特集2

第2章 組み込みRustではおなじみのベアメタル環境とは違う！

std Rustを使う前に知っておきたい…no_stdとの違い

中林 智之

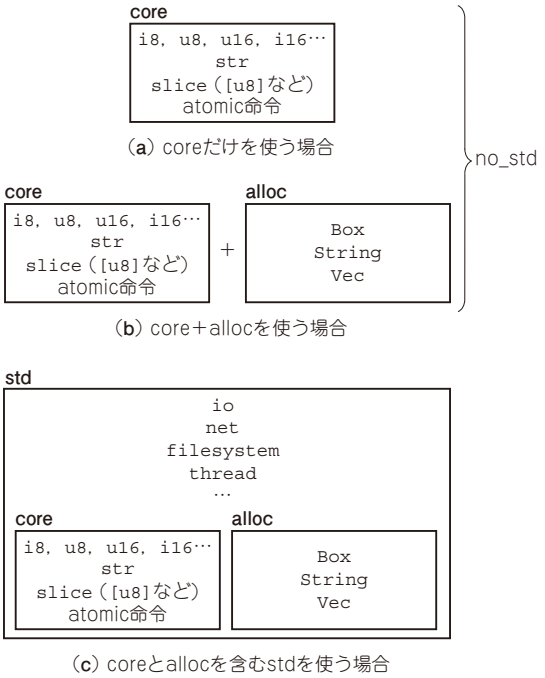

図1 Rustの標準ライブラリには3通りの構成がある

Rustの開発環境は主にPCで使えるフルセットのstdと，マイコン向けのサブセットno_stdがあります．有志の活動により，ESP32はマイコンであるにも関わらずstdも使えるようになりました．ESP32で使えるRustの開発環境はstdが使えるesp-idf-halクレート（ライブラリ）・ベースのものと，no_stdなesp-halクレート・ベースのものとが存在しています．この2つの環境をどのように使い分けるとよいのでしょうか．それぞれの環境を比較してみましょう．

Rustの標準ライブラリ

Rustの標準ライブラリは**図1**のようにcoreだけを使う，core + allocを使う，coreとallocを含むstdを使う，のいずれかの状態でRustプログラムを作成します．

ホストPCで動作するRustのプログラムを書く際は特に意識することはなく，標準ライブラリの全ての機能が含まれたstdクレートを使います．

stdクレートはOSに依存した機能を提供しており，そのような機能は組み込みシステムでは使えません．そこで組み込みシステムやベアメタルでのプログラミングでは，stdのサブセットであるcoreクレートとallocクレートだけを使います．

このような，stdを使わない状態でRustプログラミングする環境をno_stdと呼んでいます．

no_std（ベアメタル）環境

● no_stdは組み込みRustに最適とも言えるけど

no_stdは組み込みRustではおなじみの環境です．ベアメタルとは，マイコンを使った組み込みシステムやOS自体，ブートローダのプログラムを書くような，OSが存在しない状態を意味しています[注1]．

no_stdな開発では，Rustの標準ライブラリであるstdを使いません．その代わりcoreのみ，またはallocを加えたstdのサブセットで開発します．

coreはどのような環境でも使えるstdのサブセットですが，coreにはOSに依存しない基本的な型やCPU命令を使ったアトミック処理のみと，最小限のものしか含まれていません．

allocはメモリ・アロケータを実装することで使えるようになるstdのサブセットです．allocにはヒープ領域のメモリを扱うスマート・ポインタBoxや可変長配列Vecのようなコレクションが含まれています．

● no_stdのメリットとデメリット

no_stdな環境での開発はstdの機能そのものが使え

注1：本誌2020年5月号 特集「C/C++後継モダン言語の研究」や，拙著『基礎から学ぶ組込みRust』でも，no_std（ベアメタル）を前提とした組み込みRustのファームウェア開発を取り上げています．