```
linux/kernel/panic.c

Copyright (C) 1991, 1992  Linus Torvalds


This function is used through-out the kernel (including mm and fs)
to indicate a major problem.
```

```
Returns how long it waited in ms
```

```
Stop ourself in panic -- architecture code may override this
```

```
Stop ourselves in NMI context if another CPU has already panicked. Arch code
may override this to prepare for crash dumping, e.g. save regs info.
```

```
panic - halt the system
@fmt: The text string to print

Display a message, then perform cleanups.

This function never returns.
```

```
        Disable local interrupts. This will prevent panic_smp_self_stop
        from deadlocking the first cpu that invokes the panic, since
        there is nothing to prevent an interrupt handler (that runs
        after setting panic_cpu) from invoking panic() again.
```

```
        It's possible to come here directly from a panic-assertion and
        not have preempt disabled. Some functions called from here want
        preempt to be disabled. No point enabling it later though...

        Only one CPU is allowed to execute the panic code from here. For
        multiple parallel invocations of panic, all other CPUs either
        stop themself or will wait until they are stopped by the 1st CPU
        with smp_send_stop().

        `old_cpu == PANIC_CPU_INVALID' means this is the 1st CPU which
        comes here, so go ahead.
        `old_cpu == this_cpu' means we came from nmi_panic() which sets
        panic_cpu to this CPU.  In this case, this is also the 1st CPU.
```

```
        Avoid nested stack-dumping if a panic occurs during oops processing
```

```
        If we have crashed and we have a crash kernel loaded let it handle
        everything else.
        If we want to run this after calling panic_notifiers, pass
        the "crash_kexec_post_notifiers" option to the kernel.

        Bypass the panic_cpu check and call __crash_kexec directly.
```

```
        Note smp_send_stop is the usual smp shutdown function, which
        unfortunately means it may not be hardened to work in a panic
        situation.
```

```
        Run any panic handlers, including those that might need to
        add information to the kmsg dump output.
```

```
        If you doubt kdump always works fine in any situation,
        "crash_kexec_post_notifiers" offers you a chance to run
        panic_notifiers and dumping kmsg before kdump.
        Note: since some panic_notifiers can make crashed kernel
```

```
         more unstable, it can increase risks of the kdump failure too.

         Bypass the panic_cpu check and call __crash_kexec directly.
         ...

         We may have ended up stopping the CPU holding the lock (in
         smp_send_stop()) while still having some valuable data in the console
         buffer.  Try to acquire the lock then release it regardless of the
         result.  The release will also print the buffers out.  Locks debug
         should be disabled to avoid reporting bad unlock balance when
         panic() is not being callled from OOPS.
         ...
                Delay timeout seconds before rebooting the machine.
                We can't use the "normal" timers since we just panicked.
         ...
                This will not be a clean reboot, with everything
                shutting down.  But if there is a chance of
                rebooting the system it will be rebooted.
    ...                 Make sure the user can actually press Stop-A (L1-A)
         ...

 print_tainted - return a string to represent the kernel taint state.

 'P' - Proprietary module has been loaded.
 'F' - Module has been forcibly loaded.
 'S' - SMP with CPUs not designed for SMP.
 'R' - User forced a module unload.
 'M' - System experienced a machine check exception.
 'B' - System has hit bad_page.
 'U' - Userspace-defined naughtiness.
 'D' - Kernel has oopsed before
 'A' - ACPI table overridden.
 'W' - Taint on warning.
 'C' - modules from drivers/staging are loaded.
 'I' - Working around severe firmware bug.
 'O' - Out-of-tree module has been loaded.
 'E' - Unsigned module has been loaded.
 'L' - A soft lockup has previously occurred.
 'K' - Kernel has been live patched.

 The string is overwritten by the next call to print_tainted().
 ...

 add_taint: add a taint flag if not already set.
 @flag: one of the TAINT_* constants.
 @lockdep_ok: whether lock debugging is still OK.

 If something bad has gone wrong, you'll want @lockdebug_ok = false, but for
 some noteworthy-but-not-corrupting cases, it can be set to true.
 ...

 It just happens that oops_enter() and oops_exit() are identically
 implemented...
 ...  This CPU may now print the oops message ...        We need to stall this
            This CPU gets to do the counting ...
                  This CPU waits for a different one ...

 Return true if the calling CPU is allowed to print oops-related info.
 This is a bit racy..
 ...

 Called when the architecture enters its oops handler, before it prints
 anything.  If this is the first CPU to oops, and it's oopsing the first
 time then let it proceed.

 This is all enabled by the pause_on_oops kernel boot option.  We do all
 this to ensure that oopses don't scroll off the screen.  It has the
 side-effect of preventing later-oopsing CPUs from mucking up the display,
 too.

 It turns out that the CPU which is allowed to print ends up pausing for
 the right duration, whereas all the other CPUs pause for twice as long:
 once in oops_enter(), once in oops_exit().
 ...
         can't trust the integrity of the kernel anymore: ...

 64-bit random ID for oopses:
 ...

 Called when the architecture exits its oops handler, after printing
```

```
everything.
                This thread may hit another WARN() in the panic path.
                Resetting this prevents additional WARN() from panicking the
                system on this thread.  Other threads are blocked by the
                panic_mutex in panic().
                Just a warning, don't kill lockdep.

Called when gcc's -fstack-protector feature is used, and
gcc detects corruption of the on-stack canary value
```

Source: [redacted] 9/03/2016