Entrega1 - Sistemas Operativos II

Delfina Martín - Ignacio Litmanovich - Agustín Diaz

22 de marzo de 2021

Ejercicio 1

A continuación describimos brevemente de qué forma se vinculan los items presentados con la abstracción, el aislamiento y/o la administración de recursos de los sistemas operativos.

- Los sockets son un mecanismo que **abstrae** tanto el hardware como las capas inferiores (a la de transporte) del modelo TCP/IP en una comunicación.
- La memoria virtual permite que los dispositivos de almacenamiento secundario (magnético, óptico, de estado sólido, etc) sean una extensión de los de almacenamiento primario a través de una **abstracción**. En términos de **administración de recursos** el sistema operativo se debe encargar de que se esté priorizando el uso de la memoria principal por sobre la secundaria dado que esta última es de acceso más lento.
- Las **abstracciones** que permiten pedir memoria de forma dinámica al sistema operativo como malloc y calloc entre otras, esconden detalles como

  - qué tanta memoria hay disponible: solamente indican si hay o no hay disponible de la cantidad fija pedida,

  - en qué sección específica de memoria está si es que están solamente en una pieza de hardware (o si se está virtualizando memoria),

  - cuáles son los mecanismos a partir de los cuales se pagina y/o segmenta la memoria.
- Las señales son un mecanismo clave para poder **administrar los recursos** de una máquina de forma de que sea eficiente y a la vez no se produzcan fallas en el caso de que varios procesos compitan por o compartan el mismo recurso.
- La gestión de permisos y usuarios por un lado **aisla** los componentes de hardware de la máquina anfitriona de forma de que diferentes personas usuarias puedan hacer uso de ellos de forma transparente pero con potencial baja de rendimiento y por el otro debe encargarse de la **administración de recursos** para evitar que se produzcan fallas en el acceso compartido: cada quien puede acceder a sus directorios personales sin necesidad de saber en qué porción o porciones específicas de la memoria están almacenados.
- Los bloqueos obligan a ciertos procesos a quedar a la espera de un recurso que está siendo usado por otro u otros procesos y juega un rol fundamental en la **administración de recursos**. Un bloqueo se puede dar porque o bien los recursos están temporariamente agotados o bien porque algún otro proceso requiere acceso exclusivo del recurso en cuestión.
- Los archivos y en general el sistema de directorios **aislan** a las personas usuarias entre sí y **abstraen** el hardware subyacente: no es relevante en qué parte de la memoria física están almacenados realmente esos datos, de qué forma están fragmentados, etc.

Ejercicio 2

Dos buenas razones para estudiar sistemas operativos desde el punto de vista de una persona desarrolladora son:

- permite optimizar y diseñar algoritmos que se ajustan mejor a los objetivos y requerimientos del SO en cuestión
- permite comprender qué tan seguros son y en qué punto las vulnerabilidades que tienen representan un riesgo

Desde el punto de vista de una persona que administra un sistema, es de utilidad entender y poder comparar las virtudes y desventajas de los sistemas existentes. Además, incorporar nociones básicas de sistemas operativos la habilita a resolver diferentes tipos de problemas que pueden surgir como discos dañados, programas que no responden, etc.

Ejercicio 3

El problema del jardín ornamental se presenta cuando diferentes personas del mundo real quieren realizar alguna operación de forma concurrente sobre una misma cuenta bancaria. La solución del problema del jardín ornamental extrapolada al caso real de operaciones bancarias permite que se preserven las invariantes que debe cumplir el saldo en una cuenta luego de que diferentes personas operaron sobre la misma.

Por ejemplo, si dos personas quieren depositar dinero en una misma cuenta de forma concurrente, el cálculo del saldo debe garantizar que las operaciones de lectura y escritura de cada dispositivo que esté haciendo el depósito se hagan de forma atómica.

Ejercicio 4

Apartado 1

NachOS no es de propósito general, se ejecuta sobre un proceso en un sistema operativo anfitrión y tiene objetivos diferentes a los de los SO convencionales: está diseñado para enseñar, tiene objetivos pedagógicos.

Apartado 2

NachOS simula una arquitectura similar a MIPS

Apartado 3

Algunas de las características principales de la arquitectura de NachOS son:

- RISC (*reduced instruction set computer*)
- Contiene 40 registros entre los cuales destacamos
  - Puntero de pila (*stack pointer*)
  - Registro de punto flotante
  - Contador de programa (*program counter*)
  - Contador del siguiente programa (*next program counter*), utilizado para el retraso de ramas condicionales
  - Registro objetivo de carga retrasada
  - Valor a cargar en una carga retrasada
  - Valor especial para representar una dirección virtual errónea luego de una traducción
- Memoria paginada: por defecto la arquitectura especifica 32 páginas físicas cada una de las cuales tienen el tamaño del sector de disco de nachOS. Además, soporta virtuaización a través de una tabla de páginas linear ó TLB (*translation lookaside buffer*) pero no ambas al mismo tiempo.
- Varias:

  - Al comienzo de su ejecucuón nachOS alloca memoria y la pone a disposición a través de una variable mainMemory. Las rutinas readMem() y writeMem() traduen direcciones virtuales a direcciones físicas (ver siguiente apartado).

  - NachOS ejecuta código a nivel usuario una instrucción a la vez: la rutina translate() permite mapear direcciones virutales (a nivel VM) a direcciones reales de la máquina anfitriona.

  - La rutina run() toma una instrucción, la ejecuta y continúa con la siguiente. Y repite esto indefinidamente, es decir, funciona como un bucle infinito.

  - NachOS incorpora un depurador que puede ser invocado luego de cada ejecución de intrucción a nivel usuario.

Apartado 4

En las funciones ft1 y ft2 del main.c trasladamos las lineas YIELD(t1); y YIELD(t2); a la última posición dentro de los bucles for respectivamente. De esta forma el extraer el valor numérico en count (unsigned c = count;) y el incremento y guardado del valor incrementado (count = c + 1;) se hacen se forma atómica y no pueden introducirse fallas en la lectura y escritura de la variable global debido al acceso compartido concurrente.

Estos cambios se pueden visualizar en el archivo main.c adjunto.