ISEP – LEEC - Sistemas Computacionais 2017/2018

Ficha 7 – Implementação de fila de mensagens usando semáforos

Considere a API de filas de mensagem da norma POSIX (`man 7 mq_overview`, `man mq_send`, `man mq_receive`).

Pretende-se implementar uma biblioteca de fila de mensagens para aplicações *multi-thread*, de acordo com as declarações e comentários apresentados abaixo (ficheiro `mymq.h`).

```
typedef struct
{
	int slot_size; //maximum slot size. Set by mymq_init.
	int number_of_slots; //number of slots. Set by mymq_init.
	void *slots; //malloc(slot_size*number_of_slots).
	int *msg_size; //indicates the actual size of the message stored on each slot
	int oldest_slot;		//index of the oldest element in the queue
	sem_t sem_msgs_in_queue; //send() increments this, receive() waits on this
	sem_t sem_free_slots;	//receive() increments this, send() waits on this
	pthread_mutex_t access;	//the accesses to the message queue
				//must be mutually exclusive
}
mymq_t;


//This function must be called to initialize the queue. Returns -1 on error.
int mymq_init(mymq_t *mq, int slot_size, int number_of_slots);

void mymq_send(mymq_t *mq, void *data, int size);

//Returns size of received message  (in bytes)
int mymq_receive(mymq_t *mq, void *data, int size);

//Releases all resources used by the queue
int mymq_destroy(mymq_t *mq);
```

Ao contrário das funções da norma POSIX, não se pretende suportar mensagens com diferentes níveis de prioridade. De resto, pretende-se que as funções `mymq_send` e `mymq_receive` tenham um comportamento análogo ao comportamento *default* das funções `mq_send` e `mq_receive` da norma POSIX. O ficheiro `mymq_test.c` apresenta um exemplo de utilização das funções a implementar.

Sugestões de implementação:

- Os bloqueios de fila cheia, no caso da função `mymq_send`, e de fila vazia, no caso da função `mymq_receive`, deverão ser geridos com os semáforos declarados na estrutura `mymq_t` (campos `sem_free_slots` e `sem_msgs_in_queue` respetivamente).
- O campo `slots` deverá apontar para um bloco de memória a ser usado como um *buffer* circular, onde serão armazenadas as mensagens. Um *buffer* circular é um vetor em que os dados são armazenados e consumidos de forma circular, isto é, quando se atinge o último elemento do vetor (índice `number_of_slots-1`, neste caso), o próximo elemento a considerar será o primeiro (índice 0). Nesse sentido:
    - O campo `oldest_slot` deverá ser incrementado (usando aritmética "modular") sempre que uma mensagem é lida da fila:

```
oldest_slot = (oldest_slot + 1) % number_of_slots.
```

- Novas mensagens deverão ser guardadas na posição indicada pela seguinte expressão:

  `(oldest_slot+número_de_mensagens_na_fila) % number_of_slots`.

  O número de mensagens na fila é indicado pelo semáforo `sem_msgs_in_queue` (usar a função `sem_getvalue` para consultar o valor do semáforo).

- O campo `msg_size` deverá também armazenar o endereço base de um vetor. Neste caso, trata-se de um vetor de inteiros que deverá ter o mesmo número de elementos do vetor apontado por `slots` e deverá ser indexado da mesma forma. A finalidade deste vetor é indicar, para cada *slot*, o tamanho da mensagem armazenada. Deverá ser iniciado a zero.

A função `mymq_init` deverá ser usada para iniciar corretamente cada um dos campos da variável do tipo `mymq_t` apontada pelo parâmetro `mq`.

De seguida, apresenta-se um cenário de utilização da fila, com resultados esperados:

```
mymq_t mq;
mymq_init(&mq, 10, 3); //3 slots of 10 bytes
mymq_send(&mq, "Test", 4);
```

Slots	TestXXXXXX	XXXXXXXXXX	XXXXXXXXXX
msg_size	4	X	X
oldest_slot	0		
sem_msgs_in_queue	1		
sem_free_slots	2		

```
X – valor/byte indefinido

mymq_send(&mq, "Test2", 5);
```

slots	TestXXXXXX	Test2XXXXX	XXXXXXXXXX
msg_size	4	5	X
oldest_slot	0		
sem_msgs_in_queue	2		
sem_free_slots	1		

```
mymq_send(&mq, "Test3", 5);
```

slots	TestXXXXXX	Test2XXXXX	Test5XXXXX
msg_size	4	5	5
oldest_slot	0		
sem_msgs_in_queue	3		
sem_free_slots	0		

```
char buffer[10];
mymq_receive(&mq, buffer, 10);
```

slots	TestXXXXXX	Test2XXXXX	Test5XXXXX
msg_size	4	5	5
oldest_slot	1		
sem_msgs_in_queue	2		
sem_free_slots	1		

```
mymq_send(&mq, "TestFinal", 9);
```

slots	TestFinalX	Test2XXXXX	Test5XXXXX
msg_size	9	5	5
oldest_slot	1		
sem_msgs_in_queue	3		
sem_free_slots	0		

1 – Implemente as 4 funções descritas.
2 – Teste a biblioteca desenvolvida com o programa do ficheiro `mymq_test.c`.