Assignment 3 Design Document

Alex Yeh

February 11, 2023

1 Description of Assignment

This assignment is to create "The Game of Life" using a two-dimensional grid of cells that represent a unvierse. Each cell has two possible states: dead or alive. The game progresses through "generations". There are three rules that determine the state of the universe after each generation:

1. Any live cell with two or three live neighbors survives.
2. Any dead cell with exactly three live neighbors becomes a live cell.
3. Any other cells die, either due to loneliness or overcrowding.

2 Files to be included in directory "asgn2":

1. universe.c
    - This C file implements the Universe ADT.
2. universe.h
    - This header file specifies the interface to the Universe ADT.
3. life.c
    - This C file contains the main function and other functions necessary to complete my implementation of the Game of Life.
4. Makefile
    - This file directs the compilation process of life.c.
5. README.md

- This file is in Markdown format and describes how to use my program and Makefile. It also lists and explains the different command-line options that my program accepts.

6. DESIGN.pdf

   - This file is a PDF version of this design document for assignment 4. It describes my design and design process for my program with pseudocode and images.

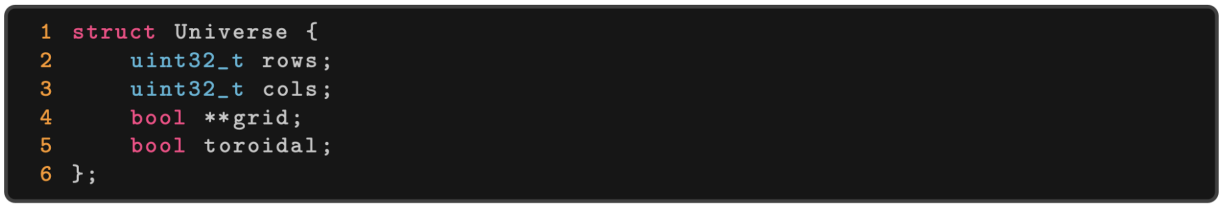
7. WRITEUP.pdf

   - This file is a PDF version of my writeup for assignment 4. It includes everything that I learned from the assignment, how I made use of the ncurses library, the insights I obtained about the process of compiling/linking from this assignment, and my understading of Conway's game of life.

3 Pseudocode

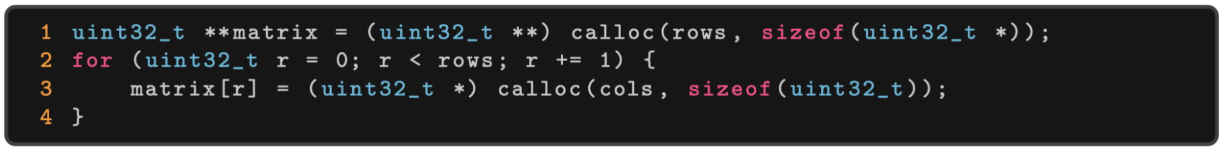
3.1 universe.c

Define struct of Universe:



```c
struct Universe {
    uint32_t rows;
    uint32_t cols;
    bool **grid;
    bool toroidal;
};
```

The photo above is the definition of the struct of the Universe ADT from the assignment 4 pdf that I used in my universe.c.



```c
uint32_t **matrix = (uint32_t **) calloc(rows, sizeof(uint32_t *));
for (uint32_t r = 0; r < rows; r += 1) {
    matrix[r] = (uint32_t *) calloc(cols, sizeof(uint32_t));
}
```

The photo above is pseudocode for allocating memory for a matrix of uint32_ts. I followed this concept write my constructor function that creates a Universe which is outlined below.

Universe *uv_create(uint32_t rows, uint32_t cols, bool toroidal)

```
    Allocate memory for the universe
    Initialize the rows of the universe
    Initialize the columns of the universe
    Initialize the toroidal boolean of the universe
    Allocate memory for the grid of the universe
    Loop through the rows of the grid
        Allocate memory for each row of the grid using the columns of the universe
    Return universe
```

uv_delete(Universe *u)

```
    Loop through the rows of the grid
        Free the memory of each row of the grid
        Set the row to NULL
    Free the memory of the grid
    Set the grid to NULL
    Free the memory of the universe
    Set the universe to NULL
```

uv_rows(Universe *u)

```
    Return the number of rows of the universe
```

uv_cols(Universe *u)

```
    Return the number of columns of the universe
```

The following is a static function I created to check if a cell was out of bounds. It is used in the following functions.

static bool out_of_bounds(Universe *u, uint32_t r, uint32_t c)

```
    If the row is greater than the number of rows of the universe or the column is greater than the number of columns of the universe
        Return true
    Return false
```

uv_live_cell(Universe *u, uint32_t r, uint32_t c)

```
    If the cell is out of bounds
        Return false
    Set the cell at the row and column to true
```

uv_dead_cell(Universe *u, uint32_t r, uint32_t c)

```
    If the cell is out of bounds
        Return false
    Set the cell at the row and column to false
```

uv_get_cell(Universe *u, uint32_t r, uint32_t c)

```
    If the cell is out of bounds
```

```
        Return false
    Return the cell at the row and column
```

bool uv_populate(Universe *u, FILE *infile)

```
    Initialize the row and column variables
    Use fscanf in a while loop to read in the rows and columns of the universe while the file is
not at the end
        If the row and column are out of bounds
            Return false
        Set the cell to live
    Return true
```

uint32_t uv_census(Universe *u, uint32_t r, uint32_t c)

```
    Initialize the count variable of live neighbors
    Loop from x = -1 to x = 1
        Loop from y = -1 to y = 1
            If x and y are both 0 (the cell itself)
                Continue
            Set row variable equal to the row plus x
            Set column variable equal to the column plus y
            If the universe is toroidal
                Add the number of rows to the row variable
                Mod the row variable by the number of rows
                Add the number of columns to the column variable
                Mod the column variable by the number of columns
            If out of bounds
                Continue
            Add the cell at the row and column to the count variable of live neighbors
    Return the count
```

void uv_print(Universe *u, FILE *outfile)

```
    Loop through the rows of the grid
        Loop through the columns of the grid
            If the cell is live
                Print a 'o'
            Else
                Print a '.'
        Print a newline
```

3.2 life.c

define OPTIONS “tsn:i:o:”

Create static void function called “program_usage” to print out help message

```
**int main(int argc, char **argv)**
    Set the default number of generations to 100
    Set the default input file to stdin
    Set the default output file to stdout
    Initialize opt equal to 0
    Initialize rows equal to 0
    Initialize columns equal to 0
    Set boolean variable “toroidal” to false
    Set boolean variable “ncurses” to true
    While loop while opt is not equal to -1
        Switch statement for opt
            Case ’t’:
                Set toroidal to true
                Break
            Case ’s’:
                Set ncurses to false
                Break
            Case ’n’:
                Set number of generations (int32_t to account for negative numbers) to user in-
put using the strtoul function
                If the number of generations is less than 0
                    Print error message to stderr
                    Return 1
                Turn generations back into a uint32_t
                Break
            Case ’i’:
                Set input file to user input using fopen(optarg, “r”) with “r” meaning read
                If the input file is null
                    Print error message to stderr
                    Return 1
                Break
            Case ’o’:
                Set output file to user input using fopen(optarg, “w”) with “w” meaning write
                Break
    If the result of fscanf to read in the rows and columns of the universe is not equal to 2
        Print error message to stderr
        Return 1        Create two universes (A and B) using uv_create, the rows and columns read
in, and toroidal as the third parameter to check if the universe is toroidal
    If the result of populating universe A with uv_populate is false
        Delete the universes using uv_delete
        Print error message to stderr
        Return 1
    If ncurses is true
        Initialize the ncurses screen
        Hide the cursor
        Loop for the number of generations
        If ncurses is true
            Clear the window
            Iterate though the rows
                Iterate through the columns
                    If the cell is live
                        Print a 'o' using the mvprintw function
            Refresh the window
            Sleep for 50000 microseconds
        Loop through the rows of the grid
            Loop through the columns of the grid
                Set count equal to the census of the cell at the row and column of universe A
                If count equals 3 (dead cell has 3 live neighbors)
                    Set the cell at the row and column to live
                Else if count equals 2 and is live
                    Set the cell at the row and column to live
                Else
                    Set the cell at the row and column to dead
    Create a temporary universe and set it equal to A
    Set A equal to B
    Set B equal to the temporary universe
    If ncurses is true
        Close the ncurses screen
    Output universe A to the specified output file using uv_print
    Delete the universes using uv_delete
    Close the input and output files
    Return 0
```

3.3 Makefile

```
Set EXECBIN equal to "life"
Set CC equal to "clang"
Set CFLAGS equal to "-Wall -Wpedantic -Werror -Wextra -Ofast -g -gdwarf-4"
Set LFLAGS equal to "-lncurses"
Set SOURCES equal to all available .c files
Set OBJECTS equal to all available .o files
Set PHONY targets to "all", "clean" and "format"
Set "all" to EXECBIN
Set “life” dependencies on object files life.o and universe.o
    Set rule for source files and use CFLAGS when building executable
Set default rule for creating .o from .c files
Set rule for make clean
Set rule for make format
```