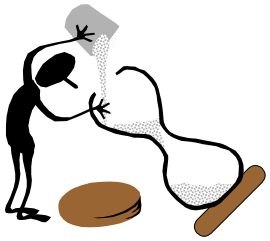
Universidad del Cauca
Departamento de Telemática

Sistemas de Tiempo Real



Administración del Tiempo

Dr. Ing. Álvaro Rendón Gallón
Popayán, abril de 2011

Temario

- Relojes, retardos y temporizadores
- El tiempo en POSIX 1003.1
- Relojes y temporizadores en POSIX 1003.1b
- El tiempo en RT-Java

Departamento de Telemática

Relojes, retardos y temporizadores

- La noción del tiempo en programación está basada en cuatro requerimientos:
  - Acceso a un reloj
  - Retardo de las tareas
  - Programación de temporizadores
  - Especificación y planificación de los plazos

- Los 4 están muy interrelacionados
- El último es la esencia de la programación en tiempo real
  - Se tratará en el capítulo de Planificación

Departamento de Telemática

Relojes

- Los programas deben disponer de mecanismos para:
  - Conocer la hora (y la fecha)
  - Medir el paso del tiempo

- Dos mecanismos:
  - Incluir una primitiva de reloj en el lenguaje (Ada)
  - Construir un controlador del dispositivo de reloj asociado al procesador
- Se requiere un sistema de referencia
  - Con un origen (Época). UNIX: 00:00, 01/01/1970 UT
  - Con una escala de tiempo. UNIX: segundos
    long time; /* El fin del mundo en febrero de 2038!! */

Relojes

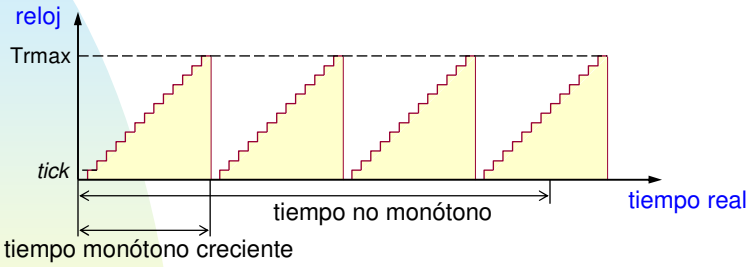
Departamento de Telemática

- Sistemas de referencia (tiempo absoluto)
  - Internos: Dispositivos asociados al procesador
  - Externos: GPS
- Los relojes internos están compuestos por:
  - Un oscilador: Base de tiempo
  - Un contador: Acumula impulsos del oscilador
- Características más importantes:
  - Precisión: Diferencia con un reloj de referencia
  - Estabilidad: Variación con el tiempo
  - Granularidad: Período del oscilador (*tick*). µseg/mseg
  - Resolución: Unidad mínima del contador. 1 nseg
  - Intervalo: Capacidad máxima del contador (Trmax)

Relojes

Departamento de Telemática

- El contador se puede desbordar. Se reinicia la cuenta

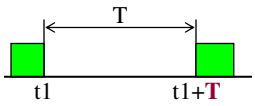


- **Tiempo monótono creciente**: El valor del reloj se incrementa de manera permanente
- **Tiempo no monótono**: El valor del reloj tiene saltos hacia atrás (e.g. Problema Y2K)
- El tiempo de los relojes debe ser monótono creciente

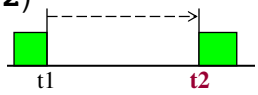
Departamento de Telemática

Retardos

- Las tareas deben poder retrasarse por un cierto período de tiempo
- Se ponen en cola para ser activadas más adelante, en lugar de hacer espera ocupada
- Retardo relativo
  - Intervalo referido al instante de la llamada (**T**)



- Retardo absoluto
  - Instante futuro (**t2**)



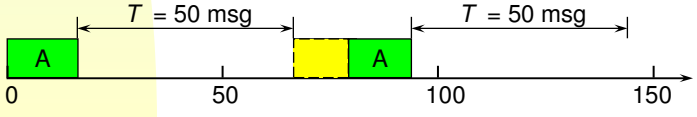
Departamento de Telemática

Retardos

- Se pueden usar para implementar tareas periódicas

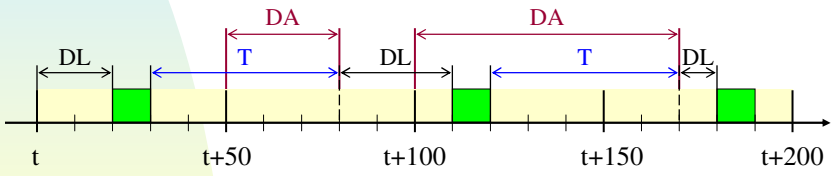
```
T=50;
while (TRUE) {
  tarea();
  sleep(T); /* Suspensión de T unidades */
}
```

- El intervalo es aproximado (al menos T)
  - No considera el tiempo de ejecución de la tarea
  - Puede haber tareas de mayor prioridad en cola, que retrasan la activación



Deriva en los retardos

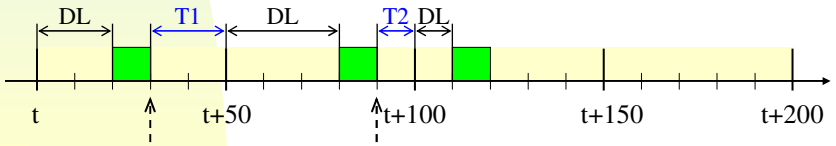
- **Deriva local**: Tiempo adicional que dura el retardo
- **Deriva acumulada**: Desplazamiento obtenido en invocaciones sucesivas por la suma de las derivas locales



- La deriva local no se puede evitar, pero sí la deriva acumulada

Eliminación de deriva acumulada

```
T=50;
siguiente= time();
while (TRUE) {
  sleep(siguiente - time());
  tarea();
  siguiente= siguiente + T;
}
```



sigui	0	50	100
sleep	0	20	10

Se suma DL por tareas de mayor prioridad

Departamento de Telemática

Temporizadores

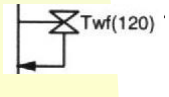
- La restricción temporal más simple, es el requerimiento de reconocer la **no ocurrencia** de un evento y reaccionar en consecuencia
- En general, una temporización es una restricción sobre el tiempo que una tarea permanece a la espera de un evento
- Las temporizaciones son a menudo incluidas en las primitivas de sincronización y comunicación

  sem_timedwait(), pthread_cond_timedwait(), mq_timedreceive()

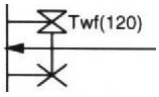
Departamento de Telemática

Temporizadores

- Dos tipos:
  - Un sólo disparo
  - Periódicos
- Tienen asociados tres eventos:
  - Activar: Se fija valor, tipo, etc.
  - Cancelar: El evento esperado llega a tiempo
  - Expirar: Normalmente es una señal (e.g. SIGALRM)



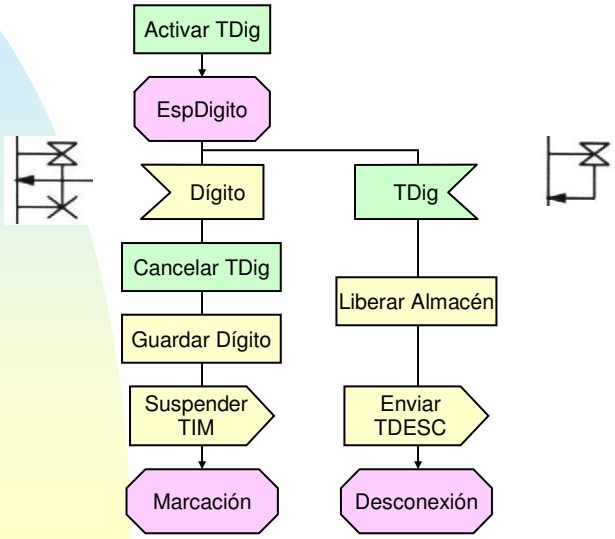
Activación-expiración



Activación-cancelación

Departamento de Telemática

Temporización en SDL



Departamento de Telemática

Temario

- Relojes, retardos y temporizadores
- **El tiempo en POSIX 1003.1**
- Relojes y temporizadores en POSIX 1003.1b
- El tiempo en RT-Java

Departamento de Telemática

El tiempo en POSIX 1003.1

- Reloj-calendario
  - Un solo reloj, con una resolución de 1 seg
  - Operaciones para:
    - Consultar el reloj
    - Conversión
    - Obtener tiempo de procesamiento
- Funciones de temporización
  - Temporizador
  - Retardo
  - Suspensión

Departamento de Telemática

Reloj-Calendario

- Consultar el reloj-calendario

  time_t **time**(time_t *relcal);

  relcal: Dirección para el resultado:
  Número de segundos transcurridos de la Época (00:00:00, 01/01/1970 UT/GMT)

  retorna: Ídem

Funciones de temporización

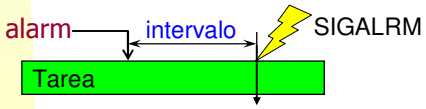
- Temporización
  - Programar una señal de alarma

  unsigned **alarm**(unsigned intervalo)

  intervalo: Tiempo de temporización en segundos.
  Cuando transcurre el intervalo, se genera la señal SIGALRM.
  Si intervalo=0, se cancela la temporización previa

  retorna: 0, si no hay temporizaciones previas
  tiempo restante, si había temp. previas

  Si hay una temporización pendiente, se reprograma



Funciones de temporización

- Retardo
  - Suspender la ejecución durante un intervalo de tiempo

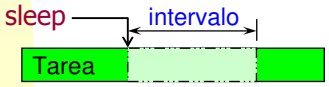
  unsigned **sleep**(unsigned intervalo)

  intervalo: Tiempo de suspensión en segundos

  retorna: 0, si se agota el retardo
  tiempo restante, si se interrumpe el retardo

  Puede ser interrumpido por una señal que se atiende con un manejador o que termina el proceso

  Algunas implementaciones están basadas en alarm(), por lo que puede haber interacciones con la señal SIGALARM.



Funciones de temporización

- Suspensión
  - Suspender la tarea hasta la recepción de una señal

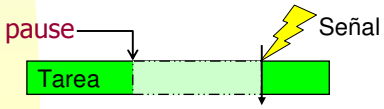
  int **pause**(void)

  retorna: -1: error

  Termina cuando se recibe una señal que se atiende con un manejador o que termina el proceso

  Algunos SOTR usan esta función para implementar sleep()

  Puede presentarse una condición de carrera entre la verificación del estado de una señal y la invocación de pause(): usar sigprocmask() y sigsuspend()



Temario

- Relojes, retardos y temporizadores
- El tiempo en POSIX 1003.1
- **Relojes y temporizadores en POSIX 1003.1b**
- El tiempo en RT-Java

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Los servicios de tiempo provistos por POSIX básico son insuficientes o inadecuados para las aplicaciones de tiempo real
- Éstas requieren:
  - Soporte para relojes adicionales
  - Mayor resolución
  - Posibilidad de enterarse del desbordamiento de los temporizadores
  - Posibilidad de utilizar otras señales distintas de SIGALRM para indicar la expiración de los temporizadores

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Puede haber varios relojes. Tienen nombres.
  Al menos uno: CLOCK_REALTIME
  Opcional: CLOCK_MONOTONIC
- La resolución de la representación del tiempo es de 1 nanosegundo. Especificación del tiempo:

```
struct timespec {
  time_t   tv_sec;     /* No. segundos */
  long     tv_nsec;    /* No. nanosegundos */
};
```

  t = (tv_sec * 10^9 + tv_nsec) nseg
- El valor máximo de la granularidad de CLOCK_REALTIME es 20 mseg (20.000.000 nseg)
- Los temporizadores se crean en forma dinámica

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Operaciones
  - Gestión de los relojes
  - Gestión de temporizadores
    - Relativos o absolutos
    - Un solo disparo o periódicos
  - Retardo de alta resolución

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Gestión de los relojes
  - Consultar un reloj

    int **clock_gettime**(clockid_t reloj,
    struct timespec *valor)
  - Fijar el valor de un reloj

    int **clock_settime**(clockid_t reloj,
    const struct timespec *valor)

    No es válida para CLOCK_MONOTONIC
  - Obtener la resolución de un reloj

    int **clock_getres**(clockid_t reloj, struct timespec *res)

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Gestión de temporizadores
  - Crear un temporizador para un proceso

  int **timer_create**(clockid_t reloj,
  struct sigevent *evento, timer_t *temporizador)

  reloj: Reloj utilizado por el temporizador
  evento: Señal a enviar en la expiración
  temporizador: ID del temporizador creado
  retorna: 0: éxito, -1: error

  Si evento= NULL y reloj= CLOCK_REALTIME,
  la señal enviada es SIGALRM
  Las señales de expiración no se ponen en cola!.
  Hay un contador de desbordamientos

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

```
struct sigevent {
   ...
   int            sigev_notify
   int            sigev_signo
   union sigval   sigev_value
   ...
}
```

sigev_notify: Tipo de notificación
SIGEV_SIGNAL= Enviar señal
SIGEV_NONE= No enviar notificación

sigev_signo: Número de la señal a enviar

sigev_value: Valor a entregar con la señal (POSIX 1003.1b)

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

– Borrar un temporizador

int **timer_delete**(timer_t temporizador)

temporizador: Temporizador

retorna: 0: éxito, -1: error

Si está activado, lo cancela

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

– Activar/cancelar un temporizador

int **timer_settime**(timer_t temporizador, int bands,
const struct itimerspec *valores,
struct itimerspec *valoresant)

temporizador: Temporizador
bands: 0= Valor relativo; TIMER_ABSTIME= Absoluto
valores: Valores de la temporización
valoresant: Valores anteriores

```
struct itimerspec {
   struct timespec  it_interval  // periodo
   struct timespec  it_value     // primera expiracion
}
```

it_interval=0: Un solo disparo, con T= **it_value**
it_value=0: Se cancela el temporizador

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

– Obtener el tiempo restante de un temporizador

int **timer_gettime**(timer_t temporizador,
struct itimerspec *valores)

temporizador: Temporizador
valores: Tiempo hasta la próxima expiración y período

– Obtener el contador de desbordamiento

int **timer_getoverrun**(timer_t temporizador)

temporizador: Temporizador
retorna: Contador de desbordamientos

Relojes y temporizadores en POSIX 1003.1b

Departamento de Telemática

- Retardo de alta resolución

– Suspende la ejecución por un número de nseg

int **nanosleep**(const struct timespec *tiempo,
struct timespec *remanente)

tiempo: Duración del retardo en nanosegundos

remanente: Tiempo restante cuando la función retorna antes de la expiración

retorna: 0: éxito, -1: error

Puede ser interrumpido por una señal que se atiende con un manejador o que termina el proceso

Pocos sistemas tienen una granularidad de nseg (normal: µseg)

La duración solicitada es redondeada hacia arriba

Departamento de Telemática

Temario

- Relojes, retardos y temporizadores
- El tiempo en POSIX 1003.1
- Relojes y temporizadores en POSIX 1003.1b
- **El tiempo en RT-Java**

Departamento de Telemática

El tiempo en RTJava

- La especificación de RTJava define la clase HighResolutionTime para expresar el tiempo con una exactitud de nanosegundos.

  HighResolutionTime es una clase abstracta que no se usa de manera directa. Se usan sus clases heredadas:

  - AbsoluteTime: Tiempo desde la Época
  - RelativeTime: Tiempo relativo
  - RationalTime: Frecuencia, expresada por: un numerador long (número de ocurrencias), y un denominador RelativeTime (intervalo de las ocurrencias)

Java Community Process

Departamento de Telemática

El tiempo en RTJava

- Los constructores para las clases heredadas son:
  - AbsoluteTime ()
  - AbsoluteTime (AbsoluteTime time)
  - AbsoluteTime (java.util.Date date)
  - AbsoluteTime (long millis, int nanos)
  - RelativeTime ()
  - RelativeTime (RelativeTime time)
  - RelativeTime (long millis, int nanos)
  - RationalTime (int frequency) // interval=1seg
  - RationalTime (int frequency, long millis, int nanos)
  - RationalTime (int frequency, RelativeTime interval)

Departamento de Telemática

El tiempo en RTJava

- Métodos para consultas y comparaciones:
  - Int HighResolutionTime.**compareTo**(HighResolutionTime time)
  - Boolean HighResolutionTime.**equals**(HighResolutionTime time)
  - java.util.Date AbsoluteTime.**getDate**()
  - java.lang.String AbsoluteTime.**toString**()
- Métodos para modificaciones de tiempo, de acuerdo a la clase utilizada:
  - AbsoluteTime **absolute**(Clock clock)
  - RelativeTime **relative**(Clock clock)
  - **add** (long millis, int nanos)
  - RelativeTime **subtract**(RelativeTime time)

Departamento de Telemática

El tiempo en RTJava

- La clase abstracta Clock ofrece los métodos para consulta de características y hora de los relojes del sistema.
- Puede haber muchos tipos de relojes.
- Hay un reloj de tiempo real monótono.
- Los métodos de la clase son:
  - **getRealtimeClock**() // retorna el reloj de tiempo real
  - **getResolution**()
  - **getTime**(AbsoluteTime time)
  - **setResolution**(RelativeTime resolution) // si el HW lo permite

Departamento de Telemática

El tiempo en RTJava

- La clase Timer permite crear y gestionar **temporizadores** para el sistema.
- El constructor de la clase es

  Timer(HighResolutionTime time, Clock clock, AsyncEventHandler handler) // time= tiempo de disparo
- Algunos métodos de la clase Timer son:
  - void **destroy**()
  - void **disable**()
  - void **enable**()
  - void **fire**(): No debe llamarse
  - Clock **getClock**()
  - AbsoluteTime **getFireTime**()
  - boolean **isRunning**()
  - void **reschedule**(** time)
  - void **start**()
  - boolean **stop**()

** Acepta cualquier clase de tiempo entre HighResolutionTime, AbsoluteTime o RelativeTime

Departamento de Telemática

El tiempo en RTJava

- La clase Timer tiene las subclases OneShotTimer y PeriodicTimer, cuyos constructores respectivamente son:
  - **OneShotTimer**(HighResolutionTime time, Clock clock, AsyncEventHandler handler)
  - **PeriodicTimer**(HighResolutionTime time, RelativeTime interval, Clock clock, AsyncEventHandler handler)
- La clase PeriodicTimer incluye los métodos adicionales:
  - RelativeTime **getInterval**()
  - void **setInterval**(RelativeTime interval)

Departamento de Telemática

Referencias

- A. Burns, A. Wellings. “Real-Time Systems and their Programming Languages”. Addison-Wesley. 1992.
- B.O. Gallmeister. “POSIX.4. Programming for the Real World”. O´Reilly, 1995.
- The Open Group. “The Open Group Base Specifications Issue 6 (IEEE Std 1003.1-2001)”. The Single UNIX Specification C950. 2001. http://www.opengroup.org/publications/catalog/t950x.htm
- The Real-Time for Java Expert Group. “The Real-Time specification for Java”. Addison-Wesley. 2000. http://www.rtj.org
- Oracle. “Sun Java Real-Time System”. 2010. http://java.sun.com/javase/technologies/realtime/rts/
- José Luis Villarroel Salcedo. “Transparencias del Curso Sistemas de Tiempo Real”. Universidad de Zaragoza. 2002
- Sebastián Sánchez Prieto (Chan). “Transparencias del curso Arquitectura de Computadores”. Universidad de Alcalá. 2002.