

Threads et al.

Jornadas técnicas del GUL

¿Qué es un thread?

- Definición 1:
 - Hilo de control
- Definición 2:
 - Proceso que comparte memoria
- Definición 3
 - Cooperativos versus expulsivos

¿Qué es un thread?

- Un hilo de control
- Tiene un estado, en general registros
 - mínimo: SP y PC

¿Qué es un thread?

- **Proceso:**
 - Tipo de datos dentro del kernel
 - Abstracción que da el sistema operativo
 - Programa en ejecución (Tanenbaum)
 - Planificado de forma expulsiva

¿Qué es un thread?

- **Thread:**
 - Puede conocerlo el kernel o no
 - Colaborativo o expulsivo
 - Comparten memoria
 - Puede haber uno o más por proceso N:M
 - one to one
 - many to many
 - etc.
- Hay threads que se implementan como procesos que comparten memoria.

Green Threads

V.S.

Red Threads

- La razón original de los threads (“los procesos son pesados”) ya no es válida en muchos casos
- Los threads se implementan como procesos
- Green Threads: N:1 espacio de usuario
- Red Threads: 1:1 kernel

¿Qué es un thread?

espacio de usuario

- Un proceso en el que vivir
- Un hilo de ejecución: Contador de programa, función main
- Un contexto de ejecución: Pila, contador de programa, registros, pila, pila de señales
- crear/salvar/recomponer estado
- Problema señales y otros

¿Qué es un thread? en el kernel

- Como fork() pero para crear compartiendo: clone() o rfork()
- Se puede compartir memoria
CLONE_VM RFMEM
- Descriptores de ficheros
- Grupo de threads
- Espacios de nombres
- etc.

Threads expulsivos, threads colaborativos

- Depende del lenguaje/uso
- **Expulsivos**, cada cierto X se les echa (da igual lo que estén haciendo), señales, el kernel
- **Colaborativos** llaman a funciones (explícitas o no) que le echan, `thread_yield()`, `sched_yield()`, `send()`, `down()`...

Condiciones de carrera

- Los threads comparten memoria
- Pueden acceder a la vez
- Leo/modifico/escribo entrelazado
- Necesito mecanismos de sincronización y comunicación de threads, semáforos, variables-condición, monitores, señales, locks, qlocks, rwlocks, rendezvous, canales...

Muchos mecanismos

- En linux futex()
- En Plan 9 rendez()
- Similares, implementan variables condición.
- Implementan semáforos, canales...
- 3 cosas sincronización, comunicación, exclusión mutua

Canales

- Si sabes usar ls|wc sabes usarlos
- Síncronos, pasan mensajes, normalmente un tipo de datos, en C un puntero
- sencillos
- send() recv() alt()
- Con y sin buffering
- ¿Me bloquearé?

Semáforos

- Caja de fichas
- Coger ficha (down, wait)
- Soltar ficha (up, signal)

Pthreads en Linux

- Tiene semáforos
- Implementado con `futex()` + `clone()`
- No lo recomiendo
- Es una implementación 1:1 con procesos ligeros `clone()`
- La última versión se llama NPTL (native POSIX Threads Library), la versión original era LinuxThreads
- ¿He dicho ya que no lo recomiendo?

Thread library, p9ports

- Dos versiones, una con tas + setcontext/getcontext/makecontext
- setjmp -> sigsetjmp -> setcontext
- La otra tira directamente de pthreads
- Usa procs, threads + canales
- Es trivial de usar, mirar la implementación es muy instructivo
- Tas -> ensamblador xchg
- Contexto, SP, PC, sigstack