LABORATORI DE SISTEMES OPERATIUS

Jordi Garcia

Pedro Marcuello

Àlex Ramírez

Toni Cortés

PRÀCTICA D'ONION (Primera etapa):

- ## Objectiu

L’objectiu de la pràctica d’ONION és dissenyar i implementar una capa de software que es recolzi directament sobre el hardware del PC i que ofereixi a l’usuari els serveis de més baix nivell del sistema operatiu, concretament:

- Gestió de crides a sistema.
- Gestió de temps.
- Gestió de processos: entorns i fluxos d’execució.
- Sincronització i comunicació entre processos.
- Entrada/Sortida sobre dispositius.

El resultat que es vol obtenir és l'implementació d'algunes de les funcionalitats del nivell nucli d’ONION. L'accés als serveis del sistema es farà a través de la llibreria del sistema, la qual implementarà el mecanisme d’entrada al sistema mitjançant un *trap*.

ETAPA 1: - Crides al sistema (3 classes)

Primera crida al sistema: *escriure_pantalla*

L'objectiu de la primera part de la pràctica és crear un fluxe d'execució (en el nostre cas una rutina) que escrigui per pantalla a través d'una crida al sistema. El codi del fluxe serà senzillament un bucle infinit que escriurà un caracter per pantalla utilitzant la crida `escriure_pantalla` de la llibreria `onionlib.lib` i les rutines internes del vostre sistema operatiu. La descripció d'aquesta crida és:

```
SINTAXI:      int escriure_pantalla(fil, col, car, atr)
              int fil, col;
              char car, atr;
```

`DESCRIPCIÓ:` Aquesta rutina escriu per pantalla el caracter `car` amb el atribut `atr`, a la posició indicada per les coordenades `fil` (corresponent a la fila) i `col` (corresponent a la columna).

`RETORN:` Retorna un zero si no hi ha hagut error d'escriptura o un -1 altrament.

Aquest serà el primer servei que oferirà el sistema operatiu que esteu escrivint. Per a implementar aquest servei, a part d'usar els fitxers ja existents `salres.asm`, `new_maq.asm` i `solib.lib` que us haureu copiat al vostre disquet de treball, heu de crear els següents fitxers:

- `boot.c`
- `fluxes.c`
- `onionlib.asm`
- `sistema.c`
- `bfs.c`
- `nucli.c`

Per facilitar les compilacions, us aconsellem que adapteu el fitxer `makefile` de la pràctica 0.

El fitxer `boot.c` contindrà la declaració de les variables globals del sistema i el programa principal que, després d'inicialitzar les variables i de programar adequadament el vector d'interrupcions, posa en marxa el sistema. El fitxer `fluxes.c` conté el procés d'usuari que s'ha d'anar executant. Inicialment aquest procés executa un bucle infinit que escriu un caracter per pantalla utilitzant la crida al sistema `escriure_pantalla`.

El fitxer `onionlib.asm` conté el codi que efectivament fa la crida al sistema `escriure_pantalla` mitjançant un *trap*. Com sabeu, el processador amb què treballeu té la instrucció `INT <num>` per a generar *traps*, on `<num>` és un enter dins el rang [0..255] que identifica el servei que es vol sol.licitar. Nosaltres triarem un únic nombre de servei, el `0x40`, per a que el nostre sistema operatiu atengui totes les peticions. A partir del fitxer `onionlib.asm` crearem la llibreria de sistema `onionlib.lib`.

El fitxer `sistema.c` conté la rutina `trap()` que analitza quin servei és sol.licitat per l'usuari. Per connectar aquesta rutina amb la interrupció `0x40` usareu la funció `vector_int()` que teniu a la llibreria `solib.lib`. La capçalera d'aquesta rutina és:

```
void vector_int(int num, void (*nomrut)());
```

La rutina `trap()` inicialment ha de salvar el contexte del procés sol.licitant usant la rutina `salvar()` que teniu al fitxer `salres.asm`. A continuació ha d'avaluar quin servei sol.licita l'usuari (de moment només s'ofereix la crida `escriure_pantalla`) i executarà la funció corresponent del nivell sistema `escriure_pantalla_sis()`, també implementada dins d'aquest fitxer. I finalment ha de restaurar el contexte del procés usant la rutina `restaurar()` que hi ha al fitxer `salres.asm`.

La rutina `escriure_pantalla_sis` accedeix als paràmetres i crida a la rutina de nivell BFS `escriure_pantalla_bfs()` implementada en el fitxer `bfs.c`. En el nivell BFS simplement es crida a la rutina de nivell nucli `escriure_pantalla_nuc()`, implementada en el fitxer `nucli.c`. La rutina de nivell sistema passa els paràmetres a la de nivell BFS, i la de nivell BFS els passa a la de nivell nucli. Així doncs, la rutina `escriure_pantalla_nuc` comprova els paràmetres (és a dir, si les coordenades fila i columna a les que es vol escriure estan dins del requadre de la pantalla) i, si no hi ha cap error, accedeix efectivament a la pantalla usant la rutina `escriure_pantalla_hw()` que teniu disponible a la llibreria `solib.lib`. A més, la rutina en el nivell de nucli és la encarregada de retornar el resultat.

Nota: Cal inicialitzar en el programa principal la variable global `pantalla` de la següent manera:

```
pantalla = ini_pantalla();
```

per a que funcioni correctament la funció `escriure_pantalla_hw`.