Pep1

Diccionario:
MPPAL: Memoria principal

1. Introducción a los computadores

- Un módulo de IO transfiere datos desde dispositivos externos hacia el procesador y la memoria. Contiene **buffers:** zonas de almacenamiento internas que mantienen temporalmente los datos hasta que se puedan enviar.
- ## Registros del procesador:
  - **Visibles al usuario:** Permiten al programador minimizar referencias a memoria principal optimizando el uso de registros.
    - **Registros de datos:** Pueden usarse con cualquier instrucción de máquina que realice operaciones sobre datos.
    - **Registros de dirección:** Contienen direcciones de MPPAL
      - Registro índice, puntero de segmento y puntero de pila}
  - **De control y estado:** Usados por el procesador para controlar su operación y por rutinas privilegiadas del SO para controlar la ejecución del programas
    - PC: Dirección de la próxima instrucción que se leerá en memoria
    - IR: Contiene la última instrucción leída.
  - **Program Status Word (PSW):**
    - En el contexto de los sistemas operativos, el PSW (Program Status Word) o EFLAGS (en sistemas basados en arquitecturas x86) es un registro en la CPU que contiene varios indicadores (flags) que determinan el estado actual del procesador. Este registro es fundamental para el manejo del control de operaciones dentro de la CPU, así como para la gestión de la ejecución del programa.
    - El PSW típicamente incluye flags como el de carry, zero, sign, overflow, entre otros. Estos indicadores son utilizados por el procesador para realizar decisiones sobre el flujo de ejecución del programa.
    - Por otro lado, en arquitecturas como x86, el registro EFLAGS realiza funciones similares pero con características adicionales adecuadas para estas arquitecturas. Este registro incluye flags de estado y control, **como el interrupt flag (que permite o deshabilita las interrupciones)** y el direction flag (que controla la dirección de cadenas de datos).
    - **¿Dónde está el PSW?** se almacena típicamente en un registro especial dentro del procesador, no en memoria física o virtual. Este registro es una parte integral de la unidad central de procesamiento (CPU) y es accesible solo a nivel del hardware de la máquina.
    - Dado que los registros son más rápidos que la memoria para acceder y modificar, el almacenamiento del PSW en un registro permite que la CPU responda rápidamente a

cambios en el estado del procesador y controle la ejecución del programa de manera eficiente. Esto es esencial para operaciones que requieren una alta velocidad, como la evaluación de condiciones y la toma de decisiones sobre el flujo de control del programa.
    - Así, el PSW no se encuentra en la memoria física ni en la memoria virtual, sino directamente en la arquitectura del procesador, facilitando un acceso más rápido y directo que es crucial para el rendimiento del sistema.

- # Interrupciones:
    - Las interrupciones son mecanismos mediante los cuales el hardware o el software pueden señalar al procesador que un evento importante requiere atención inmediata. Aquí te describo los tipos de interrupciones que mencionaste:

    1. **Interrupciones de programa**: Estas interrupciones ocurren debido a operaciones en el código del programa que requieren la intervención del sistema operativo. Pueden ser generadas por errores en el programa, como divisiones por cero, violaciones de acceso a la memoria, o explícitamente por instrucciones de software diseñadas para este propósito, como las llamadas al sistema.
    2. **Interrupciones por temporizador**: Son generadas por el reloj del sistema a intervalos regulares para permitir al sistema operativo realizar ciertas tareas de mantenimiento. Estas interrupciones son esenciales para la multitarea preemptiva, donde el sistema operativo necesita interrumpir programas en ejecución para dar tiempo de CPU a otros procesos.
    3. **Interrupciones de I/O**: Estas interrupciones son iniciadas por dispositivos de entrada/salida (como discos duros, teclados, redes, etc.) para notificar al procesador que una operación de entrada/salida se ha completado y que los datos están listos para ser procesados o que un dispositivo está listo para recibir más datos. Esto permite que el CPU continúe realizando otras tareas mientras espera que las operaciones de I/O se completen.
    4. **Interrupciones por fallo del hardware**: Son causadas por errores físicos en el hardware, como fallas en la memoria, errores en el disco, sobrecalentamiento del procesador, entre otros. Estas interrupciones alertan al sistema operativo de problemas graves que podrían requerir que el sistema detenga operaciones, realice tareas de diagnóstico o se reinicie.

Cada tipo de interrupción permite al sistema operativo gestionar recursos de manera eficiente y asegurar una respuesta adecuada ante diversas condiciones de operación, mejorando la estabilidad, rendimiento y seguridad del sistema.

Una excepción es un tipo específico de interrupción que es generada internamente por el procesador debido a eventos extraordinarios que ocurren mientras se ejecutan las instrucciones del programa. Las excepciones pueden clasificarse de varias maneras, y aunque están relacionadas con las interrupciones, tienen diferencias significativas en cuanto a su origen y manejo.

- **Diferencias entre interrupciones y excepciones:**
    - **Interrupciones**: Son señales generadas por dispositivos de hardware o software que le indican al procesador que debe detener la ejecución actual para atender un evento. Generalmente, las interrupciones son asíncronas respecto a la ejecución del programa, lo que significa que pueden ocurrir en cualquier momento.
    - **Excepciones**: Son interrupciones generadas internamente por el procesador debido a condiciones especiales que surgen durante la ejecución de instrucciones, como errores de

programación o problemas de hardware. Son sincrónicas, ocurriendo en un punto específico de la ejecución del programa.

- **Tipos de excepciones:**
  1. **Trap (Trampa)**: Es una excepción que es intencionalmente provocada por el programa para invocar funciones del sistema operativo, a menudo a través de una instrucción específica. Las trampas son manejadas de manera que el programa puede continuar después de que se procesa la trampa.
  2. **Fault (Fallo)**: Ocurre cuando hay un error que puede ser corregido sin terminar el programa. Un ejemplo común es un fault de página, que sucede cuando un programa accede a una parte de la memoria que no está actualmente en la memoria física. El sistema operativo puede manejar este fault cargando la página necesaria en la memoria física y permitiendo que el programa continúe.
  3. **Abort**: Es una señal generada por condiciones de error graves, generalmente relacionadas con fallas de hardware, y no permite que el programa continúe. Un ejemplo sería un error de paridad de memoria.
- **Interrupciones de hardware específicas:**
  - **NIM (Non-Maskable Interrupt)**: Una interrupción NIM es un tipo de señal de hardware que no puede ser ignorada o desactivada (non-maskable). Es utilizada para manejar eventos críticos como errores de hardware severos.
  - **INTR (Interrupt Request)**: Es la línea de solicitud de interrupción estándar a través de la cual los dispositivos de hardware pueden solicitar atención del procesador. A diferencia de las NIM, las INTR generalmente pueden ser enmascaradas o desactivadas por el software.
- En resumen, tanto las interrupciones como las excepciones son fundamentales para el manejo de eventos en sistemas informáticos, pero difieren en su origen, sincronicidad y manejo por parte del sistema operativo. Las excepciones son más específicas para el contexto de la ejecución del programa, mientras que las interrupciones pueden ser más generales y originarse externamente.
- ## Manejador de Interrupciones:
  - El manejador de interrupciones es una función específica o un conjunto de funciones dentro del sistema operativo diseñadas para procesar interrupciones. Cuando una interrupción ocurre, el hardware del sistema pausa la ejecución del programa actual y transfiere el control al manejador de interrupciones apropiado. Este manejador es responsable de determinar la causa de la interrupción, ejecutar el código necesario para atender esta interrupción y luego retornar el control al programa que estaba en ejecución o a otro programa, según la planificación del sistema operativo.

2. Procesos

- ## PCB - Process Control Block:

- es una estructura de datos esencial en sistemas operativos que se utiliza para almacenar toda la información necesaria para gestionar un proceso específico. Cada proceso en el sistema tiene asociado un PCB, que es utilizado por el sistema operativo para mantener el seguimiento de los procesos a medida que cambian de estado, por ejemplo, de ejecución a espera o terminado.

1. **Identificador (PID - Process ID)**: Es un número único asignado a cada proceso para identificarlo de manera unívoca dentro del sistema. El PID es crucial para operaciones como la asignación de recursos, manejo de prioridades, y seguimiento de la actividad del proceso.
2. **Estado del Proceso**: Indica el estado actual del proceso. Los estados comunes incluyen listo (ready), ejecutando (running), esperando (waiting), y terminado (terminated). Este campo es vital para la planificación y manejo de procesos del sistema operativo.
3. **Prioridad**: Algunos sistemas operativos utilizan prioridades para decidir el orden de ejecución de los procesos. Un proceso con una prioridad más alta generalmente será ejecutado antes que uno con prioridad más baja.
4. **Contador de Programa (PC - Program Counter)**: Almacena la dirección de la próxima instrucción que el proceso debe ejecutar. Es fundamental para la reanudación correcta del proceso después de una interrupción o cambio de contexto.
5. **Punteros a Segmentos de Memoria**: Estos incluyen punteros al código del programa, datos del programa y pila del proceso. Estos punteros son esenciales para asignar y gestionar la memoria que el proceso puede utilizar.
6. **Datos de Contexto**: Incluyen los valores de los registros del procesador en el momento en que el proceso fue interrumpido o cambió su estado. Esta información es crucial para poder reanudar la ejecución del proceso exactamente donde se dejó.
7. **Información de Estado de I/O**: Detalles sobre los archivos y dispositivos de entrada/salida que el proceso está utilizando, incluyendo punteros a buffers de datos, contadores de I/O y estados de las operaciones de I/O.
8. **Información de Auditoría**: Puede incluir detalles sobre el uso de recursos del proceso, como el tiempo de CPU consumido, cantidad de operaciones de entrada/salida realizadas, y otros registros utilizados para monitorizar y auditar el comportamiento del proceso.

- # Modelo de Estados de un Proceso
- # Espacio virtual de direcciones de un proceso
  - Este modelo permite a cada proceso operar como si tuviera acceso a una gran cantidad de memoria continua y exclusiva, independientemente de la memoria física disponible realmente en el sistema.
  - ## Concepto Básico
    El espacio virtual de direcciones es la vista de la memoria que un proceso tiene durante su ejecución. Este espacio es gestionado por el sistema operativo y se presenta al proceso como un rango contiguo de direcciones que comienza desde una dirección base (generalmente cero) hasta un límite máximo que puede ser varios gigabytes, dependiendo de la arquitectura del sistema y la configuración del sistema operativo.
    ## Componentes del Espacio Virtual de Direcciones

1. **Texto (Código)**: Esta sección contiene el código ejecutable del proceso. Es generalmente marcado como de solo lectura para prevenir que el proceso modifique su propio código.
2. **Datos**: Incluye variables globales y estáticas utilizadas por el programa. Esta área puede expandirse durante la ejecución del proceso a medida que se asignan y liberan variables.
3. **Heap (Montón)**: Es una región de memoria que se utiliza para la asignación dinámica de memoria durante la ejecución del proceso. El heap crece dinámicamente a medida que el proceso solicita más memoria para sus operaciones a través de llamadas como `malloc` en C o `new` en C++.
4. **Pila (Stack)**: Cada proceso tiene una pila que contiene el marco de la pila para las llamadas a funciones, incluyendo los parámetros de la función, las direcciones de retorno y las variables locales. La pila crece y se encoge dinámicamente con cada llamada y retorno de función.

3. Planificadores

FIFO: Entre cada cambio de proceso hay que considerar el tiempo de planificación del kernel el cual en ilustraciones suele ser omitido.

$$U = \frac{tiempo\_proceso}{tiempo\_proceso + overhead}$$

SPN: Shortest proces next. Es apropiativo (no lo suelta) favorece procesos pequeños y puede generar inanición para procesos largos pero depende de los procesos que lleguen al sistema.

En comparacion a FIFO que procesos largos pueden nunca dejar que se ejecute un sgte proceso

SRT: Version no apropiativa de SPN. El proceso puede ser sacado a la fuerza por el SO. "Voy a planificar siempre el mas corto pero considerando incluso a lo que le falta al que esta dentro de la cola"

Si llega un nuevo proceso con tiempo de servicio < al tiempo que le queda al proceso en ejecucion, este último es desapropiado y se planifica el proceso nuevo.

el tiempo de servicio en la practica no es conocido

¿como el so sabe que llego un proceso nuevo a la cola? SO entra al procesador y pregunta "llego alguien nuevo"?

HRRN:

$$R = \frac{\omega + s}{s} = \frac{\omega}{s} + 1$$

ω_i : es el tiempo de espera del proceso

Se elige al proceso con el R más grande

a mayor s su pendiente es más chica

Cada vez que haya que planificar voy a comparar todas las rectas y verificar quien tiene el mayor R
Favorece a procesos pequeños y favoresco a los que llevan más tiempo esperando y estas dos reglas en conjunto generar que el algoritmo no cause inanición

FEEDBACK: Utiliza distintos niveles (colas) todos los procesos son admitidos a la cola de listos. En todas las colas excepto la ultima se planifica en RR, la ultima en FIFO (no lo suelta a no ser que haga IO). Para así entrar apropiativamente al procesador.

Mientras hayan procesos en la cola de más alto nivel estaŕe planificando

¿Cómo es posible que el SO se de cuenta que ha llegado un proceso mas chico? *System Timer* -> interrupcion que ocurre repetidamente muchas veces. Se interrumpe al procesador y el so tiene la oportunidad en esos instantes de tiempo de entrar al procesador, echar un look y salir. Mas bien no entra el kernel

proceso ingresa al espacio del kernel

El cuanto de tiempo no llega a ser afectado por esta interrupcion debido a lo pequeña que llega a ser (aunque si uno quiere ser muy purista lo podria incluir como overhead)

O a este proceso se le acabo el quantum, lo saca y planifica a otro proceso

HZ: numero de ticks por segundo