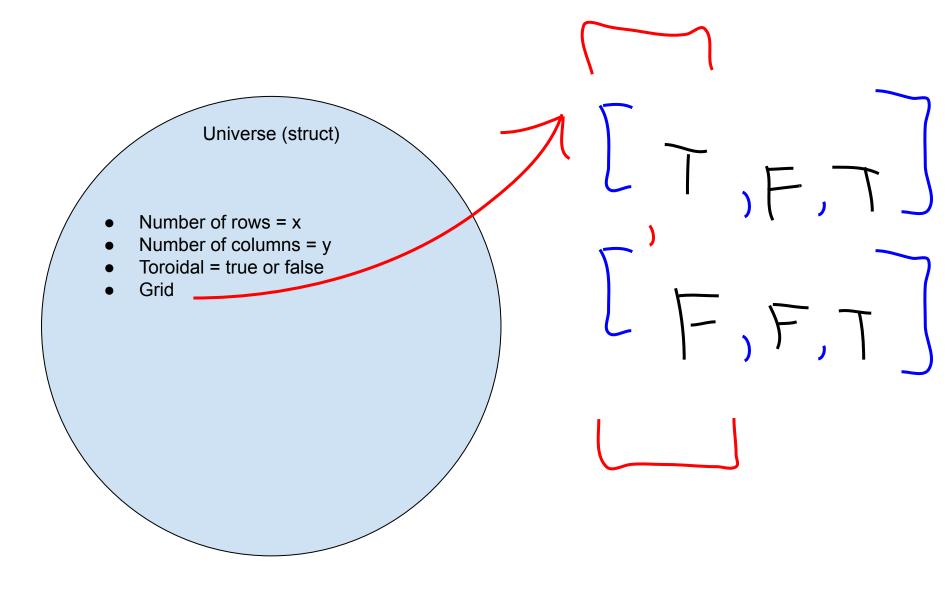


Design Document Overview Alexander Ragland CSE 13



## universe.c

```
struct Universe:
                                                                        bool uv populate(Universe *u, FILE *infile);
                                                                                While loop (until EOF)
{...};
                                                                                       fscan()
Universe *uv create(uint32 t rows, uint32 t cols, bool toroidal);
                                                                                       L = \{data\}
       Universe *uv = (Universe *) malloc(sizeof Universe)
                                                                                       R = \{data\}
                                                                                       If linenumber == 1:
       If (memory alloc success)
               uv-> rows = rows:
                                                                                               u->rows = L
               uv-> cols = cols;
                                                                                               u->cols = R
               uv-> toroidal = toroidal;
                                                                                       If (L > u - rows) or (R > u - rows)
                                                                                               print(failed to populate universe: cell not within
               uv-> grid = (bool **) calloc(rows, sizeof (bool *))
               For (r=0,r< rows,r+=1)
                                                                        bounds of universe)
                       uv->grid[r] = (uint32 *) calloc(cols, sizeof uint32t)
                                                                                               Return false
       Return u;
                                                                                       uv live cell(u, L, R)
                                                                                Return true:
void uv delete(Universe *u);
       For loop (free each row)
                                                                        uint32 t uv census(Universe *u, uint32 t r, uint32 t c);
       Free grid itself
                                                                                If not toroidal
                                                                                Neighbor Counter = 0
uint32 t uv rows(Universe *u);
                                                                                For loop (r-1, r+1)
       Return u->rows:
                                                                                       For loop (c-1, c+1)
uint32 t uv cols(Universe *u);
                                                                                               If u[r][c] is not the cell itself and it is true (alive)
       Return u->cols:
                                                                                                       Neighbor counter ++
void uv live cell(Universe *u, uint32 t r, uint32 t c);
                                                                                If toroidal
       Check if u->grid[r][c] is in bounds:
                                                                                       Cry in modulus
       If yes, u->grid[r][c] = true;
void uv dead cell(Universe *u, uint32 t r, uint32 t c);
                                                                        void uv print(Universe *u, FILE *outfile);
       Check if u->grid[r][c] is in bounds;
                                                                                For rows in u
       If yes, u->grid[r][c] = false;
                                                                                       For cols in u
bool uv get cell(Universe *u, uint32 t r, uint32 t c);
                                                                                               If u[r][c] is true
       Check if u->grid[r][c] is in bounds; return false if not
                                                                                                       Print o
       return (bool) u->grid[r][c];
                                                                                               If u[r][c] is false
                                                                                                       Print.
```

## life.c

```
#define OPTIONS ""
Set defaults
main():
   getopt()...
   Declare universe
   Declare universeb
   Populate universea()...
      Ncurses example...
          Do generation stuff
      End ncurses
   uv print(universea)
```

## **Makefile**

```
CC = clang-15
CFLAGS = -Wall -Wextra -Werror -Wpedantic
OBJECTS = life.o universe.o
LINKS =
All: life
life: $(OBJECTS)
      $(CC) -o sorting sorting.o $(LINKS)
life.o: life.c
      $(CC) $(CFLAGS) -c life.c
universe.o: universe.c
      $(CC) $(CFLAGS) -c universe.c
format:
      clang-format-15 -i -style=file *.[ch]
clean:
      rm -f life *.o
```