

# Linux e a Ponte USB para NVMe Realtek RTL9210

## Resumo:

- **Sintomas:** Reinicializações repetidas de USB, erros de I/O ou discos desaparecendo no Linux.
- **Afetados:** Realtek RTL9210 (confirmado) e RTL9220 (possivelmente).
- **Causa:** Retorno ao ROM interno (**f0.01**) após falha na soma de verificação.
- **Impacto:** Instabilidade permanente, nenhuma ferramenta de reflash disponível para Linux.
- **Solução:** Apenas utilitários OEM do Windows podem restaurar o firmware – Realtek bloqueia alternativas de código aberto.

## Prefácio

Em 2025, deveria ser completamente razoável inicializar um Raspberry Pi a partir de um SSD conectado via USB. No entanto, devido às peculiaridades do firmware da Realtek, esse objetivo razoável se tornou uma aventura. Após meses de instabilidade inexplicável – reinicializações aleatórias, discos desaparecendo, sistemas de arquivos corrompidos – o autor esgotou todas as soluções comuns: cabos novos, hubs alimentados, atualizações de kernel, ajustes de USB e otimização de firmware. A descoberta veio apenas quando o ChatGPT respondeu a uma pergunta estranha feita tarde da noite: “É possível que a ponte USB para NVMe tenha revertido para um firmware antigo?”

## Introdução

Se o seu enclosure NVMe baseado em Realtek de repente se tornar instável após semanas de funcionamento impecável – reinicializações repetidas de USB, erros de I/O ou discos desaparecendo – você não está sozinho. Esse padrão surgiu em várias marcas, desde unidades sem nome até OEMs conhecidos como Sabrent e Orico. O denominador comum: **chips de ponte USB para NVMe Realtek RTL9210 e possivelmente RTL9220.**

No início, tudo funciona. Então, aparentemente sem motivo, o dispositivo começa a se desconectar sob carga ou durante uso prolongado, especialmente em sistemas Linux ou Raspberry Pi. A verdadeira causa não é o SSD nem a fonte de alimentação – é o próprio controlador de firmware que retorna silenciosamente ao seu **código de backup embutido no ROM**, uma versão que a Realtek ainda entrega internamente como **f0.01**.

## O Mecanismo Oculto – Retorno do Firmware por Design

Os chips de ponte da Realtek armazenam seu firmware operacional e dados de configuração em um flash SPI externo. Ao ligar, o controlador verifica uma soma de verificação simples. Se essa soma não corresponder, ele se recusa a carregar o firmware externo e, em vez disso, inicializa a partir de seu ROM interno.

Esse firmware de backup é antigo e defeituoso. Ele carece de várias correções de estabilidade USB e melhorias na gestão do estado do link presentes em revisões posteriores, levando à sequência clássica que todo usuário Linux reconhece:

```
usb 3-2: reinicialização do dispositivo USB de alta velocidade número 2  
usando xhci-hcd  
usb 3-2: leitura do descritor do dispositivo/64, erro -71  
Aviso EXT4-fs (dispositivo sda2): Erro de I/O ao escrever no inode ...
```

A soma de verificação pode se tornar inválida quando os dados de configuração são reescritos – por exemplo, quando a ponte atualiza suas configurações de gerenciamento de energia ou UAS – e o dispositivo perde energia durante a escrita. A próxima inicialização detecta uma soma de verificação corrompida e retorna permanentemente ao firmware do ROM.

Nesse ponto, seu “enclosure NVMe de alto desempenho” comporta-se exatamente como o mais barato dos invólucros sem nome, pois internamente agora executa o mesmo código base defeituoso gravado no silício.

## Verificação do Problema

Você pode confirmar esse estado facilmente no Linux:

```
lsusb -v | grep -A2 Realtek
```

Uma ponte Realtek saudável relata uma revisão de firmware (**bcdDevice**) acima de 1.00. Uma que retornou mostra:

```
bcdDevice f0.01
```

Essa assinatura **f0.01** significa que o controlador está inicializando a partir do ROM – e nenhuma quantidade de desconexão, reformatação ou ajustes no kernel corrigirá isso.

Esse mecanismo de retorno foi **confirmado no RTL9210**. O **RTL9220** parece compartilhar a mesma arquitetura de design e layout de firmware, então pode exibir comportamento idêntico, mas isso permanece **provável, não comprovado**.

## Por que Você Não Pode Consertar Isso Sozinho

Em princípio, a solução é trivial: reflashar o firmware correto no SPI. Na prática, a Realtek torna isso impossível.

A empresa fornece um atualizador de código fechado para Windows para OEMs e integradores. Nada é oferecido aos usuários Linux. Desenvolvedores da comunidade realizaram engenharia reversa de utilitários de flash compatíveis (**rtsupdater**, **rtl9210fw**, **rtsupdater-cli**) que permitiam a restauração completa do firmware a partir de sistemas Linux – até que a Realtek emitiu **notificações de remoção DMCA** para suprimi-los.

Não há justificativa plausível de propriedade intelectual para bloquear tais utilitários: eles não expõem microcódigo, apenas orquestram a sequência de atualização via USB. As remoções da Realtek não foram por proteção. Foram ideológicas.

## O Custo de uma Ideologia

Não se trata de idealismo de código aberto. Trata-se da **hostilidade ideológica de um fornecedor de hardware contra sistemas abertos** que quebra dispositivos comercializados como *compatíveis com Linux*.

A resistência da Realtek à documentação e ferramentas abertas persiste há duas décadas, abrangendo Wi-Fi, Ethernet, áudio e agora controladores de armazenamento. Esse isolamento pode passar despercebido em um mundo apenas Windows, mas torna-se tóxico quando os mesmos chips são integrados em produtos multiplataforma como o **Sabrent EC-SNVE**, que exibe abertamente o logotipo do Linux em sua embalagem.

Ao proibir utilitários de flash para Linux e bloquear a manutenção da comunidade, a Realtek efetivamente **criminalizou a autorreparação**. As consequências se espalham para fora:

- Os usuários Linux veem o hardware “suportado” degradar-se em instabilidade.
- OEMs como Sabrent e Orico enfrentam custos desnecessários de RMA e garantia.
- A reputação de longa data da Realtek por má compatibilidade com Linux é reforçada novamente.

No final, não é o código aberto que quebra os dispositivos da Realtek – é a **hostilidade da Realtek ao código aberto** que os quebra.

## Um Caminho Racional Adiante

A solução não exige uma mudança ideológica, apenas pragmatismo. A Realtek poderia:

1. Lançar um atualizador de linha de comando assinado pelo fornecedor para Linux (sem necessidade de divulgação do código-fonte).
2. Publicar o algoritmo de soma de verificação para que os integradores possam validar imagens de flash com segurança.
3. Adotar um modo estilo DFU que aceite atualizações via armazenamento em massa USB, independentemente do sistema operacional.

Cada um desses passos evitaria custos de garantia, protegeria as relações com OEMs e restauraria a confiança nos chips de ponte da Realtek entre usuários profissionais do Li-

nux – de construtores de estações de trabalho a desenvolvedores de Raspberry Pi.

## O que Você Pode Fazer

Se você suspeita que seu enclosure retornou ao firmware do ROM:

- Verifique com **lsusb -v | grep bcdDevice**.
- Se mostrar **f0.01**, informe o problema ao seu OEM.
- Inclua o trecho de **dmesg** e aponte para este mecanismo de retorno documentado.
- Peça ao seu fornecedor para escalar o problema para a Realtek, citando a necessidade de um atualizador compatível com Linux.

A política de firmware da Realtek não apenas incomoda os entusiastas; ela cria perdas financeiras tangíveis para seu próprio ecossistema. Quanto mais cedo essa realidade for reconhecida dentro da empresa, mais cedo os usuários Linux e parceiros OEM poderão parar de desperdiçar tempo em ciclos de RMA evitáveis.

## Respostas dos Fabricantes

Tanto a Realtek quanto a Sabrent foram convidadas a fornecer declarações sobre o problema de retorno do firmware descrito acima. Suas respostas – se recebidas – serão adicionadas aqui.

## Apêndice – Identificação de Dispositivos Afetados

Controlador	ID do Fornecedor	ID do Produto	Notas	Status
RTL9210	0x0bda	0x9210	Ponte USB 3.1 Gen 2 10 Gb/s	<b>Confirmado</b> comportamento de retorno
RTL9220	0x0bda	0x9220	Ponte USB 3.2 Gen 2×2 20 Gb/s	<b>Possível</b> , arquitetura semelhante

Assinatura de retorno do firmware: **bcdDevice f0.01**

Revisões estáveis conhecidas: **1.23 – 1.31**