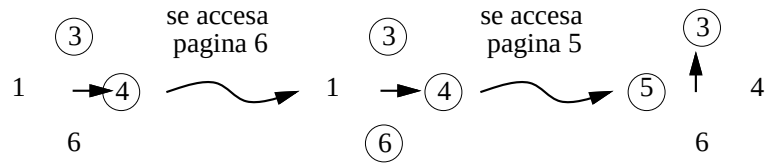
CC41B : Sistemas Operativos
Examen–Semestre Primavera'2002
Prof.: Luis Mateu.

Pregunta 1 (30%)

- **Parte a.-** La siguiente figura muestra los estados sucesivos de la memoria al realizar 2 accesos en un sistema de memoria virtual que usa *la estrategia del reloj para el reemplazo de páginas.*



  Las páginas que tienen el bit de referencia en 1, aparecen encerradas en una circunferencia. La posición del puntero está señalada por la flecha. En la figura se observa una primera transición de estados cuando se accesa la página 6 (residente) y una segunda transición al accesar la página 5 (no residente). A continuación se accesan las siguientes páginas de memoria:

  4 7 5 3 4 1 5

  Siguiendo el mismo esquema de la figura, muestre los estados de la memoria después de realizar cada uno de los accesos indicados.

- **Parte b.-** En un computador se dispone de 10 páginas reales para los procesos y 4 para el núcleo del sistema operativo. El sistema operativo utiliza la estrategia del *working set* para administrar la páginas. En él se ejecutan 3 procesos simultáneamente. El proceso A ocupa 5 páginas virtuales y su working set es el conjunto de páginas {0, 2, 3, 4}, el proceso B ocupa 6 páginas y su w-s es {1, 2, 4, 5} y el proceso C ocupa 4 páginas y su w-s es {0, 1, 3}.

  Haga un diagrama mostrando una posible asignación de la memoria y escriba el contenido de las tablas de páginas de c/u de los procesos. En las tablas Ud. debe indicar si la página vitual se encuentra residente o no y el número de página real en que se encuentra.

Pregunta 2 (30%)

- **Parte a.-** El tamaño típico de un bloque de archivo en Unix es 1 KB mientras que en Windows es de 4 KB. Señale la principal ventaja del tamaño de bloque usado por Unix y la principal ventaja del tamaño de bloque de Windows.

- **Parte b.-** Se tiene un archivo Unix de 1 GB. El programa de la izquierda lee bloques al azar en el archivo. Para ello utiliza la función `drand48`, que entrega números aleatorios entre 0 y 1, y la llamada al sistema `lseek`, que señala al sistema de archivos a partir de qué posición se debe realizar la próxima lectura. El programa de la derecha lee la misma cantidad de bloques pero secuencialmente.

```
char buff[1024];                  | char buff[1024];
int i, fd= open( ... );           | int i, fd= open( ... );
for (i= 0; i<1024*1024; i++) {    | for (i= 0; i<1024*1024; i++)
  int bloq= drand48()*1024*1024; |   read(fd, buff, 1024);
  int pos= bloq*1024;             |
  lseek(fd, pos, SEEK_SET);       |
  read(fd, buff, 1024);           |
}                                 |
```

  El disco se publicita con un tiempo de acceso de 10 milisegundos y una tasa de transferencia de 20 MB/seg. Estime el tiempo de ejecución de ambos programas. Haga supuestos razonables y explique cómo realiza sus cálculos.

- **Parte c.-** Se tiene un archivo Unix en una partición que usa bloques de 512 bytes. Haga el diagrama con la descomposición en inodo, bloques de indirección (simples y dobles) y bloques de datos del archivo, suponiendo que su largo es:

```
(12+128)*512+1 bytes
```

Pregunta 3 (40%)

En un laboratorio químico la *abomisina* se prepara en base a *tincosa* y *fomidona*. La preparación de la fomidona se hace en el formitrón y toma 2 minutos exactamente. La tincosa en cambio se decanta en el decafor en un tiempo variable que típicamente toma 5 minutos pero puede ir de 3 hasta 20 minutos. Incluso en ocasiones esta decantación falla.

La fomidona y la tincosa se mezclan en el batiscopio resultando la abomisina. Pero para que ésta posea la calidad requerida (i) la mezcla debe hacerse en no más de 5 minutos después de haber concluido la preparación de la fomidona, y (ii) el fin de la preparación de la fomidona debe ocurrir antes que el fin de la preparación de la tincosa. El laboratorio es pobre y solo posee un formitrón, un decafor y un batiscopio.

Actualmente la preparación de la abomisina se realiza en forma secuencial mediante el siguiente procedimiento:

```
Abomisina hacerAbomisina() {
  for (;;) {
    int finFomidona, finTincosa, exito;
    Tincosa tincosa;
    Fomidona fomidona= hacerFomidona(); /* en formitron */
    finFomidona= nGetTime();
    exito= hacerTincosa(&tincosa);      /* en decafor */
    finTincosa= nGetTime();
    if (exito && finTincosa-finFomidona<=5*60*1000)
      return mezclar(tincosa, fomidona); /* en batiscopio */
  }
}
```

Este procedimiento es correcto desde el punto de vista funcional, pero es ineficiente ya que se pierde mucha fomidona y tincosa debido a las exigencias (i) y (ii).

- **Parte a.-** Escriba en nSystem una versión más eficiente de `hacerAbomisina()` que disminuya las pérdidas de fomidona y tincosa, respetando las exigencias (i) y (ii). Para lograrlo, recurra a tareas adicionales.

- **Parte b.-** El procedimiento `hacerTincosa()` sólo retorna cuando se finalizó la preparación de la tincosa. Esto es ineficiente porque si ya transcurrieron 5 minutos de haber obtenido la fomidona, no tiene sentido continuar. El procedimiento `vaciarDecafor()` hace que cualquier invocación de `hacerTincosa()` en curso retorne de inmediato entregando falso (cero). Haciendo uso de `vaciarDecafor()`, reprograme su versión de `hacerAbomisina()` para hacerlo más eficiente.