FAIL

Zach Loafman
Isilon Systems

- Errors are hard to test. You can ..
  - Inject [fake] hardware errors
    - Doesn't work for most software error testing
  - Manually change things in a debugger
    - Works, if you know what you're doing, but it's slow
  - Use fail points!
- Fail points allow you to add code points where you might want to inject failures.

Example:

```
error = func(foo, bar, blatz);
if (error) {
  /* do stuff */
}
```

Suppose you want to change the code in “Do stuff”. How well do you think you can test it?

Solution: Add a fail point

```
#include <sys/fail.h>

[…]
error = func(foo, bar, blatz);
KFAIL_POINT_CODE(FP_KERN, myfailpoint,
    error = RETURN_VALUE);
if (error) {
  /* do stuff */
}
```

Using the fail point:

```
KFAIL_POINT_CODE(FP_KERN, myfailpoint,
    error = RETURN_VALUE);
```

```
# sysctl fail_point.kern.myfailpoint
fail_point.kern.myfailpoint: off
geryon# sysctl fail_point.kern.myfailpoint=".1%return(5)"
fail_point.kern.myfailpoint: off -> .1%return(5)
```

What's this doing? When the KFAIL_POINT is encountered, 0.1% of the time the code in the third argument is executed, with the value of RETURN_VALUE set to the sysctl value.

Other example usage:

```
.1%return(5)->5%return(22)
# .1% of the time, return 5,
# 5% of the remaining time, return 22.

1%sleep(100)
# 1% of the time, sleep for 100ms

panic() / break() / print()
# panic immediately / break to debugger / print to console
```

- Caveats:
  - Incredibly easy to shoot yourself in the foot
  - Be careful with sleep()
    - You can override the sleep function if you manually build the fail point using fail_point_init/fail_point_set_sleep_fn.
    - Alternately, you can write your own sleep by wrapping RETURN_VALUE.
- Status:
  - Patch ready for CURRENT, only one failpoint added. We have hundreds at Isilon, but only a couple in the base kernel.
  - I'll send the API to freebsd-arch in the next few days