udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Hilos

Adín Ramírez
adin.ramirez@mail.udp.cl

Sistemas Operativos (CIT2003-1)
1er. Semestre 2015

Adaptado de Dave Eckhardt

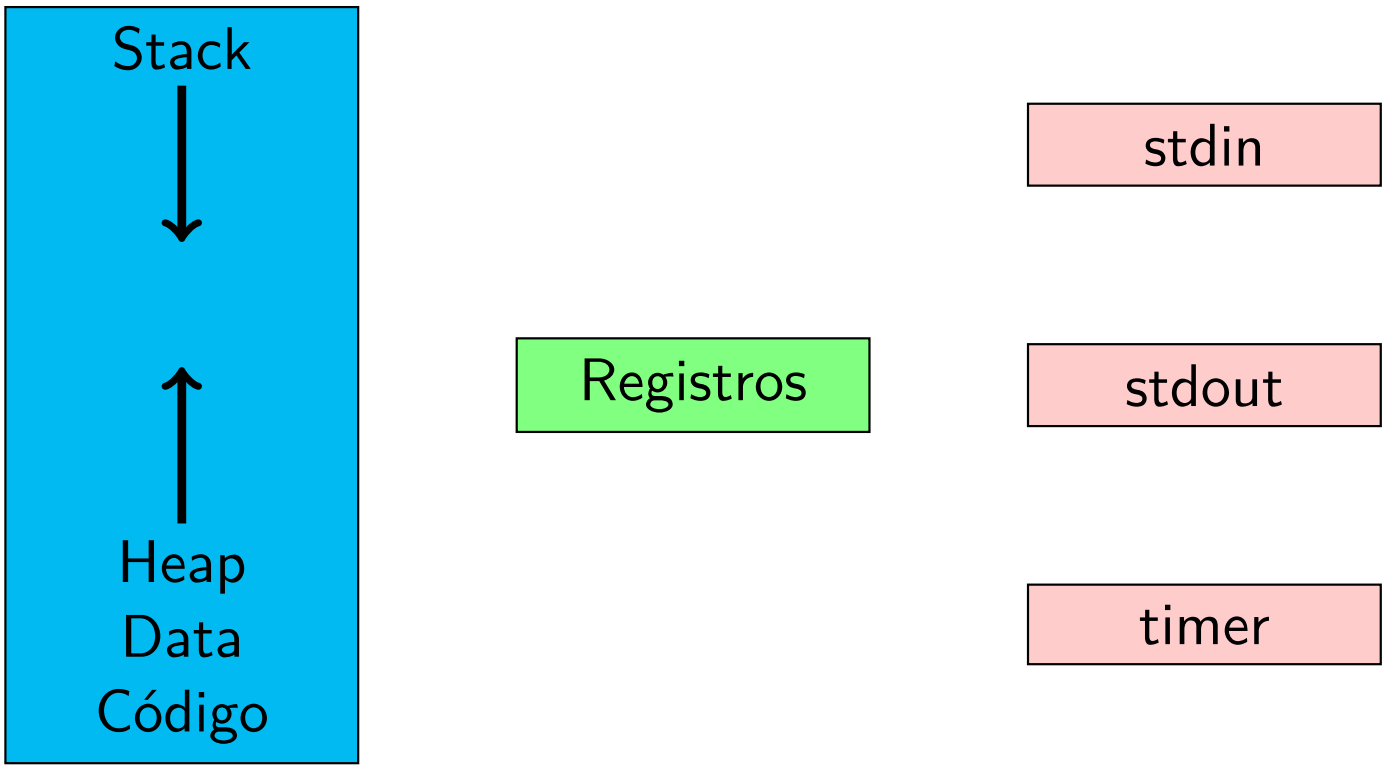
Recapitulando

- Hemos cubierto los capítulos 1–3 de OS:P+P
- No es exactamente lo mismo, revisen sus apuntes en clase
- Este tema es aproximadamente capítulo 4 de OS:P+P

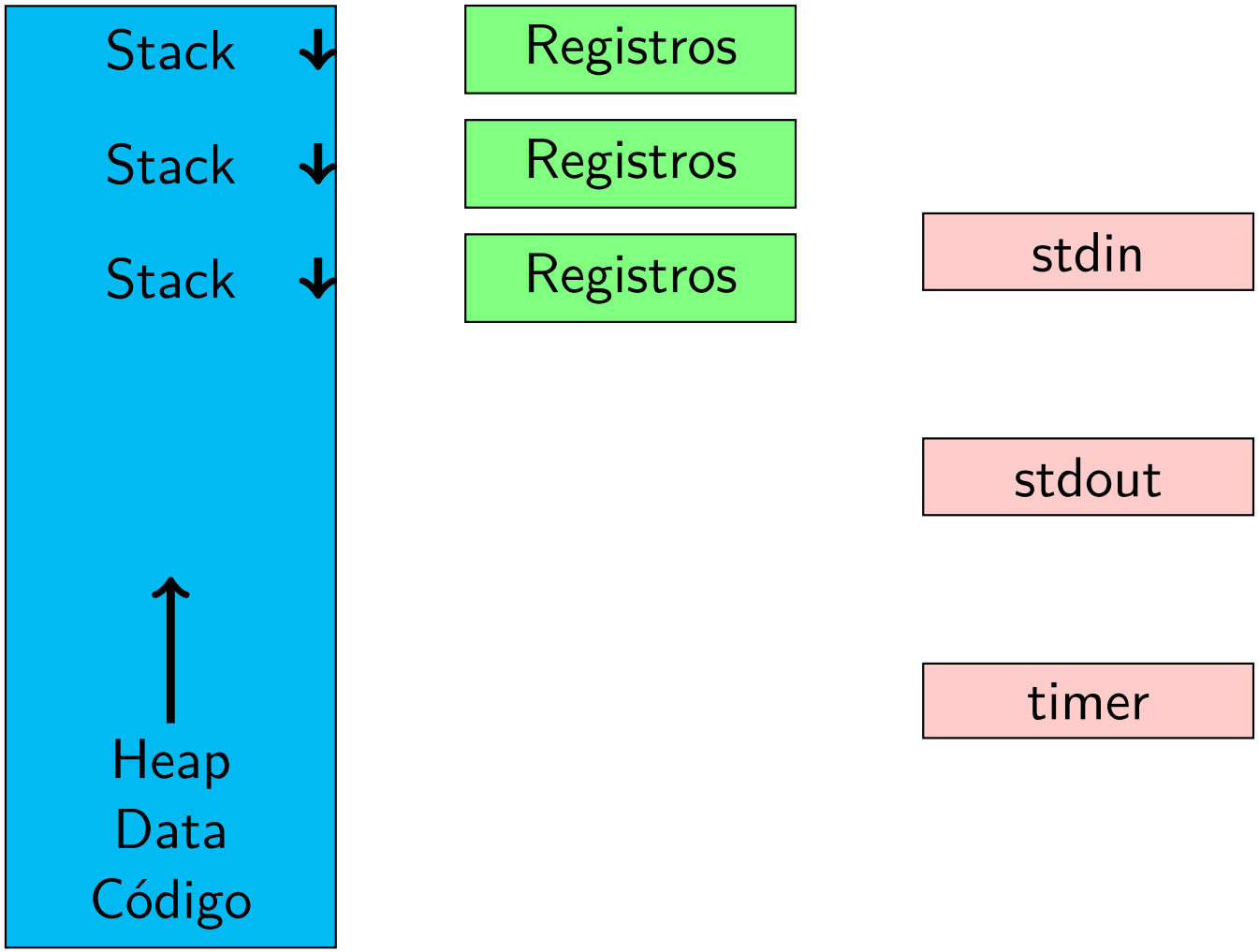
Objetivos de la clase

- Hilos o hebras (registros calendarizables)
- ¿Por qué hilos?
- Distintas versiones
- Cancelación
- Condiciones de carrera

Proceso de un hilo



Proceso de múltples hilos



¿En español?

- Tres stacks
  - Tres conjuntos de variables locales
- Tres conjuntos de registros
  - Tres punteros del stack
  - tres `%eax`, etc.
- Tres calendarizadores
- Tres potenciales malas interacciones
  - A/B, A/C, B/C
  - El patrón se complica al existir más hilos

¿Por qué hilos?

- Comparten el acceso a las estructuras de datos
- Tiempo de respuesta (sensibilidad)
- Más velocidad en procesadores múltiples

Acceso compartido a estructuras de datos

- Servidor de base de datos para varias sucursales de bancos
  - Verificar que múltiples reglas son seguidas
    - Balance de cuentas
    - Límite diario de retiros
    - etc.
  - Operaciones de múltiples cuentas
  - Muchos accesos, cada uno modifica una fracción de la base de datos
- Servidor para un juego con múltiples jugadores
  - Muchos jugadores
  - Acceso (y actualización) del estado del mundo
    - Escanear múltiples objetos
    - Actualizar uno o más objetos

Acceso compartido a estructuras de datos

- Hilo por jugador
  - Los objetos del juego están dentro de un mismo espacio de memoria
  - Cada hilo puede accesar y actualizar los objetos del juego
  - Acceso compartido a objetos del sistema operativo (archivos)
- El cambiar entre hilos es barato
  - Almacenar n registros
  - Cargar n registros

Tiempo de respuesta

- Cancelar una acción vs. procesamiento masivo
- Botón cancelar vs. descomprimiendo un archivo gigante
  - Manejar el botón del mouse durante una operación de 10 segundos
    - Mapear (x, y) para el área de cancelar
    - Cambiar el color, animar el botón, hacer un sonido
    - Verificar que el soltar del botón suceda en el área correcta de la pantalla
  - . . . sin que el descompresor entienda un click del mouse
  - Y detener el descompresor es una tarea separada
    - Los hilos permiten que el usuario pueda registrar distintas intenciones mientras siguen ejecutándose

Múltiples procesadores

- Más CPUs no pueden ayudar en un proceso de un solo hilo
- Difuminado de color en photoshop
  - Dividir la imagen en regiones
  - Un hilo de difuminado en cada región por CPU
  - Puede (a veces) dar un aumento de velocidad lineal

Tipos de hilos

- Espacio de usuario $(N : 1)$
- Hilos del kernel $(1 : 1)$
- Muchos a muchos $(M : N)$

Hilos del espacio de usuario $(N : 1)$

- Hilado interno
  - La librería de hilos agrega los hilos a un proceso
  - El cambio de hilos *solo cambia los registros*
    - Un código simple en `asm`
    - Puede solamente llamar a `yield` (entrega el CPU, y se mueve al final de la lista)

Hilos del espacio de usuario $(N : 1)$

+ No necesita cambio en el sistema operativo
- Cualquier llamada al sistema bloquea todos los hilos
  - El proceso hace una llamada al sistema
  - El kernel bloquea el proceso
  - Llamadas especiales que no bloqueen pueden ayudar
- Calendarizado cooperativo: insuficiente, extraño
  - Hay que insertar llamadas manuales a `yield`
- No puede ir más rápido en máquinas con múltiples procesadores

Hilos del kernel (1 : 1)

- Hilado soportado por el sistema operativo
  - El sistema operativo conoce la correspondencia hilo-proceso
  - Las regiones de memoria son compartidas, y las referencias se cuentan
- Cada hilo es sagrado
  - El conjunto de registros es manejado por el kernel
  - Existe un stack del kernel cuando el hilo está ejecutando código de kernel
  - Calendarización real (disparada por timer)
- Características
  - \+ El programa se ejecuta más rápido en multiprocesador
  - \+ Hilos que acaparan el CPU no obtienen todo el tiempo del CPU
  - \- Las librerías del espacio de usuario deben de re-escribirse para que sean seguras en estos hilos
  - \- Requiere más memoria del kernel
    - 1 PCB $\Rightarrow$ 1 TCB + N tCB
    - 1 k-stack $\Rightarrow$ N k-stacks

Muchos a muchos ($M : N$)

- Es un terreno medio
  - El sistema operativo provee de hilos de kernel
  - M hilos de usuario comparten N hilos de kernel
- Patrones para compartir
  - Dedicado
    - El hilo de usuario x es dueño del hilo del kernel y
  - Compartido
    - Un hilo de kernel por cada CPU (hardware)
    - Cada hilo de kernel ejecuta el siguiente hilo de usuario que sea ejecutable
  - Muchas variaciones
- Características
  - Excelente cuando el calendarizador funciona como se espera

Cancelación de hilos

- Cancelación de hilos
  - ¿Cuando?
  - No queremos el resultado de este cálculo/proceso/
  - Cancelamos la computación
- Dos tipos: asíncrono y diferido

Cancelación asíncrona

- Inmediata
- Detener la ejecución ahora
  - Ejecutar 0 instrucciones más (al menos en el espacio de usuario)
  - Liberar el stack, los registros
  - No más hilo
- Difícil de recuperar recursos (garbage-collector) —entiendase por recursos: archivos abiertos, dispositivos, etc.
- Difícil de mantener la consistencia de las estructuras de datos

Diferido

- Por favor detengase . . .
- Casi que escribimos: “Querido hilo #42, por favor deten tu ejecución. Te hechamos de menos, el siguiente ciclo terminaremos de techar. Besos y abrazos, el usuario.”
- Los hilos deben verificarse para ser cancelados
- O debemos definir puntos de cancelación seguros
  - Después de llamar `close()` está bien que me detengan

Condiciones de carrera

- ¿Qué pensamos del código?

```
ticket = next_ticket++; // 0 => 1
```

- Lo que pasa en realidad (generación de código[1])

```
ticket = temp = next_ticket; // 0
++temp; // 1 pero invisible
next_ticket = temp;
```

- ¿Recuerdan las condiciones de carrera de los procesos?
  - Revisen las diapositivas y apuntes de la clase del kernel

[1] ¿Ven la importancia de aprender compiladores?

Ley de Murphy (para hilos)

- El mundo arbitrariamente puede intercalar otra ejecución
  - Múlti procesador
    - N hilos ejecutando instrucciones **simultáneamente**
    - Pero por supuesto, los resultados se pueden intercalar
  - Procesador único
    - Un solo hilo ejecutándose a la vez
    - Pero N hilos son ejecutables, y el contador (timer) cuenta hacia cero
- El mundo decidirá la forma más dolorosa de intercalar
  - Uno en un millón, y pasa cada minuto ☹

Lo que uno espera

H_0		H_1	
`tkt = tmp = n_tkt;`	0		
`++tmp;`	1		
`n_tkt = tmp;`	1		
		`tkt = tmp = n_tkt;`	1
		`++tmp;`	2
		`n_tkt = tmp;`	2

- H_0 tiene un ticket en 0
- H_1 tiene un ticket en 1
- El resultado en `n_tkt` es 2, y nuestro jefe está feliz ☺

Pero! lo que sucede en realidad...

H_0		H_1	
`tkt = tmp = n_tkt;`	0		
		`tkt = tmp = n_tkt;`	0
`++tmp;`	1		
		`++tmp;`	1
`n_tkt = tmp;`	1		
		`n_tkt = tmp;`	1

- H_0 tiene un ticket en 0
- H_1 tiene un ticket en 0, también ☹
- El resultado en `n_tkt` es 1, y nuestro jefe está "super feliz" ☹

¿Qué paso?

- Cada hilo hizo algo razonable
  - . . . asumiendo que ningún otro hilo estaba modificando esos objetos
  - . . . es decir, asumiendo exclusión mutua
- El mundo es cruel (si algo puede salir mal, saldrá mal)
  - Cualquier confusión con el calendarizador sucederá tarde o temprano
  - La que uno espera que no suceda, sucederá
  - La que uno no esperaba que sucediera, . . .

El hack de la shell-script #!

- ¿Qué es un shell script?
- Es un archivo con varias instrucciones (dependientes de la shell)

```
#!/bin/sh
echo "Que bueno es saber sistemas operativos!"
sleep 10
exit 0
```

- O una condición de carrera esperando suceder . . .

El hack de la shell-script #!

- ¿Qué es #!?
  - Un hack
  - Llamado shebang, sha-bang, hashbang, pound-bang, hash-pling[2]
- Cuando decimos
  `execl("/foo/script", "script", "arg1", 0);`
- El archivo ejecutable `/foo/script` empieza
- Cuando encontrmaos `#!/bin/sh`, el kernel reescribe la llamada
  `execl("/bin/sh" "/foo/script", "script", "arg1", 0);`
- La shell hace `open("/foo/script", O_RDONLY, 0);`

[2] `http://en.wikipedia.org/wiki/Shebang_%28Unix%29`

La invención del `setuid`

- Set user identity
- Patente U.S. #4 135 240
  - Dennis M. Ritchie
  - 16 de enero de 1979
- Concepto
  - Un programa es almacenado con ciertos privilegios de almacenamiento
  - Cuando se ejecuta, corre con dos identidades
    - La identidad del que invoca
    - La identidad del programa mismo
  - Puede cambiar identidades a voluntad
    - Abrir archivos como el que invocó
    - Abrir otros archivos como el dueño del programa

Ejemplo de `setuid`: imprimir un archivo

- Objetivo
  - Cada usuario puedo encolar archivos
  - Los usuarios no pueden borrar los archivos de otros usuarios
- Solución
  - El directorio de las colas es del usuario `printer`
  - `setuid` como programa `encolar-archivo`
    - Crear un archivo de colas como el usuario `printer`
    - Copiar los datos del usuario (`juan`) como el usuario `juan`
  - `setuid` como programa `eliminar-archivo`
    - Permite eliminar archivos que uno encoló
  - El usuario `printer` medía el acceso a la cola del usuario `juan`

Condiciones de carrera

Proceso 0	Proceso 1
`ln -s /bin/lpr /tmp/lpr`	
	`run /tmp/lpr`
	`setuid a usuario printer`
	`start "/bin/sh /tmp/lpr ..."`
`rm /tmp/lpr`	
`ln -s /my/exploit /tmp/lpr`	
	`script = open("/tmp/lpr");`
	`execute /my/exploit`

¿Qué sucedió?

- La intención: asignar privilegios al contenido del programa
- ¿Qué paso en realidad?
  - Primero, el nombre fue mapeado a los privilegios
    - nombre ⇒ archivo, archivo ⇒ privilegios
  - Después el nombre del programa fue ligado a un archivo diferente
  - Entonces, el nombre fue mapeado a los contenidos del archivo
    - nombre ⇒ otro archivo, otro archivo ⇒ otro contenido
- ¿Cómo se resuelve?

Solucionando condiciones de carrera

- Analizar la secuencia de operaciones cuidadosamente
- Encontrar una subsecuencia que puede ser ininterrumpida
  - Sección crítica
- Utilizamos un mecanismo de sincronización
  - Continuará . . .

Puntos importantes

- Hilos: ¿qué? y ¿por qué?
- Tipos de hilos
- Razones en los distintos modos de hilos
- Condiciones de carrera
  - Asegurense de entender esta parte

Lecturas adicionales

- `setuid` demysitified
  - Hao Chen, David Wagner, Drew Dean
  - `http://www.cs.berkeley.edu/~daw/papers/setuid-usenix02.pdf`
- Cancel button problem
  - Attentiveness: Reactivity at scale
  - Gregory S. Hartman
  - `http://repository.cmu.edu/dissertations/15/`