIC_ID) {
        panic("SMP: APIC ID %d too high", apic_id);
        return;
}
```

(If you know the reason I would love to learn.)

Some fun

How many teapots does it take to make a kernel?

Some fun

Luckily only one:

kern/kern_thread.c:
panic("I'm a teapot!");

Another problem

```
panic("%s", __func__);
```

Another problem

```
void
ip6_notify_pmtu(struct inpcb *in6p, ...)
{
        struct socket *so;
..
        so =  in6p->inp_socket;
 ...
#ifdef DIAGNOSTIC
        if (so == NULL) /* I believe this is impossible */
                panic("ip6_notify_pmtu: socket is NULL");
#endif
```

Another problem

```
void
ip6_notify_pmtu(struct inpcb *in6p, ...)
{
        struct socket *so;
..
        so =  in6p->inp_socket;
 ...
KASSERT(so != NULL,
    ("%s: socket is NULL, inp=%p", __func__, in6p);
```

debug it better...

```
DB_SHOW_COMMAND(inpcb, db_show_inpcb)
{
	struct inpcb *inp;

	if (!have_addr) {
		db_printf("usage: show inpcb <addr>\n");
		return;
	}
	inp = (struct inpcb *)addr;

	db_print_inpcb(inp, "inpcb", 0);
}
```

debug it better...

```
db_print_inpcb(struct inpcb *inp, const char *name, int indent)
{
        db_print_indent(indent);
        db_printf("%s at %p\n", name, inp);
        indent += 2;
        db_print_indent(indent);
        db_printf("inp_flow: 0x%x\n", inp->inp_flow);
        db_print_inconninfo(&inp->inp_inc, "inp_conninfo", indent);
        db_print_indent(indent);
        db_printf("inp_ppcb: %p   inp_pcbinfo: %p   inp_socket: %p\n",
            inp->inp_ppcb, inp->inp_pcbinfo, inp->inp_socket);

        db_print_indent(indent);
        db_printf("inp_label: %p   inp_flags: 0x%x (",
           inp->inp_label, inp->inp_flags);
        db_print_inpflags(inp->inp_flags);
        ....
```

Dtrace

- ... to the rescue?

The idea

- that has been around for a while now....
- Enhance DDB to use CTF data to print data structures to replace some DB_SHOW_* commands.

printf debugging is dead

- How many patches of printf debugging sessions did you throw away?
- Dtrace has a learning curve but as of late you do not need a special kernel anymore (on HEAD).

The real printf

- Rather than adding printfs add SDT probes and they will still be there the next time you need to debug this problem in three years.
- You can turn them on individually on demand.
- Save your "D scripts".

The real printf

```
SDT_PROVIDER_DECLARE(opencrypto);
SDT_PROBE_DEFINE5(opencrypto, deflate,
deflate_global, bad, bad, "int", "int", "int", "int",
"int");

  error = decomp ? inflateInit2(&zbuf, window_inflate) :
    deflateInit2(&zbuf, Z_DEFAULT_COMPRESSION, Z_METHOD,
           window_deflate, Z_MEMLEVEL, Z_DEFAULT_STRATEGY);
       if (error != Z_OK) {
               SDT_PROBE3(opencrypto, deflate,
                   deflate_global, bad,
                   decomp, error, __LINE__);
              goto bad;
       }
```

The real printf

inflate.d:

```
opencrypto:deflate:deflate_global:bad
{

        printf("[%s:%s:%s:%d:%s] decomp=%d error=%d
           avail_in/mode=%d avail_out/total_out=%d\n",
           probeprov, probemod, probefunc, arg2, probename,
           arg0, arg1, arg3, arg4);
}
```

Request

If you are a **developer**: please fix the code as you touch it!

If you are a **user** hitting this: make the developer fix the code!

Questions?

panic("BSDCan2012:Fri:1230:
Did I go over the time limit?
Action=run, it's lunch break!");