Tema 0.1: Repaso de conceptos de SO

Soluciones a las cuestiones de repaso planteadas en el Tema 0.1

1. Interfaz hardware-Sistema Operativo:

   a. **Dibujar un esquema básico de los componentes de un computador y sus interconexiones: CPU, memoria y dispositivos.**
   Podemos tomar como referencia la arquitectura tipo PC, similar a la de la Figura 1.1 del Stallings.

   b. **Diferencia entre interrupciones y excepciones ¿Cómo se gestionan?**
   Interrupción: evento externo al procesador que necesita atención (se notifica activando el voltaje del pin INT del procesador). Son ajenas a la actividad en ejecución cuando se producen por lo que no debe modificarse el contexto actual de ejecución.
   Excepción: evento interno al procesador que necesita atención (normalmente debido a un error en la ejecución de la instrucción actualmente en ejecución). Como no son ajenas a lo que estaba haciendo la CPU, si será necesario tocar el contexto actual de ejecución.
   La gestión de interrupciones y excepciones es muy similar. Ver Apartado 1.4 del Stallings.

   c. **En una arquitectura de modo dual de funcionamiento ¿cómo obtenemos un servicio del SO?**
   La biblioteca del lenguaje en uso (por ejemplo, la biblioteca estándar de C) funciona como envoltura a la instrucción INT que genera una trampa (excepción) que conmuta la CPU de modo usuario a modo Kernel. Para ello, hay que salvar en la pila de usuario del proceso el contexto de ejecución del proceso. Cuando pasamos a modo kernel, la rutina (hay una única puerta de entrada) que atiende las llamadas al sistema determina la validez de los parámetros de la llamada y transfiere el control a la rutina kernel que sabe cómo atender la llamada. La ejecución del proceso llamador en modo kernel utiliza la pila kernel para apilar llamadas a función posteriores.
   Funcionalmente es como una llamada a función, pero a nivel de ejecución tiene una penalización mucho más alta.

   d. **Diferencia entre entradas/salidas programadas y e/s gestionadas por interrupción.**
   Stalling, Apartado 1.7, Epígrafes “E/S programada” y “E/S dirigida por interrupciones”. Diferencia en recuadro rojo de la Figura 1.

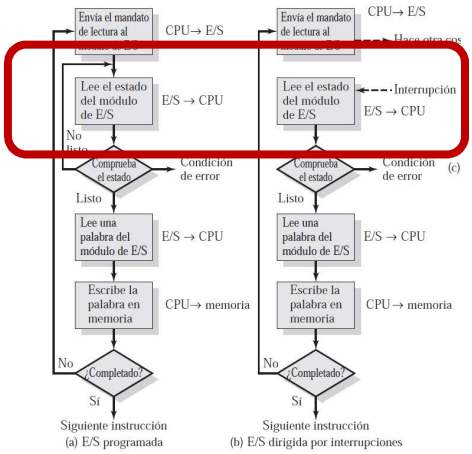

Figura 1.- (a) E/S programada y (b) E/S dirigida por interrupciones. Fuente: W. Stallings. *Sistemas Operativos. Aspectos internos y principios de diseño*. 5ª Edición, Pearson, 2005.

2. Procesos y sus estados:

a. ¿Cómo implementa un SO el concepto (abstracción) proceso?
Stallings, Apartado 3.1 (¿Qué es un proceso?)

b. ¿Qué es la imagen de un proceso?
Stallings, Tabla 3.4.

c. ¿A qué estado pasa un proceso que solicita un servicio del SO por el que posiblemente debe esperar? ¿cómo alcanza dicho estado?
En general, la forma voluntaria de ceder la CPU porque vamos a esperar un evento en invocar una llamada al sistema bloqueante (*sleep*, *wait*, *read*, etc.). Esto hace que el proceso pase al Estado bloqueado (el proceso no podrá tomar el control de la CPU de nuevo hasta que ocurra el evento por el que espera). Este estado lo alcanza invocando alguna de las llamadas antes indicadas que finalmente invocan a la función interna del sistema que bloquea al proceso.

d. ¿A qué estado(s) pasa el proceso que está actualmente ejecutándose cuando se produce una interrupción?
Cuando se produce una interrupción, el proceso actualmente en Ejecución puede bien seguir en el estado que estaba (no hay cambio de estado), bien puede pasar a Preparado. **Jamás pasará a Bloqueado** (él no ha solicitado un servicio del SO).
Revisar de la pregunta 1.b cómo se gestiona una interrupción.

e. Los sistemas operativos actuales se construyen para que los procesos cooperen en la multiprogramación. ¿Cómo se hace para que cooperen?
De forma voluntaria: cuando un proceso que solicita un servicio del sistema operativo, este suele ceder el control de la CPU (se bloquea) para permitir que otros se bloqueen.
De forma involuntaria: si el proceso no se bloque el sistema operativo le quitará el control cada cierto tiempo (tiempo-compartido) para que el resto pueda ejecutarse.