Informazio - Sistemen Arkitektura

Telekomunikazio Teknologiaren Ingeniaritzako Gradua. 3. maila.

2022ko apirilaren 4a Iraupen osoa: 55’

NAN:_____ Izen-abizenak:_____

1.- Ondoko kodeak s semaforo bat inplementatzen du, hasieran s_lock=false den aldagai partekatu batez eta TSL agindua erabiltzen duen test-and-set funtzioaren bidez, atal kritikorako sarbidea kontrolatuz. Zuzena al da down & up eragiketen funtzionamendua? Komentatu kodea lerro-zenbakiei erreferentzia eginez. (1,5 puntu)

```
01 wait(s){   //down(s)
02  while (test-and-set(&s_lock));
03  s = s - 1;
04  if (s < 0){
05    s_lock = false;
06    wait_prozesua_blokeatu ();
07  }
08  else
09    s_lock = false;
10 }
```

```
11 signal(s){   //up(s)
12  while (test-and-set(&s_lock));
13  s = s + 1;
14  if ( s <= 0){
15    wait_prozesua_desblokeatu ();
16  }
17  s_lock = false;
18}
```

wait(s)/down(s) funtzioaren if-else agindua ondoan dagoen kodeaz sinplifikatzen badugu, ba al dago hobekuntzarik? Erabili lerro-zenbakiak zure azalpenean. (0,5 puntu)

```
21 wait(s){   //down(s)
22  while (test-and-set(&s_lock));
23  s = s - 1;
24  s_lock = false;
25  if (s < 0){
26    bloquear_proceso_wait();
27  }
28 }
```

2.a) Memoria antolatzeko segmentazio hutsa (orrikapenik gabea) erabiltzen badugu, zeintzuk izango dira helbibide logiko baten eremuak? Nola itzultzen da helbide logikotik helbide fisikora? (1 puntu)

2.b) Nolakoak dira helbide logiko baten eremuak segmentazio orrikatua erabiltzen bada, eta nola itzultzen da kasu honetan helbide logikotik fisikora? (1 puntu)

3.- 4Gbyteko pendrive bat 4Kbyteko blokeetan formateatzen da. Bi kasuetan (a eta b) blokeen indizeak 32 bitekoak direla kontuan hartuta, kalkulatu fitxategi baten tamaina maximoa blokeetan:

a) Pendrivea UNIX motako fitxategi-sistema batekin formateatzen bada, eta i-nodoetan 10 bloke-helbide zuzen, zeharkako erakusleen bloke sinple baten indizea eta zeharkako bikoitz baten indizea badaude. (1 puntu)

b) Windows/MS-DOS fitxategi-sistema baterako FAT batez formateatzen bada. Kasu honetan, kalkulatu gainera FAT taulak zenbat RAM beharko duen. (1 puntu)

4.- **A** eta **S** izeneko aita-seme prozesuek, **File** izeneko fitxategiko 350. bytean partekatzen dute irakurtzeko/idazteko kokapena. File fitxategiaren i-nodo zenbakia 30 da. **I** izeneko hirugarren prozesu independente batek ere fitxategi bera 200. kokapenean irakurtzeko irekita dauka. Adierazi grafikoki behar diren datu-egituren taulak eta nola erlazionatu behar diren, hiru prozesuok fitxategi berean kokaleku propioak (partekatuak edo independenteak) izateko gauza izan daitezen. (2 puntu)

5.- Erlojuaren kontrolatzailearen lau eginkizun nagusiak zeintzuk diren labur azaldu banan-banan. Hurrengo kodea Minix-en erloju-atazari dagokio. Identifikatu bertan lau eginkizunok, eta euren hasiera- eta amaiera-tartea adierazi kode-lerroen zenbakiak erabiliz. (2 puntu)

```
3199 PRIVATE do_clocktick()
3200 {
3201 /* This routine called on every clock tick. */
3202
3203         register struct proc *rp;
3204         register int t, proc_nr;
3205
3206         /* To guard against race conditions, first copy 'lost_ticks' to a local
3207          * variable, add this to 'realtime', and then subtract it from 'lost_ticks'.
3208          */
3209         t = lost_ticks;                                 /* 'lost_ticks' counts missed interrupts */
3210         realtime += t + 1;                              /* update the time of day */
3211         lost_ticks -= t;                                /* these interrupts are no longer missed */
3212
3213         if (next_alarm <= realtime) {
3214                 /* An alarm may have gone off, but proc may have exited, so check. */
3215                 next_alarm = MAX_P_LONG;        /* start computing next alarm */
3216                 for (rp = &proc[0]; rp < &proc[NR_TASKS+NR_PROCS]; rp++) {
3217                         if (rp->p_alarm != (real_time) 0) {
3218                                 /* See if this alarm time has been reached. */
3219                                 if (rp->p_alarm <= realtime) {
3220                                         /* A timer has gone off. If it is a user proc,
3221                                          * send it a signal. If it is a task, call the
3222                                          * function previously specified by the task.
3223                                          */
3224                                         proc_nr = rp - proc - NR_TASKS;
3225                                         if (proc_nr >= 0)
3226                                                 cause_sig(proc_nr, SIGALRM);
3227                                         else
3228                                                 (*watch_dog[-proc_nr])();
3229                                         rp->p_alarm = 0;
3230                                 }
3231
3232                                 /* Work on determining which alarm is next. */
3233                                 if (rp->p_alarm != 0 && rp->p_alarm < next_alarm)
3234                                         next_alarm = rp->p_alarm;
3235                         }
3236                 }
3237         }
3238
3239         accounting();                           /* keep track of who is using the cpu */
3240
3241         /* If a user process has been running too long, pick another one. */
3242         if (--sched_ticks == 0) {
3243                 if (bill_ptr == prev_ptr) sched();      /* process has run too long */
3244                 sched_ticks = SCHED_RATE;       /* reset quantum */
3245                 prev_ptr = bill_ptr;                    /* new previous process */
3246         }
3247
3248 }
```