計算機プログラミング'95 ＃5

久野　靖*

1995.10.4

1　画面端末プログラム (2)

1.1　反転モードの追加

前回は画面端末制御の基本部分だけで終わってしまったので、今回はもう少しアプリケーションらしいプログラムが作れるようにしてみる。その前に、screen モジュールを少し改良して、次の機能を入れてみよう。

- 強調 (反転) 表示ができるようにする。
- 文字列全体を書けるようにする。

このため、インタフェースを次のように変更する (いちど作ったのを変更するのはあまりよくないが。

```
scrinit() --- 画面パッケージの初期設定
scrend()  --- 画面パッケージの後しまつ
scrlines() --- 画面の行数を返す
scrcols()  --- 画面の文字数を返す
scrclear() --- 画面をクリアする
scrputc(int l, int c, int x, int h)
  --- l 行目の c 文字目に文字 x を入れる。その際 h で強調の有無を指定
scrputs(int l, int c, char *s, int h) --- 文字列を順次 put
scrputl(int l, int c, char *s, int h) --- 文字列を入れ行末までクリア
scrupdate() --- 画面を更新する
scrpos(int l, int c) --- カーソルを l 行目の c 文字目に
```

まずは、強調表示のやりかたを説明する。まず各文字をバイトでなく整数 (32 ビット) で表し、上位のビットに強調の有無を記録する。そして、scrpudate で画面端末に書き出す時、次のエスケープシーケンスで画面のモードを制御する。

```
ESC [ 7 m  --- 反転モードに
ESC [ 0 m  --- 通常モードに
```

ではプログラムを見てもらおう。

*筑波大学大学院経営システム科学専攻

```
/* screen.c --- screen package, with reverse */
#include <stdio.h>
#include <termios.h>
static char tibuf[1024];
static int **line;
static int **lineb;
static int co, li;
static struct termios ttb1, ttb2;
#define HL 0x80000000

/* scrinit -- initialize screen */
scrinit() {
  int i;
  if(tcgetattr(0, &ttb1) != 0) {
    fprintf(stderr, "cannot get termnal controls.\n"); exit(1); }
  ttb2 = ttb1;
  ttb2.c_iflag |= IGNBRK;
  ttb2.c_iflag &= ~ICRNL&~ISTRIP&~IXOFF;
  ttb2.c_oflag &= ~ONLCR&~OCRNL&~ONLRET;
  ttb2.c_cflag |= CS8;
  ttb2.c_cflag &= ~PARENB;
  ttb2.c_lflag |= TOSTOP;
  ttb2.c_lflag &= ~ISIG&~ICANON&~ECHO&~ECHOE&~ECHOK&~ECHONL&~ECHOCTL&~ECHOPRT;
  for(i = 0; i < NCCS; ++i) ttb2.c_cc[i] = -1;
  ttb2.c_cc[VMIN] = 1;
  ttb2.c_cc[VTIME] = 0;
  if(tgetent(tibuf, "ansi") <= 0) {
    fprintf(stderr, "cannot find terminfo entry.\n"); exit(1); }
  li = tgetnum("li");
  co = tgetnum("co");
  line = (int**)malloc(sizeof(char*)*li+1);
  lineb = (int**)malloc(sizeof(char*)*li+1);
  for(i = 1; i <= li; ++i) {
    line[i] = (int*)malloc(sizeof(int)*(co+1));
    lineb[i] = (int*)malloc(sizeof(int)*(co+1));
  }
  scrclear();
  tcsetattr(0, TCSANOW, &ttb2);
}

/* scrend -- finalize screen */
scrend() {
  tcsetattr(0, TCSADRAIN, &ttb1);
}

/* scrgetc -- read 1 char from keyboard */
```

```
scrgetc() {
  char buf[1]; read(0, buf, 1); return buf[0];
}

/* scrlines -- return no. of lines */
scrlines() { return li; }

/* scrcols -- return no. of columns */
scrcols() { return co; }

/* scrclear -- clear screen */
scrclear() {
  int i, j;
  for(i = 1; i <= li; ++i) {
    for(j = 1; j <= co; ++j) line[i][j] = ' ';
    for(j = 1; j <= co; ++j) lineb[i][j] = ' ';
  }
  printf("\033[2J"); fflush(stdout);
}

/* scrputc -- put a charcter on the screen */
scrputc(int l, int c, int x, int h) {
  x &= 0x7f; if(h) x |= HL;
  if(1 <= l && l <= li && 1 <= c && c <= co) line[l][c] = x;
}

/* scrputs -- put a string on the screen */
scrputs(int l, int c, char *s, int h) {
}

/* scrputl -- put a string on the screen */
scrputl(int l, int c, char *s, int h) {
}

/* scrupdate -- update screen actually */
scrupdate() {
  int i, j, l, r, h = 0;
  for(i = 1; i <= li; ++i) {
    for(l = 1; l <= co; ++l)
      if(line[i][l] != lineb[i][l]) break;
    for(r = co; r >= l; --r)
      if(line[i][r] != lineb[i][r]) break;
    if(l <= r) {
      printf("\033[%d;%dH", i, l);
      for(j = l; j <= r; ++j) {
        if(line[i][j]&HL) {
```

```
          if(!h) { printf("\033[7m"); h = 1; } }
        else {
          if(h) { printf("\033[0m"); h = 0; } }
        putchar(line[i][j]&0x7f); lineb[i][j] = line[i][j];
      }
    }
  }
  if(h) printf("\033[0m");
}

/* scrpos -- position cursor */
scrpos(int l, int c) {
  printf("\033[%d;%dH", l, c); fflush(stdout);
}
```

練習 1 簡単なメインプログラムを書いて反転表示がうまく行くことを確認せよ。

練習 2 scrputs と scrputl を実現せよ。メインプログラムを適宜直してテストすること。

1.2 メニュー機能

次に、これを利用してメニュー機能を実現してみる。そのインタフェースは次のようにする。

```
menuselect(char *mes, char *menu[], int n)
```

ここで mes はメニュー画面に表示されるメッセージ、menu はメニューの各選択項目を表す文字列が並んだ配列、n は選択肢の数。これらを指定して menuselect を呼ぶと menuselect はメニューを表示し、ユーザに項目を選択させ、選択番号を値として返す。これを使った簡単なプログラムを示しておこう。

```
/* t52.c --- menuselect test. */

char *m1[] = {
  "0. Male.",
  "1. Female.",
  "2. Neutral" };

main() {
  int i;
  scrinit();
  i = menuselect("Choose your sex.", m1, 3);
  scrend();
  printf("your selection was: %d\n", i);
}
```

練習 3 この menuselect をあなたならどう実現するか。設計せよ。

練習 4 menuselect を利用して、ユーザに簡単な選択式アンケート (たとえば出身学科とか好きな食品とか) を問い合わせるプログラムを書いてみよ。

練習 5 その場合、ひたすら書くのでなく、プログラムをうまく構造化できるアイデアはないか? あれば試してみよ。

1.3 フォーム機能の実現

入力のもう 1 つの形態は、フォーム (フィールドに値を記入してもらうようなもの) である。こちらも次のインタフェースを設計した。

```
formfill(char *mes, char *form[], int n)
```

ただし、配列 form はフィールド名と記入用の配列がペアで入るようにするので、大きさは menu の時の倍になる。またこちらは戻り値はなしになる。さっそくメインプログラムを見ていただく。

```
/* t53.c -- form filling test. */

char f_name[100] = "", f_age[100] = "", f_occ[100] = "";
char *f1[] = {
  "Name:", f_name,
  "Age:", f_age,
  "Occupation:", f_occ };

main() {
  int i;
  scrinit();
  i = formfill("Fill your info.", f1, 3);
  scrend();
  printf("your name:%s, age:%s, occupation:%s\n", f_name, f_age, f_occ);
}
```

練習 6 この formfill をあなたならどう実現するか。設計せよ。

練習 7 menuselect と formfill を利用して、簡単なアンケートのプログラムを書いてみよ。

練習 8 この種のプログラムを作る上で、そのほかにどんな「部品」があるといいと思うか? 立案し、作成してみよ。