ファイル調査4：バイナリパッチをあてる

2025-10-05 Copilotが生成

これは何？

バイナリファイルの特定位置（オフセット）に対して、任意のバイト列を書き込む処理を PowerShell で試してみた記録。テキストファイルの置換とは違い、バイナリを直接いじるので慎重さが必要。位置と内容が分かっていれば、スクリプトで一括処理も可能。

試したコード（共通構文）

```powershell
# ファイルをバイナリとして読み込む
$bytes = [System.IO.File]::ReadAllBytes(".\target.bin")

# パッチ定義（オフセットとバイト列のペア）
$patches = @(
    @{ Offset = 0x10; Data = [byte[]](0xFE,0xED,0xFA,0xCE) },
    @{ Offset = 0x20; Data = [byte[]](0xBA,0xAD,0xF0,0x0D) },
    @{ Offset = 0x30; Data = [byte[]](0xDE,0xAD,0xBE,0xEF) }
)

# パッチ適用
foreach ($patch in $patches) {
    $offset = $patch.Offset
    $data = $patch.Data
    for ($i = 0; $i -lt $data.Length; $i++) {
        $bytes[$offset + $i] = $data[$i]
    }
}

# 上書き保存
[System.IO.File]::WriteAllBytes(".\target.bin", $bytes)
```

PowerShell 7.5での動作

PowerShell 7.5 では、上記コードはそのまま動作する。`pwsh` 環境でも `[System.IO.File]::ReadAllBytes()` や `WriteAllBytes()` は問題なく使える。ハッシュテーブルの扱いも改善されていて、補完も効きやすい。

確認には `Format-Hex` を使うと便利。

```powershell
Get-Content .\target.bin -Encoding Byte | Format-Hex
```

PowerShell 5.1での動作

Windows PowerShell 5.1 でも基本的に同じコードで動作する。ただし`Format-Hex` は 5.1 以降で追加されたコマンドレットなので、古い環境では使えない場合がある。

また`[byte[]]` のキャストやハッシュテーブルの扱いがやや厳密なので、環境によっては `$patch.Data` の部分で型が正しく扱われているか確認するのが吉。

気づいたこと

- オフセット指定でのバイナリ編集は、パターン検索より確実。
- 複数箇所のパッチも、配列で定義すればスッキリ書ける。
- 範囲外アクセスには注意`$offset + $data.Length` が `$bytes.Length` を超えないように。
- バックアップは忘れずに。壊れたら復旧できない。

まとめ

PowerShell でもバイナリパッチは可能。7.5 でも 5.1 でも基本的な構文は共通で、位置と内容が分かっていればスクリプトで一括処理できる。まずは壊してもいいファイルで試してみるのが安心。