

```
# Included files:
* julia.c

# Ignored C and H files
* compute_julia_pixel.c
* julia.h
* CMakeCCompilerId.c
* write_bmp_header.c

commit ddc4a71e028a0cd813dc25566bc28ac6a01ba0d2
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Fri Sep 29 00:31:40 2023 +0200

🔥 julia.c: fix clang-tidy

TP/TP2-julia/julia/julia.c | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit cd04f367313b58b13ea14bd21418c3cbb01d5b77
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Fri Sep 29 00:27:13 2023 +0200

🎨 julia.c clang-format

TP/TP2-julia/julia/julia.c | 55 ++++++-----
1 file changed, 26 insertions(+), 29 deletions(-)

commit aalcd168ed713270a4c385d396fd7841cd5b25dd
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Fri Sep 29 00:25:59 2023 +0200

question 5

TP/TP2-julia/julia/julia.c | 82 ++++++-----
1 file changed, 67 insertions(+), 15 deletions(-)

commit ede9ca77d6a2b510aeea8a8df0e79107d13a20d1
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Thu Sep 28 16:34:16 2023 +0200

question 2

TP/TP2-julia/julia/julia.c | 77 ++++++-----
1 file changed, 61 insertions(+), 16 deletions(-)

commit 987ca318eb3a5fb90eef3d69666eafdba3b7fc4a
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Thu Sep 28 16:33:55 2023 +0200

🐶 add vgcore

TP/TP2-julia/julia/.gitignore | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

commit 0b9bcfeb4b4d7970be8a062914603f5c2ceec3df
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Thu Sep 28 16:18:36 2023 +0200

🎨 1_exercice2.c: coherent arguments var naming

TP/TP1-threads/1_exercice2.c | 10 ++++-----
1 file changed, 5 insertions(+), 5 deletions(-)

commit cb9fb3dd3a6b37bffc801524ecd1e9bf09a0d362
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Thu Sep 28 15:50:38 2023 +0200

🐛 julia.c: fix included provided code

TP/TP2-julia/julia/julia.c | 2 ++
1 file changed, 2 insertions(+)

commit d623cdf54ab4224b8de34d5554ccbfe8b2b2c8d2
Author: AlexisHamon <alexis.hamon24@gmail.com>
Date:   Thu Sep 28 15:50:13 2023 +0200

🚧 julia.c: add question 2 provided code

TP/TP2-julia/julia/julia.c | 6 +++++
1 file changed, 6 insertions(+)
```

TP2 : Julia, threads, et répartition de charge

Programmation multi-threadée

Exercice 1: La fractale de Julia

Question 1:

Question 2: Implémentez un découpage 1D

Question 3: Implémentez maintenant un découpage 2D

Question 4:

Question 5:

```

*****
***** Fichier julia.c *****
*****

*/

#define _GNU_SOURCE

// #define ceild(n, d) (((n) < 0) ? -((-n) / (d)) : ((n) + (d)-1) / (d))
#define ceild(n, d) (((n) + (d)-1) / (d))
#define min(x, y) ((x) < (y) ? (x) : (y))

#include "julia.h"
#include <pthread.h>
#include <sched.h>
#include <stddef.h>
#include <stdlib.h>

struct julia_it {
    // Those are constants between threads
    int size_tile;
    int n_tile_height;
    // Those needs to be accessed threw a critical space
    pthread_mutex_t mutex;
    int i;
};

struct julia_it *juliaAlloc(int width, int height, int size_tile) {
    struct julia_it *it = malloc(sizeof(struct julia_it));
    it->size_tile = size_tile;
    int n_tile_width = width / size_tile + 1;
    it->n_tile_height = height / size_tile + 1;
    pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
    it->mutex = mutex;
    it->i = n_tile_width * it->n_tile_height;
    return it;
}

int juliaGet(struct julia_it *it) {
    pthread_mutex_lock(&it->mutex);
    int i = (it->i ? (it->i--) : (0));
    pthread_mutex_unlock(&it->mutex);
    return i;
}

int juliaGetYmin(struct julia_it *it, int i) {
    return ((i - 1) % it->n_tile_height) * it->size_tile;
}

int juliaGetYmax(struct julia_it *it, int i) {
    return (((i - 1) % it->n_tile_height) + 1) * it->size_tile;
}

int juliaGetXmin(struct julia_it *it, int i) {
    return ((i - 1) / it->n_tile_height) * it->size_tile;
}

int juliaGetXmax(struct julia_it *it, int i) {
    return (((i - 1) / it->n_tile_height) + 1) * it->size_tile;
}

void juliaFree(struct julia_it *it) { free(it); }

int cpucount() {
    cpu_set_t cpuset;
    sched_getaffinity(0, sizeof(cpuset), &cpuset);
    return CPU_COUNT(&cpuset);
}

struct img {
    int width;
    int height;
    unsigned char *pixel;
};

struct thread_args {
    struct img *img;
    struct julia_it *it;
};

void *threadCalc(void *ptr) {
    struct thread_args *args = (struct thread_args *)ptr;
    int i;

    while ((i = juliaGet(args->it))) {
        int y_upb = min(args->img->height, juliaGetYmax(args->it, i));
        int x_upb = min(args->img->width, juliaGetXmax(args->it, i));
        for (int y = juliaGetYmin(args->it, i); y < y_upb; ++y) {
            for (int x = juliaGetXmin(args->it, i); x < x_upb; ++x) {
                compute_julia_pixel(x, y, args->img->width, args->img->height, 1.0,
                                     &args->img->pixel[(y * args->img->width + x) * 3]);
            }
        }
    }

    return NULL;
}

int main(int argc, char **argv) {
    if (argc < 2)
        return 1;
    int n = atoi(argv[1]);

    if (n < 1)
        return 0;

    struct img *img = malloc(sizeof(struct img));
    img->width = 2 * n;
    img->height = n;
    img->pixel = malloc(img->width * img->height * 3 * sizeof(unsigned char));

    FILE *out = fopen("outfile.bmp", "w");
    if (out == NULL) {
        perror("Cannot open outfile");
        exit(1);
    }

    int cpu_count = cpucount();

    struct thread_args *args = malloc(sizeof(struct thread_args));
    args->img = img;
    args->it = juliaAlloc(img->width, img->height, 32);

    pthread_t *pth_t = malloc(cpu_count * sizeof(pthread_t));

    for (unsigned th = 0; th < cpu_count; ++th)
        pthread_create(&pth_t[th], NULL, threadCalc, (void *)args);

    for (unsigned th = 0; th < cpu_count; ++th)
        pthread_join(pth_t[th], NULL);

    free(args->it);
    free(args);
    free(pth_t);

    write_bmp_header(out, img->width, img->height);
    fwrite(img->pixel, img->width * img->height * 3, 1, out);
    fclose(out);

    free(img->pixel);
    free(img);
    return 0;
}

////////// clang-tidy output

```