HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Laboratoire n°02

Limites de Linux dans un contexte temps réel

Département : TIC

Unité d'enseignement PTR

Auteur: **Alexandre Iorio**

Professeur: **Alexandre Corbaz**

Assistant : **Catel Torres Arzur Gabriel**

Salle de labo : **A05-b**

Date : **05.03.2024**

1. Introduction

Dans le cadre de ce laboratoire, nous allons expérimenter l'interface des timers POSIX et les limites de Linux
en tant que système temps réel.

2 Analyse de difference entre deux codes

Dans un premier temps, nous executons le code `gettimeofday.c` qui va executer une boucle qui execute le fonction `gettimeofday()` et ecrire la sortie dans le fichier `results.txt` .

code `gettimeofday.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>

#define NB_MESURES 30

int main (int argc, char **argv)
{
    struct timeval tv;
    int i;

    for (i = 0; i < NB_MESURES; ++i) {
        gettimeofday(&tv, NULL);
        printf("%2d : %ld.%06ld\n", i, tv.tv_sec, tv.tv_usec);
    }

    return EXIT_SUCCESS;
}
```

results.txt

```
 0 : 4109.653766
 1 : 4109.655552
 2 : 4109.655565
 3 : 4109.655574
 4 : 4109.655582
 5 : 4109.655591
 6 : 4109.655599
 7 : 4109.655607
 8 : 4109.655615
 9 : 4109.655624
10 : 4109.655632
11 : 4109.655640
12 : 4109.655648
13 : 4109.655656
14 : 4109.655664
15 : 4109.655673
16 : 4109.655681
17 : 4109.655689
18 : 4109.655697
19 : 4109.655705
20 : 4109.655713
21 : 4109.655722
22 : 4109.655730
23 : 4109.655738
24 : 4109.655746
25 : 4109.655754
26 : 4109.655762
27 : 4109.655770
28 : 4109.655779
29 : 4109.655787
```

Puis, nous allons modifier le code dans `gettimeofday2.c` et utiliser un tableau de `struct timeval` puis, dans une autre boucle, print les valeurs du tableau puis on va sortir les valeurs dans `results2.txt` .

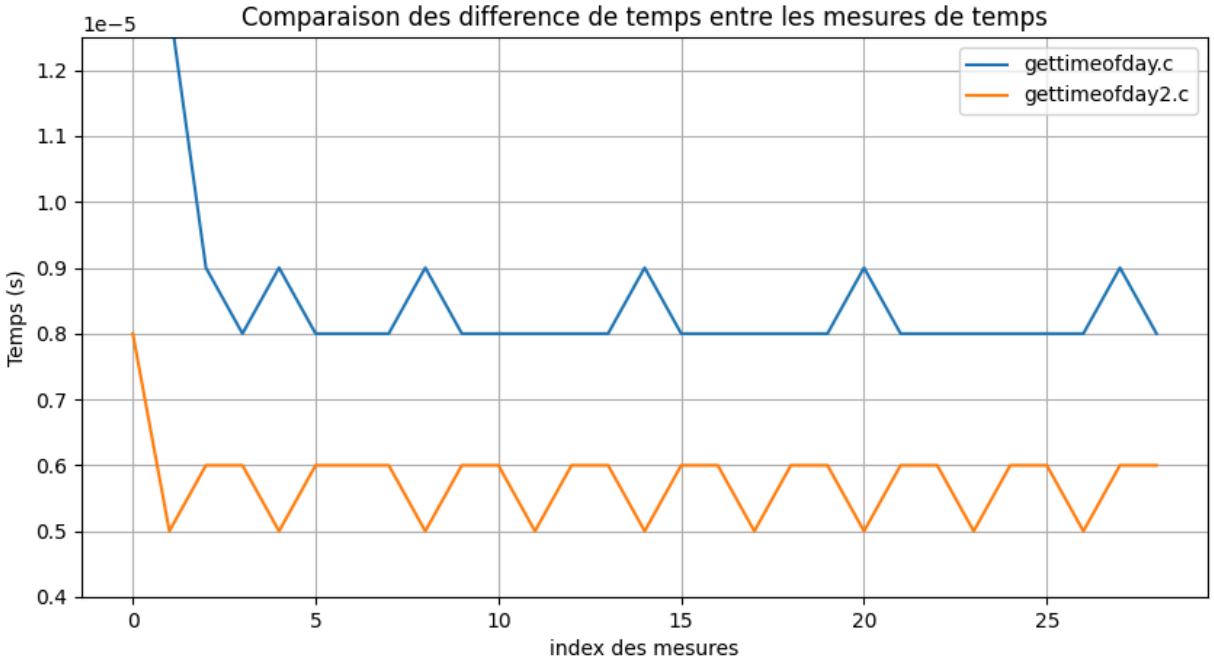
code `gettimeofday2.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>

#define NB_MESURES 30

int main(int argc, char **argv)
{
    struct timeval tv[NB_MESURES];
    int i;

    for (int i = 0; i < NB_MESURES; i++)
    {
        gettimeofday(&tv[i], NULL);
    }

    for (int i = 0; i < NB_MESURES; i++)
    {
        printf("%2d : %ld.%06ld\n", i, tv[i].tv_sec, tv[i].tv_usec);
    }

    return EXIT_SUCCESS;
}
```

results2.txt

```
 0 : 4111.520325
 1 : 4111.520333
 2 : 4111.520338
 3 : 4111.520344
 4 : 4111.520350
 5 : 4111.520355
 6 : 4111.520361
 7 : 4111.520367
 8 : 4111.520373
 9 : 4111.520378
10 : 4111.520384
11 : 4111.520390
12 : 4111.520395
13 : 4111.520401
14 : 4111.520407
15 : 4111.520412
16 : 4111.520418
17 : 4111.520424
18 : 4111.520429
19 : 4111.520435
20 : 4111.520441
21 : 4111.520446
22 : 4111.520452
23 : 4111.520458
24 : 4111.520463
25 : 4111.520469
26 : 4111.520475
27 : 4111.520480
28 : 4111.520486
29 : 4111.520492
```

Afin d'analyser au mieux les differences, nous avons calculé la différence entre deux valeurs consécutives dans les deux fichiers puis établis nous avons tracé le résultat dans un graphique.



On voit clairement que le temps ecoulé entre 2 mesures consécutives est plus grand dans l'éxecution du premier code `gettimeofday.c` que dans l'éxecution du deuxieme code `gettimeofday2.c` . Cela est surement dû au fait que le premier code utilise un seul `struct timeval` et print les valeurs dans la même boucle alors que le deuxième code utilise un tableau de `struct timeval` et fait le travail de printer les valeurs dans un deuxième temps.

La granularité ainsi que la précision des valeurs de temps est de l'ordre de la `microseconde` dans les deux cas.

3 Horloges Posix

Nous allons repeter l'experience avec les horloges POSIX. Nous allons executer le code `gettimeofday3.c` qui va executer une boucle qui execute le fonction `clock_gettime()` sur les quatres horloges POSIX

```
CLOCK_REALTIME
CLOCK_MONOTONIC
CLOCK_PROCESS_CPUTIME_ID
CLOCK_THREAD_CPUTIME_ID
```

code `gettimeofday3.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

#define NB_MESURES 30

const char *posix_clocks[] = {
    "CLOCK_REALTIME",
    "CLOCK_MONOTONIC",
    "CLOCK_PROCESS_CPUTIME_ID",
    "CLOCK_THREAD_CPUTIME_ID"
};

const clockid_t posix_clock_ids[] = {
    CLOCK_REALTIME,
    CLOCK_MONOTONIC,
    CLOCK_PROCESS_CPUTIME_ID,
    CLOCK_THREAD_CPUTIME_ID
};

int main(int argc, char **argv) {
    struct timespec ts[NB_MESURES];
    int i;

    for (i = 0; i < sizeof(posix_clocks) / sizeof(posix_clocks[0]); ++i) {
        struct timespec res;
        if (clock_getres(posix_clock_ids[i], &res) != 0) {
            perror("clock_getres failed");
            return EXIT_FAILURE;
        }

        for (int j = 0; j < NB_MESURES; ++j) {
            if (clock_gettime(posix_clock_ids[i], &ts[j]) != 0) {
                perror("clock_gettime failed");
                return EXIT_FAILURE;
            }
        }

        char filename[256];
        snprintf(filename, sizeof(filename), "%s.txt", posix_clocks[i]);
        FILE *fp = fopen(filename, "w");
        if (fp == NULL) {
```

```
            perror("Failed to open file");
            return EXIT_FAILURE;
        }

        for (int j = 0; j < NB_MESURES; ++j) {
            fprintf(fp, "%2d : %ld.%09ld\n", j, ts[j].tv_sec, ts[j].tv_nsec);
        }

        fclose(fp);
    }

    return EXIT_SUCCESS;
}
```

et ecrire la sortie dans le fichier portant le `<nom horloge>.txt` .

Voici le resultat obtenus

CLOCK_REALTIME.txt

```
 0 : 8446.132220770
 1 : 8446.132226670
 2 : 8446.132232550
 3 : 8446.132238190
 4 : 8446.132243880
 5 : 8446.132249610
 6 : 8446.132255400
 7 : 8446.132261070
 8 : 8446.132266690
 9 : 8446.132272310
10 : 8446.132277950
11 : 8446.132283560
12 : 8446.132289190
13 : 8446.132294810
14 : 8446.132300430
15 : 8446.132306060
16 : 8446.132311680
17 : 8446.132317300
18 : 8446.132322930
19 : 8446.132328570
20 : 8446.132334200
21 : 8446.132339830
22 : 8446.132345460
23 : 8446.132351130
24 : 8446.132356760
25 : 8446.132362400
26 : 8446.132368020
27 : 8446.132373670
28 : 8446.132379290
29 : 8446.132384910
```

CLOCK_MONOTONIC.txt

```
 0 : 8446.147386030
 1 : 8446.147392250
 2 : 8446.147397990
 3 : 8446.147403650
 4 : 8446.147409330
 5 : 8446.147414990
 6 : 8446.147420640
 7 : 8446.147426320
 8 : 8446.147432010
 9 : 8446.147437670
10 : 8446.147443330
11 : 8446.147448980
12 : 8446.147454630
13 : 8446.147460300
14 : 8446.147465930
15 : 8446.147471580
16 : 8446.147477300
17 : 8446.147483040
18 : 8446.147488690
19 : 8446.147494370
20 : 8446.147500020
21 : 8446.147505660
22 : 8446.147511320
23 : 8446.147517110
24 : 8446.147522770
25 : 8446.147528390
26 : 8446.147534050
27 : 8446.147539740
28 : 8446.147545400
29 : 8446.147551040
```

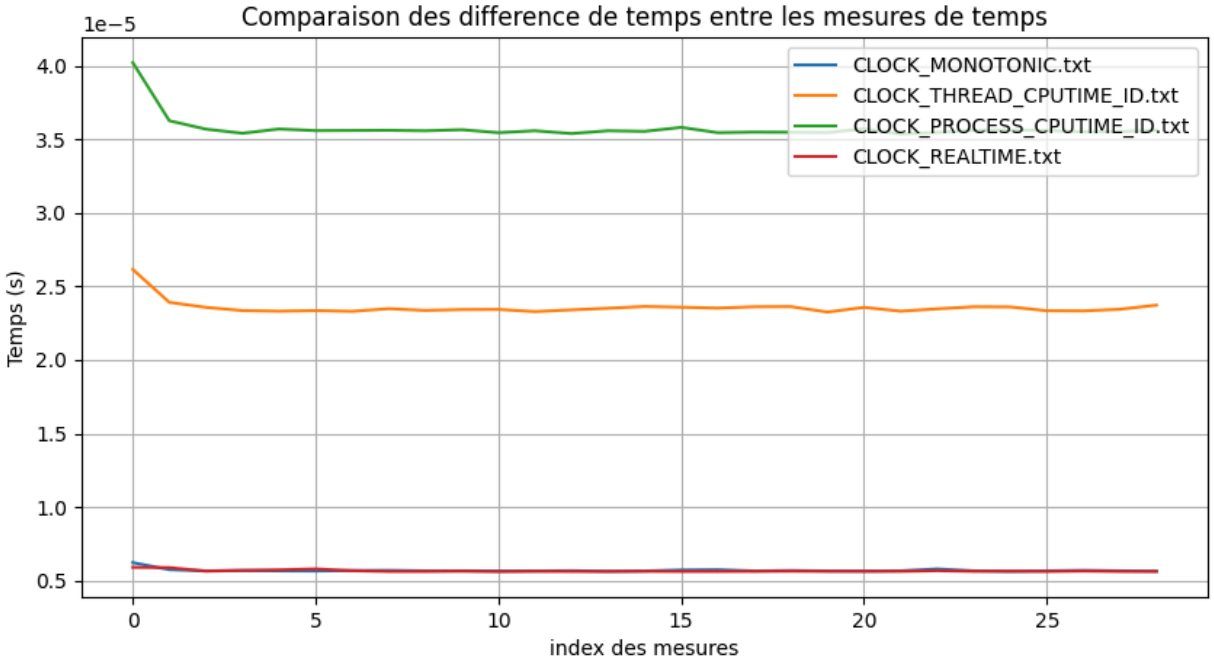
CLOCK_PROCESS_CPUTIME_ID.txt

```
 0 : 0.051119500
 1 : 0.051159720
 2 : 0.051195990
 3 : 0.051231690
 4 : 0.051267110
 5 : 0.051302820
 6 : 0.051338420
 7 : 0.051374030
 8 : 0.051409650
 9 : 0.051445240
10 : 0.051480900
11 : 0.051516360
12 : 0.051551950
13 : 0.051587350
14 : 0.051622940
15 : 0.051658490
16 : 0.051694310
17 : 0.051729770
18 : 0.051765270
19 : 0.051800760
20 : 0.051836230
21 : 0.051871950
22 : 0.051907320
23 : 0.051942800
24 : 0.051978410
25 : 0.052014060
26 : 0.052049660
27 : 0.052085180
28 : 0.052120710
29 : 0.052156330
```

CLOCK_THREAD_CPUTIME_ID.txt

```
 0 : 0.064892760
 1 : 0.064918920
 2 : 0.064942840
 3 : 0.064966420
 4 : 0.064989780
 5 : 0.065013100
 6 : 0.065036460
 7 : 0.065059770
 8 : 0.065083260
 9 : 0.065106630
10 : 0.065130060
11 : 0.065153500
12 : 0.065176790
13 : 0.065200200
14 : 0.065223720
15 : 0.065247360
16 : 0.065270950
17 : 0.065294480
18 : 0.065318100
19 : 0.065341730
20 : 0.065364990
21 : 0.065388570
22 : 0.065411890
23 : 0.065435370
24 : 0.065458990
25 : 0.065482600
26 : 0.065505950
27 : 0.065529290
28 : 0.065552740
29 : 0.065576470
```

Afin d'analyser au mieux les differences, nous avons calculé la différence entre deux valeurs consécutives dans les deux fichiers puis établis nous avons tracé le résultat dans un graphique.


Comparaison des difference de temps entre les mesures de temps
1e−5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
Temps (s)
0
5
10
15
20
25
index des mesures
CLOCK_MONOTONIC.txt
CLOCK_THREAD_CPUTIME_ID.txt
CLOCK_PROCESS_CPUTIME_ID.txt
CLOCK_REALTIME.txt