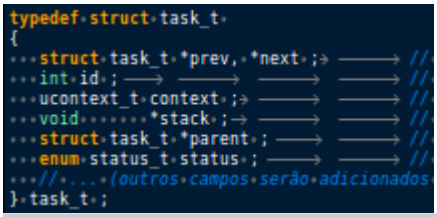

Carlos Maziero

Construção de semáforos

O objetivo deste projeto é implementar **semáforos clássicos** em nosso sistema. As funções a implementar são descritas na sequência.

Cria um semáforo

```
int sem_create (semaphore_t *s, int value)
```

Inicializa um semáforo apontado por `s` com o valor inicial `value` e uma fila vazia.

A chamada retorna 0 em caso de sucesso ou -1 em caso de erro.

Requisita um semáforo

```
int sem_down (semaphore_t *s)
```

Realiza a operação *Down* no semáforo apontado por `s`. Esta chamada pode ser bloqueante: caso o contador do semáforo seja negativo, a tarefa corrente é suspensa, inserida no final da fila do semáforo e a execução volta ao *dispatcher*; caso contrário, a tarefa continua a executar sem ser suspensa.

A chamada retorna 0 em caso de sucesso ou -1 em caso de erro (semáforo inexistente ou destruído).

Libera um semáforo

```
int sem_up (semaphore_t *s)
```

Realiza a operação *Up* no semáforo apontado por `s`. Esta chamada **não é bloqueante** (a tarefa que a executa não perde o processador). Se houver alguma tarefa aguardando na fila do semáforo, a primeira da fila deve ser acordada e retornar à fila de tarefas prontas.

A chamada retorna 0 em caso de sucesso ou -1 em caso de erro (semáforo inexistente ou destruído).

Destrói um semáforo

```
int sem_destroy (semaphore_t *s)
```

Destrói o semáforo apontado por `s`, acordando todas as tarefas que aguardavam por ele.

Importante: as tarefas que estavam suspensas aguardando o semáforo devem ser acordadas e retornar da operação *Down* correspondente com um código de erro (valor de retorno -1).

A chamada retorna 0 em caso de sucesso ou -1 em caso de erro.

Observações:

- O arquivo `datatypes.h` contém os tipos de dados (a completar) e o arquivo `pingpong.h` os protótipos das

funções a serem implementadas em `pingpong.c`.
- Os semáforos deverão ser totalmente implementados em seu código; sua implementação não deverá usar funções de semáforo de bibliotecas externas ou do sistema operacional subjacente.
- Você deverá definir a estrutura `semaphore_t` (observe que cada semáforo deve ter seu próprio contador e sua própria fila).
- É necessário habilitar/desabilitar as trocas de contexto para que as operações sobre os semáforos sejam feitas de forma atômica (sem trocas de contexto que poderiam gerar condições de disputa nas estruturas de dados do semáforo). Para isso, use o controle de preempção definido nos projetos anteriores.

Sua implementação deve funcionar com estes dois arquivos de teste:

- pingpong-semaphore.c e sua saída
- pingpong-racecond.c e sua saída

Outras informações

- Duração estimada: 4 horas.
- Dependências:
  - Gestão de Tarefas
  - Dispatcher
  - Preempção por Tempo
  - Tarefa Main
  - Operador Join
  - Sleeping
