Pregunta 1

Parte a.- La siguiente es la implementación concreta de la estrategia del *working set* para un núcleo clásico de Unix en una máquina monocore:

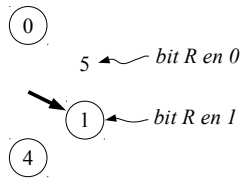
```
/* Invocada para recalcular el working set */
void computeWS(Process *p) {
  int *ptab= p->pageTable;
  for (i= p->firstPage;
       i<p->lastPage; i++) {
    if (bitR(qtab[i])) {
      setBitWS(&qtab[i], 1);
      setBitR(&qtab[i], 0);
    }
    else {
      setBitWS(&qtab[i], 0);
    }
  }
}

/* Invocada cuando ocurre un page-fault */
void pagefault(int page) {
  Process *p= current_process;
  int *ptab= p->pageTable;
  if (bitS(ptab[page])) {
    /* la página se encuentra en disco */
    send(ws, &page);
  }
  else {
    /* dirección inválida */
    segfault(page);
  }
}
```

```
void WSStrategy() { /* proceso daemon */
  Process *p;
  int page= *(int*)receive(&p);
  /* p gatilla el page-fault de la página page */
  Iterator *it= processIterator();
  for (;;) {
    Process *q= nextProcess(it);
    int *qtab= q->pageTable;
    for (i= q->firstPage;
         i<q->lastPage; i++) {
      if (bitV(qtab[i]) &&
          !bitWS(qtab[i]) &&
          !bitR(qtab[i])) {
        /* se reemplaza página i de q */
        savePage(q, i);
        setBitV(&qtab[i], 0);
        setBitS(&qtab[i], 1);
        int *ptab= p->pageTable;
        setRealPage(&ptab[page],
                    realPage(qtab[i]));
        setBitV(&ptab[page], 1);
        loadPage(p, page);
        setBitS(&ptab[page], 0);
        purgeTlb(); /* invalida la TLB */
        purgeL1(); /* invalida cache L1 */
        reply(p);
        page= *(int*)receive(&p);
    } }
    if (!hasNext(it))
      reset(it);
} }
```

Suponga que la MMU implementa el bit D (*dirty*). Modifique la función *WSStrategy* para que haga un uso eficiente del bit D.

Parte b.- Compare las dos estrategias de paginamiento en demanda vistas en el curso desde el punto de vista de (i) sobrecosto en tiempo de ejecución cuando la memoria física sobra, (ii) page-faults cuando hay penuria de memoria pero hay un solo proceso en ejecución, (iii) page-faults cuando hay penuria de memoria y hay muchos procesos en ejecución.

Parte c.- Considere un sistema Unix que implementa la estrategia del reloj. El sistema posee 6 páginas reales disponibles y corre un solo proceso. La figura indica el estado inicial de la memoria, mostrando las páginas residentes en memoria, la posición del cursor y el valor del bit R.



Dibuje los estados por los que pasa la memoria para la siguiente traza de accesos a páginas virtuales: 8 4 6 1 0 2.

Pregunta 2

Parte I. (3 puntos) Se necesita programar un driver para el dispositivo */dev/lectescr*. Este dispositivo actúa como el dispositivo */dev/mem* visto en clases, excepto que un proceso escritor necesita exclusividad mientras mantenga el dispositivo abierto (no admite simultaneidad con otros escritores o lectores). Se deben admitir múltiples procesos lectores con el dispositivo abierto simultáneamente. Si un proceso abre el dispositivo y se va a violar una de estas restricciones, el proceso debe esperar en el *open* hasta que no se viole ninguna restricción.

Reprograme las funciones correspondientes a *open* y *release* del dispositivo */dev/mem* para adaptarlas a los requerimientos de */dev/lectescr*. Declare las variables globales que necesite señalando cómo se deben inicializar en la función *lectescr_init* (pero no programe completamente esta última).

Recuerde que Ud. puede determinar si el dispositivo fue abierto en modo escritura usando la condición *filp->f_mode & FMODE_WRITE*. Puede usar los semáforos del núcleo o los monitores de la tarea 3. Su solución puede sufrir de hambruna. No se preocupe por las interrupciones mientras espera que ocurra algún evento.

Parte II. (1,5 puntos) Considere un sistema operativo que implementa la estrategia del working set. Para el área de paginamiento se usa un solo disco que tiene una tasa de transferencia de 100 MB/segundo y un tiempo de acceso de 5 milisegundos.

Estime (a) el máximo número de *page-faults* por segundo que se puede atender cuando hay penuria de memoria; y (b) cuanto tiempo tomaría llevar a disco (swap) un proceso cuyas páginas residentes en memoria totalizan 250 MB.

Parte III. (1,5 puntos) Haga un diagrama que muestre el inodo, los bloques de indirección y los bloques de datos de un archivo de 4152 KB (1038 * 4KB), ubicado en una partición Unix con bloques de 4 KB.