**CC41B: Sistemas Operativos**
**Tarea 1 - Semestre Primavera'2005**
**Plazo de entrega: Martes 6 de Septiembre**
**Prof.: Luis Mateu**

## El spooler de impresión

Un sistema computacional dispone de p impresoras. Para imprimir un texto en la impresora id (un valor entre 0 y p-1) se usa el procedimiento void `seqPrint(char *txt, int id)`, en donde `txt` es la información a imprimir. Si dos procesos invocan simultáneamente `seqPrint` utilizando la misma impresora el resultado es incorrecto (es decir, se produce un data-race).

Para resolver el problema de más arriba Ud. debe programar un spooler de impresión que ofrece una API que permite imprimir despreocupándose del momento en que se puede iniciar una impresión y del número de la impresora que se debe utilizar. El spooler acepta solicitudes de impresión prioritarias (`printJob`), en cuyo caso la tarea solicitante se bloquea hasta que el job haya sido impreso, o peticiones de impresión de baja prioridad (`submitJob`), en que la tarea solicitante no se bloquea, si no que continúa trabajando, pero solicita más tarde la confirmación de impresión (`waitJob`). Si el job no se ha impreso aún se transforma en un requerimiento prioritario. El spooler se encarga de imprimir los jobs invocando seqPrint de tal forma que no se produzcan data-races. La API es la siguiente:

- `Spooler *makeSpooler(int p)`: crea y retorna un nuevo spooler de impresión que soporta `p` impresoras.
- `void printJob(Spooler *sp, char *txt)`: Solicita a `sp` la impresión de alta prioridad de job `txt`. Este procedimiento retorna cuando finaliza la impresión de `txt`. Las solicitudes son servidas en orden de llegada.
- `Job *submitJob(Spooler *sp, char *txt)`: Deposita en `sp` una petición de impresión de baja prioridad del job `txt`. Este procedimiento retorna de inmediato un descriptor creado para el job. Toda invocación de `submitJob` tiene asociada una invocación de `waitJob`. Las peticiones son servidas en orden de llegada, pero sólo cuando no hay solicitudes de alta prioridad en espera.
- `void waitJob(Spooler *sp, Job *pjob)`: Espera hasta que haya concluido la impresión de un job previamente depositado con `submitJob`. Si el job ya fue impreso, este procedimiento retorna de inmediata. Si no ha sido impreso, el job adquiere alta prioridad y se considera que su llegada es el instante de la invocación de waitJob.
- `void destroySpooler(Spooler *sp)`: Destruye el spooler de impresión `sp` que no tiene jobs pendientes.

## Requerimientos

Ud. debe implementar la API del spooler de impresión en nSystem. Para realizar la impresión, Ud. debe crear `p` tareas servidoras, una por cada impresora disponible, que se encargan de imprimir finalmente los jobs invocando seqPrint (procedimiento dado) en una impresora que esté disponible. Para la sincronización entre las tareas que requieren imprimir y las tareas servidoras Ud. debe utilizar los monitores a la Java de nSystem2005.

## Recursos

En `http://www.dcc.uchile.cl/~lmateu/CC41B` Ud. encontrará:

- `nSystem2005.tgz`: la distribución de nSystem que incluye los monitores a la Java.
- `t1.tgz`: archivos para resolver la tarea que incluyen un `makefile`, encabezados para los procedimientos, un programa de prueba que incluye la implementación de `seqPrint` y una consola con el resultado que se espera que entregue el programa de prueba cuando la implementación del spooler de impresión es correcta.

## Plazo de entrega

La tarea se entrega en U-cursos. El plazo de entrega vence el Martes 6 de Septiembre. Se descontará medio punto por día de atraso hábil.