Sistemas Operativos

Sincronización de Threads
uso de timeouts y
patron productor/consumidor

Pablo Jaramillo

Diapositivas basadas en las de
Diego Madariaga (2022-2)

1.- Impresora Compartida

Impresora compartida

contexto

Se tiene una impresora en el Toqui controlada por el servidor de Anakena, Anakena utiliza threads para controlar la impresora, donde el thread contiene la información sobre que imprimir, si el servidor permitiera que dos o más threads operaran la impresora a la vez se mezclarian las lineas de los documentos. Los threads que interactuan con la impresora corresponden a conexiones a los computadores del Toqui.

Nosotros tenemos que evitar dicho problema hacienda que cada thread que solicite utilizar la impresora deba tener acceso exclusivo a esta antes de ocuparla u que notifique cuando termine de utilizarla. Por lo que la conexión se ve más o menos así:

```
tarea(){
    ...
    obtenerImpresora();
    ...  /* utilizar impresora*/
    devolverImpresora();
    ...
}
```

Impresora Compartida

objetivo

Además se nos pide ahorrar electricidad tal que si la impresora no se usa en 5 minutos después de su última tarea esta entrará en modo de bajo consumo, esto lo podemos hacer con la función `modoBajoConsumo()`. Para volver a utilizar la impresora se debe invocar `modoUsoNormal()`.

Implemente `obtenerImpresora()`, `devolverImpresora()` e `inicializarImpresora()` considerando todo lo mencionado.

2.- Detector de plagio

(Función masParecidas)

Detector de plagio

contexto

Para detectar copias/plagio en tareas un professor utiliza la función **masParecidas**, la cual encuentra las 2 tareas más similares de un conjunto. Esta recibe una arreglo de **n** tareas de alumnos, 2 punteros **pi** y **pj** con las direcciones de las variables de donde se almacenan los indices de las 2 tareas más parecidas.

```
void masParecidas(Tarea *tareas, int n, int *pi, int *pj){
    int min = INT_MAX;
    for(int i=0; i<n ; i++){
        for(int j=0; j<i; j++){
            int similitud = compTareas(tareas[i], tareas[j]);
            if(similitud < min){
                min = similitud;
                *pi=i;
                *pj=j;
    }}}}
```

Detector de plagio

objetivo

La función `compTareas` se tiene y entrega un coeficiente de similitud (o distancia), tal que 0 es que son iguales. Una comparación toma tiempo variable desde casi inmediato hasta varios minutos y se tienen que hacer $O(n^2)$ comparaciones, lo cual tarda mucho tiempo de CPU.

Se nos pide paralelizar `masParecidas` considerando una maquina octa-core. Mas de esto no sería factible por temas de memoria. Queremos desempeño continuo de los cores hasta que se terminen todas las comparaciones.

Detector de plagio

hints

- Utilize el Patrón Productor/Consumidor
- El productor deberá generar Jobs con pares de tareas para comparar
- Los múltiples consumidores deberán extraer los Jobs para realizar comparaciones