Conceptos Básicos de Diseño de Sistemas

Luis Daniel Benavides 22-02-2017

Importante Leer

- Capítulo 2 de “Principles of Computer system Design" by Saltzer and Kaashoek

Conceptos Básicos de Diseño de Sistemas

Estrategias básicas de diseño de sistemas

Atacando complejidad

- Modularización: divida y conquiste
- Abstracción: separación de interfaces e implementación
- División por capas
- Jerarquía: Interconexión en forma de árbol
- Uniendo todo: nombres para hacer conexiones
- * Iteración
- * Manténgalo simple

Las tres abstracciones fundamentales

Las tres abstracciones fundamentales

- Memoria
- Interpretes
- Enlaces de comunicación

Memoria

- Algunas veces llamada almacenamiento
  - *write(name, value)*
  - *value <——— read (name)*
- Ejemplos
  - Dispositivos de memoria de hardware
    - RAM chip, Flash memory,Disco magnético, CD
  - Sistemas de memoria de alto nivel
    - RAID, Sistema de archivos, Sistema de gestión de bases de datos

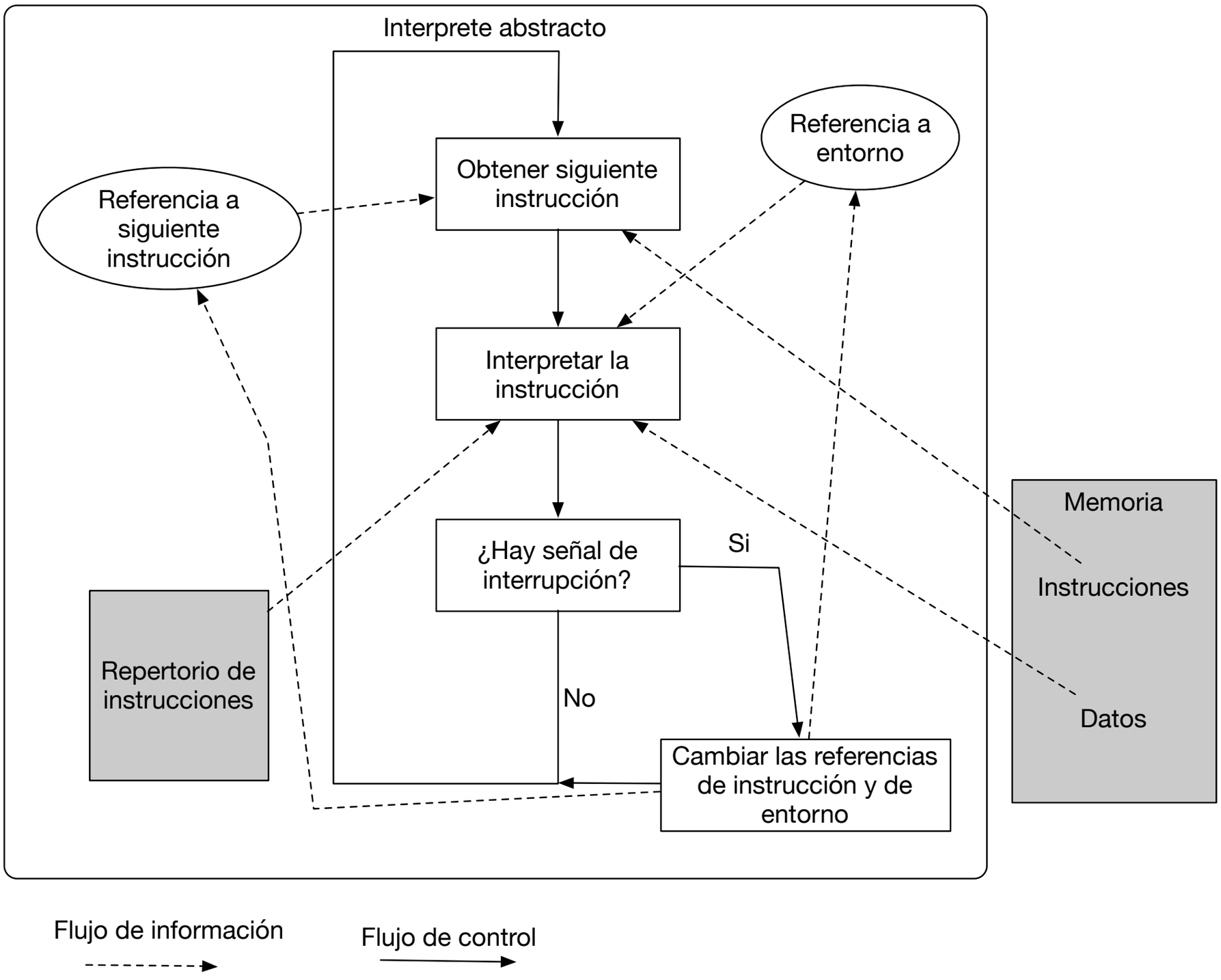
Coherencia y atomicidad del Read/Write

- Coherencia: READ es simple igual al mas reciente WRITE
- Atomicidad Antes-o-Después: Todo READ o WRITE ocurre completamente antes o después de cualquier otro READ or WRITE
- Amenazas a la coherencia y atomicidad:
  - Concurrencia (Palabra synchronized en JAVA)
  - Almacenamiento remoto (demoras pueden alterar el orden de los WRITES o READS)
  - Mejoramiento de Desempeño (Memorias Caché, optimización de los compiladores)
  - Tamaño de celdas no acomodada tamaño del valor
  - Almacenamiento replicado

Interpretes

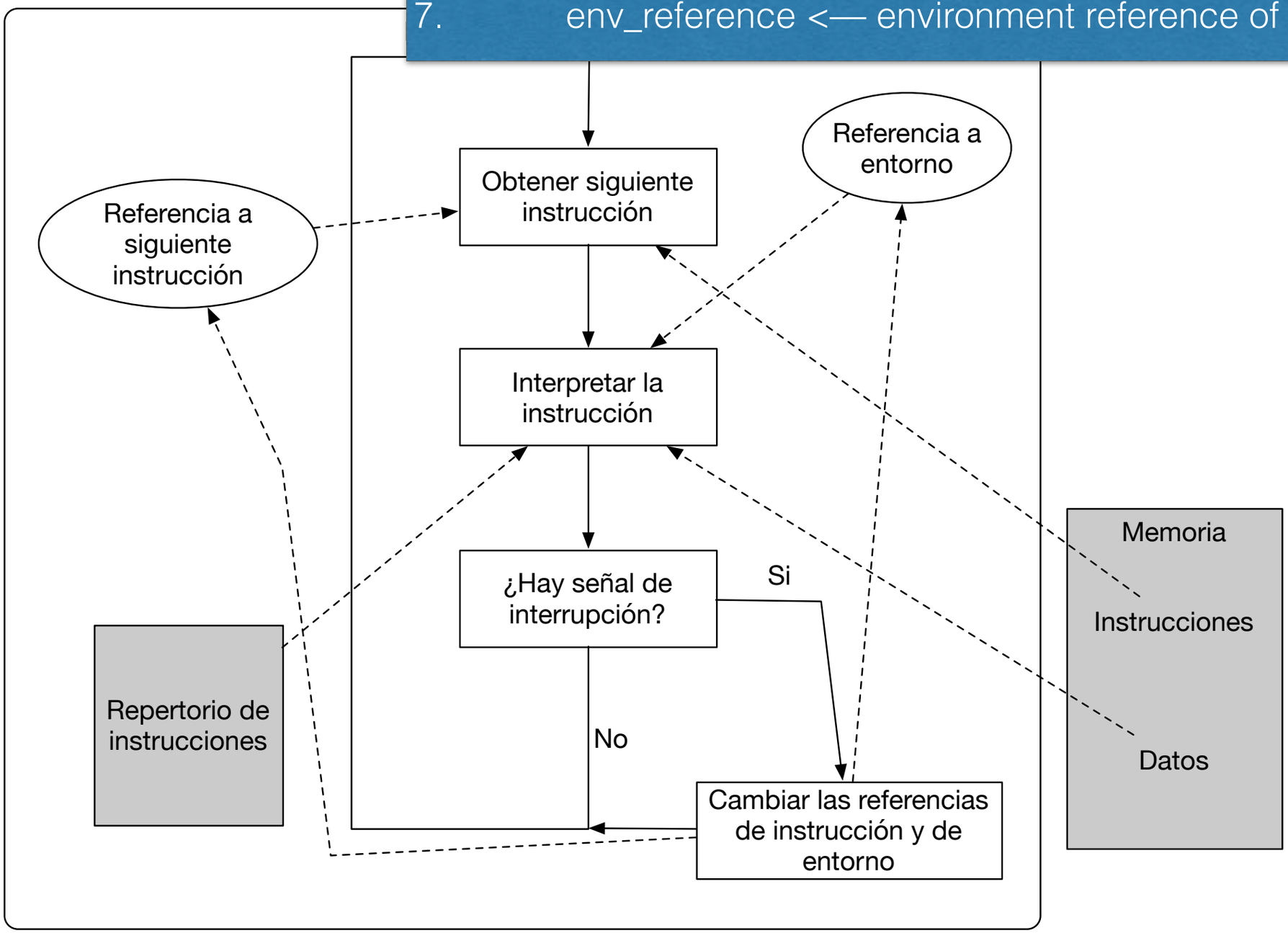
- El elemento activo de los sistemas de información.
- Se pueden modelar con
  - Una referencia a la siguiente instrucción
  - Un repertorio de acciones
  - Una referencia a un entorno
- Ejemplos
  - Hardware: Pentium 4, PowerPC 970, controlador de disco, controlador de pantalla
  - Software: Java, LISP, Pascal, C++, Smalltalk, Excel, Internet Explorer, Firefox

Interprete abstracto



Interpre

```
1.  procedure INTERPRET()
2.    do forever
3.      Instruction <— READ (instruction_reference)
4.      Perform instruction in the context of env_reference
5.      if interrupt_signal = TRUE then
6.          instruction_reference <— entry point of int_handler
7.          env_reference <— environment reference of int_handler
```



Capas de Interpretes

- Generalmente los interpretes se organizan en capas

- La capa inferior es el hardware

- Cada capa le entrega un repertorio de instrucciones a la capa superior (API)

- **Ejercicio:** Puede representar en un dibujo el concepto de capas de interpretes tomando como ejemplo un programa calendario escrito en java?

Enlaces de comunicación

- Proveen un mecanismos para mover información entre componentes físicamente separados
- Un modelo simple
  - *send (nombre_Del_Enlace, espacio_memoria_salida)*
  - *receive (nombre_Del_Enlace, espacio_memoria_entrada)*
- Ejemplos
  - Hardware: par trenzado, cable coaxial, fibra óptica
  - Alto nivel: ethernet, Universal Serial Bus, la internet, el sistema telefónico, el *pipe* de Unix.

Nombres en sistemas de computo

Nombres

- Los sistemas manipulan objetos
- Los nombres referencian objetos
- Ejemplos
    - R5 (registro de procesador)
    - 174FFF (Dirección de memoria)
    - escuelaing.edu.co (nombre de punto de conexión de red)
    - 18.72.0.151 (dirección de punto de conexión de red)
    - alice (nombre de usuario)
    - /proyecto/planeación/plan.doc
    - http://www.escuelaing.edu.co/sistemas/respuestasaProblemas.txt

Nombres desde la perspectiva de objetos

- Computador manipula objetos
- Objeto puede ser estructurado, es decir está conformado por otros objetos
- Un objeto puede usar otros objetos por copia o por referencia
- ¿Recuerda cuál es la diferencia?
- Nombres me sirven …
    - Como herramienta de comunicación y organización
    - Para desacoplar un objeto de los otros. El enlace se puede hacer más tarde en el tiempo
- Escriba algunos ejemplos del suyo de nombres para comunicarse y para desacoplar.

Esquema de nombres

- El diseñador crea un esquema de nombres
  - Espacio de nombres, alfabeto y sintaxis de los nombres
  - Algoritmo de mapéo de nombres, asocia nombres con valores
    - Universo de valores, un valor puede ser un objeto, o otro nombre del mismo o otro espacio de nombres
- La resolución de nombres depende de un contexto
  - Ej.: Búsqueda de nombre en directorio de una ciudad (contexto=ciudad)
  - Ej.: Búsqueda relativa de un archivo en un subdirectorio

API simple para el manejo de nombres

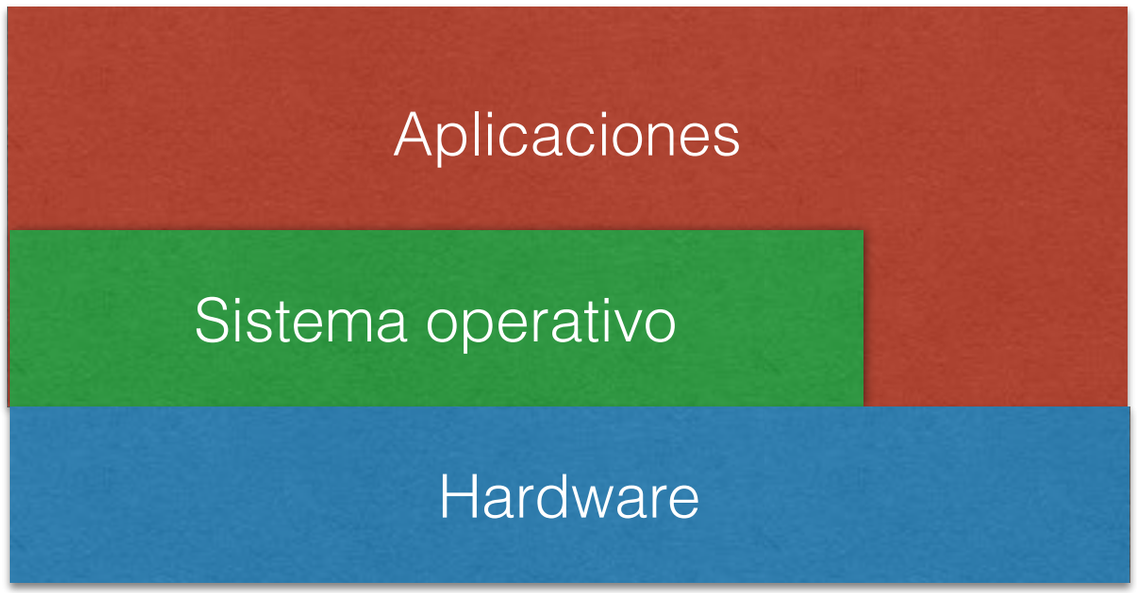
- Value <— RESOLVE(name, context)
- status <— BIND (name, value, context)
- status <— UNBIND (name, context)
- list <— ENUMERATE (context)
- result <— compare(name1, name2)

¿Cómo resolver nombres?

- Nombre bien conocido, ej., google.com
- Broadcast: publicar el nombre en un espacio de alta difusión
- Búsqueda, por ejemplo en un buscador
- Broadcast búsqueda, ej., Gritar “Alguien conoce un nombre para... (algo)”
- Resolver el nombre de un espacio de nombres hacia otro espacio de nombres, ej., DNS (www.gg.com -> 200.12.12.1)
- Presentación, alguien me presenta
- Encuentro físico, ej., una cita

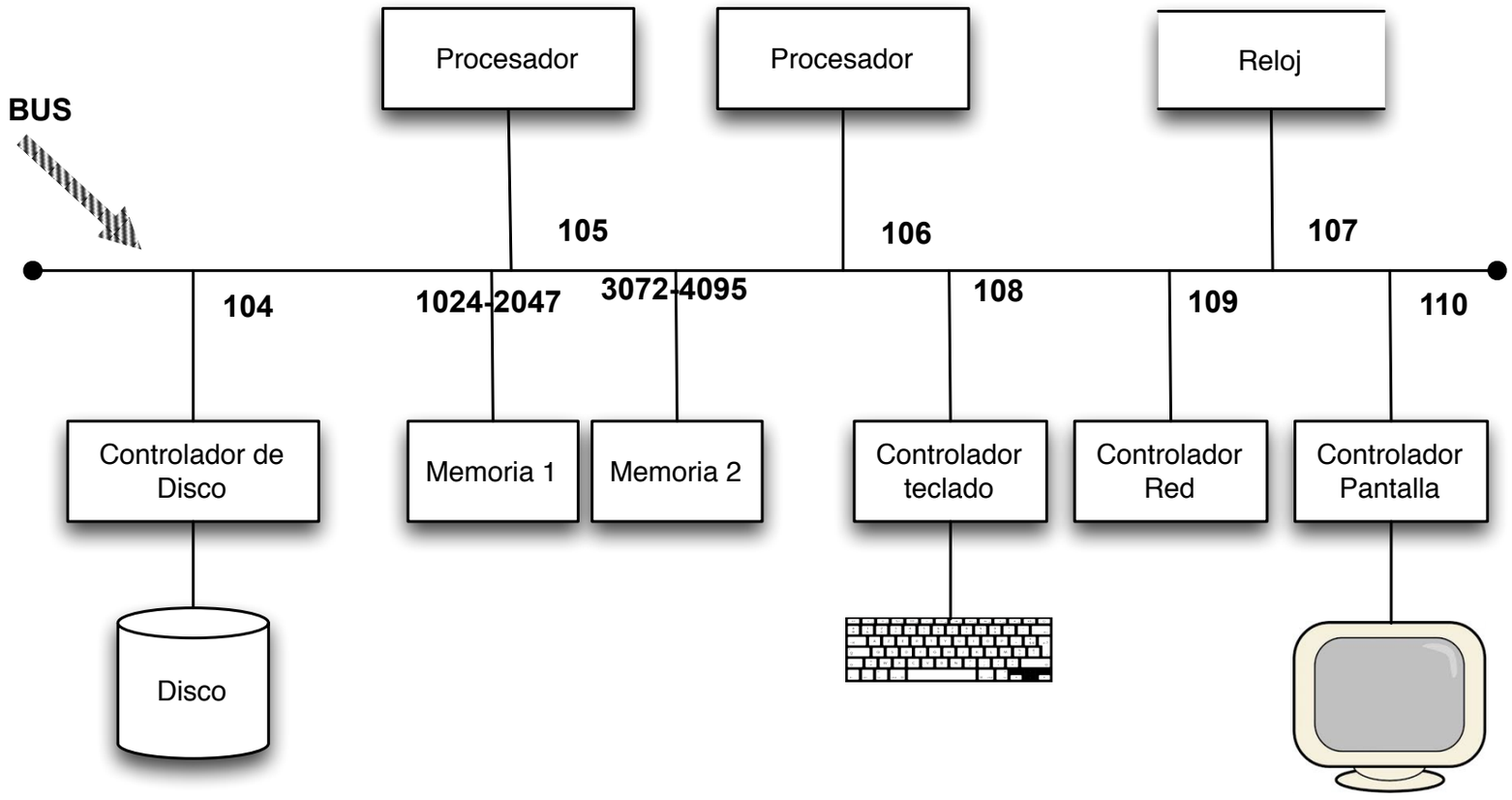
Organizando sistemas con nombres y capas

La organización típica de un computador



Una capa de hardware: el Bus

- La capa de hardware de un computador típico está construida por módulos que implementan las tres abstracciones

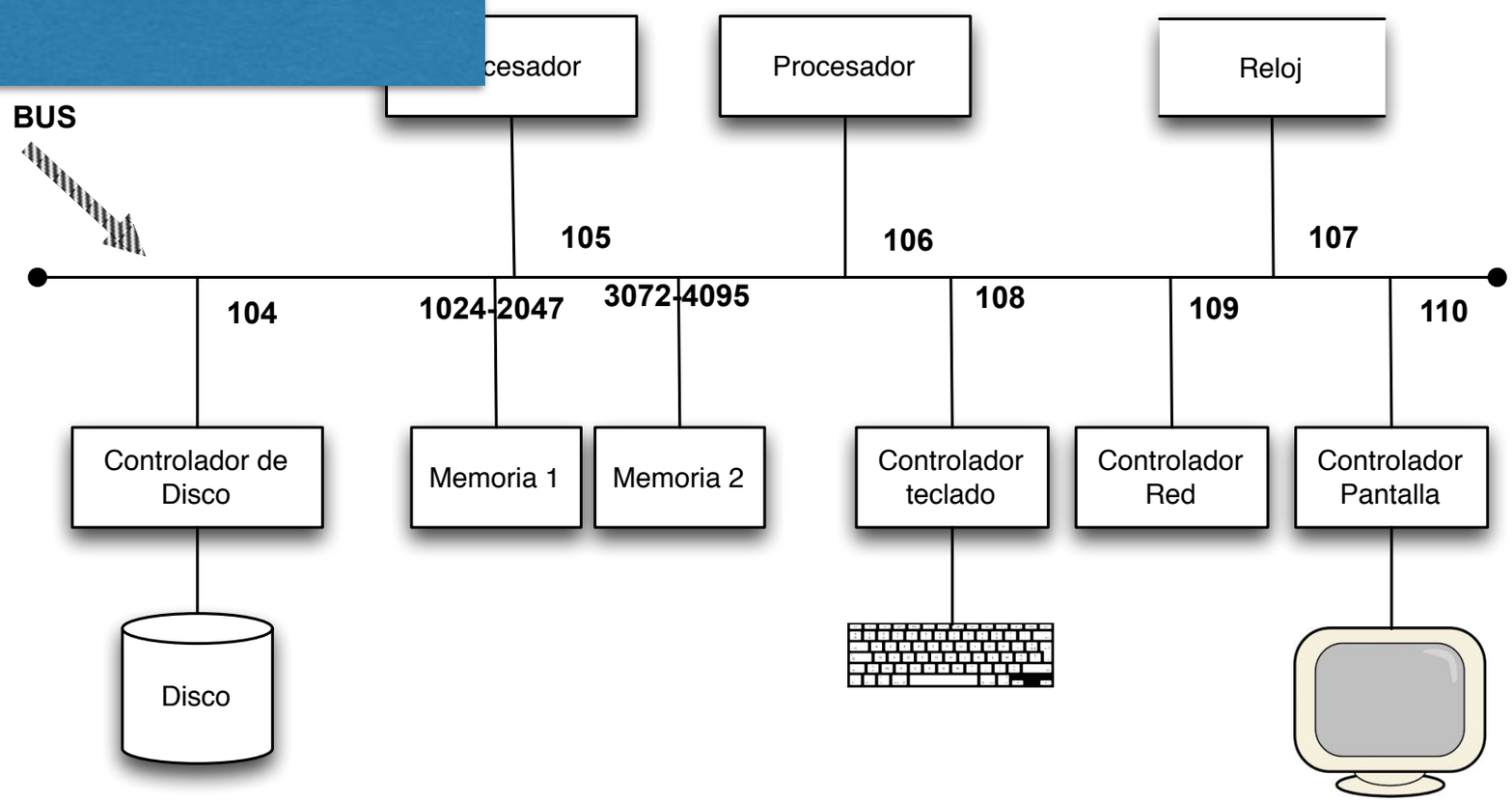


Una capa de hardware: el Bus

Ejemplo de comunicación en el bus
El procesador 2 quiere leer el contenido del espacio de memoria 1742 en el registro 1:

Mensaje 1: **{1742, READ, 106}**
En la memoria: **value <— READ(1742)**
Respuesta 1: **{106, value}**

vare de un computador típico está
ódulos que implementan las tres

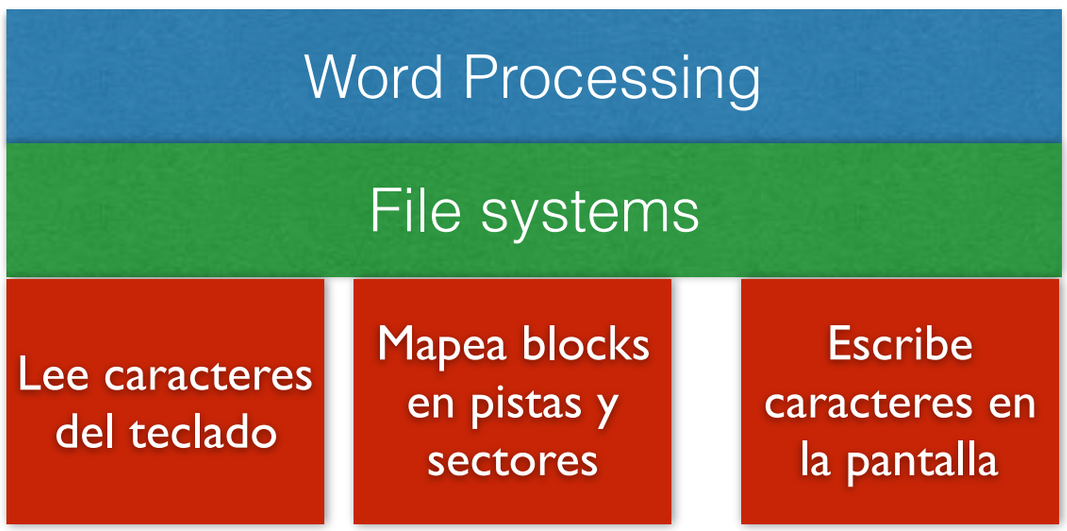


Una capa de software: El sistema de archivos

- En Unix todo es un archivo, inclusive dispositivos externos proveen una interfaz de archivo

- El sistema se organiza por capas

- Se utilizan nombres de directorios y de archivos para referenciar archivos y dispositivos



Ejemplo de uso del sistema de archivos en Unix

- Programa simple para leer del teclado, almacenar en un archivo e imprimir en la pantalla.

```
character buf

file <— OPEN(“/Users/Alumno/Documento de ARQUITECTURA.doc”,READWRITE)

input <— OPEN (“keyboard”, READONLY)

display <— OPEN (“display”, WRITEONLY)

While not End_Of_File(input) do

  READ (input, buf, 1)

  WRITE (file, buf, 1)

  WRITE (display, buf,1)

CLOSE (file)

CLOSE (input)

CLOSE (display)
```

Ejercicios

Ejercicio 1

- Usted fue contratado por la empresa de telefonía móvil para implementar el servicio de reenvío de llamadas. Usted ha estado evaluando qué hacer si alguien dirige llamadas a un número y el dueño de ese número a su vez dirige llamadas a otro número. Hasta ahora usted piensa en dos posibilidades:

    - **Sígame**. Pedro olvidó su teléfono en casa y dirige el teléfono al de María. Ana que está cuidando al bebe de Pedro y se está quedando sin batería dirige su teléfono al de Pedro. Juan llama al teléfono de Ana, el teléfono de Pedro suena, y Ana contesta

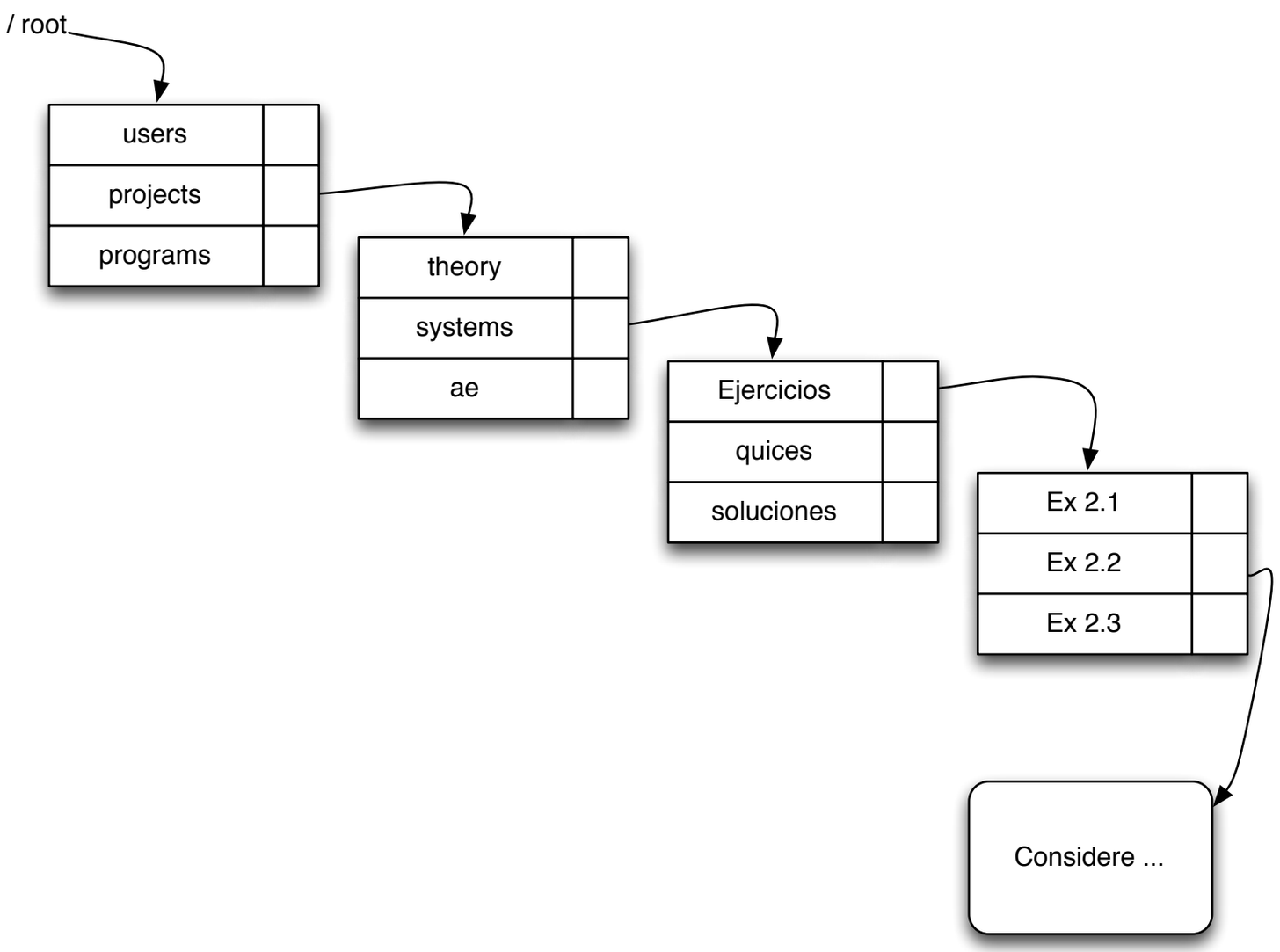
    - **Delegación.** Pedro olvidó su teléfono en casa y dirige el teléfono al de María. Ana le pide a Pedro que atienda sus llamadas durante el fin de semana y dirige su teléfono al de Pedro. Juan llama al teléfono de Ana, el teléfono de María suena, María contesta y le pasa la llamada a Pedro.

- Usando la terminología de la sección de nombres describa estas dos posibilidades

- ¿Qué podría salir mal si Pedro ya ha dirigido su teléfono al de María antes de que Ana dirija el teléfono al de él?

- La compañías de teléfonos usan generalmente Delegación y no Sígame. ¿Por qué?.

Ejercicio 2



Considere la siguiente parte del sistema de archivos.

Si le dan el siguiente path-name: */projects/systems/Ejercicios/Ex 2.2*

Usted va a resolver el tercer componente del nombre, es decir *Ejercicios.*

1) Identifique el contexto donde debe buscar el nombre y la referencia de contexto que permite localizar este contexto
2) ¿Cuál de los siguientes términos aplican a esta referencia de contexto: por defecto, explícito, por objeto, built-in (quemado), por-objeto, por nombre?

Fin