null ポインターの逆参照が引き起こす未定義の動作

この記事は、インテル® デベロッパー・ゾーンに公開されている「Null Pointer Dereferencing Causes Undefined Behavior」の日本語参考訳です。

私は最近、C/C++ で P が null ポインターの場合に &P->m_foo 式を使用することは正しいかどうかという質問について大きな討論を思いがけなく引き起こすことになりました。プログラマーのコミュニティーは 2 つのグループに分かれました。一方のグループは正しくないと主張し、もう一方は正しいと主張しました。両方のグループからさまざまな根拠やリンクが提示され、あるとき私は明確にすべき点があることに気付きました。そのため、私は非公開のメーリングリストを通じて Microsoft* の MVP エキスパートと Visual C++* 開発チームに連絡をとりました。彼らからは、この記事を準備するにあたって多くの支援を受けました。興味のある方は、この続きを是非お読みください。答えを最後まで待てない人へ。このコードは正しくありません。

討論の履歴

この討論は、PVS-Studio アナライザーによる Linux* カーネルのチェックに関する記事から始まりました。チェックでは何もする必要はありませんでした。ポイントは、この記事で次の Linux* コードを引用したことです。

```
static int podhd_try_init(struct usb_interface *interface,
        struct usb_line6_podhd *podhd)
{
  int err;
  struct usb_line6 *line6 = &amp;podhd-&gt;line6;

  if ((interface == NULL) || (podhd == NULL))
    return -ENODEV;
  ....
}
```

私は、このコードが未定義の動作を引き起こすと考えて、コードは危険であると述べました。

その後、私の考えに反論する読者から大量の email とコメントが寄せられ、その説得力のある論拠に思わず納得してしまいそうになりました。例えば、このコードが正しい証拠として、offsetof マクロの実装が指摘されました。一般的な実装を次に示します。

```
#define offsetof(st, m) ((size_t)(&((st *)0)->m))
```

ここでは null ポインターの逆参照が行われていますが、コードは問題なく動作します。null ポインターによるアクセスが行われていないため問題はないと述べた email もありました。

私はだまされやすい性格ですが、疑わしい情報はダブルチェックしようとする一面も持ち合わせています。私は問題の調査を開始し、最終的に短い記事「Reflections on the Null Pointer Dereferencing Issue (null ポインターの逆参照問題に対する意見)」を執筆しました。

すべては私が正しかったことを示していました。そのようなコードを記述することはできないのです。この記事では、私の結論に対する有力な証拠を提示するのではなく、標準規格の適切な項目を抜粋して引用しました。

この記事を公開した後、再び大量の email による反論が寄せられました。そこで私は、これを最後に結論を出すべきであると考え、言語のエキスパートに質問して回答を得ることにしました。この記事は、彼らの回答を要約したものです。

C 言語

C 言語では、'podhd' が null ポインターの場合、'&podhd->line6' 式は未定義の動作を引き起こします。

C99 標準規格では、'&' address-of 演算子について次のように述べています (6.5.3.2 "Address and indirection operators (アドレスと間接演算子)"):

*The operand of the unary & operator shall be either a function designator, the result of a [ ] or unary * operator, or an lvalue that designates an object that is not a bit-field and is not declared with the register storage-class specifier. (単項 & 演算子のオペランドは、関数指示子、[ ] または単項 * 演算子の結果、あるいはビット・フィールドではなく register ストレージクラス指定子で宣言されないオブジェクトを指定する lvalue のいずれかであるものとします。)*

式 'podhd->line6' は、明らかに関数指示子、[] または * 演算子の結果ではなく lvalue 式です。しかし、'podhd' ポインターが null の場合、式はオブジェクトを指定しません (6.3.2.3 "Pointers (ポインター)"):

If a null pointer constant is converted to a pointer type, the resulting pointer, called a null pointer, is guaranteed to compare unequal to a pointer to any object or function.(null ポインター定数がポインター型に変換された場合、生成されるポインター (null ポインター) は任意のオブジェクトや関数のポインターに対して等しくないことが保証されます。)

評価されたときに lvalue がオブジェクトを指定しない場合、動作は未定義です (C99 6.3.2.1 "Lvalues, arrays, and function designators (lvalue、配列、関数指定子)"):

*An lvalue is an expression with an object type or an incomplete type other than void**; if an lvalue does not designate an object when it is evaluated, the behavior is undefined.**(lvalue はオブジェ*

*クト型または void 以外の不完全型を含む式です。**評価されたときに lvalue がオブジェクトを指定しない場合、動作は未定義です。**)*

つまり、要約すると次のようになります。

-> がポインターで実行されると、オブジェクトが存在しない lvalue が評価され、その結果、動作は未定義です。

C++ 言語

C++ 言語でも、状況は全く同じです。'podhd' が null ポインターの場合、'&podhd->line6' 式は未定義の動作を引き起こします。

以前の記事で参照した WG21 のディスカッション (232. Is indirection through a null pointer undefined behavior?) は多少の混乱をもたらしました。この議論に参加したプログラマーたちは、この式は未定義の動作ではないと主張しました。しかし、C++ 標準規格で "polhd" が null ポインターの "poldh->line6" の使用を認めている節を見つけられた人はいませんでした。

'polhd' ポインターはオブジェクトを指定しなければいけないという基本的な制約 (5.2.5/4 の第 2 項) を満たしていません。C++ オブジェクトは nullptr をアドレスにすることはできません。

まとめ

```
struct usb_line6 *line6 = &podhd->line6;
```

podhd ポインターが 0 の場合、このコードは C でも C++ でも正しくありません。ポインターが 0 の場合、未定義の動作を引き起こします。

プログラムが動作するのは単に運がいいだけです。未定義の動作は、プログラマーが想定した方法でのプログラム実行を含む、さまざまな形になります。プログラムが動作した場合、たまたまそれが未定義の動作の特別なケースの 1 つであったということにすぎません。

そのようなコードを記述することはできません。ポインターは逆参照を行う前に確認する必要があるのです。

追加情報とリンク

- 'offsetof()' 演算子の慣用句的な実装を考える場合、コンパイラー実装は機能を実装するために可搬性のない手法を使用できることを考慮に入れる必要があります。コンパイラーのライブラリー実装が 'offsetof()' の実装に null ポインター定数を使用していたとしても、'podhd' が null ポインターの場合に '&podhd->line6' を使用しても問題ないということにはなりません。
- GCC で未定義の動作が発生しないと仮定して最適化を行うと、null チェックは削除されます。カーネルは、そうしないようにコンパイラーに指示するスイッチを指定してコンパイルします。例として、エキスパートは記事「What Every C Programmer Should Know About Undefined Behavior #2/3 (なぜすべての C プログラマーは未定義の動作について知っておくべきなのか #2/3)」に言及していました。
- null ポインターの類似した利用が TUN/TAP ドライバーのカーネル開発でも行われていたのは興味深いことです。「Fun with NULL pointers」を参照してください。類似性は当てはまらないと一部

の人に思わせる主な違いは、TUN/TAP ドライバーのバグでは、null ポインターがアクセスした構造体フィールドは、単にフィールドのアドレスを取得する代わりに変数を初期化する値として明示的に取得されることです。しかし、標準 C である限り、null ポインターでフィールドのアドレスを取得することは未定義の動作になります。

- P == nullptr の場合に &P->m_foo を記述しても問題ないケースはあるでしょうか? はい。例えば、sizeof(&P->m_foo) のように、sizeof 演算子の引数の場合は問題ありません。

謝辞

この記事は、全面的に信頼できるエキスパートのおかげで執筆することができました。執筆にご協力いただいた次の方々に感謝します。

- **Michael Burr 氏**は、Windows* サービス、ネットワーク、デバイスドライバーを含む、システムレベルおよび組込みソフトウェアを専門とする C/C++ エンスージアストです。StackOverflow コミュニティーで C および C++ に関する質問によく回答しています (より簡単な C# の質問に回答することもあります)。Visual C++* の Microsoft* MVP アワードを 6 回受賞しています。
- **Billy O'Neal 氏**は、C++ 開発者であり、StackOverflow の貢献者です。現在は Microsoft* の Trustworthy Computing チームのソフトウェア開発エンジニアで、以前は、Malware Bytes や PreEmptive Solutions を含む、さまざまなセキュリティー関連企業で働いていました。
- **Giovanni Dicanio 氏**は、Windows* オペレーティング・システム開発を専門とするコンピューター・プログラマーです。イタリアのコンピューター雑誌に、C++、OpenGL* その他のプログラミング言語に関するコンピューター・プログラミングの記事を執筆しています。いくつかのオープンソース・プロジェクトのコード生成にも貢献しています。Microsoft* MSDN フォーラム (および StackOverflow) では、C および C++ プログラミングの問題解決に役立つ回答を行っています。Visual C++* の Microsoft* MVP アワードを 8 回受賞しています。
- **Gabriel Dos Reis 氏**は、Microsoft* の主任ソフトウェア開発エンジニアです。研究者でもあり、C++ 委員会のメンバーを長い間務めています。研究対象には、高信頼性ソフトウェア用のプログラミング・ツールが含まれます。Microsoft* に入社する前は、Texas A&M University の助教授でした。高信頼性計算数学に対するコンパイラーの研究および教育活動により 2012 年の NSF (米国国立科学財団) キャリアアワードを受賞しています。C++ 標準化委員会のメンバーです。

参考文献

1. ウィキペディアの Undefined Behavior。
2. A Guide to Undefined Behavior in C and C++。Part 1、2、3。
3. ウィキペディアの offsetof。
4. LLVM ブログ。What Every C Programmer Should Know About Undefined Behavior #2/3.
5. LWN.net。Fun with NULL pointers。Part 1、2。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください