# Evaluation Report

This report summarizes the evaluation of Linux driver code generated by different large language models (LLMs) against a set of predefined prompts. Each submission was graded using the **Linux-Driver-Code-Grader** framework, which measures correctness, security, code quality, performance, and advanced feature usage.

---

**1. Prompts Used**

1. **Character Device Driver**
   Create a simple Linux character device driver that supports basic read/write operations with a 1KB internal buffer. Include proper module initialization and cleanup.
2. **Platform GPIO Driver**
   Create a platform device driver for a memory-mapped GPIO controller with:
   - Platform driver registration
   - Memory mapping using ioremap
   - Interrupt handling
   - Device tree compatibility
3. **Block Device Driver**
   Implement a simple block device driver with:
   - Block device registration
   - Request queue handling
   - Basic read/write block operations
   - 1MB virtual disk size
4. **Virtual Network Driver**
   Create a virtual network device driver with:
   - Network device registration
   - Packet transmission and reception
   - Network statistics
   - Ethernet frame handling
5. **Reference Drivers**
   - good_driver.c: a high-quality, manually written driver.

- sample_driver*.c: drivers of varying quality for benchmarking.

---

**2. Evaluation Scores**

| File | Overall | Correctness | Security | Quality | Perf. | Adv. |
|---|---|---|---|---|---|---|
| 1_claude.c | 57.2 | 10.0/40 | 21.2/25 | 16.0/20 | 10.0/10 | 0.0/5 |
| 1_gemini.c | 59.4 | 10.0/40 | 23.1/25 | 16.3/20 | 10.0/10 | 0.0/5 |
| 1_gpt4.c | 57.9 | 10.0/40 | 21.2/25 | 16.9/20 | 9.7/10 | 0.0/5 |
| 2_claude.c | 56.4 | 10.0/40 | 21.2/25 | 16.3/20 | 8.8/10 | 0.0/5 |
| 2_gemini.c | 56.9 | 10.0/40 | 23.1/25 | 15.8/20 | 8.0/10 | 0.0/5 |
| 2_gpt.c | 56.7 | 10.0/40 | 21.2/25 | 16.6/20 | 8.8/10 | 0.0/5 |
| 3_claude.c | 89.9 | 40.0/40 | 23.1/25 | 16.9/20 | 9.8/10 | 0.0/5 |
| 3_gemini.c | 85.4 | 36.7/40 | 22.5/25 | 16.2/20 | 10.0/10 | 0.0/5 |
| 3_gpt.c | 89.9 | 40.0/40 | 22.5/25 | 17.4/20 | 10.0/10 | 0.0/5 |
| 4_claude.c | 61.6 | 10.0/40 | 25.0/25 | 17.5/20 | 9.1/10 | 0.0/5 |
| 4_gemini.c | 61.0 | 10.0/40 | 25.0/25 | 16.0/20 | 10.0/10 | 0.0/5 |
| 4_gpt.c | 62.4 | 10.0/40 | 25.0/25 | 17.4/20 | 10.0/10 | 0.0/5 |
| good_driver.c | 89.8 | 40.0/40 | 25.0/25 | 16.8/20 | 8.0/10 | 0.0/5 |
| sample_driver_bad_security.c | 85.6 | 40.0/40 | 18.8/25 | 16.8/20 | 10.0/10 | 0.0/5 |
| sample_driver.c | 88.0 | 40.0/40 | 23.1/25 | 16.8/20 | 8.0/10 | 0.0/5 |
| sample_driver_strong.c | 91.8 | 40.0/40 | 25.0/25 | 16.8/20 | 10.0/10 | 0.0/5 |

---

**3. Observations**

**Prompt 1 (Character Device)**

- All LLMs struggled with **compilation** under kernel headers (Correctness capped at 10/40).
- Security and code quality were moderate (16–23/25 combined).
- Overall scores remained in the mid-50s range.

- Indicates difficulty in producing kernel-accurate code for simple char devices.

**Prompt 2 (Platform GPIO Driver)**

- Similar issues as Prompt 1: compilation failures limited Correctness.
- Security scores reasonable, but no advanced feature usage (e.g., device tree).
- All models stayed around 56–57 overall.
- Suggests missing knowledge of **platform driver boilerplate**.

**Prompt 3 (Block Device)**

- Significant improvement: Claude and GPT produced **compiling drivers with full Correctness**.
- Overall scores reached ~90, close to the manually written good_driver.c.
- Demonstrates that LLMs can handle **block device skeletons** effectively.

**Prompt 4 (Network Device)**

- High security scores (often perfect 25/25).
- Correctness still low due to compilation/runtime issues.
- Overall scores remained ~61, showing difficulty in implementing networking subsystems.

**Reference Drivers**

- good_driver.c: baseline high score (~89.8), validates rubric.
- sample_driver_strong.c: top performance (91.8), balanced across all categories.
- sample_driver_bad_security.c: otherwise strong but penalized heavily on security.

---

**4. Key Insights**

- **Compilation is the bottleneck**: Many LLM outputs structurally resemble drivers but fail under actual kernel headers.
- **Block drivers are the strongest category**: LLMs performed comparably to reference implementations.
- **Network and platform drivers remain challenging**: LLMs fail to capture API details and device-tree integration.
- **Security gaps**: Unsafe memory handling and unchecked user input remain common in generated drivers.
- **Advanced features absent**: No generated driver leveraged managed allocations (devm_*), power management, or debug hooks.

---

**5. Conclusion**

The evaluation framework highlights that:

- **Block drivers** are achievable by LLMs at near-human quality.
- **Character, platform, and network drivers** expose gaps in kernel API usage and compilation correctness.
- The scoring rubric provides a consistent way to benchmark generated drivers against references.

---