Fichier Réponses TP2 Qualité Algo

Dans le fichier daemon.c :

- int create_daemon() → complexite de 4
- static void ping_request() → complexite de 2
- static void send_check() → complexite de 5
- static int check_keep_working() → 4
- void daemon_work() → complexite de 3

Dans le fichier db-sqlite.c :

- int db_connect() et int db_disconnect() → complexite de 1
- int insert_hourly_report(....) → 1

Dans le fichier ping-report.c :

- int main() → complexite de 4
  Comment on fait ? Dans la fonction il y a un switch avec 3 cas differents et un default. Chaque cas est un chemin possible et le default en est un aussi alors la complexite cyclomatique de cette fonction est de 4.

Dans le fichier stats.c :

- get_ping_from_temp_log() → compexite de 9
- `write_ping_log(char* new_ping) → complexite de 5`

- `set_stats_ping() → complexite de 13 (12+1)`
    1. `if(fd != NULL)`
    2. `while(getline(&read_line,&n,fd) != -1) (imbriqué dans le if)`
    3. `if(read_line == NULL) (imbriqué dans le while)`
    4. `if(strcmp(read_line,"LOSS") == 0) (imbriqué dans le while)`
    5. `else (associé au if(strcmp...))`
    6. `if(ping < 0.1) (imbriqué dans le else)`
    7. `else (associé au if(ping < 0.1))`
    8. `if(ping > max) (imbriqué dans le else précédent)`
    9. `if(ping < min) (imbriqué dans le else précédent)`
    10. `if(ping > 100.0) (imbriqué dans le else précédent)`
    11. `if(read_line != NULL) (à la fin, après la boucle)`
    12. `else (associé au premier if(fd != NULL))`

Dans le fichier utils.c :

- `write_pid_file() → complexite de 2`
- `remove_file(char* filename) → complexite de 1`

Nouvelle version de la fonction process_ping_line :

Sa complexite est diminué car j'ai factoriser en deux fonctions : read_ping_line() qui est a 3 de complexité et process_ping_line qui est a 5.

```
/* Helper to process a single ping line */
static void process_ping_line(const char* line,
                              double* sum, double* max, double* min,
                              int* nb_high, int* nb_loss, int* nb_ping) {
    if(strcmp(line, "LOSS") == 0){
        (*nb_loss)++;
        return;
    }

    double ping = strtod(line, NULL);
    if(ping < 0.1) return; // ignore null ping
```

```
    (*nb_ping)++;
    (*sum) += ping;
    if(ping > *max) *max = ping;
    if(ping < *min) *min = ping;
    if(ping > 100.0) (*nb_high)++;
}

void set_stats_ping() {
    FILE* fd = fopen("/var/log/ping-report/all-ping.log","r");
    if(!fd){
        perror("stats : ");
        return;
    }

    double sum = 0.0, max = 0.0, min = 100.0;
    int nb_high = 0, nb_loss = 0, nb_ping = 0;
    char* read_line = NULL;
    size_t n = 0;

    while(getline(&read_line, &n, fd) != -1){
        if(read_line){
            process_ping_line(read_line, &sum, &max, &min, &nb_high, &nb_loss, &nb_ping);
            free(read_line);
            read_line = NULL;
            n = 0;
        }
    }

    fclose(fd);

    double mean = (nb_ping > 0) ? (sum / (double)nb_ping) : 0.0;

    insert_hourly_report(mean, max, min, nb_high, nb_loss, nb_ping);
```

Nouvelle version de read_ping_line :

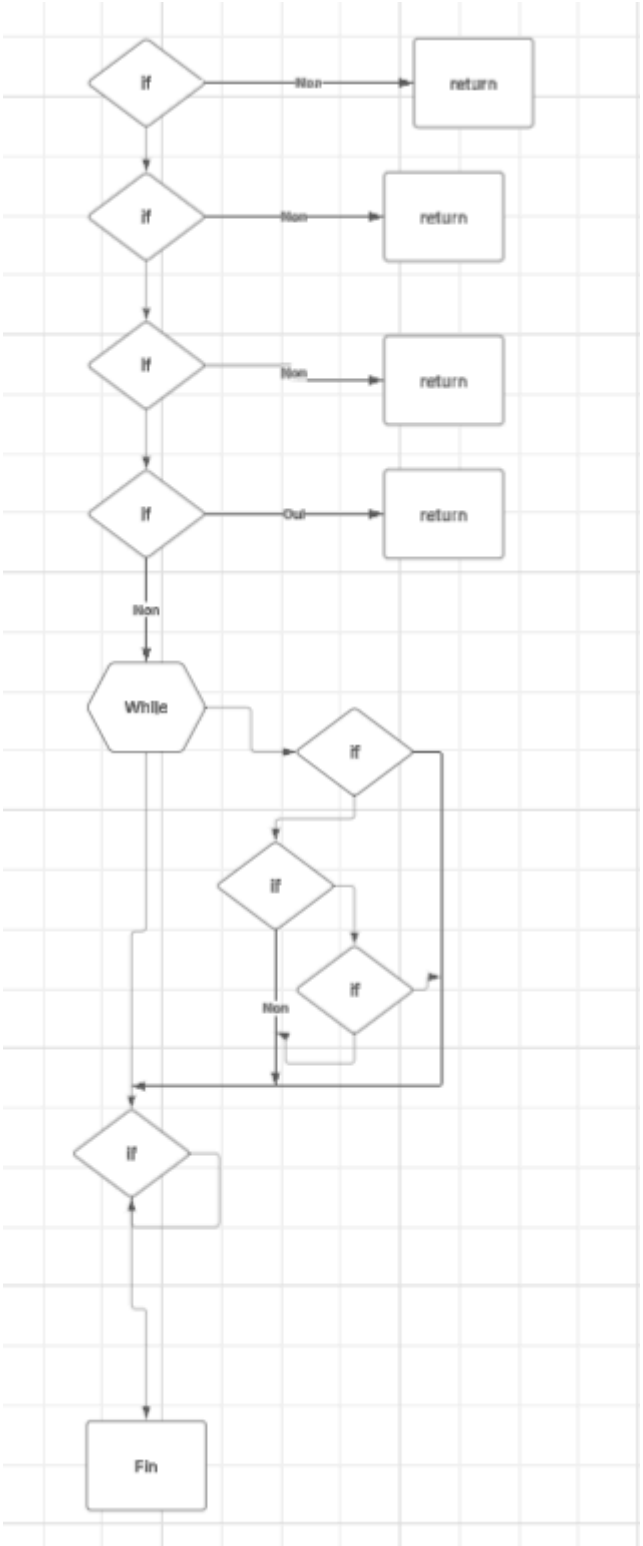
Pareil j'ai découpé en deux fonctions :
`get_ping_from_temp_log()` : complexité = 5

`set_stats_ping() : complexité = 7`

/* Helper to read the ping line from file */

```
static char* read_ping_line(FILE* fd, regex_t* p_reg, size_t nmatch){
    char* read_line = NULL;
    regmatch_t* pmatch = malloc(sizeof(*pmatch) * nmatch);
    if(!pmatch) return NULL;

    char* ping = NULL;
    while(getline(&read_line, &(size_t){0}, fd) != -1){
        if(!read_line) break;
        if(regexec(p_reg, read_line, nmatch, pmatch, 0) == 0){
            int start = (int) pmatch[1].rm_so;
            int end = (int) pmatch[1].rm_eo;
            size_t size_ping = (size_t)(end - start);
            ping = malloc(size_ping + 2);
            if(ping){
                strncpy(ping, &read_line[start], size_ping);
                ping[size_ping] = '\n';
                ping[size_ping+1] = '\0';
            }
            free(read_line);
            break;
        }
        free(read_line);
    }
    free(pmatch);
    return ping;
}

char* get_ping_from_temp_log(){
    FILE* fd = fopen("/var/log/ping-report/last-ping.log","r");
    if(!fd) return NULL;

    regex_t p_reg;
    if(regcomp(&p_reg, "time=(.*) ms", REG_EXTENDED) != 0){
        fclose(fd);
        return NULL;
    }

    char* ping = read_ping_line(fd, &p_reg, 2);

    regfree(&p_reg);
    fclose(fd);
    return ping;
}
```

Le diagramme sur la fonction `get_ping_from_temp_log` :



Le diagramme sur la fonction `et_stats_ping()` :