Mezouar Mikael Université Gustave Eiffel Peinzi Ferdyl

Rapport TP3 : Perfectionnement en programmation C

08/03/2021 L2 informatique

Introduction :

Dans ce TP, nous avons implémenté un chronomètre muni des fonctionnalités suivantes, minuteur, comptage de tour , avertissement , quitter, etc … L'interface du chronomètre a été fait avec la librairie Ncurses. Nous avons durant ce TP pris en main la notion de pré-assertions que nous avons vu en cour dans la semaine. De plus nous avons pu découvrir certaine fonction qui permettent de manipuler le temps dans un programme. Dans ce rapport nous allons voir comment nous avons implémenté ce chronomètre.

Exercice 1:

1) Pour cette fonction , nous avons fais une première version avec les fonction gettimefday() qui ne fonctionnait pas . Puis a la fin nous avons fais une fonction qui renvoie la valeur entière de l'intervalle entre 'début' et 'fin' , le tout que nous avons converti en milliseconde.

```c
int intervalle_ms(struct timeval debut, struct timeval fin) {
    return ((fin.tv_sec - debut.tv_sec) * 1000 + (fin.tv_usec - debut.tv_usec) / 1000);
}
```

2)Pour ces fonctions , nous avons simplement utiliser un tableau de conversion pour les implémenter puis nous avons ajuster chaque fonction le return au cas par cas.Ces fonction convertisse un intervalle de temps dans une unité de mesure.

```c
int nb_ms_vers_centiemes(int nb_ms){

    int temps;

    temps = (nb_ms / 10) % 100;

    return temps;

}
```

Les autres fonctions marchent de la même manière .

3) Dans cette partie on a réalisé un chronomètre basique sur la sortie standard, nous affichant les heures, les minutes, les secondes et les centièmes. Pour ce faire il suffit juste de reprendre les 4 fonctions faites précédemment et de les utiliser une boucle while. Dans cette boucle on affiche sur une ligne les 4 unités de temps demandées pour le chronomètre.

4)Pour le ChronoMoyen.c , nous avons crée une fonction menu_chrono() qui suit le pseudo code donné dans l'énoncé du TP , puis nous avons fait une fonction affiche() qui s'occupe de l'affichage de la fenêtre avec les infos nécessaire au fonctionnement du chrono avec la librairie Ncurses.

```c
void menu_chrono(void) {
    struct timeval temps_debut;
    struct timeval temps_fin;
    int touche;
    int duree_totale;
    int etat;

    duree_totale = 0;
    etat = 0;  /*0=pause et 1=en marche*/
    while (1) {
        touche = getch();
        if (touche == ' ') {
            if (etat == 0) {
                etat = 1;
                gettimeofday(&temps_debut, NULL);
            }
            else
                etat = 0;
        }
        usleep(1000);
        if (etat == 1) {
            gettimeofday(&temps_fin, NULL);
            duree_totale += intervalle_ms(temps_debut, temps_fin);
            gettimeofday(&temps_debut, NULL);
        }
        affiche(duree_totale);
        refresh();
    }
}
```

Exercice 2:

1)Nous avons crée une structure Chronomètre avec tout les paramètres nécessaire listé au début de l'exercice.

```c
typedef struct chrono {
    int etat;
    int duree_tot;
    int avert;
    int der_tour;
    int nb_tour;
    int tps_par_tour[N_TOUR];
} Chronometre;
```

2)Pour cette fonction nous avons juste fais une fonction qui renvoie un chronomètre initialisé avec les paramètres de base au début du lancement du chrono .

```c
Chronometre initialiser_chrono(void) {
    Chronometre chrono;
    int i;

    chrono.etat = 0;
    chrono.duree_tot = 0;
    chrono.avert = 20000;
    chrono.nb_tour = 0;
    chrono.der_tour = 0;
    for (i = 0 ; i < N_TOUR ; i++)
        chrono.tps_par_tour[i] = 0;
    return (chrono);
}
```

3)Pour cette fonction void nous avons affiché le temps du Chronomètre en fonction des coordonnées x et y de la fenêtre .

```c
void afficher_duree(int y, int x, int nb_ms) {


    int cent;
    int sec;
    int min;
    int heures;

    assert(y >= 0);
    assert(y < LINES);
    assert(x >= 0);
    assert(x < COLS);
    assert(nb_ms >= 0);

    cent = nb_ms_vers_centiemes(nb_ms);
    sec = nb_ms_vers_secondes(nb_ms);
    min = nb_ms_vers_minutes(nb_ms);
    heures = nb_ms_vers_heures(nb_ms);

    mvprintw(y, x, "%2d", heures);
    printw(" : ");
    printw("%2d", min);
    printw(" : ");
    printw("%2d", sec);
    printw(" : ");
    printw("%2d", cent);
}
```

4)Pour la fonction affiche_interface() nous avons choisi de recopier l'interface proposé au début du TP , celle_ci utilise la fonction afficher_duree() pour afficher les différents tours et le chrono principal . Ensuite pour l'affichage des infos pour l'utilisateur , nous avons ajusté en fonction de la fenêtre avec des espaces .

```c
void affiche_interface(Chronometre chrono) {
    int i;

    assert(chrono.duree_tot >= 0);
    mvprintw(0, COLS / 2 - 8, "***  Chronometre  ***");
    for (i = 0 ; i < chrono.nb_tour ; i++) {
        mvprintw(LINES / 2 - (i + 1), COLS / 2 - 17, "Tour %3d : ",
                 chrono.der_tour - i);
        afficher_duree(LINES / 2 - (i + 1), COLS / 2 - 6,
                       chrono.tps_par_tour[i]);
    }
    afficher_duree(LINES / 2, COLS / 2 - 6, chrono.duree_tot);
    mvprintw(LINES / 2 + 1, COLS / 2 - 22, "Avertissement :");
    afficher_duree(LINES / 2 + 1, COLS / 2 - 6, chrono.avert);
    for (i = 0 ; i < COLS ; i++)
        mvaddch(LINES - 8, i, '-');
    mvprintw(LINES - 7, 0, "Espace : Lancer / Pause");
    mvprintw(LINES - 6, 0, "r      : Reinitialiser");
    mvprintw(LINES - 5, 0, "t      : Tour");
    mvprintw(LINES - 4, 0,
             "F1/F2  : Incrementer / Decrementer les heures avertissement");
    mvprintw(LINES - 3, 0,
             "F3/F4  : Incrementer / Decrementer les minutes avertissement");
    mvprintw(LINES - 2, 0,
             "F5/F6  : Incrementer / Decrementer les secondes avertissement");
    mvprintw(LINES - 1, 0, "q      : Quitter");
}
```

5)Pour la fonction afficher_flash() , nous avons déclaré 2 couleurs , puis nous avons alterner l'affichage d'un rectangle de couleur entre les 2 couleurs.

```c
void afficher_flash(void) {

    int col;
    int i;
    int j;

    init_pair(1, COLOR_BLUE, COLOR_BLACK);
    init_pair(2, COLOR_RED, COLOR_WHITE);
    attron(COLOR_PAIR(1));
    col = 0;
    while (col < 200) {
        if ((col % 2) == 1) {
            attroff(COLOR_PAIR(1));
            attron(COLOR_PAIR(2));
        }
        else {
            attroff(COLOR_PAIR(2));
            attron(COLOR_PAIR(1));
        }
        for (i = 0; i < LINES; i++){
            for (j = 0; j < COLS; j++)
                mvaddch(i, j, '*');
        }
        refresh();
        usleep(DELAI * 10);
        col++;
    }
    attroff(COLOR_PAIR(1) | COLOR_PAIR(2));
    clear();
}
```

6)Pour cette fonction nous avons suivi le pseudo code indiqué dans la question .

```c
void ajoute_tour(Chronometre *chrono) {
    int i;

    assert(chrono != NULL);
    chrono->der_tour++;
    if (chrono->nb_tour < N_TOUR)
        chrono->nb_tour++;
    for (i = chrono->nb_tour - 1 ; i > 0 ; i--)
        chrono->tps_par_tour[i] = chrono->tps_par_tour[i - 1];
    chrono->tps_par_tour[0] = chrono->duree_tot;
}
```

Exercice 3:

1) 2) 3) 5) (réponse groupée)
Pour les questions de cette exercice nous avons gérer toute les commandes avec un switch() et des fonctions pour l'incrémentation avec les touches F.

```c
    while (1) {
        switch (touche) {
            case ' ' : change_etat(&chrono, &temps_debut); break;
            case 't' : ajoute_tour(&chrono); break;
            case 'r' : reinitialiser_chrono(&chrono);
                       clear(); break;
            case 'q' : return ;
            case KEY_F(1) : heure_avert(&chrono, 1); break;
            case KEY_F(2) : heure_avert(&chrono, 0); break;
            case KEY_F(3) : minute_avert(&chrono, 1); break;
            case KEY_F(4) : minute_avert(&chrono, 0); break;
            case KEY_F(5) : seconde_avert(&chrono, 1); break;
            case KEY_F(6) : seconde_avert(&chrono, 0); break;
            default : break;
        }
```

```c
void change_etat(Chronometre *chrono, struct timeval *temps_debut) {
    assert(chrono != NULL);
    assert(temps_debut != NULL);
    if (chrono->etat == 0) {
        chrono->etat = 1;
        gettimeofday(temps_debut, NULL);
    }
    else
        chrono->etat = 0;
}
```

```c
void minute_avert(Chronometre *chrono, int m) {
    assert(m >= 0);
    assert(m <= 1);
    assert(chrono != NULL);
    if (m)
        chrono->avert += (60 * 1000);
    else
        chrono->avert -= (60 * 1000);
}
```

Les fonctions seconde_avert() et heure_avert() fonctionnent de la même manière que minute_avert() ci-dessus.

4)

```c
if (chrono.avert == chrono.duree_tot) {
    afficher_flash();
}
```

Cette condition permet d'activer le flash pour la condition est vrai .

Conclusion :

Pour conclure ce TP nous a permis de travailler sur la manipulation du temps dans un programme avec certaine fonction que nous avons découvert , notamment la manipulation d'un intervalle de temps , ensuite cela nous a permis encore une fois de manipuler et d'approfondir nos connaissances sur la librairie Ncurses , enfin nous avons pu utilisé les pré-assertions pour la première fois , celles ci sont très utiles pour dé bugué les programmes plus rapidement et donc de trouver plus facilement les erreurs . Durant le TP nous avons rencontrer quelque problème comme la compréhension de la fonction gettimeofday() et plus précisément de ce qu'elle renvoyé précisément (pour l'implémentation de la fonction intervalle_ms()), ensuite nous avons aussi rencontrer quelque problème de conversion de temps que nous avons pu résoudre .

Annexe 1 :

Pour le chronomètre simple :

gcc ChronoSimple.c -lncurses -pedantic -Wall -ansi -o chronosimple

gcc ChronoMoyen.c -lncurses -pedantic -Wall -ansi -o chronomoyen

gcc Chrono.c -lncurses -pedantic -Wall -ansi -o chrono

Annexe 2 :

Tout nos programmes se compilent avec les commandes en annexe 1 , puis on les exécutes avec la commande ./nom_de_exécutable .