CC41B Sistemas Operativos

Examen – Semestre Primavera 2005

Prof.: Luis Mateu

Pregunta 1 (40%)

Un *broadcast* (o difusión amplia) es un mensaje que envía una tarea a todo el resto de las tareas. El siguiente ejemplo muestra los broadcasts en acción:

En el nMain se inicializa la siguiente variable global:

```
Channel chan; ... chan= makeChannel();
```

Tareas productoras A y B:

```
for (;;) {
  char msg[80];
  int size= produce(msg);
  int listeners= broadcast(
       chan, msg, size, 20);
  ...
}
```

Tareas consumidoras X, Y y Z:

```
int id= 0; char msg[80];
for (;;) {
  if (listen(chan, id, msg, 80)
       >=0)
    consume(msg);
  id++;
}
```

Una o más tareas emiten mensajes por medio de un canal. Para ello se usa el procedimiento **broadcast**, indicando canal, área de memoria que almacena el mensaje, tamaño y tiempo de espera. El canal enumera consecutivamente los mensajes emitidos (0, 1, 2, etc.). Este procedimiento se bloquea durante exactamente el tiempo de espera indicado, y entrega el número de tareas que escucharon el mensaje.

Varias tareas escuchan los mensajes mediante el procedimiento **listen**, indicando canal, número de mensaje a leer, área de memoria en donde dejar el mensaje y tamaño máximo. Una tarea siempre solicita los mensajes en el orden 0, 1, 2, etc. Cuando se invoca **listen**: (i) si el mensaje fue emitido y el tiempo de espera aún no transcurre, listen coloca el mensaje en el área de mensaje y entrega su largo, (ii) si se excede el tiempo de espera, el mensaje es ignorado y se entrega -1, y (iii) si el mensaje aún no se emite, se espera indefinidamente hasta que se emita y se opera como en (i).

Implemente el tipo de datos *Channel* y los procedimientos *makeChannel*, *broadcast* y *listen*. Ud. *debe* utilizar los monitores de nSystem.

Indicaciones: Implemente broadcast mediante 2 llamadas al monitor. Entremedio invoque nSleep para dejar pasar el tiempo de espera. Suponga que Ud. dispone de un tipo de datos, como HashMap en Java, que asocia enteros con objetos cualesquiera.

Pregunta 2 (30%)

El siguiente algoritmo ordena ascendentemente un arreglo de enteros usando el método por inserción:

```
int i, j, n= ..., *a= ...;
for (i=0; i<n; i++) {
  int kmin= i;
  for (j= i+1; j<n; j++)
    if (a[kmin]>a[j]) kmin= j;
  swap(&a[i], &a[kmin]);
}
```

a) El algoritmo se ejecuta en una computador con 8 MB de memoria real, el sistema operativo implementa *paginamiento en demanda* con páginas de 4KB, el tamaño del arreglo es de unos 16 MB (i.e. n=4 millones de elementos) y la memoria ocupada por el programa y el sistema operativo es marginal. Haga una *estimación* del número de *page-faults* al ejecutar este programa.

b) Suponga que el tamaño del arreglo es de 128 KB (i.e. 32 mil elementos) y la TLB (*translation lookaside buffer*) de la MMU (*memory management unit*) posee 16 entradas. Haga una *estimación* del número de desaciertos en la TLB al ejecutar el programa.

c) Se le ha pedido a Ud. implementar un sistema operativo con procesos livianos y scheduling *non preemptive*: un proceso P sólo cede el procesador a otro proceso Q invocando explícitamente la llamada al sistema: *yield(Q)*. Discuta si tiene sentido y cuáles son los beneficios al implementar (i) paginamiento en demanda, (ii) la estrategia del reloj, (ii) la estrategia del *working set*.

Pregunta 3 (30%)

a) Suponga que 0 corresponde a la pista más interna de un disco y 99 a la más externa. En un instante dado la cola de scheduling de disco almacena requerimientos para las pistas 50, 5, 55, 95, 45. Las peticiones se hicieron en ese orden y el cabezal se encuentra en la pista 0. Haga un gráfico que muestre la posición del cabezal versus tiempo de atención de requerimientos considerando las estrategias de scheduling: (i) first come first served, (ii) shortest seek time first, y (iii) método del ascensor. incluya las 3 curvas en el mismo gráfico para poder comparar la velocidad de atención. Para cada estrategia *calcule* el desplazamiento total del cabezal, medido en número de pistas.

b) Indique cuanto es el espacio perdido por *fragmentación interna* para archivos de: (i) 512 bytes en una partición FAT16 de 256 MB, (ii) 512 bytes en una partición FAT16 de 1 GB, (iii) 8KB en una partición FAT32 de 2 GB, (iv) 8 KB en una partición Unix de 1 GB.