Sistemas Operativos Avanzados (MUII)
Ejercicio sobre procesos y sincronización (7-mayo-2021)

Se desea desarrollar un módulo de un sistema operativo de propósito general tipo UNIX que sirva para acceder a un dispositivo de E/S, que denominaremos "*myDevice"*.

Para que el usuario pueda interaccionar con este dispositivo, dentro del módulo se definen dos operaciones, que denominaremos *interact_input* e *interact_output*.

La primera operación programa el dispositivo de E/S para llevar a cabo una lectura de los datos asociados a dicho dispositivo. Cuando los datos están listos, el dispositivo genera una interrupción.

De manera análoga, la segunda operación programa el dispositivo de E/S para llevar a cabo una escritura de los datos asociados a dicho dispositivo. Se trata de una llamada bloqueante, que sólo se completa cuando los datos se hayan copiado al dispositivo. En esta circunstancia (cuando se haya completado la operación de escritura), el dispositivo también genera una interrupción.

Cuando se produzcan ambas interrupciones, todas las operaciones que se deben llevar a cabo por parte del sistema operativo son urgentes.

Por otro lado, el funcionamiento del dispositivo *"myDevice"* requiere llevar a cabo operaciones de *backup*, no programadas por el usuario, sino que se llevan a cabo de forma periódica dentro del sistema operativo. La operación se denomina *backup_device* y es una operación bloqueante, donde se hace una copia de todos los datos en otro dispositivo de respaldo.

Finalmente, en relación al dispositivo *"myDevice"*, existe una excepción que puede producirse cuando el usuario intenta escribir un conjunto de datos superior a una cantidad *(#MAXSIZE*), que denominaremos excepción de tamaño máximo.

Se pide:

1. Asociar cada una de las actividades (*interact_input*, *interact_output*, interrupciones asociadas a ambas operaciones, *backup_device* y excepción de tamaño máximo) a un contexto de ejecución (rutina de llamada al sistema, rutina de excepción, rutina de interrupción, rutina de interrupción software de sistema, rutina de interrupción software de proceso, proceso de núcleo o actividad en modo usuario), justificando la elección.
2. Indicar cuál de las siguientes actividades necesitará acceder a las funciones *Bloquear()* y/o *Desbloquear()* que se han visto en clase.
3. Escribir en pseudocódigo las operaciones *interact_input*, *interact_output*, *backup_device* y las rutinas asociadas a las interrupciones, haciendo uso de las funciones *Bloquear()* y *Desbloquear()*. Como guía, el nivel de detalle del pseudocódigo sería similar al de la operación *fallo_página()*, en la página 45 de las transparencias utilizadas en clase.
4. Modificar el apartado anterior para el caso en el que la operación *interact_output* no sea bloqueante y que haya una operación nueva, que se denomine *test_output*, que, una vez que un proceso haya programado una

operación de escritura no bloqueante, deje al proceso bloqueado hasta que la operación de escritura se complete.

Se pretende que las actividades asociadas al módulo se puedan utilizar en un sistema multiprocesador.

Se pide:

5. Modificar las implementaciones en pseudocódigo del apartado 3, utilizando *spinlocks* donde considere, para evitar problemas de sincronización en dicho sistema multiprocesador.

Utilizamos el módulo y el sistema operativo en general en un multiprocesador, compuesto por dos procesadores con núcleos expulsivos.

6. Dibujar la traza de ejecución de ambos procesadores para los procesos A, B, C y D, para el siguiente escenario, indicando si los procesos están en modo usuario o modo sistema, si hay cambio de contexto voluntario o involuntario y las diferentes rutinas de eventos que tienen lugar (incluyendo rutinas de interrupción software de sistema y de proceso, si fuera necesario), teniendo en cuenta que:

- La rutina de interrupción software de proceso de planificación se denomina rutSProcPlan.
- La rutina de tratamiento de la llamada a sistema XXX se denomina rutXXX.
- La rutina de interrupción entre procesadores se denomina rutIPI.
- Sólo están en marcha los procesos de usuario A, B, C y D. La planificación de los procesos se hace por prioridad y el orden de prioridad, de mayor a menor, es: A, B, C y D.
- El proceso A solicita la operación *interact_input* y se bloquea. Eventualmente vendrá una interrupción asociada al dispositivo *"myDevice",* que despertará al proceso A, poniéndolo en estado listo para ejecutar, y planificándolo, cuando se pueda. Dicho proceso continuará su ejecución en modo usuario.
- El proceso B realiza una llamada al sistema *sem_wait()* sobre un semáforo que está cerrado, por lo que se quedará bloqueado. Cuando se desbloquee, el proceso realizará una llamada al sistema *exit()* y terminará.
- El proceso C es nuevo. El proceso C solicita la operación *interact_output*. Dentro de la ejecución de la rutina de dicha operación se produce una excepción de tamaño máximo, que provoca que el proceso C finalice.
- El proceso D es nuevo. El proceso D realiza una llamada getpid(). A continuación, pasa a ejecutar en modo usuario durante un determinado intervalo de tiempo. Posteriormente, realiza una llamada a sistema *sem_post()*, abriendo el semáforo por el que estaba bloqueado B. A continuación, el proceso continuará su ejecución en modo usuario.

7. Si en medio de la traza del apartado anterior, se lleva a cabo una operación de *backup*, ¿cómo se vería modificada dicha traza? Dibujar la misma, incluyendo los procesos que sean necesarios para llevar a cabo dicha operación.

Plazo y modo de entrega

El plazo de entrega del trabajo es el **8 de junio de 2021**.

La entrega se realiza en *triqui* ejecutando el mandato:

entrega.soa procesos_sincro.2021

Este mandato realizará la recolección del directorio *~/DATSI/SOA/procesos_sincro.2021* de los ficheros *autor.txt*, con los datos del alumno, y *memoria.pdf*, que debe incluir la solución del ejercicio.

Para cualquier duda sobre el ejercicio, consulte a María S. Pérez Hernández (*mperez@fi.upm.es*).