Unix/Linux 화면 조작 (curses.h)

http://blog.naver.com/neutrino37/110009868351
http://blog.naver.com/iku88/120019415939
출처: http://blog.naver.com/kamvo.do?Redirect=Log&logNo=60003596240

제10장 화면조작

10.1. 서론

CRT/VDU 단말기의 화면을 다루는 도구들을 소개한다. 화면을 다루는 도구에는 밀접하게 관련된 두개 의 C 라이브러리가 있다.

1. curses
이것은 하드웨어에 의존하지 않고 화면을 조작하는 방법이다. 이 라이브러리는 단말기의 하드웨어에 무관한 자료구조인 윈도우(window)를 제공한다. curses 라는 이름 역시 최적화 커서 이동 (cursor motion optimization)에서 따온 것이다. 이것은 AT&T 시스템 V 인터페이스 정의 (SVID) 의 Issue 2 에 정의되어 있다.

2. terminfo
이 라이브러리는 curses 와는 달리 하드웨어와 밀접한 연관성을 가진 루틴을 제공한다. terminfo 역시 SVID 의 Issue 2 에 정의되어 있다.
curses 와 terminfo 둘다, 현재의 버전은 terminfo 라는 각각의 단말기의 특성을 기록해 두는 데이타베이스(database)를 사용한다.

/usr/lib/terminfo/<c>/<name>

<name> 은 실제 단말기의 이름이고 <c> 는 <name> 의 첫글자이다.
/usr/lib/terminfo/v/vt100 은 단말기 vt100 의 특성들을 기록해두는 화일이다.

curses 와 terminfo 의 루틴들은 먼저 환경 변수(environment variable) TERM 을 보고 단말기의 이름을 알아낸 다음, terminfo 데이타베이스에서 그 단말기에 대응하는 화일을 찾는다. termcap 라이브러리 terminfo 는 AT&T 에서 새로이 만들어 낸 것이다. 어떤 UNIX 시스템에서는 terminfo 대신 termcap 이나 termlib 가 있을 수도 있다. termcap 은 단말기 명세가 /etc/termcap 이라는 화일에 기록되어있다. 새로운 단말기에 대한 명세를 만들려면 그 화일을 사용자가 편집(edit)하면 된다. 소프트웨어 개발을 쉽게하기 위해 terminfo 라이브러리는 termcap 라이브러리보다 상위의 호환성을 가진다.

10.2. curses 라이브러리 : 개요

프로그래머는 윈도우라는 자료구조를 통해서 모든 작업을 하게 된다. 윈도우의 자료형은 WINDOW 인데, 이것은 표준 인크루드 화일(standard include file)인 curses.h 에 정의가 되어있다.
프로그래머는 newwin 이라는 루틴을 사용해서 새로운 윈도우를 만들 수 있다. 그리고 전역에서 사용할 수 있는 표준 윈도우 stdscr 도 있다. curses 에 대해 자세하게 설명하기 전에 두가지 점을 먼저 지적하고 넘어 가려고 한다.

1. curses 의 루틴들이 자동적으로 프로그램과 링크되는 것은 아니다. 따라서 컴파일할 때,사용자가 링크시켜야 한다.

$cc -o scrnprog scrnprog.c -lcurses
terminfo 에 대해서도 같은 방법으로 링크 한다.

2. curses 는 C 언어의 매크로(macro)를 이용 해서 정의되는 것이 많다. 따라서 프로그래머는 그것들을 사용할 때 주의해야 한다. 그리고 컴파일시 예상치 못한 오류 메시지에 대비해야 한다.

10.3. curses 의 일반적인 구조

모든 curses 프로그램은 기본적인 구조로 구성된다.

```
#include <curses.h>
main()
{
 initscr();
 /* main body */
 endwin();
 exit(0);
}
```

curses.h 는 curses 루틴을 사용하는 모든 프로그램에 항상 포함(include) 되어야 한다. 시스템 V 에서는 curses.h 에 헤드화일 terminfo.h 가 자동적으로 포함된다. curses.h 에는 curses 의 자료구조와 중요한 매크로들이 들어있다. initscr 루틴은 모든 curses 함수에 앞서서 수행시켜야 한다. 이 루틴은 curses 의 자료구조를 초기화하고 UNIX 환경의 TERM 변수를 통하여 단말기의 종류를 결정한다. 그렇게 함으로서 원래의 단말기 상태로 돌아갈 수 있고 커서는 맨 아래 왼쪽에 위치하게 된다.

```
#include <curses.h>
main()
{
 initscr();
 refresh();
 endwin();
}
```

refresh 또는 더 일반적인 버전인 wrefresh 를 수행시켜야만 윈도우의 내용이 실제화면 (physical screen) 에 나타나게 된다. refresh 는 표준윈도우 stdscr 의 내용을 실제화면에 표시하고, wrefresh 는 특정 윈도우의 내용을 실제화면에 표시한다.

10.4. 모드 지정

curses 루틴을 사용하는 프로그램에서는 initscr 을 수행시킨 다음 단말기의 입출력 모드를 지정한다.

```
echo(); /* enable echoing */
noecho(); /* disable echoing */
```

단말기의 반향(echo) 기능을 작동시키거나 정지 시킬 수 있다. 디폴트(default)로는 반향 기능 을 작동시킨다.

```
nl(); /* enable CR-NL mappings */
```

monl(); /* disable mappings */

nl 은 출력시 개행문자(newline) 가 newline/ carriage-return 으로 바뀌어서 출력되거나, 입력시 newline/carriage-return 이 개행문자로 바뀌어서 입력된다. default 는 nl 이다.

cbreak(); /* enable CBREAK mode */
nocbreak(); /* disable CBREAK mode */

cbreak 는 입력시, 인터럽트(interrupt) 와 흐름 제어 키(flow control key) 를 제외하고는 입력자료에 어떠한 작용도 가하지 않는다. raw(); /* enable RAW mode */ noraw(); /* disable RAW mode */ raw 는 단말기를 RAW 모드로 만든다. 이 모드는 CBREAK 모드와 비슷하다.

savetty(); /* save tty state */
resetty(); /* reset tty state */

단말기의 상태를 저장하거나 복구시킬 수 있다. savetty 는 현재 단말기 상태를 curses 의 내부버퍼에 저장해 준다. restty 는 바로 전에 savetty 로 저장해둔 상태를 복구시킨다.

10.5. 문자와 문자열 쓰기

curses 는 윈도우 stdscr 에 문자와 문자열을 쓰는 4 개의 루틴을 제공한다.
addch, mvaddch, addstr, mvaddstr 이다.

```
#include <curses.h>
int c, y, x;
char *string;
. .
addch(c);
mvaddch(y, x, c);
addstr(string);
mvaddstr(y, x, string);
```

addch 루틴은 stdscr 상의 현재 커서 위치에 문자 c 를 쓴다. refresh 를 해 주어야만 실제 화면에 나타난다. CTRL-C 는 ^C 처럼 쓰인다.
mvaddch 루틴은 커서를 세로로 y 번째 줄, 가로로 x 번째 열로 움직인 후에 글자를 쓴다. 맨 윗줄 왼쪽이 좌표(0,0)이다.
addstr 과 mvaddstr 은 0 으로 끝나는 문자열 string 을 윈도우 stdscr 에 쓴다.

10.6. 형식화된 출력

curses 에는 C 언어의 printf 와 비슷한 기능을 제공하는 루틴들이 있다. 표준 윈도우 stdscr 에 쓰이는 루틴들로는 printw 와 mvprintw 가 있다.

```
#include <curses.h>
char *fmt;
int y, x;    /* NB arg0, arg1 ... argn have arbitrary type */
. .
printw(fmt, arg0, arg1, ... argn);
```

mvprintw(y, x, fmt, arg0, arg1, ... argn);

10.7. 커서 이동

move 명령을 사용하여 표준 윈도우 stdscr 위 에서 커서를 움직일 수 있다.

```
#include <curses.h>
int y, x;
. .
move(y, x);
```

매개변수 y 와 x 를 사용하여 새로운 좌표에 커서를 위치시킬 수 있다. refresh 가 수행되어 야만 실제 커서 위치가 바뀐다.

getyx 루틴을 사용하여 커서의 현재 위치를 알수 있다.

```
#include <curses.h>
int y, x;
WINDOW *w;
. .
```

getyx 는 WINDO 에 대한 포인터가 첫번째 변수 로 주어져야 한다. 첫번째 변수로 표준 화면인 stdscr 을 사용하면 표준 화면상의 현재 커서 위치를 얻을 수 있다. 수행 결과는 y 와 x 변수 에 주어진다. getyx 가 실제 C 함수가 아니라 매크로(macro)이기 때문에 가능한 것이다.

10.8. 키보드 입력 : getch

curses 에서 키보드로 부터 하나의 문자를 받아 들일 경우에 getch 를 사용한다. getch 는 C 언어의 getc 루틴처럼 정수값을 돌려준다.

```
#include <curses.h>
int in_ch;
. .
in_ch = getch();
```

기능키 입력 처음 해야하는 작업은 curses keypad 루틴을 수행시켜서 단말기의 키패드(keypad)를 초기화시키는 것이다.

```
keypad(stdscr, TRUE);
```

특수키는 curses.h 에 정의된 값을 통해 돌려준다. ASCII 값과의 충돌을 막기 위해 이 값들은 8 진수 401 부터 시작된다.

```
int in_ch;
. .
in_ch = getch();
switch(in_ch)
{
  case KEY_DOWN; /* down arrow key processing */
```

```
 . .
 case KEY_UP; /* up arrow key processing */
 . .
}
```

10.9. 화면 입력 : inch

stdscr 의 특정 위치에 어떤 문자가 있는 지 알아야 될 경우가 있다. inch 는 화면의 현재 커서 위치에 있는 문자를 돌려준다. 그리고 mvinch 는 주어진 위치의 문자를 돌려준다.

```
#include <curses.h>
int in_ch;
. .
in_ch = inch();
in_ch = mvinch(y, x);
```

화면상의 글자에는 하이라이트 타입(highlight type) 같은 속성(attribute)을 지닌 경우가 있다. 이때 실제 문자만을 얻고 싶으면 curses.h 에서 정의된 상수 A_CHARTEXT 와 비트단위 AND 를 해야한다.

```
cvalue = ivalue & A_CHARTEXT;
```

10.10. 화면 편집

이미 그려진 화면을 변화시켜야 하는 경우가 종종 있다. curses 에는 이러한 경우에 유용 한 여러가지 루틴들이 있다. 이러한 루틴들을 3가지로 구분할 수 있다.
화면을 백지화(clear)시키는 것, 문자들을 지우고 화면을 재구성하는 것, 화면을 전혀 지우는 것없이 문자를 끼워 넣는 것등이 있다.

stdscr 상에서 화면의 일부분을 지우는 루틴

```
#include <curses.h>
. .
erase();
clear();
clrtobot();
clrtoeol();
```

erase 와 clear 은 둘다 표준 윈도우 stdscr 의 모든 위치에 공백(space)을 쓴다. 다른점은 clear 는 또한 다음번 refresh 가 호출될 때 화면을 자동적으로 백지화 시킨다는 점이다. clrtobot 는 현재줄의 커서 오른쪽에 있는 모든 문자와 커서 아래에 있는 모든 줄을 지운다. clrtoeol 은 현재줄에서 커서의 오른쪽에 있는 모든 문자를 지운다.

화면상의 문자들을 지울 뿐만 아니라 문자를 지우면서 생긴 공백을 메꾸기 위해서 화면을 이동시키는 루틴

```
#include <curses.h>
int y, x;
. .
delch();
```

```
mvdelch(y, x);
deleteln();
```

delch 의 경우 현재 커서 위치에 있는 문자가 지워진다. 그리고 그 공백을 메꾸기 위해 커서의 오른쪽에 있는 문자들을 한 칸씩 왼쪽으로 이동시킨다. mvdelch 도 기능이 거의 같다. 다만 먼저 커서를 주어진 위치에 이동시킨 다음 delch 와 같은 기능을 수행한다. deleteln 은 커서가 있는 줄을 지운다. 그리고 나서 그 줄 밑에 있던 모든 줄을 한 칸씩 위로 이동시킨다.

문자를 끼우는 데 관련된 루틴

```
#include <curses.h>
int c, y, x;
. .
insch(c);
mvinsch(y, x, c);
insertln();
```

insch 는 현재 위치에 문자 c 를 끼운다. 현재 커서 위치의 오른쪽에 있는 글자들은 모두 한 칸씩 오른쪽으로 이동한다. 그 줄의 마지막에 있는 문자는 잃어버리게 된다. mvinsch 는 커서의 위치를 바꾼 다음 insch 와 같은 작용을 한다. insertln 은 커서가 있는 줄의 위에 빈 줄(blank line)을 하나 집어넣는다. 그 줄 밑에 있는 줄들은 모두 한 줄씩 밑으로 이동한다. 따라서 화면의 맨 밑에 있던 줄은 화면상에서 사라지게 된다.

10.11. 영상 속성

curses 에서는 특정 모드로 글자를 화면에 나타 내고 싶으면 그 모드에 해당하는 상수와 비트 단위 OR 하면 된다.

```
addch(ch|A_BOLD);
```

는 ch 를 볼드체(bold;주위의 글들보다 더 밝은 모드) 로 나타낸다. 이외에도 여러가지 모드가 있다.
A_STANDOUT 이 모드는 글자를 집중모드로 나타낼때 쓰인다.
A_REVERSE 역전 모드 A_BOLD 글자들이 보울드로 나타난다.
A_DIM 글자들이 기본모드보다 약간 어둡게 나타난다.
A_UNDERLINE 글자 밑에 밑줄이 그어진 형태로 나타난다.
A_BLINK 글자들이 반짝인다.

```
#include <curses.h>
int atts;
. .
attrset(atts);
attron(atts);
attroff(atts);
standout();
standend();
```

attrset 은 표준 화면상에서 모드를 작동시킬때 사용하는 루틴이다.
attron 루틴은 atts 에 주어진 모드들을 작동시킨다. 이 루틴은 앞에서 지정된 모드를 바꾸지는 않는다. 마찬가지로 attroff 는 선택된 모드를 해제한다.

그리고 두 루틴 standout 와 standend 는 각각 attron(A_STANDOUT) 와 attroff(A_STANDOUT) 와 동일한 기능을 한다.

10.12. 새 윈도우 화면의 생성과 조작

새로운 윈도우를 다루는 방범을 소개한다. 윈도우를 위한 가장 기본적인 루틴은 newwin 이다.

```
#include <curses.h>
WINDOW *win;
int lines, cols, startline, startcol;
. .
win = newwin(lines, cols, startline, startcol);
```

이 루틴을 수행시키면 세로 크기가 lines, 가로 크기가 cols 인 윈도우가 생긴다.

```
wmove(win, y, x);
```
이 루틴을 수행하면 윈도우 win 상의 현재 커서 위치가 좌표(x,y)로 바뀌게 된다. 이 좌표는 표준 윈도우 stdscr 표가 아니라 윈도우 win 의 맨윗줄 맨왼쪽을 기준으로 한 좌표이다.

```
wrefresh(win);
```
은 윈도우 win 에 있는 내용들이 실제 화면에 나타난다.

10.13. curses 예 : domenu

-- 생략

10.14. 하드웨어 의존 단말기 조정 : terminfo

terminfo 라이브러리는 프로그래머가 직접 단말기의 하드웨어를 직접 제어하는 기능을 제공한다. terminfo 데이타베이스에는 각 단말기의 특성을 얻을 수 있다. 가장 크게 문제가 발생하는 경우가 패딩(padding, 속도늦춤)을 할 때다. 이단어는 단말기가 프로그램의 속도를 적당히 유지하도록 출력속도를 늦추는 것을 의미한다.

```
#include <curses.h>
#include <term.h>
main()
{
 setupterm(0, 1, 0);
 putp(clear_screen);
 reset_shell_mode();
 exit(0);
}
```

화면을 백지시키는 프로그램이다. term.h에는 clear_screen 을 포함하여 여러가지 기능을 나타내는 매크로들이 정의되어있다.
setr- pterm 은 termino 의 초기화루틴이다.
putp 을 호출하면 화면을 백지화시키기는 문자열을 화면에 내보낸다. putp 는 좀더 일반적인 기능을 갖는 루틴 tputs 의 제한된 버전이다.

reset_shell_mode 는 단말기를 원래의 상태로 되돌린다.