```
Fault injection capabilities infrastructure
===========================================

See also drivers/md/faulty.c and "every_nth" module option for scsi_debug.


Available fault injection capabilities
--------------------------------------

o failslab

  injects slab allocation failures. (kmalloc(), kmem_cache_alloc(), ...)

o fail_page_alloc

  injects page allocation failures. (alloc_pages(), get_free_pages(), ...)

o fail_make_request

  injects disk IO errors on devices permitted by setting
  /sys/block/<device>/make-it-fail or
  /sys/block/<device>/<partition>/make-it-fail. (generic_make_request())

Configure fault-injection capabilities behavior
-----------------------------------------------

o debugfs entries

fault-inject-debugfs kernel module provides some debugfs entries for runtime
configuration of fault-injection capabilities.

- /sys/kernel/debug/fail*/probability:

        likelihood of failure injection, in percent.
        Format: <percent>

        Note that one-failure-per-hundred is a very high error rate
        for some testcases.  Consider setting probability=100 and configure
        /sys/kernel/debug/fail*/interval for such testcases.

- /sys/kernel/debug/fail*/interval:

        specifies the interval between failures, for calls to
        should_fail() that pass all the other tests.

        Note that if you enable this, by setting interval>1, you will
        probably want to set probability=100.

- /sys/kernel/debug/fail*/times:

        specifies how many times failures may happen at most.
        A value of -1 means "no limit".

- /sys/kernel/debug/fail*/space:

        specifies an initial resource "budget", decremented by "size"
        on each call to should_fail(,size).  Failure injection is
        suppressed until "space" reaches zero.

- /sys/kernel/debug/fail*/verbose

        Format: { 0 | 1 | 2 }
        specifies the verbosity of the messages when failure is
        injected.  '0' means no messages; '1' will print only a single
        log line per failure; '2' will print a call trace too -- useful
        to debug the problems revealed by fault injection.

- /sys/kernel/debug/fail*/task-filter:

        Format: { 'Y' | 'N' }
        A value of 'N' disables filtering by process (default).
        Any positive value limits failures to only processes indicated by
        /proc/<pid>/make-it-fail==1.

- /sys/kernel/debug/fail*/require-start:
- /sys/kernel/debug/fail*/require-end:
- /sys/kernel/debug/fail*/reject-start:
- /sys/kernel/debug/fail*/reject-end:

        specifies the range of virtual addresses tested during
        stacktrace walking.  Failure is injected only if some caller
        in the walked stacktrace lies within the required range, and
        none lies within the rejected range.
        Default required range is [0,ULONG_MAX) (whole of virtual address
space).
        Default rejected range is [0,0).

- /sys/kernel/debug/fail*/stacktrace-depth:

        specifies the maximum stacktrace depth walked during search
        for a caller within [require-start,require-end) OR
        [reject-start,reject-end).

- /sys/kernel/debug/fail_page_alloc/ignore-gfp-highmem:

        Format: { 'Y' | 'N' }
        default is 'N', setting it to 'Y' won't inject failures into
        highmem/user allocations.

- /sys/kernel/debug/failslab/ignore-gfp-wait:
- /sys/kernel/debug/fail_page_alloc/ignore-gfp-wait:

        Format: { 'Y' | 'N' }
        default is 'N', setting it to 'Y' will inject failures
        only into non-sleep allocations (GFP_ATOMIC allocations).

- /sys/kernel/debug/fail_page_alloc/min-order:

        specifies the minimum page allocation order to be injected
        failures.

o Boot option
```

In order to inject faults while debugfs is not available (early boot time),
use the boot option:

```
	failslab=
	fail_page_alloc=
	fail_make_request=<interval>,<probability>,<space>,<times>
```

How to add new fault injection capability
----

o #include <linux/fault-inject.h>

o define the fault attributes

```
  DECLARE_FAULT_INJECTION(name);
```

  Please see the definition of struct fault_attr in fault-inject.h
  for details.

o provide a way to configure fault attributes

- boot option

  If you need to enable the fault injection capability from boot time, you can
  provide boot option to configure it. There is a helper function for it:

```
	setup_fault_attr(attr, str);
```

- debugfs entries

  failslab, fail_page_alloc, and fail_make_request use this way.
  Helper functions:

```
	init_fault_attr_dentries(entries, attr, name);
	void cleanup_fault_attr_dentries(entries);
```

- module parameters

  If the scope of the fault injection capability is limited to a
  single kernel module, it is better to provide module parameters to
  configure the fault attributes.

o add a hook to insert failures

  Upon should_fail() returning true, client code should inject a failure.

```
	should_fail(attr, size);
```

Application Examples
----

o Inject slab allocation failures into module init/exit code

```
#!/bin/bash
```

```
FAILTYPE=failslab
echo Y > /sys/kernel/debug/$FAILTYPE/task-filter
echo 10 > /sys/kernel/debug/$FAILTYPE/probability
echo 100 > /sys/kernel/debug/$FAILTYPE/interval
echo -1 > /sys/kernel/debug/$FAILTYPE/times
echo 0 > /sys/kernel/debug/$FAILTYPE/space
echo 2 > /sys/kernel/debug/$FAILTYPE/verbose
echo 1 > /sys/kernel/debug/$FAILTYPE/ignore-gfp-wait

faulty_system()
{
        bash -c "echo 1 > /proc/self/make-it-fail && exec $*"
}

if [ $# -eq 0 ]
then
        echo "Usage: $0 modulename [ modulename ... ]"
        exit 1
fi

for m in $*
do
        echo inserting $m...
        faulty_system modprobe $m

        echo removing $m...
        faulty_system modprobe -r $m
done
```

----

o Inject page allocation failures only for a specific module

```
#!/bin/bash

FAILTYPE=fail_page_alloc
module=$1

if [ -z $module ]
then
        echo "Usage: $0 <modulename>"
        exit 1
fi

modprobe $module

if [ ! -d /sys/module/$module/sections ]
then
        echo Module $module is not loaded
        exit 1
fi

cat /sys/module/$module/sections/.text >
/sys/kernel/debug/$FAILTYPE/require-start
cat /sys/module/$module/sections/.data > /sys/kernel/debug/$FAILTYPE/require-end
```

```
echo N > /sys/kernel/debug/$FAILTYPE/task-filter
echo 10 > /sys/kernel/debug/$FAILTYPE/probability
echo 100 > /sys/kernel/debug/$FAILTYPE/interval
echo -1 > /sys/kernel/debug/$FAILTYPE/times
echo 0 > /sys/kernel/debug/$FAILTYPE/space
echo 2 > /sys/kernel/debug/$FAILTYPE/verbose
echo 1 > /sys/kernel/debug/$FAILTYPE/ignore-gfp-wait
echo 1 > /sys/kernel/debug/$FAILTYPE/ignore-gfp-highmem
echo 10 > /sys/kernel/debug/$FAILTYPE/stacktrace-depth

trap "echo 0 > /sys/kernel/debug/$FAILTYPE/probability" SIGINT SIGTERM EXIT

echo "Injecting errors into the module $module... (interrupt to stop)"
sleep 1000000
```