Control de Laboratori (grup 40)

3 de Juny

Contesteu al mateix full

Nom i Cognoms:

---

1. (2 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) En què consisteix la destrucció d'un procés si aquest es troba retardat?

   - Cal alliberar la seva pila i el seu PCB i eliminar-lo de la cua de processos retardats en la qual es troba bloquejat
   - Cal alliberar la seva pila i el seu PCB i indicar que ja no hi ha cap procés retardat
   - Només cal alliberar la seva pila i el seu PCB
   - No es pot destruir un procés si està retardat

   (b) A quin valor s'han d'inicialitzar els registres CS:IP del context de kernel d'un procés quan aquest és creat?

   - Aquests registres no cal que s'inicialitzin
   - Han de contenir l'adreça de la rutina que ha d'executar aquest procés
   - Han de contenir l'adreça de la rutina *restaurar*
   - Cap de les anteriors

   (c) Quina de les següents afirmacions sobre els semàfors és verdadera?

   - Els semàfors de sistema només es poden utilitzar amb les rutines *kinit_sem*, *kwait*, *ksignal* i *kdestruir_sem*
   - Les rutines *wait* i *kwait* són crides a sistema
   - La funcionalitat dels semàfors d'usuari s'implementa a la capa Sistema, però pels semàfors de sistema es fa a la capa Nucli
   - Totes les anteriors

   (d) Quines funcions s'executaran com a resultat d'una invocació de la crida a sistema *wait*?

   - *wait_sis*, *wait_bfs* i *wait_nuc* que executarà el servei demanat
   - Només *wait_sis*, perquè els semàfors estan implementats a la capa Sistema
   - *wait_sis*, *wait_bfs*, *wait_nuc* i *kwait*, ja que la funcionalitat dels semàfors la proporcionen les rutines de semàfors del sistema
   - *wait_sis* i *kwait*, perquè les rutines de semàfors del sistema també estan situades a la capa Sistema

2. (1,5 punts) Omple aquesta taula amb la informació referent als semàfors que utilitzen la rutina *llegir_teclat_sis* i el gestor de teclat per sincronitzar-se (Nota. 1. Que la taula tingui 4 files no implica necessàriament que s'hagi d'omplir sencera. 2. A la columna *On es guarda?* no és correcte posar a la taula de semàfors)

Nom semàfor	Quants n'hi ha?	On es guarda?	Rutina que l'inicialitza	Rutina que el destrueix

3. (1,5 punts) Justifica per què el gestor de teclat ha d'inhibir explícitament les interrupcions abans de cridar a les rutines *kwait*, *ksignal* i *llegir_teclat_bfs*, i això no és necessari quan aquestes rutines són cridades des d'altres punts

4. (1 punts) Justifica per què s'han de fer servir la variable *eoi_pendent* i la rutina *final_rsi*, enlloc de cridar la rutina *eoi* directament

5. (4 punts) Indica els **8 errors** existents en les següents parts del codi de la pràctica, ja sigui eliminant i/o afegint i/o movent i/o modificant codi. No hi ha cap error de tipus sintàctic

Nucli.c

```
void modif_proc_nuc (int id_proc, struct info_proc *p_info)
{
  if (id_proc == 0)
    RUN->ctx->ax = -1;
  else {
    struct PCB *pcb;

    if ((pcb = &taula_pcbs[pid]) == NULL)
      RUN->ctx->ax = -1;
    else {
      pcb->quantum = p_info->quantum;
      pcb->t_cpu = p_info->t_cpu;
      pcb->prioritat = p_info->prioritat;
      switch(pcb->estat) {
        case PCB_RUN:
          if (pcb->prioritat < ((struct PCB *)cap(&READY))->prioritat)
            canvi_ctx((struct PCB *)primer(&READY),1);
          break;
        case PCB_READY:
          extirpar(&READY,(struct item *)pcb);
          planificacio_wakeup(pcb);
          break;
       }
       RUN->ctx->ax = 0;
    }
  }
}

void rsi_teclat ()
{
  int c;

  RUN->ctx = salvar();
  eoi_pendent = 1;
  c = llegir_teclat_hw();
  buffer_escriure(&buffer_teclat,c);
  if (!bloq_teclat)
    planificacio_wakeup(bloq_teclat);
  final_rsi();
  restaurar(RUN->ctx);
}
```

Sistema.c

```
void signal_sis(int n_sem)
{
  RUN->ctx->ax = ksignal(n_sem);
}

int ksignal(int n_sem)
{
  if (n_sem >= MAX_SEM || n_sem < 0 || !taula_semafors[n_sem].init)
    return -1;
  else {
    if(buida(&taula_semafors[n_sem].queue))
      taula_semafors[n_sem].count++;
    else {
      planificacio_wakeup_bfs((struct PCB *)primer(&taula_semafors[n_sem].queue));
    }
    return 0;
  }
}
```