Estándares POSIX

- Estándares base.
  - Sintaxis y semántica con diversos aspectos del sistema operativo
  - No especifica la implementación
  - La mayoría basados en el lenguaje C
- Interfaces en diferentes lenguajes
  - Los mismos servicios bases en otros lenguajes
  - Ada y Fortran
- Entornos de sistemas abiertos
  - Perfiles de aplicación: subconjuntos de servicios requeridos en un determinado ámbito de aplicación
  - Definen un conjunto bien definido de implementaciones de SO para áreas de aplicación específicas

Estándares base

1003.1	**Servicios básicos del Sistema Operativo**
1003.1a	Extensiones a los servicios básicos
1003.1b	**Extensiones de tiempo real**
1003.1c	**Extensiones de threads**
1003.1d	**Extensiones adicionales de tiempo real**
1003.1e	Seguridad
1003.1f	Sistema de ficheros en red (NFS)
1003.1g	Comunicaciones por red
1003.1h	Tolerancia a fallos
1003.1j	**Extensiones de tiempo real avanzadas**
1003.1m	Puntos de chequeo y reintento
1003.2	Shell y utilidades
1003.2b	Utilidades adicionales
1003.2d	Ejecución por lotes (batch)
1003.3	Métodos para probar la conformidad con POSIX
1003.21	Comunicaciones para sistemas distribuidos de tiempo real

El estándar básico 1003.1

- La gestión de procesos
- Identificación y entorno de procesos
- Notificación de eventos a través de señales
- Servicios de temporización muy primitivos
- Servicios de ficheros y directorios
- Entrada / salida
- Control de terminales
- Bases de datos para usuarios y grupos

Extensiones para TR 10031.b

- Servicios para la programación concurrente
  - Sincronización por semáforos contadores
  - Objetos de memoria compartida
  - Colas de mensajes
  - Entrada / Salida asíncrona y síncronizada
- Para un comportamiento temporal predecible
  - Planificación expulsora por políticas de prioridades fijas
  - Inhibición de memoria virtual
  - Señales de tiempo real
  - Relojes y temporizadores

La extensión de threads 1003.1c

- Comparten el mismo espacio de direcciones
- Espacio asociado pequeño
- Alta eficacia en el cambio de contexto
- Servicios básicos:
  - Creación
  - Cancelación
  - Planificación
  - Sincronización

Otras extensiones para TR

- Extensiones adicionales 1003.1d
  - Arranque rápido de procesos
  - Tiempos límite de servicios bloqueantes
  - Medida y limitación de tiempos de CPU
  - Planificación del servidor esporádico
  - Información para ficheros de tiempo real
- Extensiones avanzadas 1003.1j
  - Primitivas para sincronización de multiprocesadores
  - Gestión de memoria de diversos tipos
  - Nuevos relojes monotónico y sincronizado

Subconjuntos POSIX de TR

Perfil	Nombre	Sistema de ficheros	Múltiples procesos	Múltiples threads	Plataforma típica
PSE51	Sistema de tiempo real mínimo	No	No	Si	Sistemas empotrado pequeño, sin MMU, sin disco, sin terminal
PSE52	Controlador de tiempo real	Si	No	Si	Controlador industrial de propósito especial, sin MMU, pero con un disco o disquete y un terminal
PSE53	Sistema de tiempo real dedicado	No	Si	Si	Sistema empotrado grande, con MMU, pero sin disco
PSE54	Sistema de tiempo real multipropósito	Si	Si	Si	Computador grande con requisitos de tiempo real

Perfil mínimo de Tiempo Real

Unidades de funcionalidad del POSIX 1003.1	Opciones del POSIX 1003.1b	Opciones del POSIX 1003.1c
Proceso único Señales I/O de dispositivos Soporte de lenguaje C	Inhibición de memoria virtual Idem para rangos de direcciones Semáforos Objetos de memoria compartida Señales de tiempo real Temporizadores Colas de mensjaes I/O sincronizada	Threads Atributo de tamaño de *stack* de thread Atributo de dirección del *stack* Planificación de threads por prioridad Protocolo de herencia de prioridad Protocolo de protección de prioridad

Servicios para un sistema mínimo

- Concurrencia
  - Modelo de threads y su gestión
- Planificación
  - Políticas de planificación y sus parámetros
- Sincronización
  - Protocolos de acceso a los recursos
- Temporización
- Señales
- Funciones no reentrantes

Concurrencia (SM)

- Recursos de thread
  - Identificador
  - Pila (stack)
  - Política y parámetros de planificación
  - Datos propios (contexto, máscara de señales, lista de mutex, etc.)
- Servicios
  - Manipulación de atributos
  - Creación
  - Terminación y cancelación

Manipulación de atributos de Thread

Inicializar y destruir atributos:

```
int pthread_attr_init(pthread_attr_t* __attr);
int pthread_attr_destroy(pthread_attr_t* __attr);
```

Modificación de atributos:

```
int pthread_attr_setstacksize    (pthread_attr_t* __attr, size_t __stacksize);
int pthread_attr_getstacksize    (pthread_attr_t* __attr, size_t* __stacksize);
int pthread_attr_setdetachstate  (pthread_attr_t* __attr, int __detachstate);
int pthread_attr_getdetachstate  (pthread_attr_t* __attr);
```

Valores para el atributo `detachstate`.

- `PTHREAD_CREATE_DETACHED`
- `PTHREAD_CREATE_JOINABLE`

Manipulación de threads

Creación de threads

```
int pthread_create (pthread_t* __thread,
                    pthread_attr_t* __attr,
                    void* (*__func)(void*),
                    void* __arg);

pthread_t pthread_self( void );

int pthread_detach( pthread_t  handle );
```

Terminación y cancelación

```
int pthread_join(pthread_t handle, void **return_value);
int pthread_cancel(pthread_t __thread);
int pthread_exit(void *__exit_value);
int pthread_cleanup_push(void (*__routine)(void *), void *__arg);
int pthread_cleanup_pop( int execute );
```

Planificación (SM)

- Planificación expulsora con prioridades fijas
- Políticas:
  - SCHED_FIFO (recomendada)
  - SCHED_RR
  - SCHED_OTHER (compatibilidad)
- Atributos independientes por cada thread

Planificación (definiciones)

Políticas de planificación

- `SCHED_FIFO`
- `SCHED_RR`
- `SCHED_OTHER`

Parámetros de planificación

Estructura: `struct sched_param`

Campo único: `sched_priority`

Herencia de atributos

- `PTHREAD_INHERIT_SCHED`
- `PTHREAD_EXPLICIT_SCHED`

Planificación (funciones)

Gestión de atributos de planificación:

```
int pthread_attr_setschedpolicy(pthread_attr_t *__handle, int __sched_policy);
int pthread_attr_getschedpolicy(pthread_attr_t *__handle, int *__sched_policy);
int pthread_attr_setinheritsched(pthread_attr_t *__handle, int __inherit_sched);
int pthread_attr_getinheritsched(pthread_attr_t *__handle, int *__inherit_sched);
int pthread_attr_setschedparam(pthread_attr_t *handle,
                               const struct sched_param *sched_param );
int pthread_attr_getschedparam(pthread_attr_t *handle,
                               struct sched_param *sched_param );
```

Con el thread ya creado:

```
int pthread_setschedparam   (pthread_t handle,
                             int policy,
                             const struct sched_param *param);

int pthread_getschedparam   (pthread_t handle,
                             int *policy,
                             struct sched_param *param);
```


Sincronización (SM)

- Mutex
  - Operaciones: Bloquear, liberar
  - Atributos: Protocolo
    - PTHREAD_NONE
    - PTHREAD_PRIO_INHERITANCE
    - PTHREAD_PRIO_PROTECT
- Variables condición
  - Operaciones: Esperar, señalizar, broadcast
  - Se usa siempre junto a un mutex

Mutex y Variables Condición

Mutex

```
int pthread_mutexattr_init(pthread_mutexattr_t *mu_attr_h);
int pthread_mutexattr_destroy(pthread_mutexattr_t *mu_attr_h);
int pthread_mutexattr_setprotocol     (pthread_mutexattr_t *handle,
                                        pthread_protocol_t protocol);
int pthread_mutexattr_getprotocol     ( pthread_mutexattr_t *handle,
                                        pthread_protocol_t *protocol);
int pthread_mutexattr_setprio_ceiling ( pthread_mutexattr_t *handle,
                                        int prio_ceiling);
int pthread_mutexattr_getprio_ceiling (pthread_mutexattr_t *handle,
                                        int *prio_ceiling);

int pthread_mutex_init(pthread_mutex_t *handle, pthread_mutexattr_t
                                       *mu_attr_h);
int pthread_mutex_destroy(pthread_mutex_t *handle );
int pthread_mutex_lock(pthread_mutex_t *handle );
int pthread_mutex_unlock(pthread_mutex_t *handle );
int pthread_mutex_trylock(pthread_mutex_t *mu_h );
```

Variables condición

```
int pthread_condattr_init( pthread_condattr_t *handle);
int pthread_condattr_destroy( pthread_condattr_t *handle);

int pthread_cond_init(pthread_cond_t *handle, pthread_condattr_t *cv_attr_h);
int pthread_cond_destroy(pthread_cond_t *handle );
int pthread_cond_wait(pthread_cond_t *cv_h,  pthread_mutex_t *mu_h );
int pthread_cond_signal(pthread_cond_t *handle );
int pthread_cond_broadcast(pthread_cond_t *handle );
int pthread_cond_timedwait            (pthread_cond_t *cv_h,
                                       pthread_mutex_t *mu_h,
                                       const struct timespec *abstime );
```


Temporización (SM)

- Relojes
  - CLOCK_REALTIME (min. 20 mseg.)
- Retrasos de alta resolución
  - *nanosleep()*
- Temporizadores
  - Notifican con una señal
  - Paso de un intervalo
  - Llegada de una hora
  - Expiraciones únicas o periódicas

Temporizadores

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

int nanosleep(const struct timespec *rqtp, struct timespec *rmtp);
int clock_settime(int clock_id, const struct timespec *tp);
int clock_gettime(int clock_id, struct timespec *tp);

struct sigevent
{
    int sigev_notify;    // SIGEV_SIGNAL para señal
    int sigev_signo;
    // otros campos para crear un hilo
};

struct itimerspec
{
    struct timerspec it_interval;
    struct timerspec it_value;
};

int timer_create(int clock_id, struct sigevent* event, timert_t* timer_id);
int timer_settime(timer_t timer_id, int flags, struct itimerspec* spec,
                  struct itimerspec *ovalue);


int pthread_delay_np( const struct timespec *interval );  // No portable
```


Señales (SM)

- Independiente en cada thread
- Operaciones
  - Esperar una o varias señales
  - Modificar la máscara de señales
  - Enviar un señal a un thread

```
int sigwait( const sigset_t* sigset, int *sig );
int pthread_sigmask( int how, const sigset_t *newmask, sigset_t *prev );
int pthread_kill(pthread_t handle, int sig );
```

Funciones no reentrantes (SM)

- Existen funciones no reentrantes en la librería
- Sustitución por las equivalentes reentrantes con un #define
- Precaución con los módulos de librerías que usan funciones reentrantes