Control de Laboratori (grup 40)

29 d'Abril

Contesteu al mateix full

Nom i Cognoms:

---

1. (4 punts) Seleccioneu la resposta (una només) que considereu correcta en cadascun dels apartats. Cada resposta correcta val 0,5 punts. Cada resposta incorrecta resta 0,25.
   (a) La rutina *multiplexar* està situada a la capa...
      - Sistema
      - BFS
      - Nucli
      - Qualsevol de les anteriors

   (b) El procés nul ha de ser...
      - Un procés infinit amb la prioritat que volguem
      - Un procés finit amb prioritat 0, que és la mínima prioritat possible a Onion
      - Un procés infinit amb prioritat 1, que és la mínima prioritat possible a Onion
      - Cap de les anteriors

   (c) En què consisteix la inicialització del procés inicial?
      - Cal assignar-li un PCB i reservar-li espai per la pila, però no cal crear-li un context
      - Cal assignar-li un PCB, però no cal reservar-li espai per la pila ni crear-li un context
      - Cal assignar-li un PCB i crear-li un context, però no cal reservar-li espai per la pila
      - Cal crear-li un context i reservar-li espai per la pila, però no cal assignar-li un PCB

   (d) Podem crear un procés amb prioritat 0 cridant a *crear_pcb*?
      - No, l'únic procés amb prioritat 0 és el procés nul
      - No, la prioritat 0 no és una prioritat vàlida, ja que a Onion les prioritats comencen amb la 1
      - Sí, ja que la prioritat 0 és una prioritat vàlida
      - Cap de les anteriors

   (e) Cada cop que s'efectua un canvi de context...
      - El procés que es posa a RUN tindrà un quantum equivalent al temps que li restés del seu quantum anterior. Aquest temps està guardat al seu PCB
      - El procés que es posa a RUN tindrà un quantum sencer
      - El procés que es posa a RUN tindrà un quantum equivalent al valor que hagi quedat a la variable *tics*
      - Cap de les anteriors

(f) On està definida l'estructura PCB?

- Al boot.c, perquè és on es troba tota la inicialització d'estructures
- A sistema.h, ja que la gestió de processos es fa a nivell de Sistema
- A nucli.h, ja que la gestió de processos es fa a nivell de Nucli
- En un include propi (per exemple, pcb.h)

(g) S'ha de fer alguna cosa amb les interrupcions al boot.c?

- Cal garantir que es reprograma el vector d'interrupcions amb les interrupcions inhibides i que les variables del sistema ja estan inicialitzades quan ja és possible que arribin interrupcions
- Únicamemt cal garantir que es reprograma el vector d'interrupcions amb les interrupcions inhibides
- No cal inhibir les interrupcions en cap moment
- Cap de les anteriors

(h) Com s'implementa en la pràctica d'Onion la part de les prioritats apropiatives de la nostra política?

- En la rutina *multiplexar*, si la prioritat del primer procés de la cua de Ready és més gran que la prioritat del procés que està corrent, es fa un canvi de context
- En la rutina *multiplexar*, si la prioritat del primer procés de la cua de Ready és més gran o igual que la prioritat del procés que està corrent, es fa un canvi de context
- En la rutina *crear_pcb_nuc*, si la prioritat del procés que s'està creant és més gran que la prioritat del procés que està corrent, es fa un canvi de context
- En la rutina *crear_pcb_nuc*, si la prioritat del procés que s'està creant és més gran o igual que la prioritat del procés que està corrent, es fa un canvi de context

2. (1,5 punts) Explica com s'ha aconseguit a la pràctica que es considerés la diferenciació entre l'espai d'adreces del sistema i l'espai d'adreces de l'usuari. Detalla en quins punts es realitza el canvi entre els diferents espais d'adreces i com es fa

3. (1,5 punts) Comenta breument com s'aconsegueix que les rutines de gestió de cues que us proporcionem (preparades per treballar amb elements del tipus *struct item*), puguin treballar amb PCBs. Raona per què funciona aquesta solució

4. (3 punts) Completa el codi de la rutina *multiplexar* tal com l'has implementada a la pràctica d'Onion

```
void multiplexar() {
    run->context = salvar();
    run->tcpu++;
    tics--;









    restaurar(run->context);
}
```