Laboratório 2 – Escalonador com prioridades

Neste laboratório iremos acrescentar gradualmente recursos ao nosso RTOS como fazer uma tarefa dormir, diferenciar a prioridade de tarefas, alterar o quantum da tarefa e outros.

Exercício 1 – Fazendo uma tarefa dormir.

Você deve ter percebido que decrementar uma variável não é uma boa técnica para gastar tempo do processador. Dentre as desvantagens, podemos citar o alto consumo de energia e a ocupação elevada do processador. Uma estratégia mais interessante é fazer a tarefa dormir por um tempo pré-determinado, liberando o slot de tempo para outras tarefas executarem. Idealmente, teremos uma função **`wait(time_ms)`**, como abaixo, que fará exatamente isso.

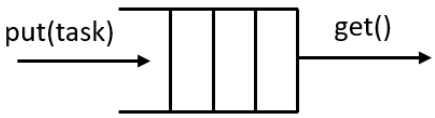
```
void blinkRED() {

    P0DIR |= BIT0;                          // Configura o LED como saída
    while(1) {                              // Task nunca termina
        wait(250);
        P1OUT ^= BIT0;
    }
 }
```

O sistema operacional deve ser responsável por gerenciar o estado das tarefas. Para isso iremos criar uma entrada na nossa estrutura *task_t.* Crie a entrada *wait* para salvarmos a quantidade de ticks que iremos aguardar.

O escalonador terá uma nova tarefa agora: Ele irá gerenciar o estado das tarefas. Antes de escolher a nova tarefa, decrementa um tick para cada tarefa que estiver em *estado de espera*. Além disso, selecione a próxima tarefa apenas se ela não estiver no estado de espera. Isso irá liberar slots de tempo ocupados por tarefas em estado de espera.

Se for necessário, crie uma tarefa *idle()* que não faz nada. Chamaremos essa tarefa de *tarefa de plano de fundo*, do inglês “background task”. Qual é a desvantagem de se ter uma tarefa *idle*?

Exercício 2 – Criando uma fila de prioridades

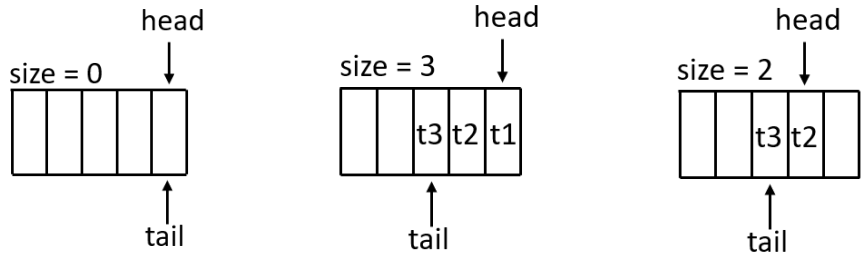


Fila de processos

O sistema de filas e prioridades de execução resolvem o problema da tarefa de plano de fundo. As tarefas podem ser colocadas na fila de prioridade normal enquanto as tarefas de plano de fundo são colocadas na fila de baixa prioridade e só serão executadas quando todas as tarefas de maior prioridade forem terminadas ou colocadas em estado de espera.

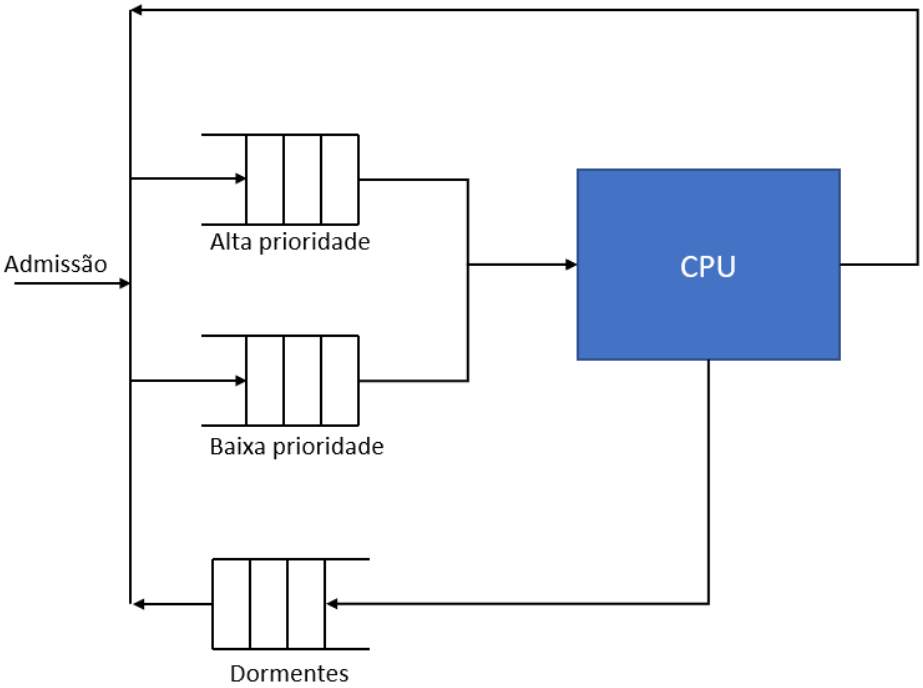
Neste exercício iremos criar a estrutura de um fila de processos. Dentro do objeto “fila” devemos criar um vetor para armazenar as tarefas (ou apenas os ponteiros para as tarefas) e três inteiros indicando o cabeçalho, rodapé e tamanho da fila (head, tail e size). Toda vez que um elemento é colocado na fila, incrementamos o ponteiro de fim de fila e armazenamos o objeto. Toda vez que retirarmos um elemento da fila, incrementamos o ponteiro de início de fila e retornamos o elemento.

A figura abaixo mostra uma fila em três momentos diferentes. No primeiro, a fila acabou de ser criada e não contém nenhum elemento. Na segunda etapa, colocamos 3 tarefas na fila. Na última etapa uma tarefa foi retirada. Perceba que os índices *head* e *tail* se movem na mesma direção e são cíclicos, ou seja, se chegarem no final, devem voltar ao início.



Exercício 3 –Prioridades das tarefas.

Use o recém-criado objeto que modela uma fila para criar um escalonador que gerencie prioridades das tarefas.



Exercício 4 – Ajuste fino entre tarefas: quantum

Uma tarefa pode ocupar mais de um tick por ciclo. Damos o nome de *quantum* à unidade que quantifica o número de ticks que uma tarefa pode ocupar numa rajada só. O *quantum* de uma tarefa pode ser visto como um ajuste fino no sistema de filas de prioridade, onde numa mesma fila, dois processos diferentes podem ter reservas diferentes de slots de tempo.