INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL
INTERDISCIPLINARIA EN INGENIERÍA Y
TECNOLOGÍAS AVANZADAS

UPIITA

SISTEMAS OPERATIVOS EN TIEMPO REAL

3MV9

ALUMNO:

REYES GARCÍA MAURICIO ITZAE

2014640223

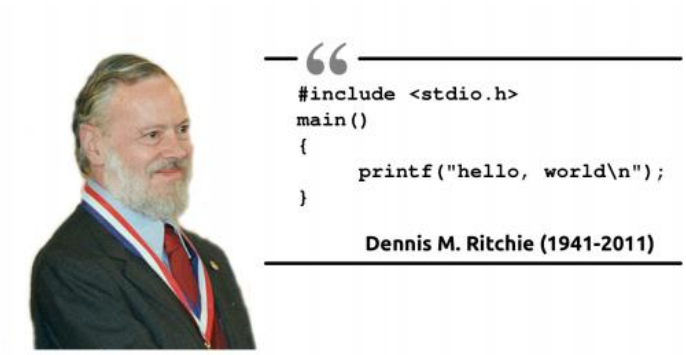
31/05/2019

INTRODUCCIÓN

POSIX (acrónimo de Portable Operating System Interface, y X viene de UNIX como seña de identidad de la API) es una norma escrita por la IEEE, que define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (o "shell").

El término fue sugerido por Richard Stallman en la década de 1980, en respuesta a la demanda del IEEE, que buscaba un nombre fácil de recordar. La traducción del acrónimo es "Interfaz de Sistema Operativo Portable".

Cada uno de los estándares que lo componen cubre diferentes aspectos de los sistemas operativos. Algunos de ellos ya han sido aprobados, mientras que otros están aún en fase de desarrollo. Los estándares POSIX se pueden agrupar en tres categorías diferentes:



Dennis MacAlistair Ritchie. Colaboró en el diseño y desarrollo de los sistemas operativos Multics y Unix, así como el desarrollo de varios lenguajes de programación como el C, tema sobre el cual escribió un célebre clásico de las ciencias de la computación junto a Brian Wilson Kernighan: El lenguaje de programación C.

Tabla I. Lista de estándares base del POSIX

POSIX.1	Interfases del sistema (estándar básico)[a,b]
POSIX.2	*Shell* y utilidades[a]
POSIX.3	Métodos para medir la conformidad con POSIX[a]
POSIX.4	Extensiones de tiempo real
POSIX.4a	Extensión de *threads*, o múltiples flujos de control
POSIX.4b	Extensiones adicionales de tiempo real
POSIX.6	Extensiones de seguridad
POSIX.7	Administración del sistema
POSIX.8	Acceso a ficheros transparente a la red
POSIX.12	Interfases de red independientes del protocolo
POSIX.15	Extensiones de colas *batch*
POSIX.17	Servicios de directorios

[a] Estándares IEEE ya aprobados
[b] Estándar ISO/IEC ya aprobado

Figura 3: GNU + Linux = GNU/Linux

DESAROLLO

Una serie de pruebas acompañan al estándar POSIX. Son llamadas "PCTS" en alusión al acrónimo "Posix Conformance Test Suite". Desde que la IEEE empezó a cobrar altos precios por la documentación de POSIX y se ha negado a publicar los estándares, se ha aumentado el uso del modelo Single Unix Specification. Este modelo es abierto, acepta entradas de todo el mundo y está libremente disponible en Internet. Fue creado por The Open Group.

- POSIX.1, Core Services (implementa las llamadas del ANSI C estándar). Incluye:
  - Creación y control de procesos.
  - Señales.
  - Excepciones de punto flotante.
  - Excepciones por violación de segmento.
  - Excepciones por instrucción ilegal.
  - Errores del bus.
  - Temporizadores.
  - Operaciones de ficheros y directorios (sobre cualquier fs montado).
  - Tuberías (Pipes).
  - Biblioteca estándar de C.
  - Instrucciones de entrada/salida y de control de dispositivo (ioctl).

- POSIX.1b, extensiones para tiempo real:
  - Planificación (scheduling) con prioridad.
  - Señales de tiempo real.
  - Temporizadores.
  - Semáforos.
  - Intercambio de mensajes (message passing).
  - Memoria compartida.
  - Entrada/salida síncrona y asíncrona.
  - Bloqueos de memoria.

- POSIX.1c, extensiones para hilos (threads):
  - Creación, control y limpieza de hilos.
  - Planificación (scheduling).
  - Sincronización.
  - Manejo de señales.

- POSIX.2, Shell y Utilidades (IEEE Std 1003.2-1992)
  - Intérprete de Comandos
  - Programas de Utilidad

Luego de 1997 el Grupo Austin realizó modificaciones a POSIX. Las especificaciones tienen el nombre de Single Unix Specification (Especificación Única de Unix)

- POSIX:2001 o IEEE Std 1003.1-2001 equivale a la versión 3 de Single UNIX Specification.
  - Las base de definiciones, Tema 6.
  - Las interfaces y encabezamientos del sistema, Tema 6.
  - Los comandos y utilidades, Tema 6.

- POSIX:2004 o IEEE Std 1003.1-2004 implica una pequeña actualización de POSIX:2001. Tiene dos correcciones técnicas de errores.

- A partir de 2009 POSIX:2008 o IEEE Std 1003.1-2008 representa la versión actual.
  - La base de definiciones, Tema 7,
  - Las interfaces encabezamientos del sistema, Tema 7.
  - Los comandos y utilidades, Tema 7.

1) Estándares Base: Definen interfases del sistema relacionadas con diferentes aspectos del sistema operativo. El estándar especifica la sintaxis y la semántica de estos servicios del sistema operativo, de modo que los programas de aplicación puedan invocarlos directamente. El estándar no especifica cómo se implementan estos servicios; de este modo, los implementadores de sistemas pueden elegir la implementación que crean más conveniente—y así competir entre ellos—, siempre que cumplan la especificación de la interfase. Todos los estándares bases desarrolladas hasta el momento lo han sido para lenguaje C. En el momento de escribir este artículo está abierto el debate sobre si los estándares base deben desarrollarse de forma independiente del lenguaje, y luego especificar interfases concretas para los diferentes lenguajes de programación. La Tabla I y la Tabla II muestran los estándares base que están siendo desarrollados por los grupos de trabajo del POSIX.

Tabla II. Estándares base POSIX adicionales

P1224	Servicios de mensajería electrónica (X.400)
P1224.1	Interfase para portabilidad de aplicaciones X.400
P1238	Interfase de comunicaciones OSI
P1238.1	Interfase OSI de transferencia de ficheros
P1201.1	Interfase gráfica a usuario (ventanas)

2) Interfases en diferentes lenguajes de programación: Son estándares secundarios que traducen a un lenguaje de programación concreto los estándares base. Los lenguajes utilizados hasta el momento son Ada, Fortran 77, y Fortran 90, además del lenguaje C, en el que se han especificado hasta el momento los estándares base. La Tabla III muestra las interfases POSIX que están actualmente en desarrollo para diferentes lenguajes de programación.

Tabla III. Lista de interfases POSIX para diferentes lenguajes de programación

POSIX.5	Interfases Ada[a]
POSIX.9	Interfases Fortran 77[a]
POSIX.19	Interfases Fortran 90
POSIX.20	Interfases Ada para las extensiones de tiempo real
[a] Estándares IEEE ya aprobados	

3) Entorno de Sistemas Abiertos: Estos estándares incluyen una guía al entorno POSIX y los perfiles de entornos de aplicación. Un perfil de aplicación es una lista de los estándares POSIX, con especificación de las opciones y parámetros necesarios, que se requieren para un cierto entorno de aplicación. El objetivo principal de los perfiles de aplicación es conseguir un conjunto pequeño de clases de implementaciones de sistemas operativos bien definidas y que sean apropiadas para entornos particulares de aplicaciones. La Tabla IV muestra la lista de estándares que están siendo desarrollados en este grupo.

Tabla IV. Lista de estándares POSIX de entornos de aplicaciones

POSIX.0	Guía al entorno POSIX de sistemas abiertos
POSIX.10	Perfil de entorno de aplicaciones de supercomputación
POSIX.11	Perfil de entorno de aplicaciones de procesado de transacciones
POSIX.13	Perfiles de entornos de aplicaciones de tiempo real
POSIX.14	Perfil de entorno de aplicaciones multiprocesadoras
POSIX.18	Perfil de entorno de aplicación de plataforma POSIX

Estandarización de Sistemas Operativos de Tiempo Real

Debido a la necesidad de conseguir la portabilidad de las aplicaciones de tiempo real, se estableció en el POSIX un grupo de trabajo de tiempo real. Este grupo desarrolla estándares para añadir al POSIX básico (o UNIX) los servicios de sistema operativo necesarios para poder desarrollar aplicaciones de tiempo real. Estas aplicaciones se caracterizan porque el funcionamiento correcto no sólo depende de los resultados del cálculo, sino también del instante en el que se generan estos resultados. Con objeto de garantizar que los cálculos se realizan en los instantes requeridos, es preciso que el sistema de tiempo real tenga un comportamiento temporal predecible, y para ello, es preciso también que los servicios del sistema operativo sean capaces de proporcionar el nivel de servicio requerido con un tiempo de respuesta acotado. El objetivo principal del grupo de trabajo de tiempo real del POSIX es "desarrollar estándares que sean los mínimos cambios y adiciones a los estándares POSIX

para soportar la portabilidad de aplicaciones con requerimientos de tiempo real". Muchas aplicaciones de tiempo real, y especialmente los sistemas empotrados, tienen restricciones físicas especiales que imponen el uso de sistemas operativos con un conjunto reducido de funciones o servicios del sistema. Por ejemplo, existen muchos sistemas que no disponen de disco duro, no tienen unidad hardware de manejo de memoria (MMU), o tienen poca memoria. Para estos sistemas es necesario que el estándar permita implementaciones que sólo soporten un subconjunto de los servicios POSIX. Los subconjuntos necesarios para las aplicaciones de tiempo real han sido abordados por el grupo de trabajo de tiempo real, que ha propuesto cuatro perfiles para entornos de aplicaciones de tiempo real: sistemas empotrados pequeños, controladores de tiempo real, sistemas empotrados grandes, y sistemas grandes con requerimientos de tiempo real. De acuerdo con estos requerimientos, el grupo de trabajo de tiempo real está actualmente desarrollando cuatro estándares:

- POSIX.4: Extensiones de tiempo real. Define interfases para soportar la portabilidad de aplicaciones con requerimientos de tiempo real. POSIX.4a: Extensión de threads. Define interfases para soportar múltiples threads o flujos de control dentro de cada proceso POSIX. POSIX.4b: Extensiones adicionales de tiempo real. Define interfases para soportar servicios de tiempo real adicionales.
- POSIX.13: Perfiles de entornos de aplicaciones de tiempo real. Cada perfil especifica una lista de los servicios que se requieren para un entorno de aplicación particular. Los estándares base POSIX.4, POSIX.4a y POSIX.4b están especificados para lenguaje C. Existe un grupo de trabajo en el POSIX dedicado a la especificación de interfases Ada, que produjo ya las interfases Ada al estándar base POSIX.1, y está actualmente desarrollando las interfases Ada para las extensiones de tiempo real, bajo el nombre POSIX.20.

EXTENSIONES DE TIEMPO REAL

Planificación de Procesos de Tiempo Real

El estándar base POSIX.1 define un modelo con actividades concurrentes denominadas procesos, pero no especifica ninguna política de planificación ni ningún concepto de prioridad. Para que las aplicaciones de tiempo real puedan ser portables, es preciso especificar políticas de planificación que permitan obtener tiempos de respuesta predecibles. El POSIX.4 define tres políticas de planificación; cada proceso, a través de un atributo de planificación, puede elegir la que desee:

• SCHED_FIFO: Es una política de planificación expulsora basada en prioridades estáticas4 , en la que los procesos con la misma prioridad se atienden en el orden de llegada (cola FIFO). Está política tendrá al menos 32 niveles de prioridad.

• SCHED_RR: Esta política es muy similar a SCHED_FIFO, pero emplea un método de rodaja temporal (round-robin) para planificar procesos de la misma prioridad. También tiene 32 niveles de prioridad como mínimo.

• SCHED_OTHER: Es una política de planificación definida por la implementación.

La planificación expulsora de prioridad estática es una estrategia de prioridad utilizada con mucha frecuencia para sistemas de tiempo real. Es muy sencilla, y permite alcanzar altos niveles de utilización del sistema si se realiza la asignación de prioridades de acuerdo con los métodos del ritmo monotónico (rate monotonic)[6] o plazo monotónico (deadline monotonic)[5]. Con las políticas de planificación especificadas en el estándar, junto a las funciones asociadas que permiten modificar y leer las políticas y prioridades de cada proceso, es posible planificar aplicaciones de tiempo real en sistemas operativos POSIX. En [9] aparece una buena introducción al diseño y análisis de este tipo de sistemas de tiempo real utilizando resultados recientes de planificación.

Inhibición de la Memoria Virtual

Aunque el estándar POSIX.1 no requiere que las implementaciones suministren mecanismos de memoria virtual, es práctica común en los sistemas UNIX el proporcionarlos. La memoria virtual presenta grandes ventajas para aplicaciones que no son de tiempo real, pero introduce una gran incertidumbre en la respuesta temporal. Con objeto de acotar los tiempos de acceso a memoria—y por tanto la respuesta temporal de la aplicación—el POSIX.4 define funciones para bloquear en memoria física o bien todo el espacio de direccionamiento de un proceso, o bien rangos seleccionados de ese espacio. Estas funciones deberán de ser utilizadas por los procesos con requerimientos temporales estrictos, así como por aquellos procesos con los que se sincronicen. De esta forma, se pueden conseguir tiempos de respuesta predecibles.

Sincronización de Procesos

El POSIX.4 define funciones para permitir la sincronización de procesos a través de semáforos contadores. Estos semáforos se identifican por un nombre que pertenece a un espacio de nombres definido por la implementación. Este espacio de nombres puede coincidir o no con el espacio de nombres de ficheros, por lo que no se hace necesaria la existencia del sistema de ficheros para utilizar los semáforos. El semáforo contador es un mecanismo de sincronización muy común, que permite el acceso mutuamente exclusivo a recursos compartidos, la señalización y espera entre procesos, y otros tipos de sincronización. Uno de los usos más comunes de los semáforos es permitir que diferentes procesos puedan compartir datos; esto se consigue en POSIX.4 utilizando objetos de memoria compartida (ver la sección 2.4), junto con los semáforos.

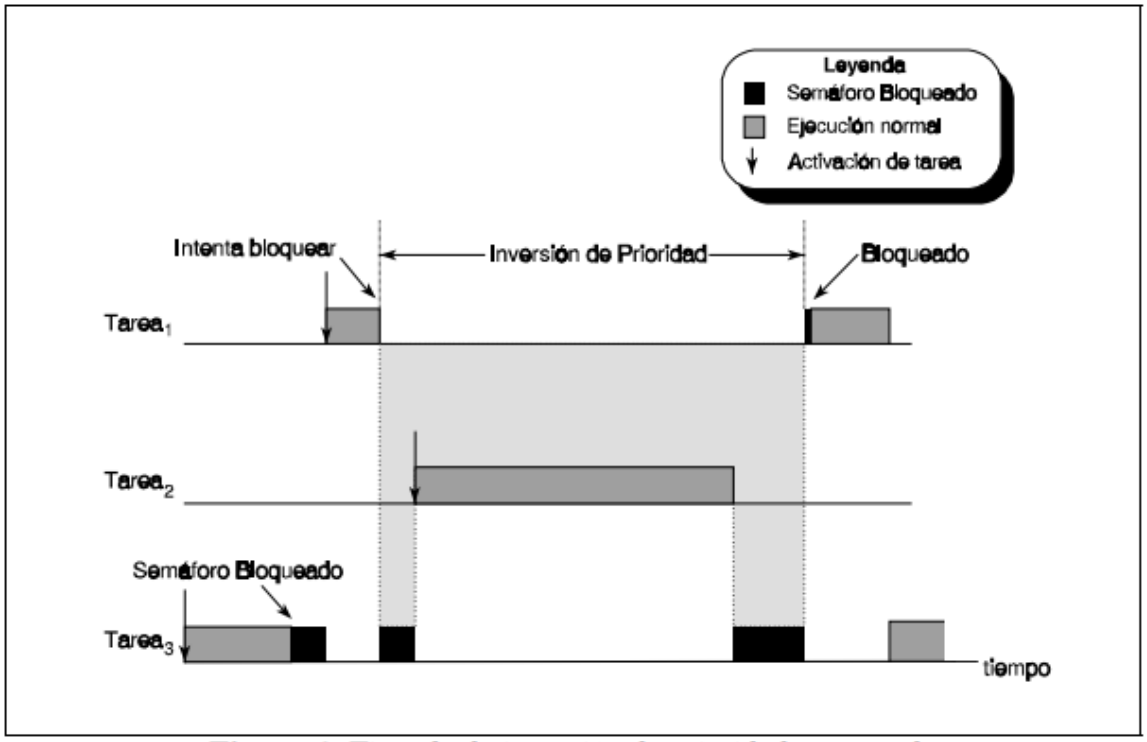


Figura 1. Ejemplo de inversión de prioridad no acotada

Memoria Compartida

Los procesos POSIX.1 tienen espacios de direccionamiento que son independientes entre sí. Sin embargo, muchas aplicaciones de tiempo real (y también muchas que no son de tiempo real) necesitan compartir grandes cantidades de datos de una manera eficiente. Esto se puede conseguir si los procesos son capaces de compartir regiones de memoria física. Con este propósito, el estándar POSIX.4 define los objetos de memoria compartida, que son regiones de memoria que pueden ser mapeadas en el espacio de direcciones de un proceso. Cuando dos o más procesos mapean el mismo objeto de memoria entonces comparten la región de memoria asociada. Si los objetos de datos que se colocan en memoria compartida requieren un acceso mutuamente exclusivo por parte de distintos procesos, se pueden utilizar semáforos para efectuar estos accesos. Al igual que en el caso de los semáforos, los objetos de memoria compartida se identifican por un nombre que pertenece a un espacio de nombres dependiente de la implementación. También es posible en POSIX.4 mapear ficheros en memoria. La información del fichero se escribe o lee en memoria principal directamente, y al cerrar el fichero el sistema actualiza la información en la memoria secundaria. Los ficheros mapeados en memoria también pueden ser compartidos por varios procesos.

Señales de Tiempo Real

El mecanismo de señales definido en el POSIX.1 permite notificar eventos que ocurren en el sistema, pero no es completamente satisfactorio para aplicaciones de tiempo real. Las señales no se almacenan en colas y, por tanto, algunos eventos se pueden perder. Las señales no están priorizadas, y esto implica tiempos de respuesta más largos para eventos urgentes. Además, los eventos del mismo tipo producen señales con el mismo número, que son indistinguibles. Puesto que muchas aplicaciones de tiempo real están basadas en el rápido intercambio de eventos en el sistema, el POSIX.4 ha extendido la interfase de señales para conseguir las siguientes características:

- Las señales de tiempo real se guardan en colas, por lo que los eventos no se pierden.
- Las señales de tiempo real pendientes de procesado se extraen de la cola en orden de prioridad, usando el número de señal como prioridad. Esto permite diseñar aplicaciones con tiempos de respuesta más rápidos ante eventos urgentes.
- Las señales de tiempo real contienen un campo adicional de información, que la aplicación puede utilizar para intercambiar datos entre el generador de la señal y el módulo que la procesa. Por ejemplo, este campo puede ser utilizado para identificar la fuente y la causa de la señal.
- El rango de señales disponibles para la aplicación ha sido expandido.

Comunicación Entre Procesos

Se especifica un mecanismo sencillo de colas de mensajes para la comunicación entre procesos. Las colas de mensajes están identificadas por un nombre perteneciente a un espacio de nombres dependiente de la implementación. Los mensajes tienen asociado un campo de prioridad, y se extraen de las colas en orden de prioridad. Esta facilidad permite minimizar la cantidad total de inversión de prioridad en el sistema. La recepción y la transmisión de mensajes puede hacerse tanto de forma bloqueante—si es necesario el proceso se suspende hasta que llegue un mensaje o haya espacio en la cola—como no bloqueante. La transmisión y la recepción no están sincronizadas, es decir, el transmisor no necesita esperar a que el receptor haya recibido el mensaje. Los tamaños máximos de mensajes y colas se pueden seleccionar durante la creación de la cola, lo que permite incrementar la predecibilidad de las operaciones con colas de mensajes.

Relojes y Temporizadores

Se define un reloj de tiempo real que debe tener una precisión de al menos 20 ms. El tiempo se representa con resolución de nanosegundos, por lo que, si una implementación dispone de un reloj hardware de alta precisión, lo puede aprovechar con plena resolución. También se pueden crear temporizadores que cuentan intervalos de tiempo utilizando como base

temporal el reloj de tiempo real u otros relojes definidos por la implementación. Cuando el intervalo especificado en un temporizador ha transcurrido, se envía una señal al proceso que lo creó. Existen diferentes opciones para los temporizadores, tales como disparo único, activación periódica, etc., que permiten manejar el tiempo de forma flexible y sencilla. Se define también una función para dormir un proceso durante un intervalo relativo especificado en nanosegundos.

EXTENSION DE THREADS

El modelo de procesos del POSIX.1 no es completamente adecuado para aquellos sistemas que requieren alta eficiencia y procesan gran cantidad de eventos en intervalos pequeños de tiempo, debido a que los procesos tienen tiempos de cambio de contexto muy elevados, el tiempo necesario para crearlos o destruirlos es muy elevado, se necesita hardware especial (MMUs) para proporcionar a cada proceso un espacio de direcciones independiente, y el modelo no es adecuado para sistemas multiprocesadores de memoria compartida. En la mayoría de los núcleos de tiempo real comercialmente disponibles para sistemas empotrados pequeños, el modelo de concurrencia está basado en tareas que comparten el mismo espacio de direccionamiento y tienen un estado asociado poco voluminoso, comparado con los procesos POSIX. El Grupo de Trabajo de Tiempo Real consideró estas características y decidió desarrollar la extensión de threads.

El estándar POSIX.4a define interfases para soportar múltiples actividades concurrentes, denominadas threads, dentro de cada proceso POSIX. Los threads definidos en el POSIX.4a tienen un estado asociado más pequeño que el de un proceso. Todos los threads que pertenecen al mismo proceso comparten el mismo espacio de direccionamiento. Pueden ser implementados con tiempos de cambio de contexto y de creación y destrucción más bajos que los de los procesos. El POSIX.4a ha sido específicamente desarrollado para abordar las necesidades de los sistemas multiprocesadores de memoria compartida. Con estas características, el modelo de threads está mucho más próximo al modelo de concurrencia de los núcleos de tiempo real comerciales que el modelo de procesos. Los threads no sólo están pensados para aplicaciones de tiempo real, sino que también pueden ser empleados para sistemas que, no siendo de tiempo real, requieren cambios de contexto eficientes y tiempos de creación/destrucción pequeños, como en aplicaciones de ventanas, software multiprocesador, etc. Los threads pueden usar todas las funciones definidas en el POSIX.4 y POSIX.1, además de las funciones definidas específicamente para threads en el POSIX.4a. Las funciones más relevantes del POSIX.4a se describen a continuación.

Control de Threads

Estas funciones permiten controlar la creación y terminación de threads, así como las funciones relacionadas con estas operaciones. Se definen funciones para crear un thread, esperar a la terminación de un thread, terminar un thread de forma normal, manejar

identificadores de threads, etc. Asimismo se definen funciones para el manejo de los atributos de creación de un thread, tales como el tamaño de stack.

Planificación de Threads

Planificación Global: Todos los threads tienen dominio de contención global—o de sistema—, y por tanto cada thread se planifica compitiendo con todos los demás threads del sistema, sin que el proceso al que pertenecen tenga ninguna importancia. El planificador, por tanto, funciona sólo al nivel de threads, y los parámetros de planificación de los procesos se ignoran.

- Planificación Local. Los threads sólo compiten con otros threads pertenecientes al mismo proceso. La planificación se realiza en dos niveles. Primero, se planifican los procesos entre sí. Después, los threads del proceso (o procesos) seleccionado compiten entre ellos por el uso de la CPU.
- Planificación Mixta. Algunos threads tienen dominio de contención global o de sistema, y otros tienen dominio de contención local o de proceso. La planificación se realiza a dos niveles. En primer lugar, se planifican los procesos y los threads de dominio global. En segundo lugar, si es necesario, se planifican los threads con dominio local pertenecientes al proceso seleccionado.

Tanto los sistemas de planificación global como los de planificación mixta ofrecen las mejores perspectivas para la mayoría de las aplicaciones de tiempo real, ya que permiten planificar todos los distintos objetos concurrentes que tengan requerimientos temporales estrictos, al mismo nivel. Los sistemas con planificación mixta pueden además realizar planificación local para algunos threads concretos. La planificación local es normalmente mucho más eficiente y rápida que la planificación global. Sin embargo, sólo debe utilizarse para grupos de threads cuya prioridad sea globalmente menor o mayor que las prioridades de otros grupos de threads en el sistema, es decir, cuando ningún otro thread en el sistema necesite tener una prioridad efectiva situada entre los niveles de prioridad de los threads del grupo. El motivo de esta restricción es que la prioridad del proceso, y no la prioridad de los threads, es la que se utiliza para planificar el grupo de threads con dominio local.

Sincronización de Threads

Se definen dos primitivas de sincronización para threads: los mutex o secciones mutuamente exclusivas, y las variables condicionales. Los mutex se utilizan en la sincronización de threads para el acceso mutuamente exclusivo a recursos compartidos. Son similares a los semáforos, pero requieren que el thread que bloquea el mutex denominado el propietario de ese mutex sea el mismo que los libera. Las variables condicionales se pueden utilizar para espera y señalización de eventos entre threads, aunque su uso está ligado al de los mutex en una forma similar a las secciones críticas condicionales . La espera a una variable condicional

se puede especificar con un tiempo máximo de espera (timeout). Ambas primitivas de sincronización pueden ser opcionalmente utilizadas por threads pertenecientes a diferentes procesos.

Los mutex se definen con tres protocolos de sincronización opcionales:

- **NO_PRIO_INHERIT:** La prioridad del thread no depende de sus relaciones de propiedad sobre ningún mutex, prácticamente, se dice que un thread es propietario del mutex que bloquea.
- **PRIO_INHERIT:** El thread propietario de un mutex hereda las prioridades de los threads que están en espera de adquirir el mutex. Este es el protocolo de herencia básica de prioridad.
- **PRIO_PROTECT:** Cuando un thread adquiere un mutex, hereda la prioridad denominada techo de prioridad del mutex, que se define generalmente como la prioridad de la tarea de prioridad más alta que puede bloquear ese mutex. La aplicación asigna el techo de prioridad a cada mutex; con los techos de prioridad adecuados, el funcionamiento es el denominado protocolo de protección de prioridad, también denominado emulación del protocolo de techo de prioridad.

La inversión de prioridad no acotada se puede evitar utilizando los protocolos de herencia básica de prioridad o de protección de prioridad, y así se pueden conseguir altos niveles de utilización en sistemas con requerimientos de tiempo real estricto. El protocolo de protección de prioridad, con las definiciones apropiadas de techo de prioridad, se puede utilizar también para evitar un tipo de inversión de prioridad especial que aparece en los sistemas multiprocesadores, denominada bloqueo remoto. En aparece una descripción detallada del bloqueo remoto y la sincronización en multiprocesadores.

En el POSIX.4a se definen otras funciones para el manejo de datos asociados a cada thread, la cancelación de threads, envío de señales a threads, así como versiones reentrantes de otras funciones definidas en el POSIX.1.

EXTENSIONES ADICIONALES DE TIEMPO REAL

El estándar POSIX.4b define extensiones adicionales de tiempo real para soportar la portabilidad de aplicaciones con requerimientos de tiempo real. La razón por la que se dividen las extensiones de tiempo real en dos estándares ha sido el facilitar una aprobación más rápida de los servicios que se consideraron esenciales para tiempo real aquellos especificados en el POSIX.4, dejando otros servicios de tiempo real también convenientes pero menos necesarios para un segundo estándar. Puesto que el POSIX.4b ha comenzado su proceso de estandarización más tarde que el POSIX.4, los servicios que se incluyen en los borradores actuales tienen más posibilidades de cambiar que los del POSIX.4.

A continuación se describen brevemente los servicios que están siendo estandarizados en el POSIX.4b:

Tiempos Límite (Timeouts)

Algunos de los servicios definidos en el POSIX.1, POSIX.4, y POSIX.4a pueden suspender al proceso que los invoca durante un período indefinido de tiempo, hasta que los recursos necesarios para completar el servicio estén disponibles. En sistemas de tiempo real estricto es importante limitar la cantidad máxima de tiempo que un proceso puede emplear esperando a que uno de estos servicios se complete. Esto permite detectar condiciones anormales, y por tanto incrementa la robustez del programa permitiendo implementaciones tolerantes a fallos. Los tiempos límite especifican la máxima cantidad de tiempo que el proceso puede estar suspendido en espera de la terminación de un servicio. Los servicios elegidos para tener tiempos límite han sido aquellos que todavía no tenían capacidad de especificar un tiempo límite, y cuyo uso se consideró más probable en los segmentos de código en los que la respuesta temporal es crítica:

- Esperar a que un semáforo se desbloquee
- Esperar a la llegada de un mensaje a una cola de mensajes
- Enviar un mensaje a una cola de mensajes
- Esperar a que un mutex sea liberado.

Relojes de Tiempo de Ejecución

Se define un reloj opcional de tiempo de CPU para cada proceso y para cada thread, y se utiliza la interfase POSIX.4 de relojes y temporizadores para manejar estos relojes. Además de la medida del tiempo de CPU, que es especialmente útil en sistemas de tiempo real para caracterizar y analizar el sistema, se pueden crear temporizadores basados en los relojes de tiempo de CPU, con objeto de detectar el consumo de una cantidad excesiva de tiempo de ejecución por parte de un proceso o thread. De esta forma, se puede detectar durante la ejecución si ha habido errores de software, o errores en la estimación de los tiempos de ejecución de peor caso. La detección de la situación en la que una tarea excede el tiempo de ejecución de peor caso asumido durante la fase de análisis es muy importante en sistemas de tiempo real robustos, porque si los tiempos asumidos no se cumplen, los resultados del análisis de planificabilidad ya no son válidos, y el sistema puede incumplir sus requerimientos temporales. Los relojes de tiempo de ejecución permiten detectar cuando ocurre un consumo excesivo de tiempo de CPU, para activar las acciones apropiadas de manejo de esta condición de error.

Servidor Esporádico

Se define una nueva política de planificación—llamada SCHED_SPORADIC—que implementa el algoritmo de planificación del servidor esporádico [10]. Esta política puede ser utilizada para procesar eventos aperiódicos al nivel de prioridad deseado, permitiendo garantizar los requerimientos temporales de tareas de prioridad inferior. El servidor esporádico proporciona tiempos de respuesta rápidos y hace predecibles los sistemas que procesan eventos aperiódicos.

Control de Interrupciones

Muchos sistemas de tiempo real requieren poder capturar interrupciones generadas por dispositivos hardware especiales, y gestionar estas interrupciones desde el programa de aplicación. Las funciones propuestas en el estándar permiten a un proceso o thread capturar una interrupción a través de la asignación de una rutina de servicio de interrupción escrita por el usuario, suspender la ejecución del proceso o thread hasta que llegue una interrupción, y proteger secciones de código de ser interrumpidas. Las interfases definidas no conseguirán una portabilidad completa de los programas de aplicación debido a las muchas diferencias existentes en los mecanismos de manejo de interrupciones de las diferentes arquitecturas. Sin embargo, la portabilidad de la aplicación se incrementará con el uso de esta interfase, ya que se establece un modelo de referencia, y además el código no portable se confina a módulos específicos claramente delimitados.

Control de Dispositivos de Entrada/Salida

En sistemas de tiempo real es frecuente interaccionar con el entorno a través de dispositivos especiales como entradas/salidas analógicas o digitales, contadores, etc. Típicamente, es el usuario responsable de la aplicación el que escribe los drivers de entrada/salida o porciones de código que acceden directamente al dispositivo hardware para estos dispositivos especiales. Una forma estandarizada de acceder a los drivers de entrada/salida para realizar operaciones de control sobre el dispositivo asociado permitiría que estas operaciones estuviesen claramente definidas. El POSIX.4b define una función que permite a un programa de aplicación transferir información de control hacia y desde el driver del dispositivo. Del mismo modo que para las funciones de control de interrupciones, los programas que utilicen la función de control de dispositivos pueden no ser completamente portables, pero su portabilidad se mejora por el uso de esta interfase que proporciona un modelo de referencia para acceder a los drivers de dispositivos.

Creación de Procesos

Otra interesante función que se define en POSIX.4b es la creación eficiente de procesos, sin necesidad de utilizar la secuencia de funciones *fork()* y *exec()* típica del UNIX, en la que primero se hace una copia del proceso original, para después destruir la copia y sustituirla por una nueva imagen de proceso. Aunque la secuencia mencionada presenta algunas ventajas en especial en lo relativo a las operaciones de herencia de descriptores de fichero entre el proceso padre y el hijo en un porcentaje muy alto de veces sería suficiente una primitiva mucho más sencilla y eficiente que simplemente crease un nuevo proceso utilizando la imagen de proceso almacenada en un determinado fichero. Esto es lo que hace la nueva función definida en el POSIX.4b, denominada *spawn(),* y que permitirá reducir el tiempo de creación de un alto porcentaje de los procesos.

PERFILES DE ENTORNOS DE APLICACIONES DE TIEMPO REAL

El estándar POSIX.1, junto con las extensiones de tiempo real y la extensión de threads, constituyen un poderoso conjunto de interfases que permiten implementar sistemas operativos capaces de dar respuesta a las necesidades de sistemas grandes con requerimientos de tiempo real. Sin embargo, para sistemas de tiempos reales empotrados y pequeños, sería deseable un subconjunto de estas interfases. Por ejemplo, muchos sistemas empotrados tienen un hardware necesariamente limitado, que hace muy difícil implementar servicios tales como el sistema de ficheros o espacios de direccionamiento independientes para los procesos. Los perfiles de entornos de aplicaciones de tiempo real (AEP según sus siglas en inglés) definidos en el estándar POSIX.13 proporcionan los subconjuntos de los servicios definidos en los estándares base que se consideran adecuados para un entorno de aplicación particular. En el estándar POSIX.13 se han definido cuatro AEPs de tiempo real:

1) **Sistema Mínimo:** Corresponde a un sistema empotrado pequeño sin necesidad de unidad de manejo de memoria (MMU), sin sistema de ficheros (sin disco), y si terminal de entrada/salida. Sólo se permite un proceso, aunque puede haber múltiples threads ejecutándose de forma concurrente.
2) **Controlador de Tiempo Real:** Corresponde a un sistema controlador de propósito especial. Es como el perfil mínimo, pero añadiendo un sistema de ficheros y un terminal de entrada/salida. Sólo se permite un proceso, aunque se permiten múltiples threads.
3) **Sistema Dedicado:** Corresponde a un sistema empotrado grande, sin sistema de ficheros. Puede tener múltiples procesos y múltiples threads.
4) **Sistema Multi-Propósito:** Corresponde a un sistema grande de tiempo real con todos los servicios soportados.

La Tabla V resume las principales características de cada uno de los perfiles de tiempo real.

Tabla V. Características de los Perfiles de Tiempo Real

Perfil	**Sistema de Ficheros**	**Múltiples Procesos**	**Múltiples *Threads***
Sistema Mínimo de Tiempo Real	NO	NO	SI
Controlador de Tiempo Real	SI	NO	SI
Sistema Dedicado de Tiempo Real	NO	SI	SI
Sistema de Tiempo Real Multi-Propósito	SI	SI	Opcional

CONCLUSIONES

Al trabajar con Linux, a menudo encontraremos referencias a POSIX. Lamentablemente, por lo general no nos detenemos a explicar su significado. Por eso, en este post explicaremos qué es POSIX y por qué es importante. De esta manera, tanto los nuevos usuarios como aquellos más experimentados tendrán una referencia a mano.

En primer lugar, POSIX significa Portable Operating System Interface. Consiste en una familia de estándares especificadas por la IEEE con el objetivo de facilitar la interoperabilidad de sistemas operativos. Además, POSIX establece las reglas para la portabilidad de programas. Por ejemplo, cuando se desarrolla software que cumple con los estándares POSIX existe una gran probabilidad de que se podrá utilizar en sistemas operativos del tipo Unix. Si se ignoran tales reglas, es muy posible que el programa o librería funcione bien en un sistema dado pero que no lo haga en otro.

Sin ir más lejos, examinemos en la última versión de la especificación IEEE 1003 (designación formal de los estándares POSIX). Al buscar la referencia sobre el comando du, podemos ver que solamente las opciones -a, -H, -k, -L, -s, y -x son obligatorias como vemos en la Fig. 1. Otras, tales como -c y -h, aparecen en la versión de du provista por el proyecto GNU. Esto significa que si utilizamos estas últimas en un script desarrollado en Linux e intentamos hacerlo funcionar en FreeBSD o AIX, es muy probable que no funcione como se esperaba.

REFERENCIAS

1. «What is POSIX?» (html). Universidad de Indiana (en inglés). Archivado desde el original el 14 de junio de 2018. Consultado el 19 de julio de 2018. «Short for "Portable Operating System Interface for uni-X", POSIX is a set of standards codified by the IEEE and issued by ANSI and ISO. The goal of POSIX is to ease the task of cross-platform software development by establishing a set of guidelines for operating system vendors to follow.»
2. Stallman, Richard (11 de mayo de 2011). «The origin of the name POSIX.». Richard Stallman (en inglés). Archivado desde el original el 11 de mayo de 2011. Consultado el 19 de enero de 2018. «It seemed to me that nobody would ever say "IEEEIX", since the pronunciation would sound like a shriek of terror; rather, everyone would call it "Unix". That would have boosted AT&T, the GNU Project's rival, an outcome I did not want. So I looked for another name, but nothing natural suggested itself to me.»
3. ↑«Welcome to the POSIX Certification web site» (html). IEEE (en inglés). Archivado desde el original el 14 de junio de 2004. Consultado el 27 de noviembre de 2018. «POSIX® Certified by IEEE and The Open Group is for products meeting the IEEE POSIX standards.»
4. «IEEE STANDARD ISO/IEC 14515-1:2000 IEEE Std 2003.1-2000 - ISO/IEC/IEEE International Standard for Information Technology -- Portable Operating System Interface (POSIX(R)) -- Test methods for measuring conformance to POSIX -- Part 1: System interfaces». IEEE (en inglés). Archivado desde el original el 19 de enero de 2018. Consultado el 19 de enero de 2018. «Purchase a copy of this standard».
5. «POSIX.1-2008 is simultaneously IEEE Std 1003.1™-2008 and The Open Group Technical Standard Base Specifications, Issue 7.». The Open Group (en inglés). Archivado desde el original el 28 de diciembre de 2017. Consultado el 19 de enero de 2018. «POSIX.1-2008 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level.»
6. «POSIX.1-2008 is simultaneously IEEE Std 1003.1™-2008 and The Open Group Technical Standard Base Specifications, Issue 7.». The Open Group (en inglés). Archivado desde el original el 27 de diciembre de 2017. Consultado el 19 de enero de 2018. «This standard has been jointly developed by the IEEE and The Open Group. It is both an IEEE Standard and an Open Group Technical Standard.»
7. «Writing POSIX-Standard Code». Microsoft TechNet (en inglés). Archivado desde el originalel 3 de junio de 2016. Consultado el 19 de enero de 2018. «This session provides a brief introduction to POSIX and some of the issues for writing code conforming to POSIX.1 environment, with emphasis on INTERIX as the development environment.»
8. https://www.uco.es/~i02samoj/docencia/pas/practica-POSIX.pdf.
9. https://www.uco.es/~i02samoj/docencia/pas/practica-POSIX.pdf.