TP2 de SOPE: etapa 2

Este trabalho foi realizado por Eduardo Campos (201604920), Miguel Gomes (201605908) e Pedro Azevedo (201603816) no âmbito da disciplina de SOPE.

Compilação e Execução

Na pasta onde se encontra o makefile principal faça:

```
make clean && make
```

Executar os dois programas , em terminais separados, primeiro o Q2 e de seguida o U2:

```
./Q2 <-t nsecs> [-l nplaces] [-n nthreads] <fifoname>
./U2 <-t nsecs> <fifoname>
```

Descrição do Programa

Utentes

Ao iniciar o programa U2 este vai preencher uma struct do tipo UserParser.

```
typedef struct
{
    int nsecs;
    char fifoname[256];
} User;
```

Após esta inicialização o programa vai criar multiplas threads ,durante o período especificado, e a tempos aleatórios. Estas threads iram criar uma struct do tipo message que por sua vez irá ser utilizada para escreverem para o FIFO público ,definido pelo utilizador.

```
typedef struct Message
{
    int i;
    int pid;
    long tid;
    int dur;
    int pl;
} Message;
```

Depois do envio da mensagem a thread espera que pela resposta do servidor que será enviada para um FIFO privado criado pela thread, terminando após recebido. A seguir ao tempo de execução terminar, o programa liberta todos os seus recursos e sai do programa com `pthread_exit`.

Quartos de Banho

Ao inicializar o Q2 este vai preencher uma struct do tipo BathroomParser, em que as flags place_f, thread_f e os seus respetivos números (nplaces e nthreads) são opcionais.

```
typedef struct
{
    int secs_f;
    int nsecs;
    int place_f;
    int nplaces;
    int thread_f;
    int nthreads;
    char fifoname[MAX_FIFONAME];

} BathroomParser;
```

Após a inicialização, o programa espera por pedidos vindos do FIFO público indicado e trata-os criando uma thread nova que será responsável pela comunicação da resposta de sucesso ou insucesso ,pelo o FIFO privado correspondente. Depois de enviar a resposta espera o tempo que foi indicado, terminando após esse mesmo. Passado o tempo de execução definido, igualmente ao programa dos utentes, este liberta os seus recursos e termina o programa com `pthread_exit`.

No caso de a `place_f` estar ativa limitamos o número de casas de banho disponíveis. Se existir uma casa de banho livre ele atribui essa casa de banho ao pedido, caso contrário o pedido retorna com um valor negativo. No caso de não estar ativa, o número da casa de banho atribuído depende do número pedidos recebido até

ao momento.

No caso de a `thread_f` estar ativa limitamos o número de threads que podem estar ativas durante o perído de execução. No caso de não estar ativa, o número de threads máximo é o número de threads que o sistema aguenta.

Em ambos os casos são recorridos semaphores e uma queue para limitar quer o número de threads e o número de casas de banho possíveis.

Notas:

1. Caso a criação de threads dê erro, o progama sai do ciclo de criação de threads liberta os seus recursos (semaphores, mutexes, fifo,etc...).
2. Os sistemas de sincronização de presentes são mutexes e semaphores. Mutexes para a impedir deadlocks ao interagir com partes críticas do programa (variáveis que as threads utilizam) e semaphores para limitar a criação de threads de acordo com o limite indicado.
3. No caso de o U2 ser executado primeiro, o programa cria na mesma as threads ,no entanto o registo do log a fase de operação será sempre CLOSD, dado que tenta abrir o FIFO apenas uma vez no início do programa e se não conseguir considera que o Quarto de Banho está fechado.