Pregunta 1

I. La siguiente es la implementación concreta de la estrategia del reloj:

```
void pagefault(int page) {
  Process *p= current_process;
  int *ptab= p->pageTable;
  if (bitS(ptab[page])) /* residente en disco */
    send(clock, &page);
  else
    segfault(page);
}
void clockStrategy() { /* un proceso daemon */
  Process *p; /* proceso que gatilla el page fault */
  int page= *(int*)receive(&p);
  Iterator *it= processIterator();
  for (;;) {
    Process *q= nextProcess(it);
    int *qtab= q->pageTable;
    for (i=q->firstPage; i<q->lastPage; i++) {
      if (bitV(qtab[i])) {
        if (bitR(qtab[i])
          setBitR(&qtab[i], 0); /* referenciada */
        else {
          /* q: proceso que cede su página */
          int real_page= realPage(qtab[i]);
          savePage(q, i); /* retoma otro proceso */
          setBitV(&qtab[i], 0);
          setBitS(&qtab[i], 1);
          int *ptab= p->pageTable;
          setRealPage(&ptab[page], real_page);
          setBitV(&ptab[page], 1);
          loadPage(p, page); /* retoma otro proceso */
          setBitS(&ptab[page], 0);
          purgeTlb(); /* invalida la TLB */
          purgeL1();  /* invalida cache L1 */
          reply(p);

          page= *(int*)receive(&p);
        }
      }
    }
    if (!hasNext(it))
      reset(it);
  }
}
```

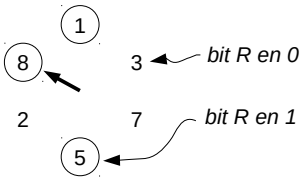
Suponga que la MMU *sí* implementa el bit D (*dirty*). Modifique la función *clockStrategy* para que haga un uso eficiente del bit D.

II.- La implementación de más arriba no funciona cuando 2 procesos comparten su área de código. (a) Dé un ejemplo de ejecución incorrecta. (b) ¿Qué cambios haría para evitar que las páginas compartidas se fuesen a disco?

III. ¿Cuál de las 2 estrategias de reemplazo de páginas vistas en clases usaría Ud. en un sistema operativo mono-proceso para implementar paginamiento en demanda? Explique por qué descarta la otra estrategia.

IV. 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: 5, 7, 4, 7, 2, 6.



Pregunta 2

a.- (3 puntos) Se requiere implementar un driver para el dispositivo */dev/buf* que sirve de buffer. Múltiples procesos productores pueden abrir el dispositivo simultáneamente y escribir un ítem usando la llamada *write*. Al mismo tiempo múltiples procesos lectores abren el dispositivo y leen un ítem usando *read*. Por simplicidad considere que un ítem consiste en un único caracter y por lo tanto los procesos siempre leen/escriben 1 caracter (es decir que el parámetro *count* es 1 y no necesita verificarlo). El buffer tiene una capacidad de solo 10 caracteres.

Las restricciones son las usuales del problema productor/consumir: un lector debe esperar cuando el buffer está vacío y un escritor espera cuando el buffer está lleno. Un caracter escrito es leído por un solo proceso lector.

Programe las funciones correspondientes a *buf_read* y *buf_write* del driver requerido. Declare las variables globales que necesite e indique cómo se inicializan. Recuerde que este problema se resuelve usando un arreglo circular de 10 ítemes. Para solucionar el problema de sincronización puede usar los semáforos del núcleo o los mutex y condiciones de la tarea 3. Ignore las señales. Torpedo:

```
ssize_t buf_read(struct file *filp,
     char *buf, size_t count, loff_t *f_pos);
ssize_t buf_write(struct file *filp,
     char *buf, size_t count, loff_t *f_pos);
int copy_to_user(char *to, char *from, int c);
int copy_from_user(char *to,char *from,int c);
void down(struct semaphore *s);
void up(struct semaphore *s);
void m_init(KMutex *mutex);
void c_init(KCondition *cond);
void m_lock(KMutex *mutex);
void m_unlock(KMutex *mutex);
int c_wait(KCondition *cond, KMutex *mutex);
void c_broadcast(KCondition *cond);
void c_signal(KCondition *cond);
```

b.- (1,5 puntos) Se tiene un archivo que no requiere bloques de indirección doble en una partición Unix con bloques de 2 KB. Se agrega un byte a este archivo y se crea el bloque de indirección doble. Haga un diagrama mostrando inodo, bloques de datos y de indirección. ¿De qué tamaño es el archivo?

c.- (1, 5 puntos) Un computador posee un disco duro con un tiempo de acceso de 10 milisegundos y una velocidad de transferencia de 100 MB/seg. Estime cuanto tiempo tomaría mostrar todas las líneas que contengan el string "jperez" en (i) un archivo de texto desordenado de 10 MB, y (ii) 1000 archivos de texto desordenado de 10 KB cada uno, en el mismo directorio pero dispersos en el disco.