Department of Computer Science

MEng Robotics and AI – 2023-2024

COMP0204: Introduction to Programing for Robotics and AI

Lab Session 8: A tutorial on <ncurses.h> in C

(This library is required for the assessment 6. So, it is important to get a good grasp of various functionalities offered by ncurses)

ncurses is a programming library providing an application programming interface (API) that allows the programmer to write text-based user interfaces in a terminal-independent manner. The ncurses library provides a range of functionalities for writing console applications which go beyond simply printing text, but without the complexity of writing a full GUI application. The most useful things ncurses provides is moving the cursor (and therefore print position) around the console, keyboard input handling, and printing with various text and background colours.

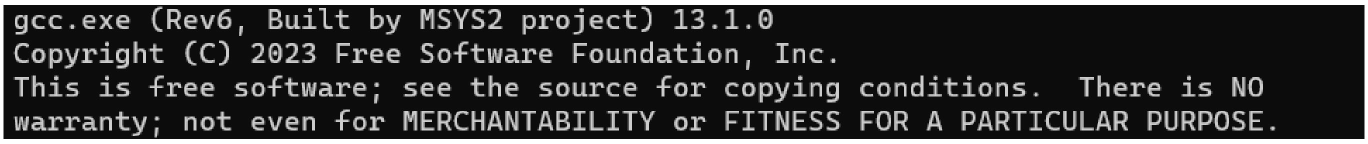
1. Installation:

Visit lecture 7 slides for instructions on the installation in Windows and macOS.

If this still doesnot work in Windows, check the gcc version. Go to terminal and type:

gcc –version

This should return:



```
gcc.exe (Rev6, Built by MSYS2 project) 13.1.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If an older gcc is installed on your laptop, you would need to install a newer version. You can follow this guide: https://www.msys2.org/

2. Getting Started:

In your program, include the necessary NCurses header files:

```
#include <ncurses/ncurses.h>                          // In windows

OR

#include <ncurses.h>                                 // In macOS
```

Note: you donot need to include <stdio.h> when using NCurses.

3. Initialize and Terminate NCurses

Initialize NCurses at the beginning of your program:

```
initscr();              // Initialize ncurses
```

And terminate it at the end:

```
endwin(); // End ncurses mode
```

4. Write a basic Hello World program

```
#include <ncurses.h>

int main() {
    // Initialize NCurses
    initscr();

    // Print text and refresh
    printw("Hello, NCurses!");
    refresh();

    // Get input
    getch();

    // Clean up and close NCurses
    endwin();

    return 0;
}
```

Compiling a .c file and running the executable from the terminal:

Windows user :

Compile:

```
gcc <filename.c -o <filename> -lncurses -DNCURSES_STATIC
```

Run:

```
.\<filename>.exe
```

macOS user :

Compile:

```
gcc <filename.c -o <filename> -lncurses
```

Run:

```
./<filename>
```

5. Printing characters, strings and numbers

In the code above, include the following lines right printw statement.

```
    addstr("------Ncurses tutorial----------\n");

    addstr("Use of addch function for printing character: ");
    addch('a');

    printw("\nUse of printw for printing numbers: %d, %f\n", 11,11.5 );
```

- addstr – ncurses function for printing string.
- addch – prints a single character
- printw – the ncurses equivalent of printf

6. Moving curser position, printing and sleeping (adding delay)

```
void moving_and_sleeping()
{
    int row = 5;
    int col = 0;

    curs_set(0);

    for(char c = 65; c <= 90; c++)
    {
        move(row++, col++);
        addch(c);
        refresh();
        napms(100);
    }

    row = 5;
    col = 3;

    for(char c = 97; c <= 122; c++)
    {
        mvaddch(row++, col++, c);
        refresh();
        napms(100);
    }

    curs_set(1);

    addch('\n');
}
```

This prints the alphabet in upper and lower case, moving one space down and across for each letter. The row and column variables are initialised to 5 (to allow for the heading) and 0, and the ncurses curs_set function is used to hide the cursor.

The for loop iterates the upper case letters, moving to the current position with move, printing the character with addch, calling refresh and then using the napms ("nap for milliseconds") function to pause for a tenth of a second.

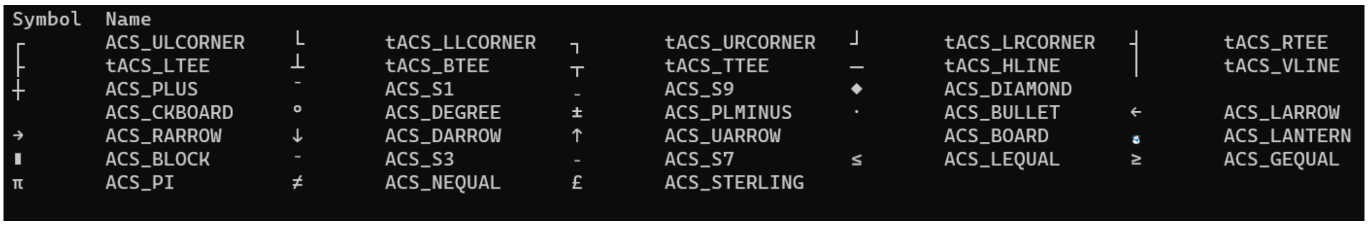
Moving and then printing is so common that the three printing functions shown in the printing function have equivalents that move and print in one function call. They have the same names prefixed with mv, and I have used mvaddch for the second loop which prints lower case letters.

7. Line graphics

The function below prints the available line-drawing characters. These can be used to add line-drawing characters to the screen.

```
void print_char_table(){
   // print name tab symbol tab
    printw("Symbol\tName\n");
    addch(ACS_ULCORNER);addch('\t');printw("ACS_ULCORNER\t");
    addch(ACS_LLCORNER);addch('\t');printw("tACS_LLCORNER\t");
    addch(ACS_URCORNER);addch('\t');printw("tACS_URCORNER\t");
    addch(ACS_LRCORNER);addch('\t');printw("tACS_LRCORNER\t");
    addch(ACS_RTEE);addch('\t');printw("tACS_RTEE\t");
    addch(ACS_LTEE);addch('\t');printw("tACS_LTEE\t");
    addch(ACS_BTEE);addch('\t');printw("tACS_BTEE\t");
    addch(ACS_TTEE);addch('\t');printw("tACS_TTEE\t");
    addch(ACS_HLINE);addch('\t');printw("tACS_HLINE\t");
    addch(ACS_VLINE);addch('\t');printw("tACS_VLINE\t");
    addch(ACS_PLUS);addch('\t');printw("ACS_PLUS\t");
    addch(ACS_S1);addch('\t');printw("ACS_S1\t\t");
    addch(ACS_S9);addch('\t');printw("ACS_S9\t\t");
    addch(ACS_DIAMOND);addch('\t');printw("ACS_DIAMOND\n");
    addch(ACS_CKBOARD);addch('\t');printw("ACS_CKBOARD\t");
    addch(ACS_DEGREE);addch('\t');printw("ACS_DEGREE\t");
    addch(ACS_PLMINUS);addch('\t');printw("ACS_PLMINUS\t");
    addch(ACS_BULLET);addch('\t');printw("ACS_BULLET\t");
    addch(ACS_LARROW);addch('\t');printw("ACS_LARROW\t");
    addch(ACS_RARROW);addch('\t');printw("ACS_RARROW\t");
    addch(ACS_DARROW);addch('\t');printw("ACS_DARROW\t");
    addch(ACS_UARROW);addch('\t');printw("ACS_UARROW\t");
    addch(ACS_BOARD);addch('\t');printw("ACS_BOARD\t");
    addch(ACS_LANTERN);addch('\t');printw("ACS_LANTERN\t");
    addch(ACS_BLOCK);addch('\t');printw("ACS_BLOCK\t");
    addch(ACS_S3);addch('\t');printw("ACS_S3\t\t");
    addch(ACS_S7);addch('\t');printw("ACS_S7\t\t");
    addch(ACS_LEQUAL);addch('\t');printw("ACS_LEQUAL\t");
    addch(ACS_GEQUAL);addch('\t');printw("ACS_GEQUAL\t");
    addch(ACS_PI);addch('\t');printw("ACS_PI\t\t");
    addch(ACS_NEQUAL);addch('\t');printw("ACS_NEQUAL\t");
    addch(ACS_STERLING);addch('\t');printw("ACS_STERLING\t");
}
```

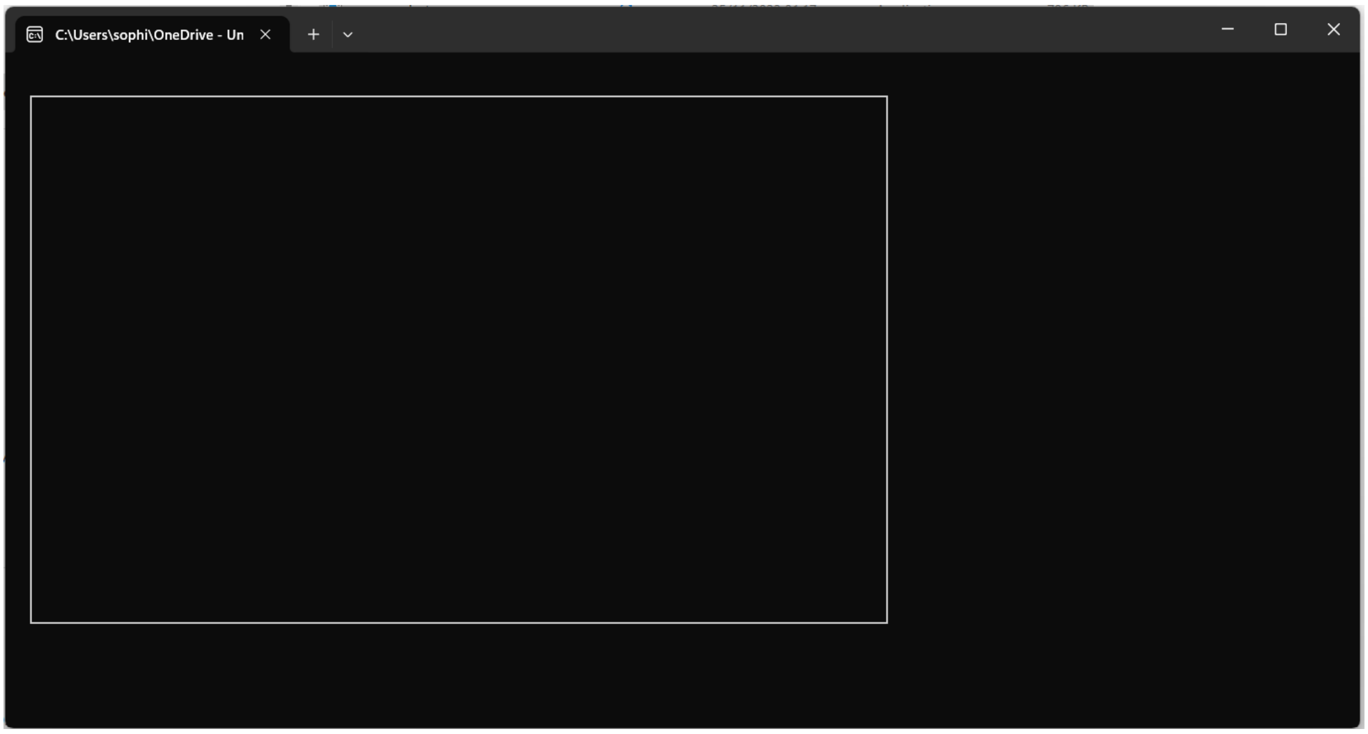
Output: Note the different characters.



```
Symbol  Name
┌       ACS_ULCORNER    └       tACS_LLCORNER   ┐       tACS_URCORNER   ┘       tACS_LRCORNER   ┤       tACS_RTEE
├       tACS_LTEE       ┴       tACS_BTEE       ┬       tACS_TTEE       ─       tACS_HLINE      │       tACS_VLINE
┼       ACS_PLUS        ⎺       ACS_S1          ⎽       ACS_S9          ◆       ACS_DIAMOND
        ACS_CKBOARD     °       ACS_DEGREE      ±       ACS_PLMINUS     ·       ACS_BULLET      ←       ACS_LARROW
→       ACS_RARROW      ↓       ACS_DARROW      ↑       ACS_UARROW              ACS_BOARD       ☃       ACS_LANTERN
▮       ACS_BLOCK       ⎻       ACS_S3          ⎼       ACS_S7          ≤       ACS_LEQUAL      ≥       ACS_GEQUAL
π       ACS_PI          ≠       ACS_NEQUAL      £       ACS_STERLING
```

Exercise 1: Write a function that uses these characters to draw a rectangle.

Expected output:



8. Printing colour

The function below shows the use of colour with ncurses.

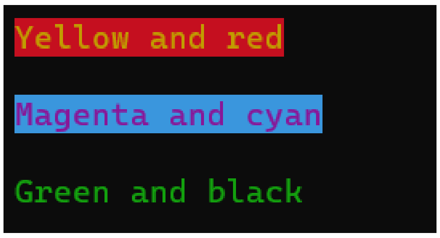
```
void colouring()
{
    if(has_colors())
    {
        if(start_color() == OK)
        {
            init_pair(1, COLOR_YELLOW, COLOR_RED);
            init_pair(2, COLOR_MAGENTA, COLOR_CYAN);
            init_pair(3, COLOR_GREEN, COLOR_BLACK);

            attrset(COLOR_PAIR(1));
            addstr("Yellow and red\n\n");
            attroff(COLOR_PAIR(1));

            attrset(COLOR_PAIR(2));
            addstr("Magenta and cyan\n\n");
            attroff(COLOR_PAIR(2));

            attrset(COLOR_PAIR(3));
            addstr("Green and black\n\n");
            attroff(COLOR_PAIR(3));
            refresh();
        }
        else
        {
            addstr("Cannot start colours\n");
            refresh();
        }
    }
    else
    {
        addstr("Not colour capable\n");
        refresh();
    }
}
```

Output:



```
Yellow and red

Magenta and cyan

Green and black
```

Not all terminals can print in colour so first we need to check using the has_colors function. If so we then need to call the start_color(). Unlike has_colors this returns OK rather than true so we need to check for that.

(In this function if colour is not available a message is printed and the function ends. For production code it might be better to provide a non-coloured fallback.)

Assuming you have 8 colours available then there are 8 * 8 = 64 possible combinations or pairs which can be used for backgrounds and foregrounds. Before using a pair it has to be set using the init_pair function which takes a number between 1 and 64 (not 0 to 63), the foreground colour and the background colour. Here three pairs are set.

This is a full list of ncurses colour constants:

- COLOR_BLACK
- COLOR_RED
- COLOR_GREEN
- COLOR_YELLOW
- COLOR_BLUE
- COLOR_MAGENTA
- COLOR_CYAN
- COLOR_WHITE

Now we can use the colour pairs to print. Firstly call attrset to set the printing attributes to the required pair, then addstr, then switch off the attribute attroff.

9. Taking arrow keys from the keypad as input.

```
#include <ncurses/ncurses.h>
int main()
{
    int ch;
    initscr();
    raw();
    keypad(stdscr, TRUE);
    noecho();

    printw("Type any character to see it in bold\n");
    while (1)
    {
        ch = getch();

        if (ch == KEY_LEFT)
            printw("Left arrow is pressed\n");
        else if (ch == KEY_RIGHT)
            printw("Right arrow is pressed\n");
        else if (ch == KEY_UP)
            printw("Up arrow is pressed\n");
        else if (ch == KEY_DOWN)
            printw("Down arrow is pressed\n");
        // if ESC is pressed, end the program
        else if (ch == 27)
            break;

        //refresh();
    }
    endwin();
    return 0;
}
```

Exercise 2:

Write a C program that add a character ‘R’ at a random location of the rectangle, then use the arrow keys to move the ‘R’ up, down, left and right. Press ‘q’ to exist.

Check the usage of **mvprintw** function.

Note: random number generation was covered in Lecture 3.

Below are links to some further tutorials and materials explaining the capabilities of ncurses.

[1] https://dev.to/tbhaxor/introduction-to-ncurses-part-1-1bk5
[2] https://www.viget.com/articles/game-programming-in-c-with-the-ncurses-library/
[3] https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/