

スタックメモリの基礎



KN0015-00 / 2024 年 10 月 20 日 / こがねさん(著)



<https://www.kumikomist.com/>

■目次

1. スタックメモリとは 1	3. メモリー領域の使い方 3
2. メモリー領域 2	4. スタックの構造 4

■文書内の記号

	取り扱いにおける禁止事項（してはいけないこと）を示しています。
	取扱における指示事項（必ずしなければいけないこと）を示しています。

	取り扱いにおける注記事項を示しています。
	取り扱いにおけるポイントを示しています。

1. スタックメモリとは

スタック (stack) メモリーとは関数の引数や戻り値、ローカル変数に関数の呼び出し元へ戻るためのアドレスを保存するために用いられます。スタックメモリーは目につきにくい存在ですが、RAM の一部を専有しておりその容量には制限があります。Microsoft Windows や Mac OS、Linux などでは RAM 容量を気にすることはそうないでしょう。しかしマイコンの RAM 容量はとても少ないのです。このため組み込みソフト開発者は、開発する製品の規模に合わせて適切なスタック領域を確保し、運用する必要があります。

2. メモリー領域

メモリー領域は図1のように、①テキスト領域、②静的領域、③ヒープ領域、④スタック領域に分かれます。このうち①がROMに、②～④がRAMに格納されます。このうち特異なのが今回のテーマであるスタック領域と、それに近いものとしてヒープ領域があります。

スタック領域とヒープ領域は、それぞれ使用するほどRAMを専有していき、領域を超えるとオーバーフローによりメモリー破壊が発生します。メモリー破壊は原因の特定が困難な不具合の1つで、最大限の注意が必要です。しかしそれを恐れてスタック領域やヒープ領域を多く取ると、RAM不足で変数の確保が難しくなってしまいます。

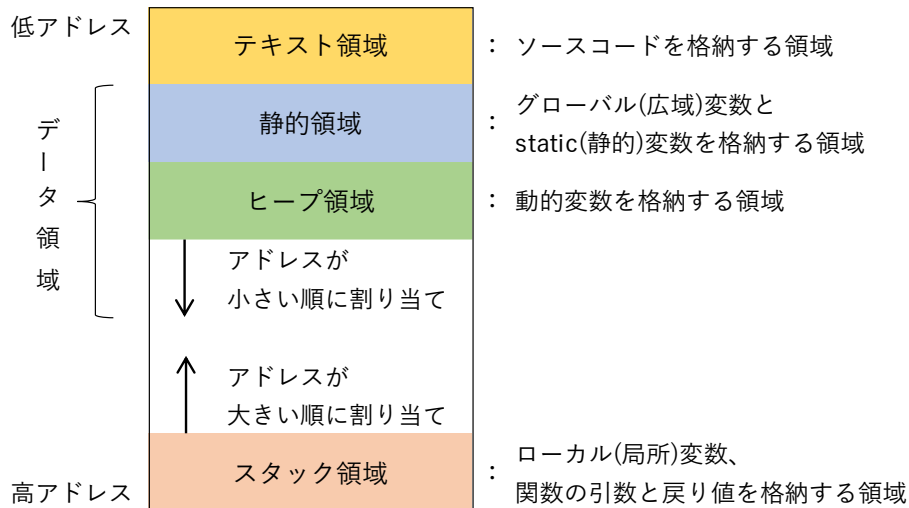


図1 メモリー領域



- ヒープは動変数 (C 言語の malloc) を使う場合に必要領域です。動的メモリーの使用は非推奨であり、ここでは説明を省略します。
- 処理系によっては、ビルド時にスタックオーバーフローを指摘してくれるものもあります。指摘された場合はスタック領域のサイズを大きくするか、スタック使用量の削減を試みてください。

3. メモリー領域の使い方

実際のメモリー領域の使い方は図 2 のようになります。

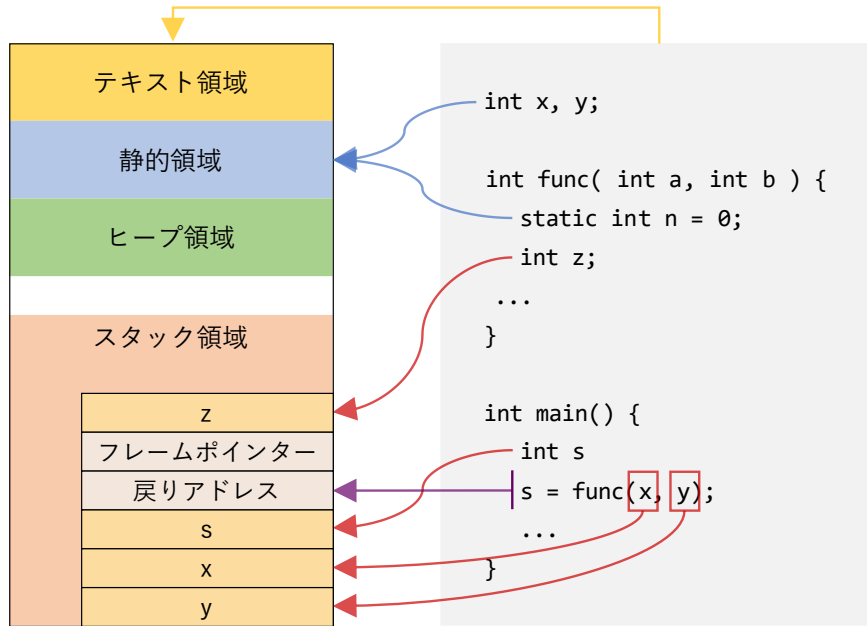


図 2 メモリー領域の割り当て例

見てのとおり、関数の引数や関数内のローカル変数はスタック領域に確保されます。したがってスタックメモリーを節約するためには、次の点に気を付ける必要があります。

- (1) 構造体を引数で渡す際は、値渡しではなくポインターで渡す。
- (2) サイズの大きい配列を関数内で宣言しない。
どうしても必要な場合は、static 変数またはグローバル変数にすることを検討してください。
- (3) 再帰関数を使用しない。または関数を深くしない（関数の中で関数を呼び出す）。

	<ul style="list-style-type: none">● サイズの大きい引数やローカル変数を採用すると、スタック領域の専有と解放に掛かるオーバーヘッドが大きくなります。処理の遅延が発生するという意味でも、サイズの大きい引数やローカル変数の使用は避けるべきです。
	<ul style="list-style-type: none">● フレームポインターはベースポインターとも呼ばれ、呼びされた関数の起点となるアドレスを表します。ローカル変数はこのフレームポインターを起点に呼び出されます。● オーバーヘッドとはコードの実行に際して、間接的に発生する処理を表します。今回の場合は、関数を呼び出して関数内の処理を実行するのが本来の処理。関数を呼び出すためにはスタック領域に引数やローカル変数枠を確保するのが間接的な処理となります。

4. スタックの構造

スタックの構造は後入れ先出し（LIFO：Last In First Out）です（図 3 (a)）。箱に入れたのとは逆順に、最後に箱に入れたものを最初に取り出します。スタックにデータを積むことをプッシュ、スタックからデータを取り出すことをポップと呼びます。

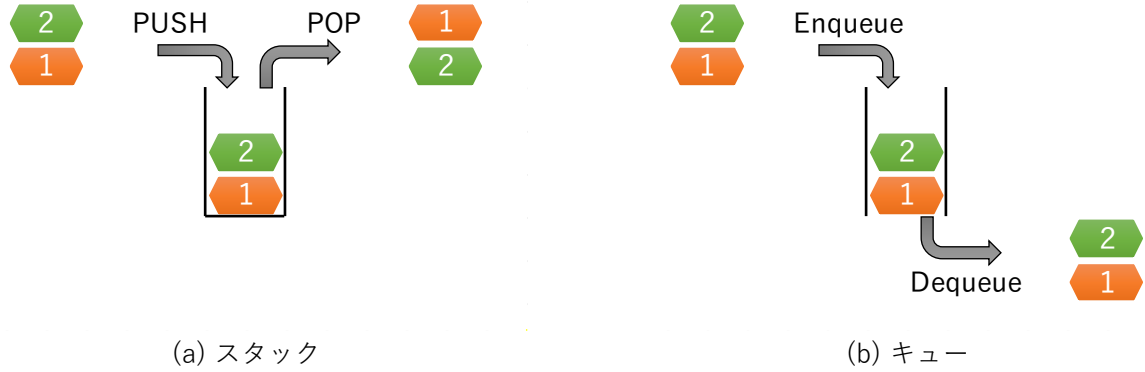


図 3 スタックとキュー

また対となるものとしてキューがあります。キューの構造は先入れ先出し（FIFO：First In First Out）です（図 3 (b)）。