LABORATORI DE SISTEMES OPERATIUS

ETAPA 2: - Multiplexar (4 setmanes)

Com a punt de partida d'aquesta etapa es farà servir el codi base proporcionat a la web de l'assignatura (http://studies.ac.upc.es/FIB/SO/Welcome.html). Aquest codi cubreix la primera etapa de la realització d'ONION, fent el pas de paràmetres de les crides al sistema mitjançant registres.

A) Multiplexar 1 procés

L'objectiu d'aquesta etapa és implementar la rutina `multiplexar()` i crear les estructures de dades necessàries per gestionar l'execució d'un procés. Per això crearem i omplirem un PCB amb la informació del procés inicial, i l'usarem a les rutines `trap()` i `multiplexar()` per salvar i restaurar l'estat del procés. Igual que en l'etapa anterior, el programa principal, després d'haver inicialitzat les variables i d'haver programat adequadament el vector d'interrupcions, executarà la rutina corresponent al procés d'usuari que hi ha al fitxer `fluxes.c`.

Aquesta tasca es pot implementar en tres parts:

- Modificar la rutina `restaurar()` per a que agafi com a paràmetre l'adreça del cim del contexte que ha de restaurar. Cal comprovar a la rutina `trap()` que la rutina `restaurar()` funciona correctament .
- Crear el PCB del procés inicial i usar-lo per guardar l'estat a la rutina `salvar()`, i per obtenir l'estat a la rutina `restaurar()`.
- Implementar la rutina de `multiplexar()`, la qual ha de salvar l'estat del procés en execució, comptar els tics, avisar al controlador d'interrupcions mitjançant una crida a la funció `eoi()`, i restaurar l'estat d'aquest mateix procés.
  Aquesta rutina s'ha de connectar a la interrupció de rellotge (la interrupció `0x08`) usant la rutina `vector_int`. El rellotge s'ha de programar per a que generi 200 interrupcions per segon cridant a la rutina `timer` amb el valor del paràmetre a *6000*.

B) Multiplexar 2 processos: *crear_pcb*

L'objectiu és multiplexar dos fluxos de manera que cadascun s'executi durant N interrupcions de rellotge (*quantum*). Això serà la primera implementació d'un algorisme de **Round Robin apropiatiu** que més endavant s'ampliarà a múltiples fluxos. El codi dels dos fluxos serà igualment un bucle infinit que escriurà un caracter en pantalla utilitzant la rutina `escriure_pantalla` de la llibreria `onionlib.lib`.

Per a crear el segon fluxe heu d'implementar la crida al sistema `crear_pcb`, la qual inicialitza les estructures de dades necessàries per a crear un nou fluxe, donada l'adreça del codi, i la prioritat i el quantum del fluxe. Aquesta rutina té la següent descripció:

```
SINTAXI:      int crear_pcb(codi, prioritat, quantum)
              void (*codi)();
              int prioritat, quantum;

DESCRIPCIÓ:   Aquesta rutina crea un nou procés amb la prioritat i quantum donats.
```

Inicia el seu PCB, li assigna una pila i la inicialitza per a que pugui començar el flux de forma correcta a partir de l'adreça `codi`.

`RETORN:` Número identificador del procés (*pid*) creat o -1 si hi ha error o bé no queden PCBs lliures.

Com que no es disposa de gestió de memòria dinàmica, la memòria necessària per a gestionar els processos (els PCBs i la pila per cada procés) estarà declarada de forma estàtica amb dades globals del sistema. El nombre de processos estarà limitat a una constant (per exemple 20). Per tant, cal definir al programa principal les següents estructures de dades:

```
pcb_struct PCBs[NPROCS];
int pila[NPROCS][STACK_SIZE];
pcb_struct *RUN, *READY;
```

L'apuntador `RUN` apunta al PCB que està en execució, inicialment el procés inicial. L'apuntador `READY` apunta al procés que està en espera, inicialment a `NUL`. Quan es crea el nou procés, l'apuntador `READY` s'actualitza d'acord amb la política d'apropiació implementada (immediata o diferida).

En concret, les tasques que ha de fer la rutina `crear_pcb_nuc` són:

- Comprovar els paràmetres.
- Trobar un PCB lliure i omplir-lo.
- Omplir la pila del procés.
- Posar-lo a `READY` o a `RUN`, segons les característiques del procés i de l'algorisme de planificació.

A més, cal modificar la rutina `multiplexar()` per tal que faci la planificació dels processos segons l'algorisme *Round Robin* apropiatiu, tenint en compte el *quantum* del procés en execució. En concret, la rutina `multiplexar()` ha de fer les següents tasques:

- Salvar els registres generals i de segment del processador a la pila del procés i canviar a l'espai d'adreces de sistema (modificar `DS`, `ES` amb el valor de `DGROUP`).
- Actualitzar els tics d'execució del procés en execució i si cal planificar aleshores canviar el contexte per el del nou flux.
- Avisar al controlador que s'ha acabat la interrupció: crida a la rutina `eoi()`.
- Restaurar el context del nou procés (i recuperar `DS`, `ES`).

C) Multiplexar N processos

La darrera part d'aquesta etapa consisteix en la multiplexació de N processos. Heu de canviar la variable `READY` i convertir-la en una cua de processos preparats per executar-se (cua `READY`). A més, heu de modificar la rutina `multiplexar()` per a que faci la gestió adeqüada de la planificació dels processos.

Ara heu d'introduïr el concepte de *prioritat* del procés. D'aquesta manera veureu que <u>només s'han d'executar els fluxos de prioritat més gran</u> i que el sistema aplica l'algorisme de *Round Robin* dintre de la mateixa prioritat. **La política d'apropiació serà diferida**. Recordeu que una de les estructures que heu d'inicialitzar ha de correspondre al procés `nul`, consistent en un bucle infinit, que tindrà la

prioritat més baixa possible i que entrarà en execució solament quan el processador no tingui cap altre procés per triar. De fet, com en molts sistemes operatius, el procés `nul` serà el mateix procés inicial després d'haver executat el codi d'inicialització.

Useu les rutines de la llibreria `solib.lib` per a la gestió eficient de cues. Per això, fixeu-vos que cal afegir un camp a l'estructura dels PCB per permetre encuar els PCB a la cua de `READY`. Per a més detalls, consulteu el document *Llibreria de suport al desenvolupament de la pràctica d´ONION* que podeu trobar al web de l'assignatura.

APENDIX: Com tornar al DOS?

El nucli del sistema operatiu ONION que implementeu substitueix al sistema operatiu MS-DOS. De tota manera, a l'hora de realitzar les proves de funcionament pot resultar convenient restaurar el MS-DOS sense haver de fer *reset* a la màquina. Per a fer-ho, podeu utilitzar les rutines `savedos()` i `restoredos()` que es troben a la llibreria `savedos.lib`. La rutina `savedos()` salva els valors originals de les entrades 0x08, 0x09 i 0x40 del vector d'interrupcions (les que utilitza ONION), així com la pila del MS-DOS i els valors d'alguns registres importants. Per a que funcioni, cal que sigui la primera crida que faci la rutina `main()` del vostre codi. Un cop fet això, quan es cridi la rutina `restoredos()`, es restauraran els valors del MS-DOS i tornarà a aparèixer el prompt.

ATENCIÓ: tingueu en compte que si per error o intencionadament heu modificat la zona de memòria utilitzada pel MS-DOS, pot ser que el sistema després de cridar a `restoredos()` sigui inestable o, senzillament, no es pugui retornar al MS-DOS. Així mateix, si utilitzeu `restoredos()` dins d'una rutina de servei a una interrupció hardware (teclat o rellotge), recordeu-vos de fer abans el corresponent `eoi()`. Altrament les interrupcions quedarien bloquejades i el sistema no funcionaria.