Explaining the dreaded "No init found." boot hang message
===

OK, so you've got this pretty unintuitive message (currently located
in init/main.c) and are wondering what the H*** went wrong.
Some high-level reasons for failure (listed roughly in order of execution)
to load the init binary are:
A) Unable to mount root FS
B) init binary doesn't exist on rootfs
C) broken console device
D) binary exists but dependencies not available
E) binary cannot be loaded

Detailed explanations:
0) Set "debug" kernel parameter (in bootloader config file or CONFIG_CMDLINE)
to get more detailed kernel messages.
A) make sure you have the correct root FS type
(and root= kernel parameter points to the correct partition),
required drivers such as storage hardware (such as SCSI or USB!)
and filesystem (ext3, jffs2 etc.) are builtin (alternatively as modules,
to be pre-loaded by an initrd)
C) Possibly a conflict in console= setup --> initial console unavailable.
E.g. some serial consoles are unreliable due to serial IRQ issues (e.g.
missing interrupt-based configuration).
Try using a different console= device or e.g. netconsole= .
D) e.g. required library dependencies of the init binary such as
/lib/ld-linux.so.2 missing or broken. Use readelf -d <INIT>|grep NEEDED
to find out which libraries are required.
E) make sure the binary's architecture matches your hardware.
E.g. i386 vs. x86_64 mismatch, or trying to load x86 on ARM hardware.
In case you tried loading a non-binary file here (shell script?),
you should make sure that the script specifies an interpreter in its shebang
header line (#!/...) that is fully working (including its library
dependencies). And before tackling scripts, better first test a simple
non-script binary such as /bin/sh and confirm its successful execution.
To find out more, add code to init/main.c to display kernel_execve()s
return values.

Please extend this explanation whenever you find new failure causes
(after all loading the init binary is a CRITICAL and hard transition step
which needs to be made as painless as possible), then submit patch to LKML.
Further TODOs:

- Implement the various run_init_process() invocations via a struct array
  which can then store the kernel_execve() result value and on failure
  log it all by iterating over _all_ results (very important usability fix).
- try to make the implementation itself more helpful in general,
  e.g. by providing additional error messages at affected places.

Andreas Mohr <andi at lisas period de>