Práctica 3

Parte Extra

Planificacion multinivel con realimentación

En esta práctica debéis implementar una política de planificación de 2 niveles con realimentación. En cada uno de los niveles se implementará la política `round robin`.

En el primer nivel se encolarán las tareas interactivas (`INTERACTIVE`), a las que asignaremos un quanto de 2. Inicialmente todas las tareas se marcan como `INTERACTIVE`. Si una tarea interactiva agota su cuanto pasará inmediatamente a formar parte de las tareas intensivas en cpu (`CPU_BOUND`), encoladas en el segundo nivel. A las tareas `CPU_BOUND` se les asignará un quanto de 4 y sólo se les dará tiempo de CPU si no hay tareas interactivas. Si una tarea `CPU_BOUND` realiza una operación de E/S antes de agotar su cuanto, pasará a considerarse dicha tarea como `INTERACTIVE`.

Se os entrega un esqueleto de la política dentro del fichero `sched_rr_dq.c` que deberéis completar atendiendo a los siguientes requisitos:

- Cuando llegue por primera vez una tarea al sistema, función `task_new_rr_dq()`, la marcaremos como `INTERACTIVE` y le asignaremos el cuanto almacenado en la variable global `rr_quantum_int`.
- Al encolar una nueva tarea, `enqueue_task_rr_dq`, se mirará si está marcada como `INTERACTIVE` o `CPU_BOUND` para guardarla en la cola adecuada.
- Cuando se busque la siguiente tarea para ser ejecutada, `pick_next_task_rr_dq()`, buscaremos primero en la lista `INTERACTIVE` y, si no hay tareas, en la lista de `CPU_BOUND`.
- Cuando se solicite una tarea para migración a través de `steal_task_rr_dq()`, haremos lo contrario, buscaremos primero en la lista `CPU_BOUND` y, si no hay tareas, en la lista de `INTERACTIVE`.
- Cuando una tarea agote su quanto, `task_tick_rr_dq()`, se marcará como `CPU_BOUND`, `INTERACTIVE` en caso contrario.

Para tener disponible una nueva cola se ha usado el campo `rq_cs_data` de la `runqueue`. Este campo se inicializa en la función `sched_init_rr_dq` suministrada con el esqueleto y se utilizará para encolar las tareas `CPU_BOUND`:

```
static int sched_init_rr_dq(void) {
  int cpu;
  runqueue_t* cpu_rq;
  slist_t *tasks_cpu_bound;

  for (cpu=0; cpu<nr_cpus; cpu++) {
    cpu_rq=get_runqueue_cpu(cpu);

    // Use rq_cs_data for cpu_bound queue tasks
    if( (tasks_cpu_bound=malloc(sizeof(slist_t))) == NULL) {
      fprintf(stderr,"Cannot allocate memory for private run queues\n");
      return 1;
    }
    // Init slist
    init_slist(tasks_cpu_bound, offsetof(task_t,rq_links));

    // Assign pointer
    cpu_rq->rq_cs_data=tasks_cpu_bound;
  }
  return 0;
}
```

Las funciones que necesiten acceder a la cola `CPU_BOUND` para, por ejemplo, encolar una tarea, lo podrán hacer hacer mediante:

```
task_t* task;
slist_t* cpu_bound_tasks = rq->rq_cs_data;
...
insert_slist(cpu_bound_tasks, task);
```

Las tareas de tipo `INTERACTIVE` se almacenarán en la cola disponible dentro de la `runqueue` al igual que se hacía en los planificadores suministrados como ejemplo:

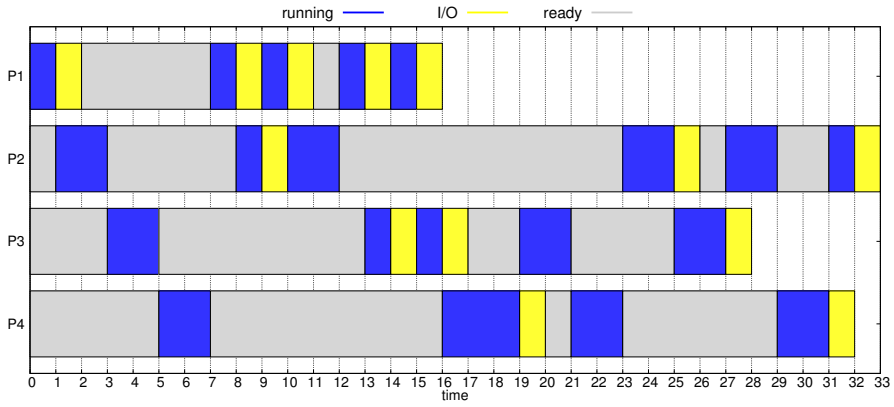
```
task_t* task;
...
insert_slist(&rq->tasks,task);
```

Recuerda: Debéis modificar el `Makefile` para incluir el nuevo fichero (`sched_rr_dq.c`) en la compilación y también el fichero de cabecera `sched.h` para que esté disponible.

Ejemplo

Ejemplo de simulación con el siguiente fichero y una sola CPU

```
P1 0 0 1 1 1 1 1 1 1 1 1 1
P2 0 0 3 1 4 1 3 1
P3 0 0 3 1 1 1 4 1
P4 0 0 5 1 4 1
```



Ejemplo de simulación con dos CPUs: