Practica 3 Creación de Procesos

En esta práctica se muestra la creación de procesos sobre un sistema operativo de tiempo real. En particular, se muestra cómo crear procesos periódicos haciendo uso de varios servicios disponibles sobre el sistema operativo MaRTE OS.

Funciones utilizadas

La función (incluir time.h)

int clock\_gettime (clockid\_t clock\_id, struct timespec \*tp);

permite obtener el tiempo transcurrido en la struct timespec cuya dirección se le pasa como segundo argumento. El argumento clock\_id puede ser CLOCK\_MONOTONIC, CLOCK\_REALTIME, CLOCK\_THREAD\_CPUTIME\_ID. La struct timespec está declarada en el archivo de cabecera time.h

struct timespec {

time\_t tv\_sec; /\*\* segundos transcurridos\*/

int tv\_nsev;/\*\* nanosegundos transcurridos después de los tv\_sec segundos \*/

};

La función (incluir misc/timespec\_operations.h)

static inline void incr\_timespec (struct timespec \*t1, const struct timespec \*t2)

incrementa el tiempo almacenado en la estructura timespec a la que apunta t1, sumándole en tiempo almacenado en la estructura timespec a la que apunta t2.

La función (incluir pthread.h)

int pthread\_attr\_init (pthread\_attr\_t \*attr);

se utiliza para inicializar la estructura de atributos de thread de la cual se pasa su dirección al llamar a esta función.

La función (incluir pthread.h)

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr,

void \*(\*start\_routine)(void \*), void \*arg);

crea un thread (un proceso ligero) a partir de la información que se le pasa como argumentos. En esta función, el parámetro thread su usa para pasar un entero por referencia, en el parámetro attr se debe pasar la dirección de una estructura pthread\_attr\_t previamente incializada con la función pthread\_attr\_init() mencionada antes. En el parámetro start\_routine, se debe pasar un apuntador a función de tipo void \* que recibe un apuntador a void, esto es, el nombre de una función void \*fn(void \*). En el parámetro arg se pasa un apuntador a void que en el código de la función fn se usa como argumento de la misma.

La función (incluir time.h)

int clock\_nanosleep(clockid\_t clock\_id, int FLAGS,

const struct timespec \*rqtp, struct timespec \*rmtp);

se utiliza para poner a dormir al thread que la llama durante el tiempo especificado en la estructura timespec a la que apunta rqtp. El cuarto parámetro rmtp puede ser NULL.

Utilización de las funciones anteriores

En la función main() del presente ejercicio práctico, primero se crea un hilo llamando a las funciones pthread\_attr\_init() y pthread\_create():

pthread\_t th;

pthread\_attr\_t attr;

CHK( pthread\_attr\_init (&attr) );

CHK( pthread\_create (&th, &attr, periodic, &period1) );

Después se llamará a la función clock\_gettime() como sigue

struct timespec next\_activ\_dummy, my\_lag;

// …

if (clock\_gettime (CLOCK\_MONOTONIC, &next\_active\_dummy) )

printf(“Error en clock\_gettime”);

Con esto se obtiene en la estructura next\_active\_dummy, el tiempo en segundos y nanosegundos transcurrido desde que inició la aplicación. Después, se inicializarán los campos tv\_sec y tv\_nsec (segundos y nanosegundos) de la estructura my\_lag.

my\_lag.tv\_sec = 0; /\* 0 segundos \*/

my\_lag.tv\_nsec = 500000000; /\* 0.5 segundo \*/

Una vez hecho esto, se llama a la función clock\_nanosleep() como se indica aquí:

If ( clock\_nanosleep(CLOCK\_MONOTONE, TIMER\_ABSTIME,

&next\_activ\_dummy, NULL) )

printf(“Error in clock\_nanosleep”);

Después, se crea un segundo thread llamando nuevamente a