岸 哲夫

コマンドのまとめ（その2）

コマンドの概略も2回目です．前回（2009年10月号，pp.135-139）はメモリ内部の調査をしましたが，今度は動きを調べます．スタックという領域には呼び出した関数のアドレスや戻り先，引き数などが記録されています．スタック領域の調査について記述します．（筆者）

1. スタックの検査について

● スタックの動作

具体的なスタックの実例を説明します．

スタックとは抽象データ型でありデータを後入れ先出しで保持するものです．データを先入れ先出しで保持するものはキューと呼ばれています．やはり代表的な抽象データ型の一つです．スタックが「棚」ならキューは「行列」です．

スタックは関数を呼び出した際に，戻りアドレスを格納したり，引き数を格納したり，ローカル変数を格納したりすることに使われます．

スタックにデータを保存したり，取り出したりするようすは，ちょうど食堂のトレーを積み重ねているのに似ています．トレー（データ）は上へ上へと積まれていき，取り出すときは，上から順に取り出します．積んであるトレー（データ）を途中から取ることはできません．

データをスタックに保存することをプッシュ（push）すると呼びます，またスタックからデータを取り出すことをポップ（pop）するといいます．

メモリ上に積まれているようすを，簡単なプログラムを使って説明します．

リスト1を実行すると**図1**のようになります．

GDBで実行し，スタックの中身を見ると**図2**のようになります．メモリにはこのような形で格納されています．

● スタックにサブルーチンの情報を保持する

さてプログラム実行中にサブルーチンの情報を持つとき，そのメモリ・エリアをコール・スタックと呼びます．前にも書いたとおり，関数を呼び出した際に戻りアドレスを格納したり，引き数を格納したり，ローカル変数を格納したりすることに使

リスト1　スタックの動作を見るプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define STACK_END  "end"    /* スタックの終わりを表すデータ */
/* 関数のプロトタイプ宣言 */
char    *push(char *);      /* スタックにデータを積む */
char    *pop(void);         /* スタックからデータを取り出す */
void    print_stack(void);  /* スタックの内容をすべて表示する */
/* グローバル変数 */
char     *stack;            /* スタック領域 2000バイトもいらないが */
int     stack_pointer = 0;  /* スタック・ポインタ */

//メイン処理
void main(void)
{
    char    buf[10];
    char    data[] = { "1112223334335556667778889999end" };
    int     x,b;
    char    *n;
    char    *dum;
    stack = (char *)malloc(2000);

    for(x = 0; x < 10; ++x){
        strncpy((char  *)buf,(const char*)data+(x*3),(size_t)3);
        dum=push(buf);
    }
    printf("積まれたスタック¥n");
    print_stack();
    printf("¥n");
    printf("取り出されたスタック¥n");
    for(x = 0; x < 10; ++x){
        n = pop();
        printf("%.*s¥n", 3, n);
    }
}
//dataの内容を3バイトで切ってstackに積む
char *push(char *d)
{
    if(d == STACK_END)
        return(STACK_END);
    strncpy(stack+(stack_pointer*3),d,3);
    ++stack_pointer;
    return(NULL);
}
//stackのstack_pointerに応じた内容を戻す
char *pop(void)
{
    if(stack_pointer == 0)
        return(STACK_END);
    --stack_pointer;
    return(stack+(stack_pointer*3));
}
//stackの内容をprintする
void print_stack(void)
{
    int     i;
    for(i = 0; i < 10; ++i){
        printf("%.*s¥n", 3, stack+(i*3));
    }
}
```


KEYWORD ── スタック，出力フォーマット，配列と構造体の表示，ブレークポイント，プロンプト，シグナル