```
/*** RE_CHECK    re_check.c   2005-11-8 ***/
  /* a=$(./re_check "$a") という形で使用*/
  /* echo $a にて  確認    値が変わって行きます*/
/*** 最終結果は res=$(echo "$a" cut -f 1) (数値です)とやって  また   ***/
/*** 表示では str_res=$(echo "$a" cut -f 2) (文字列です)とやって  使います   ***/
/***  "" のあることに注意せよ!!    フォーマットどおりにするため ***/
/***   コマンド引数で  初期設定可能にしました  ***/
/*** コマンド引数  無しだとhelp表示もに    という具合です     ***/
/*** min,maxも設定すれば自動判定、list抽出では循環します     ***/
/***  設定中の値には 変化ないとき blink 表示の試みも  ***/
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
char *help_msg0 = "\t\t\t\t\t\t\t\t\t\t";          /*** \nが  おまけの cut で邪魔をするので こんな改行表示ですいません
 ***/
char *help_msg1 = "   a=$\(./re_check \"$a\") という形で通常ループ内にて使用します  \(返り値:TAB区切り文字列)\t";
char *help_msg2 = "結果(数値)は     res_a=$\(echo \"$a\" cut -f 1) とかやって  また\t\t";
char *help_msg3 = "表示(文字列)は  disp_a=$\(echo \"$a\" cut -f 2) とかやって  使います\t\t";
char *help_msg4 = "初期設定  数値 : a=$(./re_check n_init [n_min [n_max]])  n:整数  []部分省略可\t\t";
char *help_msg5 = "         Choise : a=$(./re_check n_init List: c_str1 c_str2 ..)  c:文字列\t\t";
char *help_msg6 = "リセット更新  : a=$\(./re_check \"$a\"  -R [n])  REデータ複数使用の切り替え時\t\t\t\t";
char *help_msg7 = "    おまけ     ./re_check cut -f 1 で  このhelp_msgの頭の RE現在値だけを拾えます";
char re_read(char *cptr);
char c;
char str[256];
char *field = "%d\t%s\t%d\t%d\t%d\t%d\t%d\t%s\n";          /*** 最終結果を頭にした  ***/
/*** 1:res 2:str_res(表示用/点滅) 3:old_re 4:new_re 5:n_min 6:n_max 7:n_step 8:str_choiseの順 tab区切り ***/

main(int  argc , char *argv[ ])
{
int i,n,val ,old_re,new_re,n_min,n_max,n_step;          /* Armadilloのintは32ビットのワードで  -2,147,483,648～2,147,483,647 で
した*/
char str_val[128];
char str_blk[128];    /*** str_val と同じ文字長のblank(_)文字列用  ***/
char str_choise[256]="";
char *ptr_choise;    /*** str_choise のポインタ  ***/

                    re_read(&c); /* ポインター渡しで  ＲＥデータ貰う */

switch(argc){
case 2:             /*argcが2(引数ひとつ) ,つまりre_check "$a" のケースが次  re_check 123 という初期設定も *1 へ*/
        switch(sscanf(argv[1],field,&val,str_val,&old_re,&new_re,&n_min,&n_max,&n_step,str_choise) ) { /*変数に代入*/
        case 7:             /*7番目&n_stepまで  LIST,str_choiseが空のとき  ここまであるのが正規*/
                    old_re=new_re;                  /* re_check "$a"では  以前の値をこの後使いますので*/
                    new_re=c;
                    n=new_re - old_re;

                    if (n < -48){
                            n=n+96;
                    }
                    else if (n >= 48){
                            n=n-96;
                    }
                    else {}

         val=val+n;

                    if (val < n_min){
                            switch(n_step){
                            case 0:
                             val=n_max;
                             break;
                            default :
                             val=n_min;
                            }
                    }
                    else if (val > n_max){
                            switch(n_step){
                            case 0:
                             val=n_min;
                             break;
                            default :
```

```
                        val=n_max;
                    }
                }
                else {}

                sprintf(str_blk, "%d", val);  /*str_blkにvalの値を文字列でセーブ*/
                i=0;
                        if (strcmp(str_blk, str_val) == 0)          /* blinkさせるか 判定 */
                        {
                        while (str_blk[i] != '\0')          /*"null"に出会うまで */
                                str_val[i++]='_';       /*"_"を 一文字ずつ転記 */
                        }
                        else
                        {
                        while (str_blk[i] != '\0')          /*"null"に出会うまで */
                                str_val[i++]=str_blk[i];            /*一文字ずつ転記 */
                        }
                        str_val[i]='\0';                    /*ラストにヌル記入 */

                break;

        case 8:             /*8番目  LIST,str_choise までが有りのとき  ここまであるのが正規*/
                old_re=new_re;                  /* re_check "$a"では  以前の値をこの後使いますので*/
                new_re=c;
                n=new_re - old_re;

                if (n < -48){
                        n=n+96;
                }
                else if (n >= 48){
                        n=n-96;
                }
                else {}

                if (n < 0){ /*必ず  順番どおりひとつずつにしました */
                        n=-1;
                }
                else if (n > 0){
                        n=1;
                }
                else {}

         val=val+n;

                if (val < n_min){
                        switch(n_step){
                        case 0:
                         val=n_max;
                         break;
                        default :
                         val=n_min;
                        }
                }
                else if (val > n_max){
                        switch(n_step){
                        case 0:
                         val=n_min;
                         break;
                        default :
                         val=n_max;
                        }
                }
                else {}

/*さて  choise 結果を用意しよう " "検出させながら1文字ずつ転記する方法を*/
                        ptr_choise = str_choise;        /*str_choise[]の先頭を指す */
                                for (i = val; i > 1; --i)
                                {
                                while (*ptr_choise != ' ')
                                        ptr_choise++;       /*" "に出会うまでポインタ進める */
                                        ptr_choise++;       /*もう一つ進め  次の先頭に */
                                }
                                 i=0;
                                while (*ptr_choise != ' ')      /*" "に出会うまで */
```

```
                                        str_blk[i++]=*ptr_choise++;     /*" "str_valに一文字ずつ転記 */
                                        str_blk[i]='\0';                /*ラストにヌル記入 */
                                                i=0;
                                        if (strcmp(str_blk, str_val) == 0)          /* blinkさせるか 判定 */
                                        {
                                        while (str_blk[i] != '\0')          /*"null"に出会うまで */
                                                        str_val[i++]='_';       /*"_"を 一文字ずつ転記 */
                                        }
                                        else
                                        {
                                        while (str_blk[i] != '\0')          /*"null"に出会うまで */
                                                        str_val[i++]=str_blk[i];            /*一文字ずつ転記 */
                                        }
                                        str_val[i]='\0';                /*ラストにヌル記入 */
                        break;

        case 0:   /* re_check --help などの(文字)形  数値でないとscanfはゼロを返す */
                                        goto Help_Message;

        case 2:                 /*argcが 2 で  その引数が数値ひとつ, つまり re_check 123 という初期設定  *1 */
                        sprintf(str_val, "%d", val);  /*n2にvalの値を文字列でセーブ*/
                        new_re=c;
                        old_re=c;
                        n_min=-999999;
                        n_max=9999999;
                        n_step=1;
                        break;

        default:                /*argcが 2("$a"引数ひとつ) なのだが 正規のフォーマットではない *1 */

                                        goto Help_Message;
        }
        break;

case 1: /* re_checkだけの「コマンドライン引数」が  無いとき Help_Message  他の出力の方がいい？*/
                                        goto Help_Message;

case 3:                                 /*「コマンドライン引数」が2:  re_check "$a" -R (リセット) の形 */
case 4:                                 /*    あるいは 数値設定での min max step の値の設定*/
default:                        /* そして  re_read 3 -List: a bb ccc "d EE" F とか 列記された候補のループ式チョイス*/
        switch(sscanf(argv[1],field,&val,str_val,&old_re,&new_re,&n_min,&n_max,&n_step,str_choise) ) { /*変数に代入*/
        case 7:                 /*re_check "$a" -R (リセット) で 7番目まで(LISTが空)のとき*/
                        old_re=c;               /*最新のRE値に更新*/
                        new_re=c;
                        if(argc > 3){                                                   /*re_check "$a" -R
[n] : n無しなら RE値更新だけ  */
                         sscanf(argv[3],"%d",&val);     /* RE値更新にあわせて nあれば  初期値も再設定*/
                        if (val < n_min){       /*          val  がちゃんと min max の間に収まっていますか?  */
                                switch(n_step){
                                case 0:
                                 val=n_max;
                                 break;

                                default :
                                 val=n_min;
                                }
                        }
                        else if (val > n_max){
                                switch(n_step){
                                case 0:
                                 val=n_min;
                                 break;
                                default :
                                 val=n_max;
                                }
                        }
                        else {}
                                }
                        sprintf(str_val, "%d", val);  /*str_valにvalの値を文字列でセーブ*/
                        break;

        case 8:                 /*re_check "$a" -R (リセット)  で 8番目まで(LISTあり)のとき  */
                        old_re=c;               /*最新のRE値に更新*/
                        new_re=c;
```

```
                        if(argc > 3){                                   /*re_check "$a" -R [n] : n無しなら RE値更新だけ  */
                         sscanf(argv[3],"%d",&val);      /* RE値更新にあわせて nあれば  初期値も再設定*/
                if (val < n_min){       /*          val  がちゃんと min max の間に収まっていますか?  */
                        switch(n_step){
                        case 0:
                         val=n_max;
                         break;

                        default :
                         val=n_min;
                        }
                }
                else if (val > n_max){
                        switch(n_step){
                        case 0:
                         val=n_min;
                         break;
                        default :
                         val=n_max;
                        }
                }
                else {}
                        }
/*さて  choise 結果を用意しよう " "検出させながら1文字ずつ転記する方法を*/
                        ptr_choise = str_choise;          /*str_choise[]の先頭を指す */
                                for (i = val; i > 1; --i)
                                {
                                while (*ptr_choise != ' ')
                                        ptr_choise++;         /*" "に出会うまでポインタ進める */
                                        ptr_choise++;         /*もう一つ進め  次の先頭に */
                                }
                                 i=0;
                                while (*ptr_choise != ' ')        /*" "に出会うまで */
                                str_val[i++]=*ptr_choise++;     /*" "str_valに一文字ずつ転記 */
                                str_val[i]='\0';                /*ラストにヌル記入 */
                break;

        case 0:  /* re_check -h, --Help とかargv[1]  数値でないとscanfはゼロを返す */
                printf("%s ## Not Numeric(2)##  Show Help-Message\n" , argv[1] ); /* HELP表示*/
                exit(0);

        default:                /*re_check 3 List: a bb ccc "d EE" F 列記された候補のチョイスの初期設定 */
                                                        /*re_check 3 0 99 1  min max step式の初期設定 */

                switch(sscanf(argv[2],"Lis%s",str_val) ) { /*変数に代入     いまは仮です*/
                case 1:   /* re_check 3 "$zz"  scanfは 成功した個数を返す "Lis%s"のフォーマットに限定*/
                                for (i = 3; i < argc; ++i)
                {
                strcat(str_choise, argv[i]); /*str_choiseに文字列" "区切りでセーブ   */
                strcat(str_choise, " ");
                }

 /*さて  choise 結果を用意しよう " "検出させながら1文字ずつ転記する方法を*/
                        ptr_choise = str_choise;          /*str_choise[]の先頭を指す */
                                for (i = val; i > 1; --i)
                                {
                                while (*ptr_choise != ' ')
                                        ptr_choise++;         /*" "に出会うまでポインタ進める */
                                        ptr_choise++;         /*もう一つ進め  次の先頭に */
                                }
                                 i=0;
                                while (*ptr_choise != ' ')        /*" "に出会うまで */
                                str_val[i++]=*ptr_choise++;     /*" "str_valに一文字ずつ転記 */
                                str_val[i]='\0';                /*ラストにヌル記入 */

                old_re=c;               /*最新のRE値に更新*/
                new_re=c;
                n_min=1;
                n_max=argc-3;           /*マイナス3は  リストの数にするためです*/
                n_step=0;
                break;

/*          re_read 3 0 5 1 の形 という初期設定   */
```

```
                        default:    /*          re_read 3 0 5 1の形 という初期設定   */
                                    switch(sscanf(argv[2],"%d",&n_min) ) {
                                    case 1: /*argv[2] は  数字*/
                                    n_max=9999999;
                                    n_step=1;
                                    if(argc > 3) {sscanf(argv[3],"%d",&n_max);}
                                    if(argc > 4) {sscanf(argv[4],"%d",&n_step);}

                        if (val < n_min){       /*          val  がちゃんと min max の間に収まっていますか?   */
                                    switch(n_step){
                                    case 0:
                                     val=n_max;
                                     break;

                                    default :
                                     val=n_min;
                                    }
                        }
                        else if (val > n_max){
                                    switch(n_step){
                                    case 0:
                                     val=n_min;
                                     break;
                                    default :
                                     val=n_max;
                                    }
                        }
                        else {}

                        sprintf(str_val, "%d", val);  /*str_valにvalの値を文字列でセーブ*/
                                    old_re=c;                 /*最新のRE値に更新*/
                                    new_re=c;
                                    break;

                                    default: /*argv[2] は  数字ではない error_return*/
                                                goto Help_Message;
                                                }
                        }
            break;
            }
            break;
}

 /* この結果表示 forTESTです  fieldで  %.8などと  精度の項だけ指定しました*/
            printf(field,val,str_val,old_re,new_re,n_min,n_max,n_step,str_choise); /* 入力  結果表示 for 返り値*/
  exit(0);
Help_Message:
                                                            printf("%d\t%Xh\t<--  RE現在値です
\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t\n",c,c,help_msg0,help_msg1,help_msg0,help_msg2,help_msg0,h
elp_msg3,help_msg0,help_msg4,help_msg0,help_msg5,help_msg0,help_msg6,help_msg0,help_msg7);
                                    exit(1);
}
char re_read(char *cptr)
{
char r[2] = {0,0}; /* 入力側のバッファ */
char w[2] = "\22";   /* 出力文字(cont+R) PIC_LCD端末からのRE読み取り用*/
 FILE *input;
  FILE *output;
  struct termios initial_settings, new_settings;

  /* ttyS1ファイルディスクリプタのオープン処理 */
    input = fopen("/dev/ttyS1", "r");
    output = fopen("/dev/ttyS1", "w");
    if(!input  ||!output) {
        fprintf(stderr, "Unable to open /dev/tty\n");
        exit(1);
    }

    tcgetattr(fileno(input),&initial_settings);
    new_settings = initial_settings;
    new_settings.c_lflag &= ~ICANON;
    new_settings.c_lflag &= ~ECHO;
    new_settings.c_cc[VMIN] = 1;
    new_settings.c_cc[VTIME] = 0;
```

```
    new_settings.c_lflag &= ~SIG;
    if(tcsetattr(fileno(input), TCSANOW, &new_settings) != 0) {
        fprintf(stderr,"could not set attributes\n");
    }


            write(fileno(output), w, 1); /* 1文字出力 */
//          sleep (1e-1);
            read(fileno(input), r, 1);  /* 1文字入力 */
                        *cptr =  r[0]; /* 入力  結果  */

  tcsetattr(fileno(input),TCSANOW,&initial_settings);
  close(fileno(output));
  close(fileno(input));

}
```