

SISTEMAS OPERATIVOS I

Segundo curso Ingeniería Informática. Curso 2005-2006

Práctica 3: Planificación de la CPU.

Realizar un programa en C que simule el comportamiento de un planificador RoundRobin.

El programa recibirá, como argumentos:

- qQUANTO** El valor del cuanto (para planificaciones round robin, entre 50 y 500 ms). Si no se especifica se supone un valor de 100.
- tHZ** El número de veces por segundo que interrumpe el temporizador del sistema.(50,60,80 ó 100). Si no se especifica se supone un valor de 60.
- ffich** El nombre del fichero donde está los procesos que tiene que planificar.
- v** Muestra mas información del funcionamiento del planificador.

Por ejemplo:

```
$a.out -q100 -t60 -fdatos
```

simula un roundrobin de cuanto 100ms, los procesos a simular están en el fichero "datos".

El formato del fichero con los procesos es el siguiente:

- Una línea para cada proceso.
- Cada línea contiene el instante de llegada y la duración de las ráfagas de CPU y e/s de cada proceso, separadas por espacios en blanco o tabuladores. Dado que un proceso siempre comienza y termina con una ráfaga de CPU en cada línea debe haber un número par de numeros (instante de llegada más las ráfagas).(Se supone que los números son enteros y representan segundos).

- ejemplo:

```
1 4 15 1 7 1
2 4 20 3
6 1 40 5 20 7 32 4
```

indicaría tres procesos, que llegan en los instantes 1, 2 y 6. El proceso que llega en el instante 2 tiene una ráfaga de CPU de 4 segundos, una de e/s de 20 segundos y finalmente una de CPU de 3 segundos

COMENTARIOS

EL programa ha de contar OBLIGATORIAMENTE con los siguientes estructuras y funciones.

- **struct PROCESO:** utilizada para almacenar la información de los procesos. Debe incluir, al menos, los siguientes miembros.
 - Identificación del proceso.
 - Estado del proceso.
 - Número de rafagas y duración de cada ráfaga.
 - Duración de la ráfaga actual en tics.
 - Número tics que lleva en la ráfaga actual.
 - Número de tics CPU totales que lleva ejecutados el proceso.
 - Número de tics que lleva el proceso en la cola de listos.
- **ColaListos:** cola de procesos listos, implementada como una cola de struct PROCESO.
- **ColaEspera:** cola de procesos en espera, implementada como una cola de struct PROCESO.
- **ColaPendientes:** cola de procesos que aún no han llegado junto con sus respectivos instantes de llegada.
- **ColaTerminados:** cola de procesos que han terminado.
- **ProcesoActual:** puntero a struct PROCESO del proceso que se supone en CPU ese momento.
- **función *tic()*:** función ejecutada periódicamente, es la que llama a las demás en caso de ser necesario. Simula el temporizador del sistema.
- **función *swtch()*:** selecciona el siguiente proceso a ejecutar cuando sea necesario. Invocada por *tic()*. Es la que mueve el proceso de la CPU a la cola de listos, espera o terminados, según corresponda.

El pseudocódigo del programa que realiza la simulación es el siguiente:

```

ProcesarArgumentosEntrada;
LeerDatosProcesos;
while (not terminado){
    tic();
    nanosleep(duracion_de_un_tic);
}
ImprimirResultados;

```

- *ProcesarArgumentosEntrada*. Procesa los argumentos de entrada y determina los parámetros de la planificación. Duración del tic, duración del cuanto en tics etc.
- *LeerDatosProcesos*. A partir de los datos del fichero de entrada crea y rellena la estructura de la ColaPendientes.
- *tic()*. Se ejecuta periódicamente (su periodicidad depende del valor de HZ). Realiza las siguientes tareas:
 - Si es el instante de llegada de un proceso nuevo lo pasa a la cola de listos..
 - Aumenta un tic de CPU al proceso que en ese momento está en CPU (proceso actual). Lo hace tanto para el número de tics de la ráfaga actual como para el total del proceso.
 - Aumenta un tic en la ráfaga actual a TODOS los procesos que están en la cola de procesos en espera.
 - Aumenta un tic de tiempo en cola de listos a todos los procesos que están en la cola de listos.
 - Si algún proceso de la cola de espera acaba su ráfaga de espera lo pasa a la cola de listos.
 - Si el proceso en CPU (proceso actual) acaba su cuanto (en planificación RR) o acaba su ráfaga o hay un proceso que deba ejecutarse en la cola de listos llama a la función *swtch* (cambio de contexto).
- *swtch*. Cambio de contexto. Hace las siguientes tareas:
 - Pasa el proceso actual a "listo", a "espera" o a "terminado" según corresponda. Esto supone cambiar el estado del proceso y moverlo a la cola correspondiente.
 - Pasa el proceso que corresponda (según el algoritmo de planifi-

cación) de la cola de listos a CPU. Si no hay procesos en la cola de listos la CPU permanecerá ociosa hasta la siguiente llamada a *tic()*.

- Informa por pantalla qué proceso abandona la CPU y cuál entra en CPU.
- Si se ha especificado la opción -v informa, además, por pantalla cómo están las colas de procesos listos, de espera, de pendientes y de terminados.
- *ImprimirResultados*. Esta función debe mostrar:
 - Para cada proceso
 - * Tiempo total en CPU (en segundos y en tics).
 - * Tiempo total en cola de espera (en segundos y en tics).
 - * Tiempo total en cola de listos (en segundos y en tics). Esto constituye el "tiempo de espera" del proceso.
 - * Tiempo de retorno: Tiempo transcurrido desde que se envió el proceso hasta que termina.
 - Tiempo de espera promedio.
 - Tiempo de retorno promedio.
 - Rendimiento: Relación entre tiempo empleado por los procesos y el tiempo total transcurrido. Para estos cálculos considérese que un cambio de contexto consume 0.1 ms.
- La única salida por pantalla del simulador debe ser la realizada por *swtch* cuando cambia de contexto y la de *ImprimirResultados*.

FORMA DE ENTREGA Como en la práctica anterior.

FECHA DE ENTREGA SEMANA 23 AL 27 ENERO 2006