NETIF Msg Level

The design of the network interface message level setting.

History

The design of the debugging message interface was guided and constrained by backwards compatibility previous practice. It is useful to understand the history and evolution in order to understand current practice and relate it to older driver source code.

From the beginning of Linux, each network device driver has had a local integer variable that controls the debug message level. The message level ranged from 0 to 7, and monotonically increased in verbosity.

The message level was not precisely defined past level 3, but were always implemented within +-1 of the specified level. Drivers tended to shed the more verbose level messages as they matured.

- 0 Minimal messages, only essential information on fatal errors.
- 1 Standard messages, initialization status. No run-time messages
- 2 Special media selection messages, generally timer-driver.
- 3 Interface starts and stops, including normal status messages
- 4 Tx and Rx frame error messages, and abnormal driver operation
- 5 Tx packet queue information, interrupt events.
- 6 Status on each completed Tx packet and received Rx packets
- 7 Initial contents of Tx and Rx packets

Initially this message level variable was uniquely named in each driver e.g. "lance_debug", so that a kernel symbolic debugger could locate and modify the setting. When kernel modules became common, the variables were consistently renamed to "debug" and allowed to be set as a module parameter.

This approach worked well. However there is always a demand for additional features. Over the years the following emerged as reasonable and easily implemented enhancements

- Using an ioctl() call to modify the level.
- Per-interface rather than per-driver message level setting.
- More selective control over the type of messages emitted.

The netif_msg recommendation adds these features with only a minor complexity and code size increase.

The recommendation is the following points

- Retaining the per-driver integer variable "debug" as a module parameter with a default level of '1'.
- Adding a per-interface private variable named "msg_enable". The variable is a bit map rather than a level, and is initialized as:

  ```
  1 << debug
  ```

  Or more precisely:

  ```
  debug < 0 ? 0 : 1 << min(sizeof(int)-1, debug)
  ```

  Messages should changes from:

  ```
  if (debug > 1)
       printk(MSG_DEBUG "%s: ...
  ```

  to:

  ```
  if (np->msg_enable & NETIF_MSG_LINK)
       printk(MSG_DEBUG "%s: ...
  ```

The set of message levels is named

Old level	Name	Bit position
0	NETIF_MSG_DRV	0x0001
1	NETIF_MSG_PROBE	0x0002
2	NETIF_MSG_LINK	0x0004
2	NETIF_MSG_TIMER	0x0004
3	NETIF_MSG_IFDOWN	0x0008
3	NETIF_MSG_IFUP	0x0008

Old level	Name	Bit position
4	NETIF_MSG_RX_ERR	0x0010
4	NETIF_MSG_TX_ERR	0x0010
5	NETIF_MSG_TX_QUEUED	0x0020
5	NETIF_MSG_INTR	0x0020
6	NETIF_MSG_TX_DONE	0x0040
6	NETIF_MSG_RX_STATUS	0x0040
7	NETIF_MSG_PKTDATA	0x0080