CSC 374/407: Computer Systems II

Lecture 9

Joseph Phillips
De Paul University

2014 January 5

Overview

What is ncurses?

- A text-package that lets you control text on the screen

ncurses

- “curses” (circa 1980)
  - A screen package for BSD Unix from U.C. Berkeley
- “ncurses” (1993)
  - For “*n*ew *curses*”
- BIG IDEA:
  - Two screens
    - Virtual one written to by program
    - Physical one (user's terminal)
  - Package minimizes bandwidth by
    - Relying on programmer to `refresh()` to sync screens
    - Only sending over minimal number of changes needed

Our first ncurses program:

jphillips@localhost:~/Academic/DePaul/Classes/CSC374/lectu

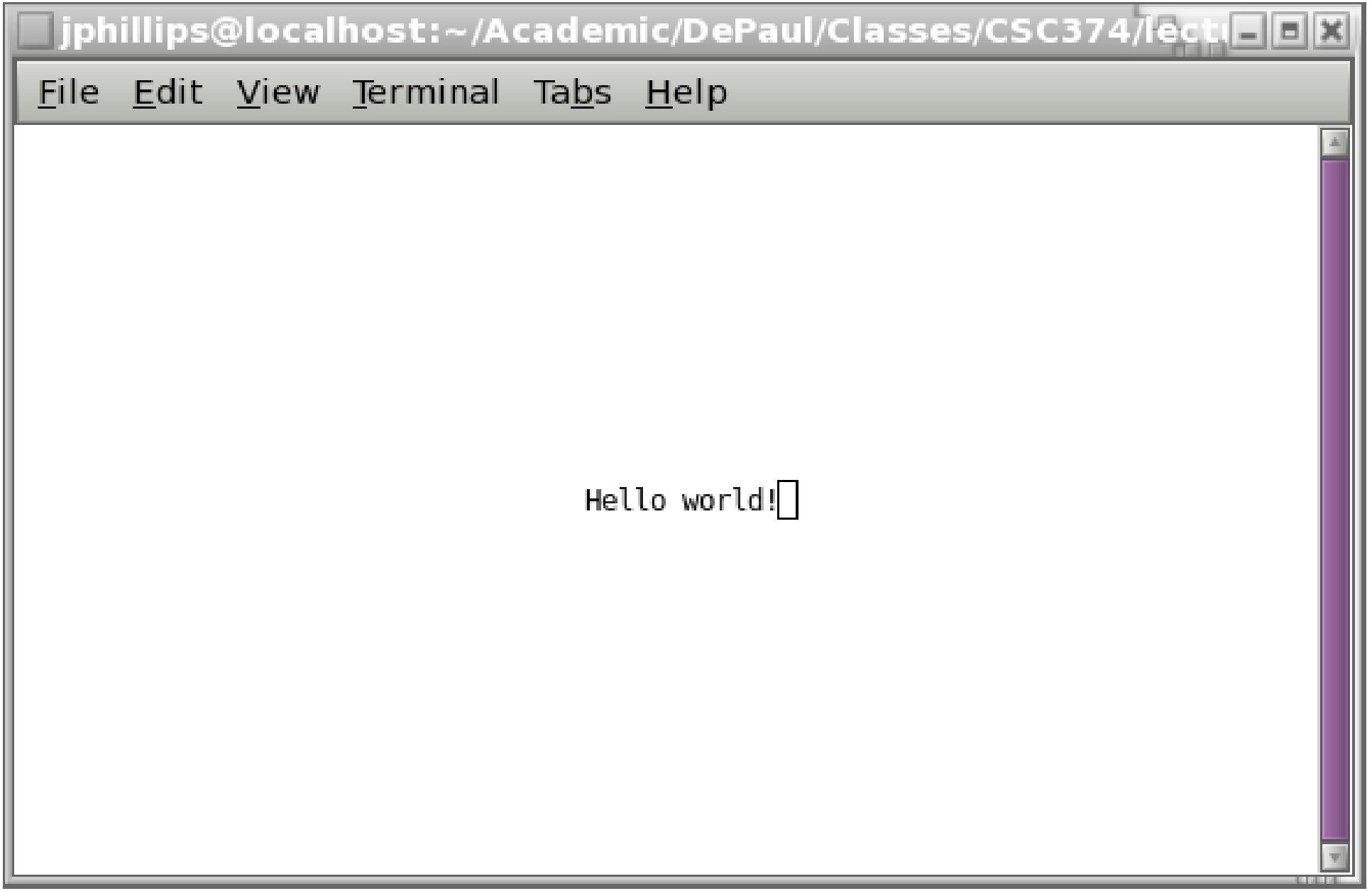
File Edit View Terminal Tabs Help

```
/*
   Compile with
   linux> gcc helloWorldNCurses.c -o helloWorldNCurses -lncurses
 */

#include <stdlib.h>
#include <ncurses.h>
#include <unistd.h>

int main ()
{
  initscr();  // Turns ncurses on
  clear();    // Clears screen
  move(10,35);// Move cursor to row 10, column 35
  addstr("Hello world!");
     // Write given text to current cursor location
  refresh();  //  VERY IMPORTANT, SEE NOTHING UNTIL DO THIS!
  sleep(10);
  endwin();   // Turns ncurses off
  return(EXIT_SUCCESS);
}
```

Our first ncurses program, cont'd



ncurses functions (1)

`initscr()`

Starts ncurses package

`clear()`

Clears screen

`move(int row, int col)`

Moves cursor to given *row* and *col* (upper left is 0,0)

`addstr(const char* str)`

Write string pointed to by *str* to current location

`addchr(char c)`

Writes character *c* to current location

`refresh()`

Sends changes need to write virtual screen to physical

`endwin()`

Ends ncurses package

Subwindows

Type `WINDOW*` refers to a window

`stdscr` refers to the whole window

Can define subwindows of whole screen

subwindows.c

jphillips@localhost:~/Academic/DePaul/Classes/CSC374/lectu

File Edit View Terminal Tabs Help

```
/*
   Compile with
   linux> gcc subwindows.c -o subwindows -lncurses
 */

#include <stdlib.h>
#include <ncurses.h>
#include <unistd.h>
#define LINE_LEN 10
int main ()
{
  char    line[LINE_LEN];
  int     i;
  WINDOW  *windPtr1, *windPtr2;
  initscr();  // Turns ncurses on
  clear();    // Clears screen
  windPtr1 = newwin(5,10, 4,10);  windPtr2 = newwin(5,10,14,50);
  scrollok(windPtr1,FALSE);
  scrollok(windPtr2,FALSE);
  for  (i = 0; i < 40;  i++)
  {
    snprintf(line,LINE_LEN," %d",i);
    waddstr( ((i%2) == 0) ? windPtr1 : windPtr2, line);
    wrefresh( ((i%2) == 0) ? windPtr1 : windPtr2);
    usleep(400000);
  }

  delwin(windPtr2); delwin(windPtr1);  // Delete windows
  endwin();                            // Turns ncurses off
  return(EXIT_SUCCESS);
}
```

subwindow.c, cont'd

jphillips@localhost:~/Academic/DePaul/Classes/CSC374/lectu

File Edit View Terminal Tabs Help

```
 0 2 4 6 8
 10 12 14
16 18 20 2
2 24 26 28
 30 32 34
```

```
 1 3 5 7 9
 11 13 15
17 19 21 2
3 25 27 29
 31 33 35
```

Using subwindows

```
WINDOW* newwin (int numRows, int
  numCols, int beginRow, int
  beginCol)
```

Makes and returns new window.

```
wrefresh(WINDOW* wPtr)
```

Refreshes just **wPtr*.

```
delwin(WINDOW* wPtr)
```

Deletes **wPtr*.

Using subwindows, cont'd

`waddstr(WINDOW* wPtr, const char* str)`

Writes `str` to `*wPtr`.

`waddch(WINDOW* wPtr, char ch)`

Writes `ch` to `stdscr`/`*wPtr`.

`wmove(WINDOW* wPtr, int row, int col)`

Moves cursor to *row*,`col` within window `*wPtr`.

`scrollok(windowPtr, TRUE)`
`scrollok(windowPtr, FALSE)`

Allows/disallows scrolling of window.

Keyboard input

Gives you more control over keyboard input

- Can get keys as they are typed, without user pressing "`Enter`"
- Can get arrow keys
- Can turn of "echoing" of typed chars

  (When would this be good?)

caseChanger.c

jphillips@localhost:~/Academic/DePaul/Classes/CSC374/lect

File Edit View Terminal Tabs Help

```
/*
   Compile with
   linux> gcc caseChanger.c -o caseChanger -lncurses
 */
#include <stdlib.h>
#include <ncurses.h>
#include <ctype.h>

int main ()
{
  int     i;
  WINDOW  *windPtr1, *windPtr2;
  initscr();  // Turns ncurses on
  clear();    // Clears screen
  noecho();   // Turn off normal echoing of chars
  windPtr1 = newwin(5,40, 4,10);  windPtr2 = newwin(5,40,14,10);
  for  (i = 0; i < 50;  i++)
  {
    int ch = getch();
    waddch(windPtr1,ch); wrefresh(windPtr1);

    int freaky;
    if  ( isupper(ch) ) freaky = tolower(ch); else freaky = toupper(ch);
    waddch(windPtr2,freaky); wrefresh(windPtr2);
  }

  delwin(windPtr2); delwin(windPtr1);  // Delete windows
  endwin();                            // Turns ncurses off
  return(EXIT_SUCCESS);
}
```

caseChanger.c cont'd

jphillips@localhost:~/Academic/DePaul/Classes/CSC374/lectu

File Edit View Terminal Tabs Help

```
Appears normal above.  But it is inverte
d below.



aPPEARS NORMAL ABOVE.  bUT IT IS INVERTE
D BELOW.
```

Misc. ncurses

`int getch()`

Gets key (without having to press Enter!)
If it ==ERR then no key was pressed.

`noecho()`

Turns off echoing of chars

`nodelay(stdscr,TRUE)`

Allows non-blocking input from keyboard
(Doesn't even wait for typed key)

`keypad (stdscr,TRUE)`

Allows keypad chars (like arrow keys)

YOUR TURN!

Write a program that writes a string diagonally, bouncing off the top/bottom/left/right if the string is too long.

Your turn, again!

Write a program that writes the text you type diagonally, bouncing off the top/bottom/left/right borders as you type it.

Combing sockets with ncurses

Our turn!

How would you design a Unix “talk” (or Internet “chat”) program?