Actividad Practica capitulo 1 UF1286

Enunciado:

Este caso práctico está diseñado para evaluar tus conocimientos sobre la ejecución de procesos, el arranque de un sistema operativo y el diseño arquitectónico de sistemas. A través de los siguientes ejercicios, deberás demostrar tu comprensión teórica y capacidad de análisis aplicado a situaciones reales en el ámbito de los sistemas operativos.

Ejercicios:

1. Una aplicación informática es ejecutada en un sistema operativo. La aplicación realiza una lectura en el disco duro del sistema. Finalmente, el usuario cierra la aplicación. Describe la secuencia de estados en los que se encuentra el proceso de la aplicación en cada momento desde que se inicia hasta que finaliza.

2. Imagine que su sistema operativo está instalado en la única partición del disco duro del sistema y que, por lo tanto, en los primeros sectores del disco se encuentra el programa que se encarga de cargar el sistema operativo. Describa la secuencia de pasos genéricos que se producen desde que arranca el equipo hasta que se produce la petición de autenticación en el sistema operativo.

3. Desea diseñar un sistema operativo en el que pueda intercambiar módulos que serán programados en diferentes tipos de lenguaje. Imagine que es uno de los ingenieros informáticos responsables de determinar el diseño arquitectónico. Presente el diseño del sistema operativo adecuado y defienda su propuesta.

Forma de Entrega:

El caso práctico deberá entregarse en formato pdf:
ApellidoNombre_CasoPractico1.pdf

1. Una aplicación informática es ejecutada en un sistema operativo. La aplicación realiza una lectura en el disco duro del sistema. Finalmente, el usuario cierra la aplicación. Describe la secuencia de estados en los que se encuentra el proceso de la aplicación en cada momento desde que se inicia hasta que finaliza.

Nuevo --- Creacion del proceso

Listo --- a la espera de ser asignado el procesador

Ejecutando --- el planificador le asigna cpu

Bloqueado ---- el proceso se bloquea para realizar operacion E/S, la lectura del disco duro

Listo --- una vez finalizada la lectura el proceso vuelve a estar a la espera de procesador

Ejecutando --- el planificador le vuelve a asignar cpu

Finalizado --- el usuario finaliza el programa y el sistema libera todos los recursos que usaba la aplicación

2. Imagine que su sistema operativo está instalado en la única partición del disco duro del sistema y que, por lo tanto, en los primeros sectores del disco se encuentra el programa que se encarga de cargar el sistema operativo. Describa la secuencia de pasos genéricos que se producen desde que arranca el equipo hasta que se produce la petición de autenticación en el sistema operativo.

Arranca el hardware bajo el control de la bios

Se hace un testeo del hardware y se carga en memoria el MBR (Registro maestro de arranque) que contiene informacion sobre las particiones creadas y la particion activa, y es cuando el MBR toma el control, buscando en la particion activa el sector de arranque del sistema, lo carga en memoria y el MBR le da el control al sistema operativo

Arranca el sistema operativo:

El sistema hace un test de los archivos propios, se crean la estructuras de datos internas que necesita el SO para su ejecución y se carga en memoria el resto del SO procediendo al proceso de Autenticación

En resumen:

1.- Arranque de bios

a.- Test de hardware

b.- Carga en memoria del MBR

c.- Carga en memoria del sector de arranque del sistema operativo tomando este el control una vez cargado

2.- Arranque del sistema operativo

a.- Test de ficheros propios

b.- Carga en memoria del resto del sistema de arranque necesario

c.- Inicio del sistema de autenticación

En linux → Bios → MBR → Kernel → se monta ROOT en modo solo lectura→ se busca, se descomprime y se monta INITRD → se inicializan los ficheros → se desmonta INITRD → se crea el dispositivo ROOT → se libera la memoria no utilizada → se pasa el control a sbin/init que carga los servicios, y las herramientas necesarias, y monta las particiones listadas en etc/fstab → y ya se procede a la autenticacion del usuario

en Windows → Bios → MBR → sector de arranque del SO → NTDLR/BOOTMGR → BOOTBCD que si hay varios sistemas operativos, los muestra en una lista para elequir con cual arrancar → se ejecuta el kernel de windows (NTOSKRNL.EXE) → Se procede a la autenticacion del usuario

3. Desea diseñar un sistema operativo en el que pueda intercambiar módulos que serán programados en diferentes tipos de lenguaje. Imagine que es uno de los ingenieros informáticos responsables de determinar el diseño arquitectónico. Presente el diseño del sistema operativo adecuado y defienda su propuesta.

Diseño:

La arquitectura se basa en un **microkernel**, que se encarga de lo esencial: planificación de procesos, comunicación entre procesos (IPC), manejo básico de memoria y drivers mínimos. Todo lo demás (gestion de sensores, GUI, domótica, entornos de agua, entornos secos, etc.)

se implementa como **módulos ampliables e intercambiables**, ejecutados en espacio de usuario con la lectura de los datos del sensor.

Elemento	Propuesta
Arquitectura base	Microkernel es la base de los SO para pequeños dispositivos como los domóticos
Modularidad	Un modulo básico de arranque con un servidor web para controlar desde cualquier dispositivo conectado a la red o a internet dependiendo del dispositivo a controlar -un modulo por cada sensor a controlar, ya sea de seguridad (volumetrico), o de luz/ruido/ vibración/ temperatura , etc … que se encargaría de sus propios procesos conectando mediante API propia con el modulo base, ademas posibilidad de adecuar los modulos existentes a nuevas caracteristicas medibles con el sensor y/o desarrollar nuevos modulos acordes a nuevos sensores o nuevos controles
Lenguajes permitidos	Cada módulo puede ser desarrollado en distintos lenguajes: Rust, C, Ada, Zig, Lua, Python, Go, Java, Kotlin, JavaScript. XML, CSS (Bootstrap), etc
Interfaces	Comunicación mediante APIs mediante wifi o cable según el tipo de sensor
Gestión de dependencias	Cada módulo tendria su mecanismo de gestion de dependencias, perteneciendo dichas dependencias al modulo base si son comunes o al propio módulo si son exclusivos de cada módulo
Seguridad	Separación clara entre kernel y la base que se conectaria con las API a los modulos que se necesite implementar. Con objeto de mitigar vulnerabilidades,

Justificación:

- **Flexibilidad multiplataforma**: permite a los desarrolladores usar los lenguajes en los que esten especializados, por ejemplo, Rust para seguridad, Python y C++ para backend, y controladores, XML HTML CSS y javascript para interfaces dinamicas tipo DOM, caracteristicos de frontend .
- **Mantenimiento individualizado**: cada módulo debería poder evolucionar por separado, teniendo un control de versiones propio
- **Mejor prueba y depuración**: los errores de un módulo no deben afectar a todo el sistema. Ideal para entornos seguros de investigación, enseñanza. Organismos gubernamentales etc
- **Escalabilidad**: dependiendo de las funciones que se deseen a controlar se añadirían mas o menos módulos, con un mínimo del modulo base y el modulo de integracion de módulos

- **Control individualizado por interfaz web:** un usuario admin, que creará a los usuarios necesarios y les dotará de un nivel de acceso que permita realizar el desempeño propio de cada usuario

En resumen, el sistema deberá permitir por ejemplo, el control completo de unas instalaciones con tanques de agua y recintos secos para diferentes especies de animales y plantas, desde la luz, temperatura, alcalinidad, acidez, riego alimentación, incluso todo el sistema domótico y de seguridad de las instalaciones incluyendo oficinas y almacenes. Dependiendo de los módulos a instalar.