Module Master M1

Systèmes temps réel et Informatique Industrielle
Chapitre VII : Gestion du temps

Projet n°04 d'exposé d'étudiants de Master M1 Informatique

Présenté par : Prof. Kholladi Mohamed-Khireddine
Département d'Informatique
Facultés des Sciences Exactes
Université Echahid Hamma Lakhdar d'El Oued
Tél. 0770314924
Email. kholladi@univ-eloued.dz et kholladi@yahoo.fr
Site Web. www.univ-eloued.dz
http://kholladi.doomby.com/ et http://kholladi.e-monsite.com/

VII – Gestion du temps

VII.1 – Horloges et chronomètres

- Exécution d'une tâche à un moment donné :
  - Horloge absolue (clock).
- Ou à intervalles réguliers :
  - Chronomètres (timers) :
- À un coup,
- À répétition.
- UNIX (POSIX.1) fournit déjà plusieurs outils.
- POSIX.4 rajoute de la précision et des fonctionnalités souvent associées à une horloge "temps réel", externe ou non.

VII.2 - Les horloges POSIX.1

time:

#include <time.h>

time_t time(time_t *what_time);

Retourne le nombre de secondes écoulées depuis le 1er janvier 1970 0h et le stocke dans what_time.

gettimeofday:

struct timeval {

time_t tv_sec;

time_t tv_usec;}

int gettimeofday(struct timeval *what_time);

La résolution est de toutes façons limitée par l'horloge de la machine (~ 0,1s).

Codage sur trente deux (32) bits ➔ bug du 15 février 2038 !

VII.3 - Les timers POSIX.1

```
#include <sys/time.h>
int setitimer(int id,
        const struct itimerval *new_value,
        struct itimerval *old_value);
int getitimer(int id,
        struct itimerval *current_value),
        struct itimerval{struct timeval it_value,
                struct timeval it_interval;}
```

id : ITIMER_REAL → signal SIGALRM

ITIMER_VIRTUAL → signal SIGVTALRM

ITIMER_PROF → signal SIGPROF

```
int alarm(const int i);
```

Envoie le signal SIGALRM au bout de i secondes.

```
int sleep(const int n);
```

Suspend le processus pendant n secondes.

Problèmes de précision et de dérive : la précision ultime est ~ un µs.

ITIMER_VIRTUAL (utilisation de la CPU par le processus) et ITIMER_PROF (utilisation de la CPU par le processus et le système au nom du processus) sont utilisés pour le profilage.

VII.4 - La gestion du temps avec POSIX.4

a - Horloges

```
#include <time.h>
int clock_settime(clockid_t id,
const struct timespec *curr_time);
int clock_gettime(clockid_t id,
        struct timespec *curr_time);
int clock_getres(clockid_t id,
        struct timespec *resolution);
int clock_nanosleep(const struct timespec *req,
        struct timespec *remaining);
```

b - Timers

```
#include <time.h>
int timer_create(clockid_t id,
        const struct sigevent *signal_spec,
        timer_t *timer_id);
int timer_settime(timer_t timer_id, int flags,
        const struct itimerspec *new_int,
        struct itimerspec *old_int);
int timer_gettime(timer_t timer_id,
        struct itimerspec *curr_int);
int timer_getoverrun(timer_t timer_id);
int timer_delete(timer_t timer_id);
int nanosleep(const struct timespec *req,
        struct timespec *remaining);
```

VII.5 - Les horloges POSIX.4

a - Horloges fournies par Linux (>2.6.12)

- CLOCK_REALTIME - Une horloge système temps réel configurable.
- CLOCK_MONOTONIC - Une horloge non configurable, toujours croissante qui mesure le temps depuis un instant non spécifié dans le passé et qui ne change pas après le démarrage du système.
- CLOCK_PROCESS_CPUTIME_ID - Une horloge qui mesure le temps CPU (utilisateur et système) consommé par le processus appelant (et tous ses threads).

- CLOCK_THREAD_CPUTIME_ID - Une horloge qui mesure le temps CPU (utilisateur et système) consommé par le thread appelant.
- Seule CLOCK_REALTIME existe partout.

Structure de stockage du temps

```
struct timespec {
        time_t tv_sec;
        time_t tv_nsec;
                };
```

```
int clock_gettime (clockid_t clock_id,
        struct timespec *tp)
```

Lit la valeur courante de l'horloge spécifiée.

```
int clock_settime (clockid_t clock_id,
        struct timespec *tp)
```

Règle l'horloge :

- Valide uniquement pour CLOCK_REALTIME

Utilisé pour spécifier l'heure à laquelle sera réveillé un thread bloqué sur un appel à clock_nanosleep avec une date absolue sur CLOCK_REALTIME.

```
int clock_getres (clockid_t clock_id,
struct timespec *tp)
```

Lit la résolution de l'horloge spécifiée.

```
int clock_nanosleep(clockid_t clock_id,
                int flags,
                const struct timespec *rqtp,
                struct timespec *rmtp)
```

Suspend la tâche :

- Jusqu'à la date spécifiée par rqtp
- Ou jusqu'à l'arrivée d'un signal géré par l'application

Si TIMER_ABSTIME est spécifié dans flags, rqtp une date absolue, sinon rqtp est un intervalle.

Si la tâche est réveillée par l'arrivée d'un signal et si TIMER_ABSTIME n'est pas spécifié et si rmtp n'est pas NULL, alors le temps restant jusqu'au réveil normal est retourné dans rmtp.

```
int nanosleep(const struct timespec *rqtp,
              struct timespec *rmtp)
```

Identique à clock_nanosleep, mais uniquement pour des intervalles de temps.

VII.6 - Les timers POSIX.4

Structure pour gérer les données relatives au timer.

```
struct itimerspec {
        struct timespec it_value;
        struct timespec it_interval;
        }
```

- it_value : date d'expiration,
- it_interval : valeur de rechargement, après expiration.

```
int timer_create(clockid_t clock_id,
        const struct sigevent *evp,
        timer_t *timerid)
```

Crée un timer basé sur l'horloge spécifiée.

evp décrit le mécanisme de notification à l'expiration du timer.

L'identificateur du timer est retourné dans timerid.

Structure sigevent

- sigev_notify :

- SIGEV_NONE : pas de notification (on suit le timer avec timer_gettime)
- SIGEV_SIGNAL : un signal temps réel est envoyé au processus
- SIGEV_THREAD_ID : le signal est envoyé à un thread
- SIGEV_THREAD : une fonction de notification est exécutée

- sigev_signo : numéro du signal envoyé

- sigev_value : valeur transportée
- union sigval sigev_value définie par :

int sival_int ;

void * sival_ptr;

- Sigev_notify_function : nom de la fonction exécutée par le thread créé (SIGEV_THREAD)
- Sigev_notify_attributes : attributs du thread créé (SIGEV_THREAD)
- Sigev_notify_thread_id : TID du thread auquel est envoyé le signal (SIGEV_THREAD_ID)

int timer_delete(timer_t timerid)

Détruit le timer.

int timer_settime(timer_t timerid,
int flags,
const struct itimerspec *value,
struct itimerspec *ovalue)

- Arme le timer.
- Si value.it_value est nul, le timer est arrêté, sinon il est démarré.
- Si value.it_interval est différent de zéro, le timer est périodique.
- Si flags est égal à TIMER_ABSTIME, value.it_value est interprété comme une date absolue de l'horloge spécifiée dans timer_create, sinon c'est interprété comme un intervalle.
- Le thread appelant soit être un thread POSIX (créé par pthread_create).
- Les valeurs courantes du timer sont stockées dans ovalue.

int timer_gettime(timer_t timerid,
struct itimerspec *value)

Stocke dans value la date d'expiration et la valeur de rechargement du timer.

int timer_getoverrun(timer_t timerid)

- Retourne le nombre de dépassements du timer depuis la dernière expiration du timer timerid.

- Nombre de fois où le timer a expiré alors qu'il n'y avait pas de tâche ayant mis en place une procédure de gestion du signal émis à l'expiration.