Zasada odświeżania ekranu

- zainicjalizuj WINDOW dla reprezentacji okna na ekranie
- wprowadź i przemieść znaki w tablicy znaków
- odśwież (refresh) dla wyświetlenia zaktualizowanego okna na ekranie
- wprowadź następne zmiany do okna
- i ponownie odśwież ekran

Plik nagłówkowy

curses składa się z biblioteki procedur w */usr/lib* i pliku nagłówkowego */usr/include/curses.h*.

Plik nagłówkowy musi być włączony do programu:

```
#include <curses.h>
```

¡curses.h¿ włącza inne pliki nagłówkowe:

```
#include <stdio.h>
#include<sgtty.h>
```

Zmienne zewnętrzne i definicje

```
#define bool char
#define reg register

#define ERR (0) /* zwracane, gdy funkcja zawodzi */
#define OK  (1) /* zwracane, gdy dziala prawidlowo */
#define TRUE (1) /* boolowska prawda */
#define FALSE (0) /* boolowski falsz */
```

LINES określa długość ekranu
COLS określa szerokość ekranu

Jeśli nie określimy ich w programie, to są one pobierane z bazy terminali.

Trzy zmienne związane z terminalem:

`Def_term`	używana do określenia domyślnego terminala. Np. może być określona, gdy **initscr()** nie może znaleźć zmiennej TERM w otoczeniu użytkownika
`My_term`	używane do określenia własnego typu terminala
ttytype	używane do zapamiętania nazwy aktualnego terminala

Pseudo-funkcje

Jeśli mamy wiele okien, to funkcje muszą pobierać w argumencie wskaźnik do okna. Jednak, gdy działamy wyłącznie w oknie domyślnym, to istnieje zbiór pseudo-funkcji oddziaływujących automatycznie na *stdscr*.

Np.

```
refresh() - odswieza stdscr
addch()   - dodaje znak do stdscr
move()    - przesuwa kursor w stdscr
```

Ich odpowiedniki działające na okna, są poprzedzane literą **w**, np.:

```
wrefresh()
waddch()
wmove()
```

Wskaźnik okna jest zawsze pierwszym argumentem.

```
#define addch(CH)  waddch(stdscr,CH)
```

Druga grupa pseudo-funkcji ma na celu skrócenie kodu:

```
#define mvaddch(Y,X,CH)    mvwaddch(stdscr, Y, X, CH)
#define mvwaddch(WIN,Y,X,CH)    wmove(WIN, Y, X);\
                                waddch(WIN, CH)
```

Pseudo-funkcje są używane również do zdefiniowania **getxy()**, **inch()**, **clearok()**, **leaveok()**, **scrollok()**, **flushok()**.

UWAGA: argument WIN we wszystkich funkcjach *curses* występuje przed współrzędnymi Y,X, a współrzędna Y zawsze przed współrzędną X.

Użycie biblioteki *curses*

Program źródłowy musi zawierać linię:

```
#include <curses.h>
```

i musi być skompilowany z:

```
cc [flagi] -lcurses nazwa-pliku
```

albo zlinkowany z:

```
ld [flagi] -lcurses nazwa-pliku
```

Jeśli *curses* are przygotowane do użycia *termcap*, to należy dodać opcję **-ltermlib**.

Narzędzia lokalizacji kursora

move() przesuwa położenie logicznego kursora w *stdscr*. Jest to *# define* makro for **wmove()**. Współrzędne (y,x) są wyliczane względem okna, nie ekranu.

getyx() inne makro, które zwraca lokalizację (y,x) kursora logicznego. Ze względu na sposób napisania makra, y i x muszą być zmiennymi całkowitymi, a nie adresami zmiennych całkowitych.

inch() zwraca znak z tablicy obrazu ekranu spod położenia kursora logicznego. Makro dla **winch(stdscr)**.

leaveok() wskaźnik boolowski _ **leave** w danym oknie. Jeśli _ **leave** jest ujstyawiony, to **refresh()** przesuwa kursor logiczny do końcowego położenia kursora fizycznego po odświeżeniu ekranu. W przeciwnym przypadku kursor fizyczny będzie pozostawiony w miejscu wskazywanym przez kursor logiczny.

Tryb pracy wyróżniony (*standout*)

standout()	rozpoczyna wyróżnianie wprowadzanych znaków
standend()	powrót do normalnego trybu wprowadzania znaków

Pseudo-funkcje dla *wstandout() i wstandend()*. W trybie wyróżniania, jeśli terminal nie ma takiego trybu, to stara się znaki podkreślać, a jeśli nie może podkreślać, to ignoruje polecenie.

Jedną z _ flag w WINDOW jest _ STANDOUT, która mówi addch() o trybie wprowadzania znaków.

curses używają najwyższego bitu bajtu dla zaznaczenia wyróżnienia znaku. Funckja **redisplay()** dokonuje konwersji znaków przed wysłaniem na ekran.

Operacje na oknach

Zasadniczą różnicą między *overlay() i overwrite()* jest to, że pierwsza pomija znaki spacji, podczas gdy druga znaki spacji również przepisuje do okna numer dwa.

touchwin() ustawia _ *firtsch = 0* i _ *lastch = (_ maxx-1)* dla każdej linii okna. To mówi funkcji **refresh()**, że każdy znak obrazu ekranu mógł być zmieniony.

Gdy pracujemy z nakładającymi się oknami, to *touchwin()* wymusza na *refresh()* wyświetlenie okna znajdującego się pod spodem.

touchwin() jest również użyteczne, gdy mamy kilka sub-okien. Można wywołać *touchwin()* dla okna-rodzica i następnie *refresh()* dla okna-rodzica.

Funkcje specjalne

mvcur(int oldy, int oldx, int newy, int newx)

przesuwa fizyczny kursor ze starej pozycji (y,x) do nowej pozycji (y,x).

unctrl(char ch)
zwraca lańcuch dwuznakowy "x̂", gdy dany jest znak CONTROL-x.
Dla korzystania z *uunctrl()* konieczne jest:

```
#include <unctrl.h>
```

scrollok(WINDOW *win, bool flag)
Ustawia flagę _ *flag* w wyspecyfikowanym oknie jako TRUE albo FALSE.

scroll(WINDOW *win)
wykonuje właściwe logiczne przewinięcie okna; jest wywoływany automatycznie w razie potrzeby. Użytkownik może wywołać *scroll()*, gdy jest to porządany przez nas efekt specjalny.

Manipulowanie terminalem

gettmode()	odczytuje status tty
savetty()	zapamiętuje status tty
resetty()	przywraca status zapamiętany przez *savetty()*

```
#include <curses.h>
#include <signal.h>
main()
{
  void die();
  /* if (startup() != ERR startup unwritten  */
  /* init stdscr, curscr and terminal  */

  initscr();

  /*  call die() if get interrupt  */

  signal(SIGINT, die);

  /*  spreadsheet function defined later  */

  spreadsheet();
  die();
}

void die()
{
  /*  ignore interrrupts  */

  sigignore(SIGINT);

  /*  move cursor to lower left  */

  mvcur(0,COLS-1, LINES-1, 0);

  /*  make terminal the way it was */

  endwin();
  exit(0);  /*  exit normally  */
}

#define VERSION 12

spreadsheet()
{
  char filename[20];

  intro(VERSION);  /* print first screen */
  /*              initialize input modes */
  initinput();
  /*             read in file, check password */

  getfileandpw(filename);

  /*  create the fields for spreadsheet  */

  makefields();
  /*   makefields also calls driver  */
}

/*  intro prints the initial screen  */

intro(version)
int version;
{
  int x1,x2,y;
  /*   make box around the screen  */

  box(stdscr, '*','^');
  move(1,2);
  addstr("SHOWME Spreadsheet Program, version: ");
  getyx(stdscr, y, x1);
  printw("%d", version);
  getyx(stdscr, y, x2);
  while (x2++ - x1 < 6)
    addch(' ');
  addstr("1/1/86");
  /* send stdscr to terminal  */
  refresh();
}
initinput()   /*  set up initial modes  */
{
  raw();
  echo();
  nonl();
}

getfileandpw(filename)
char *filename;
{
  char passwd[6];
  int i=0;
  mvaddstr(4,2, "What file do you want? ");
  refresh();
  while ((filename[i] = getch()) != '\r')
    i++;
  filename[i]='\0';
  mvaddstr(3,2, "Six character password? ");
  noecho();
  refresh();
  i=0;
  while (i<6)
    passwd[i++]=getch();
  /*  check password */
  if (passwdok(passwd)) {
    mvaddstr(4,26, "OK");
    /* readinffile(filename); readinffile unwritten*/
    refresh();
  }
  else {
    mvaddstr(4,26, "NO");
    refresh();
    die();
  }
}
/* dummy function so that example will run  */
passwdok(passwd)
char * passwd;
{
  return (1);
}

int _top, _left;

makefields()
{
  WINDOW *worksp, *field[10][10];
  int i, j;
  extern int _left, _top;

  worksp=subwin(stdscr, LINES-2, COLS-2, 1, 1);
```

```
  /* create a subwindow to work in  */

  werase(worksp);
  wmove(worksp,1,0);
  waddstr(worksp, "\t0\t1\t2\t3\t4\t5\t6\t7");
  wmove(worksp, 3, 0);
  waddch(worksp, 'a');
  mvwaddch(worksp, 5, 0, 'b');
  mvwaddch(worksp, 7, 0, 'c');
  mvwaddch(worksp, 9, 0, 'd');
  mvwaddch(worksp,11, 0, 'e');
  mvwaddch(worksp,13, 0, 'f');
  mvwaddch(worksp,15, 0, 'g');
  mvwaddch(worksp,17, 0, 'h');
  mvwaddch(worksp,19, 0, 'i');
  mvwaddch(worksp,21, 0, 'j');
  wrefresh(worksp);

  /*  display border of workspace  */

  for (i=0; i<10; i++)
    for (j=0; j<10; j++){
      field[i][j]=newwin(1,7,4+2*i, 7+8*j);
      wprintw(field[i][j], "%d %d", i, j);

      /*  create fields to work in  */
    }
  _left=0; _top = 0;
  reffields(field);
  /*  iodriver(worksp, field);  iodriver unwritten  */
}

/*  reffields refreshes the fields
    currently positioned in the work space */

reffields(field)
WINDOW *field[10][10];
{
  int i,j;
  extern int _top,_left;
  for (i=_top; i<= _top+7; i++)
    for (j=_left; j<=_left+7; j++) {
      touchwin(field[i][j]);
      wrefresh(field[i][j]);
    }
}
```