13: I/O

Sistemas Operativos 2

Ing. Alejandro León Liu

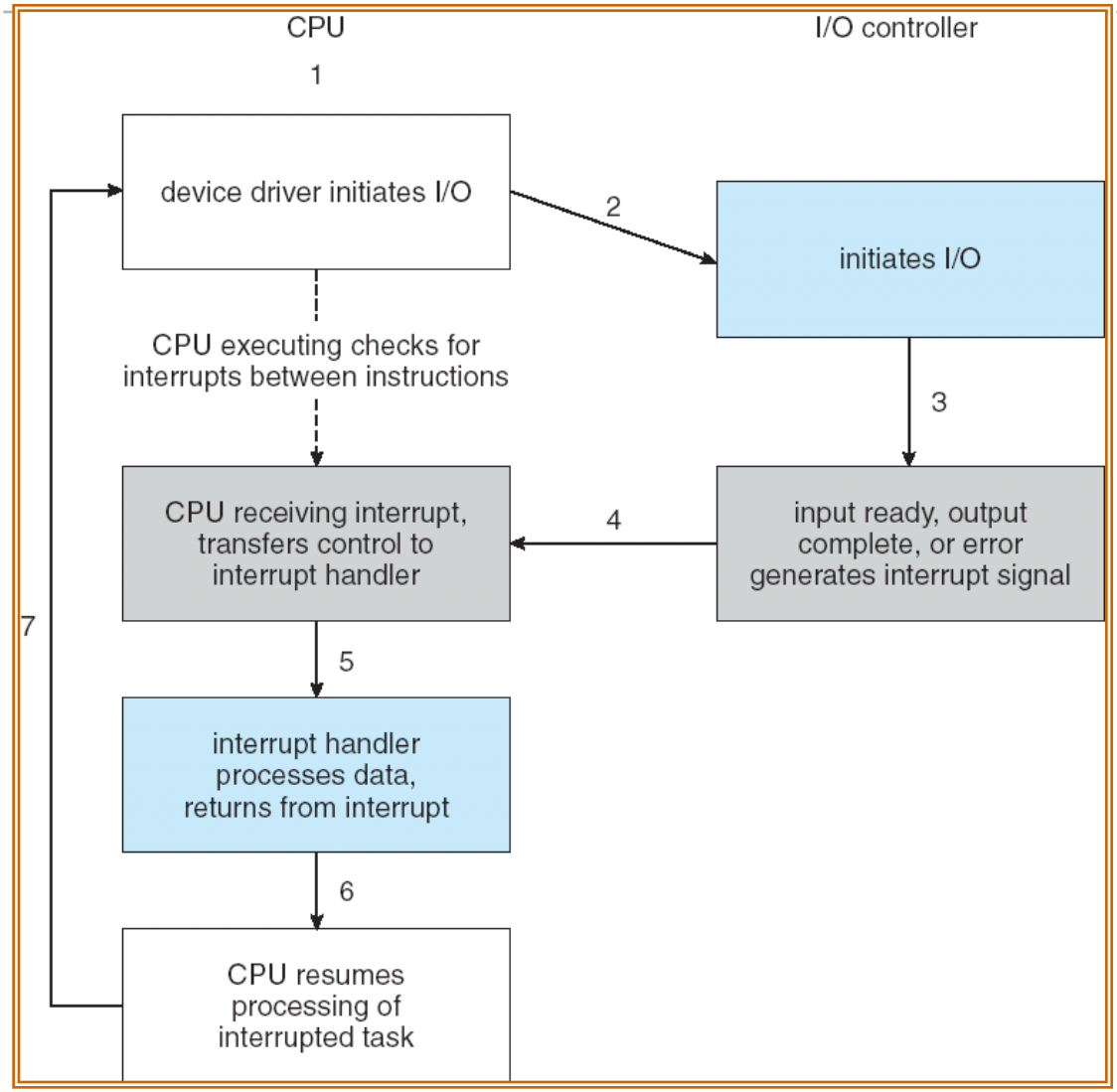
- **I/O**
- I/O API
- Kernel I/O Subsystem
- Ejemplo
- Performance

I/O

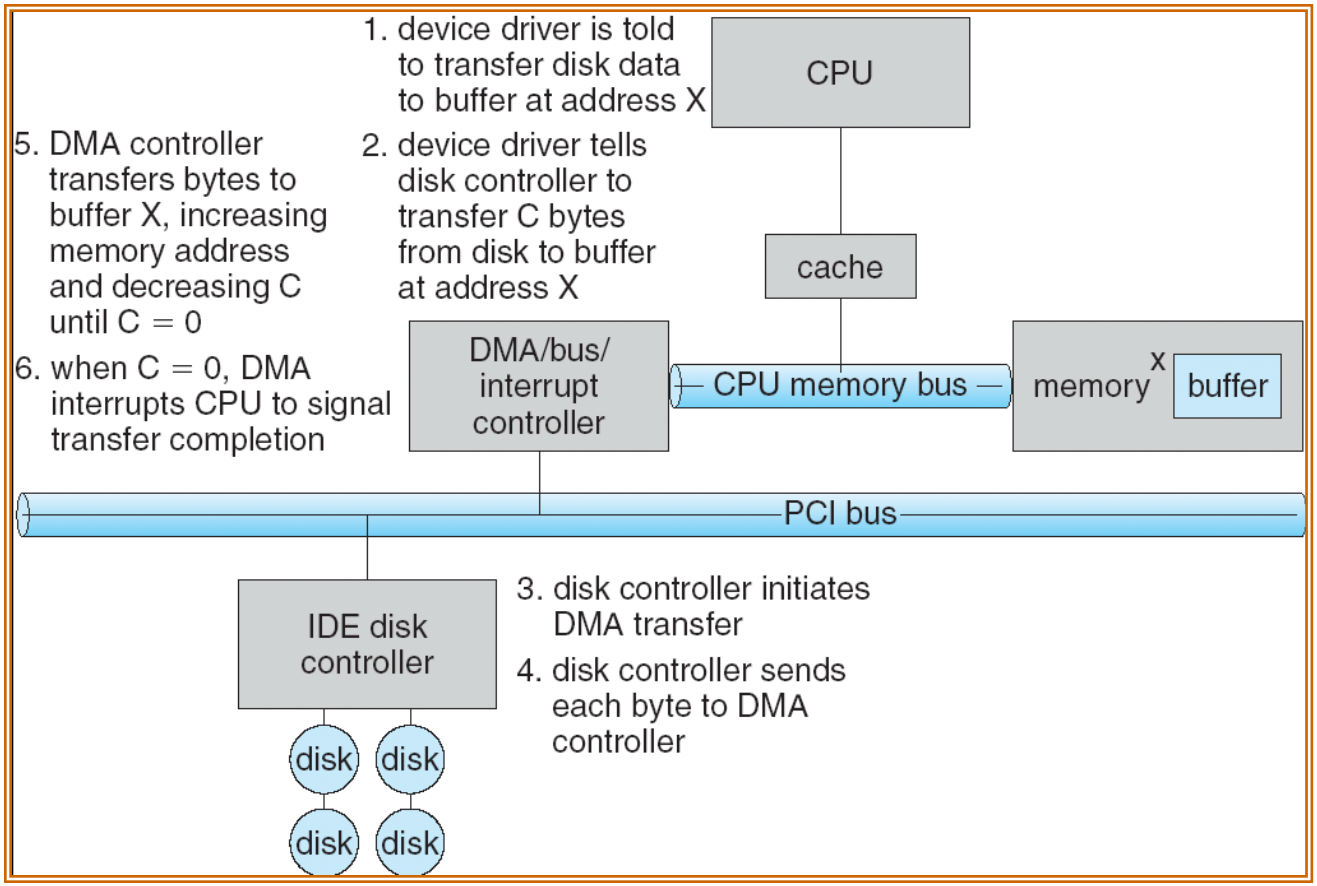
- Diversos dispositivos
  - Diferente función, diferentes características
  - S.O. debe controlarlos todos: Complejidad
  - I/O se lleva a cabo en I/O kernel subsystem
  - Drivers: encapsular detalles de dispositivos
  - Tipos de dispositivos
    - Almacenamiento
    - Comunicación (redes)
    - Interfaz con humanos.

- Puertos
- Bus
  - Subsistema para transferir datos entre componentes
- Controladores
  - Chip que controla un bus, puerto o dispositivo
- Comunicación a través de un puerto
  - Memoria compartida
  - Registros
    - Data in
    - Data out
    - Status
      - Leer status del dispositivo
    - Control
      - Enviar comandos al dispositivo

- Flujo de control
  - Polling: busy waiting
    - Útil para alta velocidad
      - Read
      - And
      - Jump
    - Costo de oportunidad
  - Interrupciones
    - Asíncrono
    - Prioridades


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

- Direct Memory Access
  - Evitar lectura/escritura de datos por registros

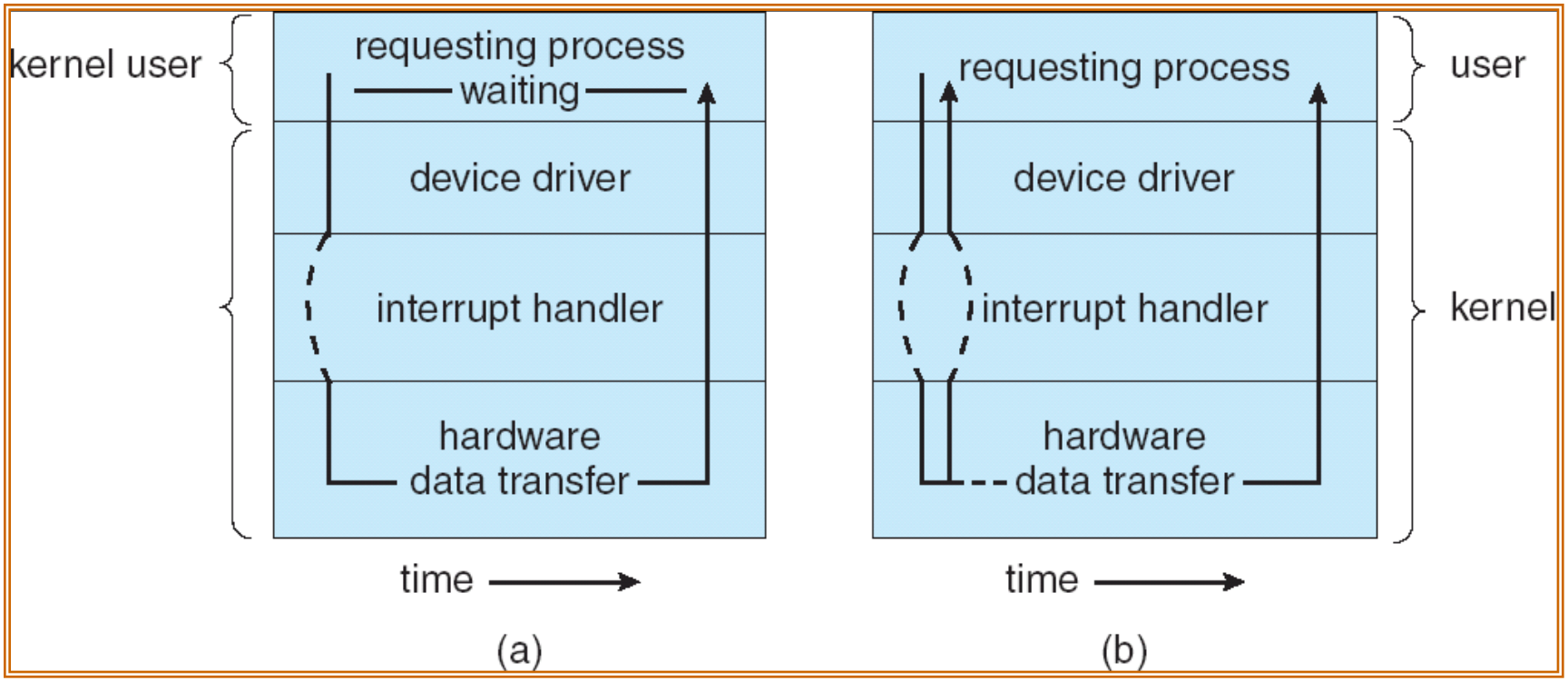


- I/O
- **I/O API**
- Kernel I/O Subsystem
- Ejemplo
- Performance

I/O API

- S.O. debe manejar varios dispositivos de forma estándar: Interfaces
  - Dispositivos por Bloques: discos
    - Read
    - Write
    - Seek
    - Memory mapped files
  - Dispositivos de caracteres
    - Get
    - Put
    - Aplicación interpreta teclas
  - Dispositivos de red: sockets
    - Connect
    - Read
    - write

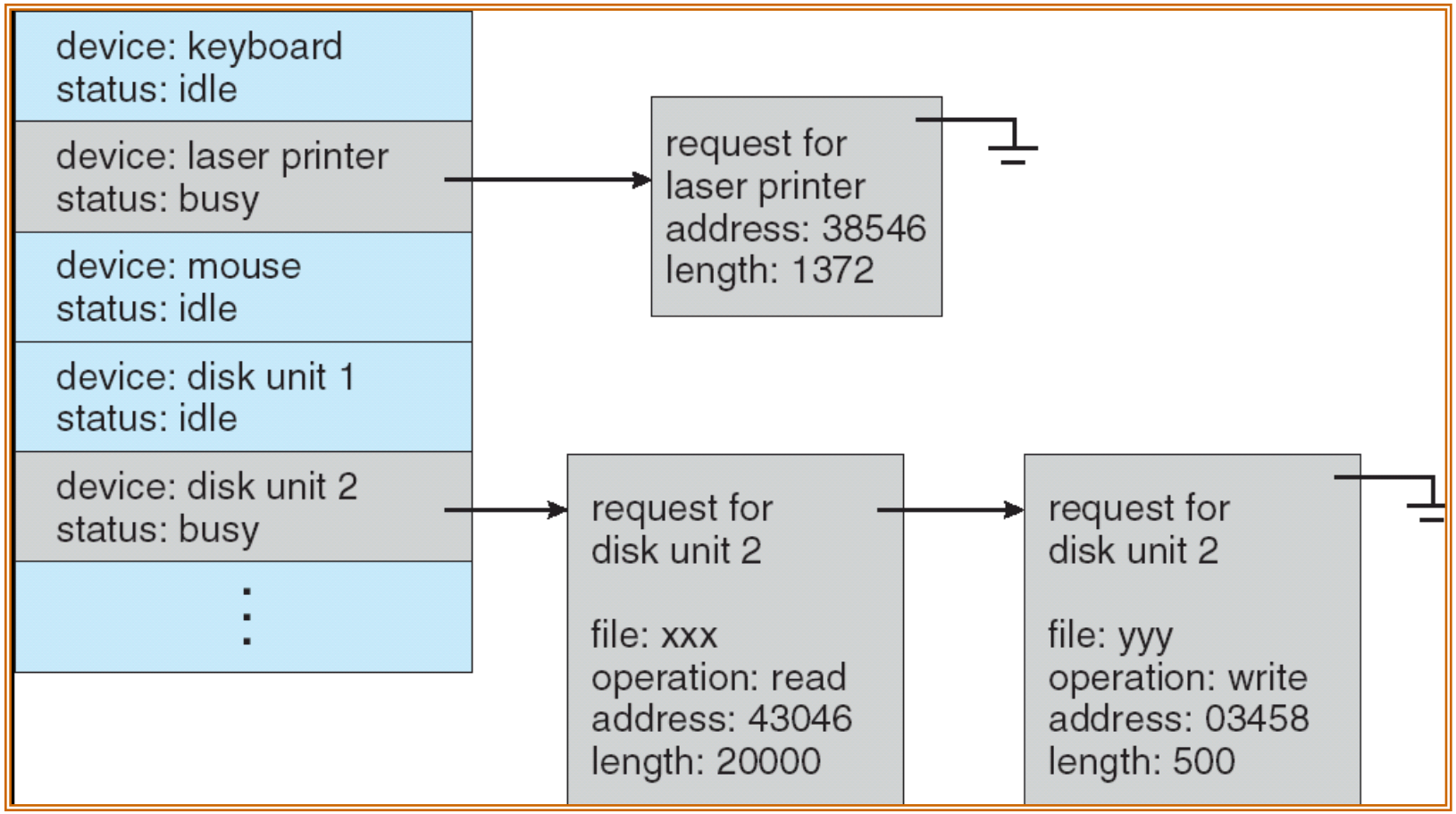
- Relojes y timers
  - Get date
  - Get elapsed time
  - Set timer
- Blocking / non blocking I/O
  - Blocking: Aplicación se ‘bloquea’ hasta terminar I/O
  - Non blocking: Aplicación puede continuar ejecutándose
    - Multithreading
    - Llamadas asíncronas
    - Listeners, delegates, etc…


kernel user
requesting process
waiting
device driver
interrupt handler
hardware
data transfer
time
(a)
requesting process
device driver
interrupt handler
hardware
data transfer
time
(b)
user
kernel

- I/O
- I/O API
- **Kernel I/O Subsystem**
- Ejemplo
- Performance

KERNEL I/O SUBSYSTEM

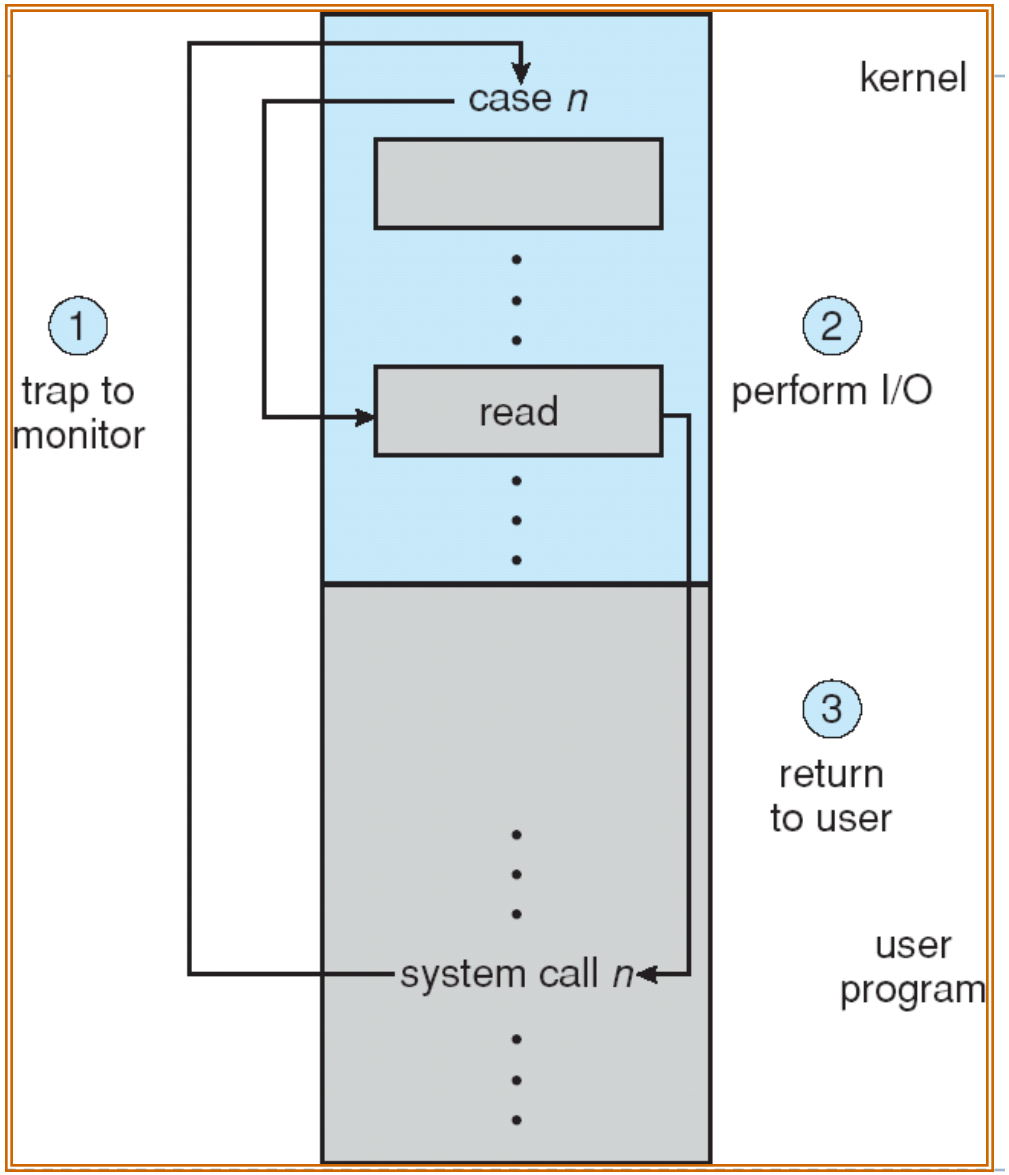
- Calendarización



- Buffers
  - Área de memoria, guarda datos mientras son transferidos.
  - Diferentes velocidades entre productor y consumidor
    - Transmitir un archivo
      - Velocidad de transmisión menor que velocidad de disco
      - Acumular bytes en memoria
      - Al completar un buffer, escribir a disco
  - Diferentes tasas de transmisión
    - Separar información en paquetes
    - Juntar paquetes en un buffer

- Transmitir datos entre kernel y aplicación
  - Aplicación ejecuta un write
  - No se ejecuta inmediatamente, aplicación cambia datos
  - Kernel escribe el write desde un buffer: evitar inconsistencias
- Cache
  - Memoria que guarda copias de los datos
  - Acceso a cache más rápido que dato original
  - Cache & buffer
    - Archivos en memoria

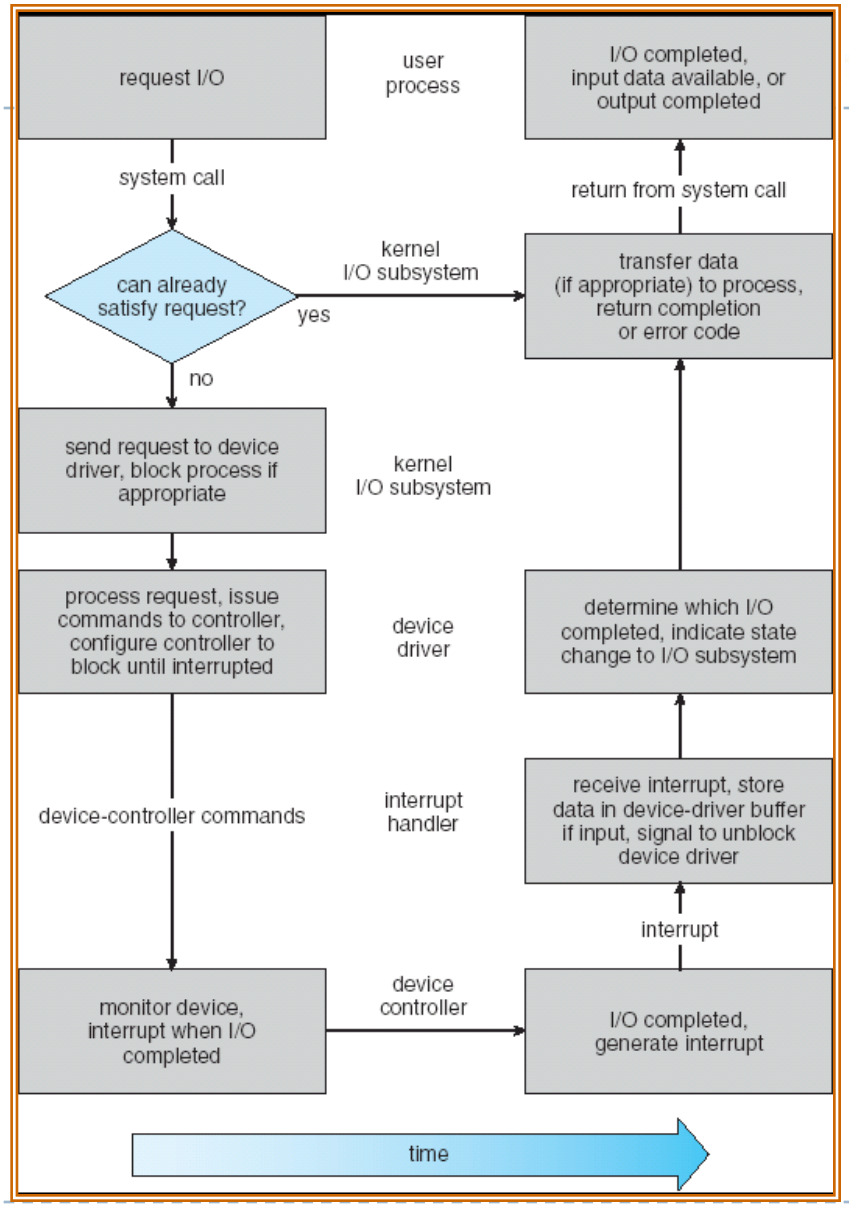
- Spooling
  - Almacena output a un dispositivo
  - Dispositivo debe recibir información sin pausas
  - Impresoras & Cintas magnéticas
  - Coordinar output concurrente a dispositivo
- Protección de I/O
  - Prevenir a usuarios ejecutar I/O
    - Instrucciones de IO son privilegiadas
  - Memoria mapeada protegida de usuarios


kernel
case n
read
1
trap to monitor
2
perform I/O
3
return to user
system call n
user program

- I/O
- I/O API
- Kernel I/O Subsystem
- **Ejemplo**
- Performance

EJEMPLO

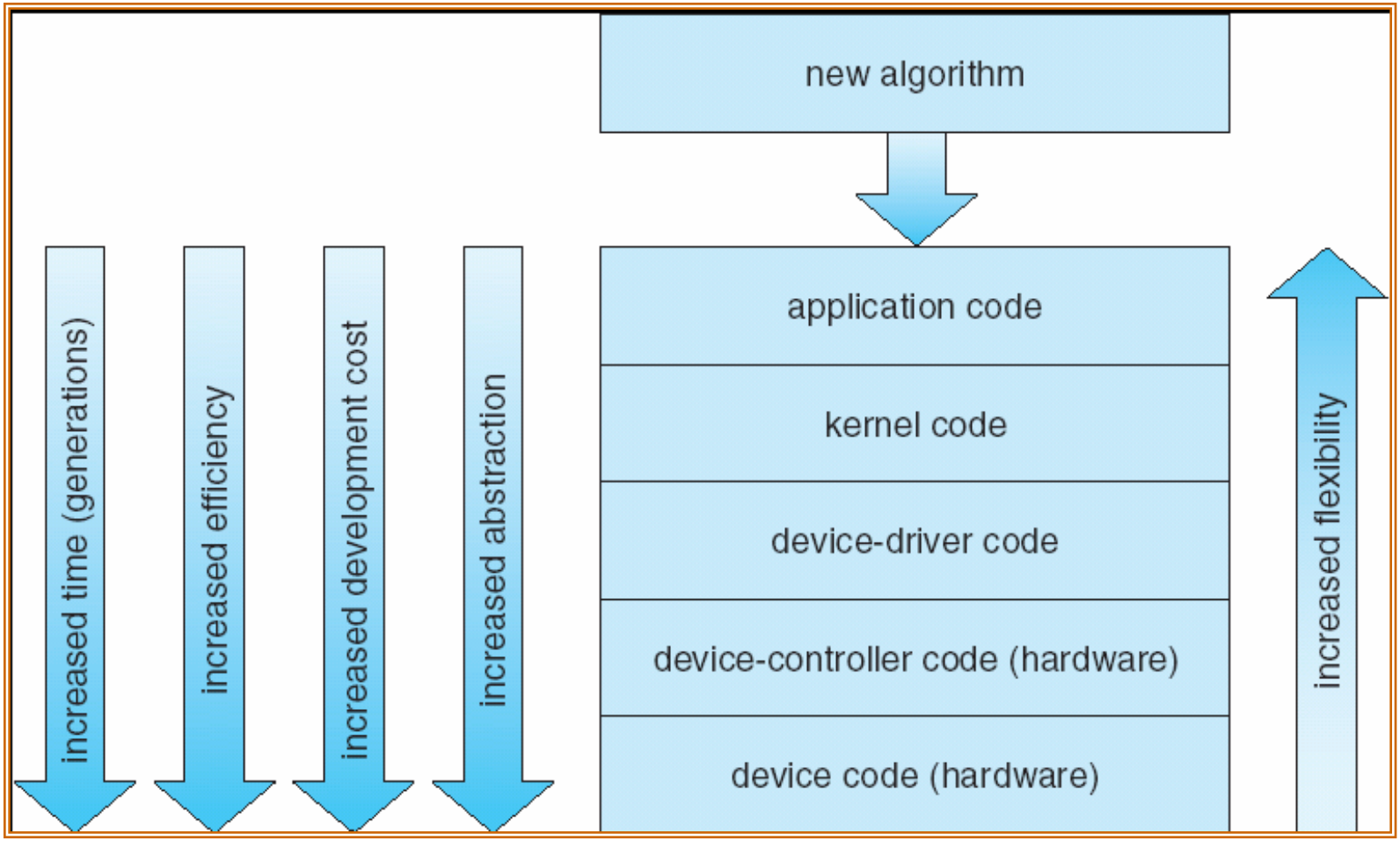
- Blocking read
  - ¿Data en Cache?
  - Encolar
  - Calendarizar
  - Alocar memoria
  - Controlador opera dispositivo
  - Polling o interupt notifica fin
  - Retornar data a proceso



- I/O
- I/O API
- Kernel I/O Subsystem
- Ejemplo
- **Performance**

PERFORMANCE

- I/O
  - Varios context switch
  - Transmitir datos: Ocupar bus del sistema
- Transferencias grandes, menos interrupciones
- Concurrencia: DMA
- Procesamiento en controladores: CPU libre


new algorithm
application code
kernel code
device-driver code
device-controller code (hardware)
device code (hardware)
increased time (generations)
increased efficiency
increased development cost
increased abstraction
increased flexibility

- Nivel de aplicación
  - Flexible
  - No causa errores del sistema
  - Más fácil desarrollo
    - Evitar reinicio de dispositivos
  - Ineficiente
    - Varias capas
    - Varios context-switch
- Nivel de kernel
  - Desarrollo complicado
  - Menos flexible
  - Peligroso

- Hardware
  - Mejor desempeño
  - Difícil reparar bugs
  - Difícil implementación