Программирование с помощью ncurses

As honour, love, obedience, troops of friends,
I must not look to have; but, in their stead,
Curses, not loud but deep, mouth-honour, breath,
Which the poor heart would fain deny, and dare not.

William Shakespeare, The Tragedy of Macbeth.

В прошлый раз мы научились управлять текстовой консолью с помощью интерфейса termios. Однако, для того, чтобы представить текстовый экран во всем великолепии, возможностей termios недостаточно. Сегодня мы поговорим о расширенном средстве управления терминалом – библиотеке ncurses. Библиотека ncurses и вправду заставляет терминал переливаться всеми цветами радуги (вот почему во всей серии статей, посвященных Unix API, эта статья – единственная, в которой вы найдете скриншоты).

Когда-то давно графические терминалы были редкостью, а пользователи текстовых терминалов хотели работать с интерфейсами, похожими не графические (и, самое главное, – использовать новое удобное средство ввода, – мышь). Специально для того, чтобы предоставить интерфейс «укажи и щелкни» пользователям текстовых терминалов, была разработана библиотека curses (ее название происходит от ее важнейшей функции – управления курсором, а вовсе не от проклятья, которая она накладывает на программистов). Изначально библиотека curses создавалась для BSD UNIX. В Linux используется открытый (на условиях MIT License) клон curses – библиотека ncurses (new curses). Приложений, использующих ncurses, в современных Linux-системах не так уж и много. Среди наиболее популярных проектов, основанных на ncurses, можно назвать Midnight Commander, текстовый Web-браузер lynx, программу чтения новостей tin и почтовый клиент mutt. Сравнительно невысокая популярность ncurses объясняется тем, что ниша ее применения сократилась. Большинство современных компьютеров поддерживают растровую графику, так что если вам нужно реализовать интерфейс «укажи и щелкни», вы, как правило, можете воспользоваться более совершенными графическими средствами. Выбирать ncurses, как платформу для нового проекта, следует только в том случае, если с одной стороны, вашей программе совершенно необходим интерфейс «укажи и щелкни», а с другой стороны, программа должна работать в условиях отсутствия графической системы.

Введение в ncurses

Основными концепциями пользовательского интерфейса программы, использующей ncurses, являются экран (screen), окно (window) и под-окно (sub-window). Экраном называется все пространство, на котором ncurses может выводить данные. С точки зрения ncurses, экран – это матрица ячеек, в которые можно выводить символы. Если монитор работает в текстовом режиме, экран ncurses совпадает с экраном монитора. Если терминал эмулируется графической программой, экраном является рабочая область окна этой программы. Окном ncurses называется прямоугольная часть экрана, для которой определены особые параметры вывода. В частности, размеры окна влияют на перенос и прокрутку строк, выводимых в этом окне. В каком-то смысле окно можно назвать «экраном в экране». На уровне интерфейса

программирования окна представлены структурами данных, по этой причине мы будем часто говорить об окне как о структуре.

В процессе инициализации ncurses автоматически создается окно stdscr, размеры которого совпадают с размерами экрана. Кроме структуры stdscr по умолчанию создается еще одна структура – curscr. Операции вывода данных ncurses модифицируют содержимое структуры stdscr, однако на экране всегда отображается содержимое окна curscr. Иначе говоря, данные, которые выводит ваша программа в окно stdscr (или в другое окно), не отображаются на экране монитора автоматически. Для того чтобы сделать результаты вывода видимыми, вы должны вызывать специальные функции обновления экрана (refresh() или wrefresh()). Эти функции сравнивают содержимое окон stdscr и curscr и на основе различий между ними вносят изменения в структуру curscr, а затем обновляют экран. Благодаря наличию окна curscr, ncurses-программе не требуется «помнить» весь свой предыдущий вывод и перерисовывать его всякий раз, когда в этом возникает необходимость. Этим программы ncurses отличаются от графических программ. В старину, когда терминалы связывались с компьютерами через модемы, использование двух окон давало дополнительное преимущество в скорости обмена данными, ведь программе нужно было передавать на терминал не копию экрана целиком, а только разницу между содержимым окон curscr и stdscr.

Хотя ваша программа может пользоваться для вывода данных исключительно окном stdscr, ваша задача по проектированию интерфейса существенно упростится, если вы будете создавать собственные окна, расположенные «внутри» stdscr. Программа, использующая ncurses, может работать с несколькими окнами одновременно, выполняя вывод в каждое из них. Кроме окон (windows) программы ncurses могут создавать под-окна (subwindows), поведение которых несколько отличается от поведения стандартных окон.

Важнейшей особенностью ncurses является возможность указать произвольную позицию курсора для вывода (и ввода) данных. Позиция курсора отсчитывается от левого верхнего угла текущего окна. Ячейка в верхнем левом углу имеет координаты (0, 0). При работе с функциями ncurses важно помнить, что первой координатой является номер строки (что соответствует y, в терминах графического программирования), а второй координатой – номер столбца (что соответствует x в графическом режиме).

В случае ошибки функции ncurses обычно возвращают константу ERR. Если функция не должна возвращать какое-то информативное значение (как, например, функция getch()), в случае успешного выполнения она возвращает значение OK.

Первая программа ncurses

Написание первой программы ncurses (она называется cursed, исходный текст вы найдете в файле cursed.c) мы начнем с перечисления заголовочных файлов.

```
#include <termios.h>
#include <sys/ioctl.h>
#include <signal.h>
#include <stdlib.h>
#include <curses.h>
```

Помимо уже знакомых нам заголовочных файлов, в программу включен файл <curses.h>, который содержит объявления функций, переменных, констант и структур данных, экспортируемых библиотекой ncurses.

Прежде чем переходить к программированию ncurses, следует рассмотреть решение одной задачи, с которой в настоящее время сталкиваются все разработчики, использующие эту библиотеку. Речь идет об изменении размеров окна терминала (под размерами окна в данном случае понимается число строк и столбцов). Пользователи настоящих текстовых терминалов редко переключали их режимы, и готовы были мириться с последствиями своих действий. В наши дни, когда экраном терминала зачастую служит окно графической программы, пользователь вправе ожидать, что при изменении размеров окна работа консольной программы не нарушится, а ее интерфейс не развалится.

Когда размеры окна терминала меняются, выполняющаяся в нем программа получает сигнал SIGWINCH. Это одновременно и хорошо и плохо. Хорошо – потому, что терминал информирует программу об изменении своих размеров, плохо – потому, что сигналы имеют особенность вмешиваться в работу программы. Например, если вы напишете программу, использующую ncurses, и не позаботитесь об обработке сигнала SIGWINCH, при изменении размеров окна терминала ваша программа может неожиданно завершиться, оставив терминал в неканоническом состоянии. Рассмотрим, как обрабатывается сигнал SIG_WINCH в программе cursed.

```
void sig_winch(int signo)
{
    struct winsize size;
    ioctl(fileno(stdout), TIOCGWINSZ, (char *) &size);
    resizeterm(size.ws_row, size.ws_col);
}
```

Функция sig_winch() представляет собой обработчик сигнала SIGWINCH. Следует отметить, что изменение размеров окна программы, работающей в текстовом режиме, представляет собой довольно нетривиальную задачу и стандартного рецепта, описывающего, что должна делать программа, когда размеры окна изменились, не существует. Разработчики ncurses, как могли, постарались упростить решение этой задачи для программистов, введя функцию resizeterm(). Функцию resizeterm() следует вызывать сразу после изменения размеров окна терминала. Аргументами функции resizeterm() должны быть новые размеры экрана, заданные в строках и столбцах. Функция resizeterm() старается сохранить внешний вид и порядок работы приложения в изменившемся окне терминала, но это ей удается не всегда, с чем мы столкнемся ниже. Необходимые для resizeterm() значения размеров окна мы получаем с помощью специального вызова ioctl(). При этом первым параметром функции ioctl() должен быть дескриптор файла устройства, представляющего терминал. Вторым параметром ioctl() является константа TIOCGWINSZ, а третьим параметром – адрес структуры winsize. Структура winsize определенная в файле <sys/ioctl.h>, включает в себя поля ws_row и ws_col, в которых возвращается число строк и столбцов окна терминала.

Перейдем теперь к функции main() программы cursed:

```
int main(int argc, char ** argv)
{
    initscr();
    signal(SIGWINCH, sig_winch);
    cbreak();
    noecho();
    curs_set(0);
    attron(A_BOLD);
    move(5, 15);
    printw("Hello, brave new curses world!\n");
    attroff(A_BOLD);
```

```
    attron(A_BLINK);
    move(7, 16);
    printw("Press any key to continue...");
    refresh();
    getch();
    endwin();
    exit(EXIT_SUCCESS);
}
```

Работа с ncurses начинается с вызова функции initscr(). Эта функции инициализирует структуры дынных ncurses и переводит терминал в нужный режим. По окончании работы с ncurses следует вызвать функцию endwin(), которая восстанавливает то состояние, в котором терминал находился до инициализации ncurses. После вызова initscr() мы устанавливаем обработчик сигнал SIGWINCH. Устанавливать обработчик SIGWINCH следует только после инициализации ncurses, поскольку в обработчике используется функция resizeterm(), предполагающая, что библиотека ncurses уже инициализирована. Функция noecho() отключает отображение символов, вводимых с клавиатуры. Функция cur_set() управляет видимостью курсора. Если вызвать эту функцию с параметром 0, курсор станет невидимым, вызов же функции с ненулевым параметром снова «включает» курсор.

Функция attron() позволяет указать некоторые дополнительные атрибуты выводимого текста. Этой функции можно передать одну или несколько констант, обозначающих атрибуты (в последнем случае их следует объединить с помощью операции «|»). Например, атрибут A_UNDERLINE включает подчеркивание текста, атрибут A_REVERSE меняет местами цвет фона и текста, атрибут A_BLINK делает текст мигающим, атрибут A_DIM снижает яркость текста по сравнению с нормальной, атрибут A_BOLD делает текст жирным в монохромном режиме и управляет яркостью цвета в цветном режиме работы монитора. Специальный атрибут COLOR_PAIR() применяется для установки цветов фона и текста. На странице man, посвященной функции attron(), вы найдете описания и других атрибутов. Все перечисленные выше атрибуты оказывают воздействие только на тот текст, который выводится после установки атрибута. Выше мы говорили о том, что окно ncurses представляет собой матрицу ячеек для вывода символов. Помимо кода символа каждая ячейка содержит дополнительные атрибуты символа. Сбросить атрибуты можно с помощью функции attroff(). Так же, как и в случае с attron(), функции attroff() можно передать несколько констант, обозначающих атрибуты, разделенных символом «|». Так же, как и установка атрибута, сброс атрибута влияет только на текст, напечатанный после сброса (текст, напечатанный ранее с установленным атрибутом, остается без изменений). В нашей программе мы сначала устанавливаем атрибут A_BOLD. Теперь, до тех пор, пока мы не сбросим этот атрибут, весь текст будет печататься жирным шрифтом. Прежде чем напечатать строку текста этим шрифтом, мы воспользуемся еще одной возможностью ncurses – выводом текста в произвольной области экрана. Функция move() устанавливает позицию курсора в окне stdscr. Первый аргумент функции – строка, второй аргумент – столбец, в котором должен находиться курсор. Следующий затем вывод текста начнется с той позиции, в которой был установлен курсор. Если попытаться поместить курсор за пределы окна, функция move() не станет выполнять никаких действий, и курсор останется на прежнем месте. Мы переводим курсор в позицию (5, 15) и выводим на экран строку "Hello, brave new curses world!" с помощью функции printw(). Функция printw() представляет собой аналог printf() для окна stdscr и имеет тот же список параметров, что и printf().

Теперь мы сбрасываем атрибут A_BOLD с помощью функции attroff(), устанавливаем атрибут A_BLINK, переводим курсор в позицию (7,16) и

распечатываем строку "Press any key to continue...". Хотя мы уже напечатали две строки, на экране терминала все еще ничего нет. Для того чтобы напечатанные нами символы стали видимыми, необходимо вызывать функцию refresh(). Функция refresh() является, в некотором роде, избыточной (действительно, почему бы не отображать распечатанный текст сразу же после вызова printw()?). Фактически функция refresh() представляет собой пережиток тех времен, когда терминал связывался с компьютером при помощи модема. Контролируя частоту вызовов refresh(), можно было сократить трафик между терминалом и компьютером (см. выше описание взаимодействия окон stdscr и curscr).

В результате всех проделанных операций на экране, в заданных позициях, появятся две строки (рис.1) – одна выделенная жирным шрифтом, другая – мигающим (к сожалению, мигание на скриншоте незаметно).

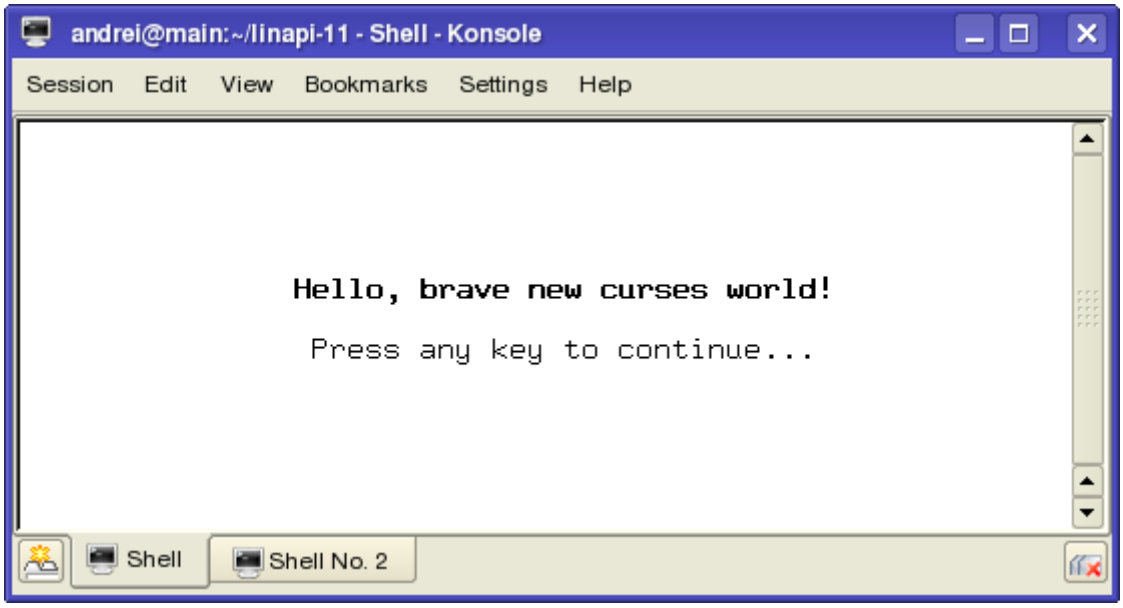

Рисунок 1. Две строки от ncurses

Функция getch(), которую мы вызываем далее, предназначена для считывания символов из потока ввода терминала. Функция считывает по одному символу и может работать в двух режимах: блокирующем (режим по умолчанию) и неблокирующем. В блокирующем режиме функция приостанавливает выполнение программы до появления символа в потоке ввода, а в неблокирующем – возвращает значение сразу же, независимо от того, есть ли символ в потоке ввода или нет (если символа в потоке ввода нет, функция getch() в неблокирующем режиме возвращает значение ERR). В режиме cbreak() (о котором подробнее будет рассказано во второй части статьи) функция, считавшая символ, передает его программе, не дожидаясь, пока пользователь нажмет [Enter]. Таким образом, программа cursed завершается сразу же после нажатия на любую клавишу.

Интерфейс программирования ncurses не является частью glibc, а вынесен в отдельную библиотеку libncurses, поэтому во время сборки программы эту библиотеку нужно подключать явным образом, например:

```
gcc cursed.c –o cursed –lncurses
```

Окна

В текстовых интерфейсах, построенных на основе ncurses, окна играют такую же важную роль, что и в графических интерфейсах. Прежде чем переходить к созданию приложений, использующих окна, необходимо внести

некоторые уточнения в описание интерфейса ncurses. Прежде всего, вы должны понимать, что при работе с ncurses вы *всегда* имеете дело с окнами. В рассмотренной выше программе cursed мы работали с окном stdscr. Выше уже отмечалось, что каждое окно ncurses описано структурой данных, однако при работе с stscr нам не приходилось иметь дело ни с какими специальными структурами. Объясняется это тем, что в программе cursed мы использовали функции (attron(), move(), printw(), attroff(), getch()), специально предназначенные для работы с окном stdscr. Поскольку эти функции работают исключительно с окном stdscr, передавать им структуру, описывающую окно, не требуется. Для работы с другими окнами нам придется использовать обобщенные варианты функций. Списки параметров обобщенных функций совпадают со списками параметров функций, предназначенных для работы с stdscr, за исключением того, что первым параметром обобщенной функции должен быть указатель на структуру WINDOW, определяющую окно, для которого вызывается функция. Например, для установки атрибутов текста в произвольном окне применяется функция wattron(). Первым параметром этой функции служит указатель на структуру WINDOW, а второй параметр wattron() полностью аналогичен параметру функции attron(). Как получить указатель на структуру WINDOW, соответствующую некоторому окну? Переменная stdscr, которую экспортирует библиотека ncurses, содержит указатель на структуру WINDOW, представляющую корневое окно stdscr. Эта переменная определена как

```
extern WINDOW * stdscr;
```

Из того, что stdscr является обычным окном ncursees, следует, что вместо функций, предназначенных специально для stdscr, мы можем использовать их обобщенные аналоги, указывая переменную stdscr в качестве идентификатора окна. Например, вызов attron(A_BOLD) эквивалентен вызову wattron(stdscr, A_BOLD). Обобщенным вариантом функции attroff() является функция wattroff(), а обобщенным вариантом функции move() функция wmove(). Вызов

```
move(5, 15);
```

из программы cursed можно заменить вызовом

```
wmove(stdscr, 5, 15);
```

Функции printw() соответствует обобщенная функция wprintw(). Функции getch() соответствует функция wgetch(), аргументом которой должен быть все тот же указатель на WINDOW. Отметим, что функциями getch()/wgetch() и printw()/wprintw() не исчерпывается многообразие функций ввода/вывода символов ncurses. В описании API библиотеки вы найдете множество других полезных функций.

Перейдем, наконец, к созданию собственных окон. Смысл создания дополнительных окон заключается в том, что мы ограничиваем область вывода текста (и область применения различных атрибутов) отдельными участками экрана. Создав в некоторой области экрана окно, мы можем быть уверены, что вывод данных, который мы выполняем в этом окне, никак не повлияет на остальной экран. Библиотека ncurses предоставляет в наше распоряжение несколько функций, создающих новые окна. Самой простой и часто используемой является функция newwin(). У функции newwin() четыре параметра. Первые два параметра соответствуют количеству строк и столбцов в создаваемом окне, а вторые два указывают положение верхнего левого угла нового окна (строка и столбец) относительно окна stdscr. Функция newwin() возвращает указатель на структуру WINDOW (или NULL в случае ошибки). После завершения работы с окном, выделенные ему ресурсы следует высвободить с помощью функции delwin(). Единственный параметр этой функции – указатель на структуру WINDOW, которую следует удалить. Отметим, что удаление окна с помощью delwin() само по себе не влияет на

содержимое экрана. Данные, выведенные в окно останутся на экране до тех пор, пока не будут перезаписаны другими данными.

Помимо функции newwin() нам будет полезно познакомиться еще с двумя функциями: subwin() и derwin(). Эти две функции предназначены для создания под-окон. Списки параметров у этих функций такие же, как и у newwin(), с той разницей, что первым параметром каждой функции является указатель на структуру WINDOW, соответствующую родительскому окну. Последние два аргумента у subwin() и derwin() интерпретируются по-разному. У функции subwin() они задают положение верхнего левого угла окна относительно экрана, а у функции derwin() – относительно родительского окна. Чем же под-окно отличается от обычного окна? Окно и его под-окно разделяют массив, в котором хранятся символы и их атрибуты. Новое под-окно наследует все атрибуты своего родителя. Эти атрибуты затем могут быть изменены, что не повлияет на атрибуты родительского окна.

Мы займемся созданием окон в программе cursedwindows (файл cursedwindows.c на диске). Тут я на всякий случай заявляю, что у меня нет ни малейшего желания оскорбить продукцию Microsoft. Список заголовочных файлов и обработчик сигнала SIG_WINCH у программы cursedwindows такие же, как и у программы cursed, так что в листинге мы их пропустим и рассмотрим только функцию main().

```
int main(int argc, char ** argv)
{
    WINDOW * wnd;
    WINDOW * subwnd;

    initscr();
    signal(SIGWINCH, sig_winch);
    cbreak();
    curs_set(0);
    refresh();
    wnd = newwin(6, 18, 2, 4);
    box(wnd, '|', '-');
    subwnd = derwin(wnd, 4, 16, 1, 1);
    wprintw(subwnd, "Hello, brave new curses world!\n");
    wrefresh(wnd);
    delwin(subwnd);
    delwin(wnd);
    move(9, 0);
    printw("Press any key to continue...");
    refresh();
    getch();
    endwin();
    exit(EXIT_SUCCESS);
}
```

В функции main() мы, как и прежде, инициализируем ncurses с помощью функции initscr() и устанавливаем обработчик SIGWINCH. Далее мы делаем курсор невидимым, как и в предыдущем примере. После этого мы должны обновить экран с помощью refresh().

Мы создаем новое окно с помощью функции newwin(). Наше окно насчитывает 6 строк и 18 столбцов и его верхний левый угол находится в ячейке (2, 4) окна stdscr. Указатель на структуру WINDOW, который возвращает функция newwin(), мы сохраняем в переменной wnd. Функция box(), которую мы вызываем далее, позволяет создать рамку вдоль границы окна. Аргументами этой функции должны быть идентификатор окна и символы, используемые, соответственно, для рисования вертикальной и горизонтальной границы. Теперь было бы логично вывести какой-нибудь текст

в окно, обрамленное рамкой, но тут возникает одна сложность. Поскольку символы рамки сами находятся внутри окна, символы текста могут затереть их в процессе вывода. Мы решаем эту проблему с помощью создания под-окна subwnd внутри окна wnd и вывода текста в это под-окно. Поскольку окно subwnd по размерам меньше, чем окно wnd, символы рамки не будут стерты.

Теперь мы можем распечатать текст, что мы и делаем с помощью функции wprintw(), указав ей идентификатор окна subwnd. Для того чтобы символы, напечатанные в окне, стали видимыми, мы должны вызвать функцию wrefresh(). Мы вызываем эту функцию только для окна wnd, поскольку оно содержит символьный массив и своего под-окна subwnd. Обратите внимание на то, что символы строки "Hello, brave new curses world!", которую мы печатаем в под-окне subwnd с помощью функции wprintw(), переносятся при достижении границы под-окна (рис. 2). После завершения работы с окнами мы можем удалить структуры wnd и subwnd с помощью функции delwin(). Весь вывод, выполненный в окне wnd, останется на экране (точнее в окне curscr) до тех пор, пока вы не перезапишете его другим выводом.

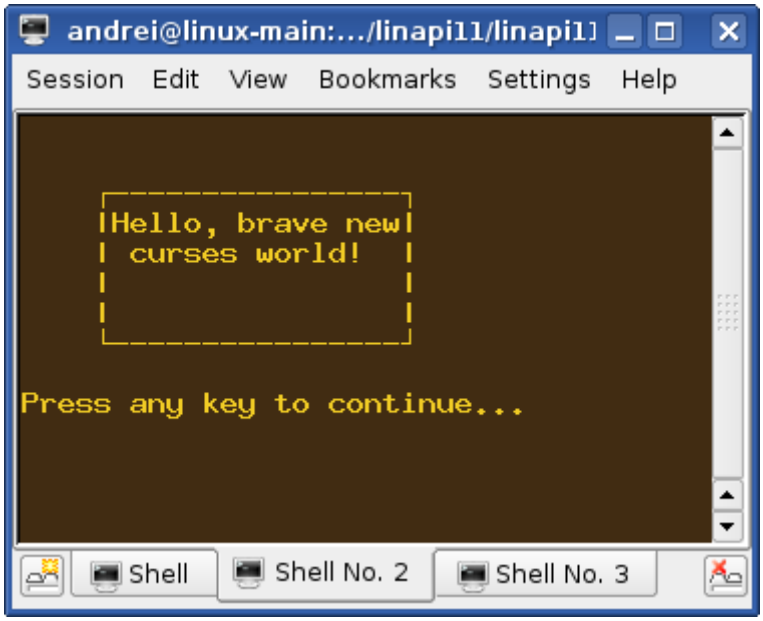

Рисунок 2. Вывод текста в окне wnd и за его пределами

Знакомство с ncurses не закончено. В следующей статье мы рассмотрим управление цветом и ввод данных средствами ncurses.