組み込みRustのライブラリ 便利クレート探偵団

第5回 heaplessによるキューとスマート・ポインタ

中林 智之

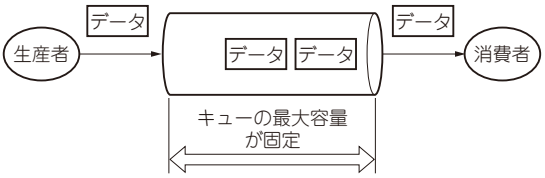

図1 SPSC (Single Producer Single Consumer) キューの動作

Rustは組み込みで使えるプログラミング言語として注目されています．本連載ではそんなRustの組み込み開発で役立つライブラリ（クレート）を紹介します．

前回（2024年7月号）から引き続きマイコンをターゲットとしたno_stdな組み込みRustでメモリ・アロケータ（C言語で言うmallocなどのメモリ確保API）なしでも便利なコレクションが使えるheapless[1]について紹介します．

前回はheaplessがどのようにうれしいクレートかという概要と，heaplessの最も基本的なコレクションのheapless::vec::Vecについて紹介しました．

今回はheaplessで使える応用的なコレクションとしてspsc::Queue，mpmc::QとBoxを紹介します．

単一生産者－単一消費者キュー… spsc::Queue

最大容量が固定のSingle Producer Single Consumer（SPSC）キューです．

SPSCキューとは，キューにデータを入れるデータの生産者（producer）とキューからデータを取り出すデータの消費者（consumer）が，それぞれ単一であることを意味しています（**図1**）．

例えば，UARTでシリアル通信することを考えると，割り込みハンドラで受信したデータをキューに入れて，別のメイン・タスクでデータを取り出して使う，というような使い方ができます．

heaplessのspsc::QueueではRustの所有権システムを利用し，複数の場所からデータを入れたり，取り出したりすることを制限しており，よりバグが発生しにくいようになっています．

リスト1 spsc::Queueの使い方①…普通のキューとして使う

```
use heapless::spsc::Queue;

let mut rb: Queue<u8, 4> = Queue::new();

assert!(rb.enqueue(0).is_ok());
assert!(rb.enqueue(1).is_ok());
assert!(rb.enqueue(2).is_ok());
assert!(rb.enqueue(3).is_err()); // full

assert_eq!(rb.dequeue(), Some(0));
assert_eq!(rb.dequeue(), Some(1));
assert_eq!(rb.dequeue(), Some(2));
assert_eq!(rb.dequeue(), None);
```

キューに追加

キューから取り出す

● プログラム動作環境の構築

今回のプログラムはspsc::QueueやBoxの概念と使い方の説明に着目するため（とくにマイコンでないと動作しないというものでもないため），ホストPC上で動作させます．

後半紹介するBoxサンプルの都合上，32ビットのx86アーキテクチャでビルド，実行しています．Linuxの場合は次のコマンドでRustのクロスビルド・ターゲットを追加してください．

```
$ rustup target add i686-unknown-
                          linux-gnu↵
```

今回作成したプログラムは参考文献(2)からダウンロードできます．

● データの入出力方法に応じて選べる2つの使い方

spsc::Queueには2つの使い方があります．

▶1，普通のキューとして

まずは普通のキューとしての使い方です．同じタスクやスレッドからキューを操作する場合はこの使い方がおすすめです．

リスト1では，enqueue()で要素を追加した順番に，dequeue()で要素を取り出せます．

1点，指定した容量の$N-1$の要素までしか格納できないことに注意しましょう[注1]．**リスト1**でQueue<u8，4>とした場合，要素は3つまで格納できます．