Projeto de Sistemas Operativos 2018-19
`CircuitRouter-Bench`
Enunciado*do exercício 1

LEIC-A / LEIC-T / LETI
IST

Os alunos devem ler primeiro o documento de visão geral do projeto, assim como os dois guias do Exercício 0, antes de lerem este guia.

O objetivo do Exercício 1 é o de desenvolver a consola `CircuitRouter-SimpleShell`, que permite lançar e gerir instâncias da aplicação `CircuitRouter-SeqSolver`, estudada no Exercício 0. Cada instância lançada calcula a solução para um problema distinto submetido pelo utilizador e corre em paralelo com as outras instâncias. No final, imprime o seu resultado num ficheiro.

As próximas secções descrevem em detalhe cada requisito.

Como ponto de partida desde exercício, deverá descarregar a solução do Exercício 0, correspondente ao arquivo `so1819-ex1-base.zip` que está disponível na página da disciplina (no fénix), na secção "Laboratórios".

1 `CircuitRouter-SeqSolver`

A versão do `CircuitRouter-SeqSolver` aperfeiçoada no Exercício 0 recebe os dados do problema (grelha e pares de pontos (origem, destino)) a partir do stdin e imprime o resultado no stdout. Neste exercício, pretende-se adaptar o `CircuitRouter-SeqSolver` para que utilize ficheiros de entrada e saída.

1.1 Passagem do problema de entrada

Esta versão deve ser adaptada para que os dados do problema passem a ser obtidos a partir de um ficheiro (`inputfile`), cujo nome é passado como (único) argumento obrigatório de linha de comandos. Eis um exemplo de uma invocação do `CircuitRouter-SeqSolver`:

```
./CircuitRouter-SeqSolver inputs/random-x32-y32-z3-n64.txt
```

Os dados do problema, contidos no ficheiro, são formatados da mesma forma que na versão anterior do `CircuitRouter-SeqSolver`.

1.2 Impressão de resultados em ficheiro

O resultado deve também passar a ser guardado num ficheiro em vez de ser impresso no stdout. Como consequência, a flag `-p` deixará de existir e deve ser removida, dado que o ficheiro de output será sempre criado. Mais precisamente, o `CircuitRouter-SeqSolver` deve ser modificado para criar um ficheiro novo, cujo nome é dado pelo nome do ficheiro de entrada sufixado por `.res`. Caso já exista um ficheiro com

*Enunciado atualizado a 29/9/2018: elimina o requisito de imprimir o tempo de execução de cada processo filho

esse nome, o ficheiro já existente deve ser renomeado para passar a ter o sufixo `.old`. Caso já exista também um ficheiro com este último nome, este deve ser eliminado.

Como exemplo, se o ficheiro de entrada se chamar `m.txt`, o ficheiro com os resultados deve chamar-se `m.txt.res` e, caso já existisse um ficheiro com o mesmo nome, este será antes renomeado para `m.txt.res.old`.

Nota importante: o formato utilizado pelo `CircuitRouter-SeqSolver` para apresentar o circuito 3D obtido deve ser tal como implementado na função `grid_print` fornecida na solução para o Exercício 0 (arquivo `so1819-ex1-base.zip`).

2 CircuitRouter-SimpleShell

Uma vez lançado, o programa `CircuitRouter-SimpleShell` consiste numa consola que permite ao utilizador executar múltiplas instâncias do `CircuitRouter-SeqSolver`, cada uma executada num processo filho para resolver um problema distinto.

O programa `CircuitRouter-SimpleShell` recebe um argumento opcional, `MAXCHILDREN`, que indica quantos processos filho podem estar em execução simultaneamente. Intuitivamente, este argumento deve ser definido de acordo com o paralelismo hardware disponível: idealmente, o valor de `MAXCHILDREN` deve ser igual ao número de contextos hardware disponíveis na máquina. No caso de `MAXCHILDREN` não ser fornecido, o `CircuitRouter-SimpleShell` corre sem limite no número de processos filho simultâneos.

Os comandos aceites são descritos de seguida.

- `run inputfile`

  Cria um novo processo filho que executará o `CircuitRouter-SeqSolver` a partir dos argumentos iniciais contidos no ficheiro `inputfile`. O novo processo filho deve correr em paralelo com o pai e com os restantes processos filho que estejam ainda ativos. No entanto, caso `MAXCHILDREN` esteja definido, não deve haver mais do que `MAXCHILDREN` processos filho simultaneamente em execução. Se o número máximo de processos filho já estiver no limite, o `CircuitRouter-SimpleShell` deve aguardar que uma das execuções termine antes de lançar uma nova execução.

- `exit`

  Termina o `CircuitRouter-SimpleShell`.

  O comando `exit` só deve terminar o `CircuitRouter-SimpleShell` depois de esperar que todos os processos filho (execuções de `CircuitRouter-SeqSolver`) terminem. Nesse momento, o processo `CircuitRouter-SimpleShell` deve imprimir no stdout, para todas as instâncias executadas do `CircuitRouter-SeqSolver`:

  - o respetivo process id (pid);
  - uma indicação da forma como terminou:
    - `OK`: terminação normal com exit;
    - `NOK`: terminação abrupta e/ou com exit code diferente de zero;
  - ~~o tempo que a execução demorou (desde o lançamento do processo filho até ao pai detetar que o filho terminou).~~

  O seguinte exemplo (em que 2 processos filho foram lançados) ilustra o formato que deve ser usado:

```
CHILD EXITED (PID=2987; return OK)
CHILD EXITED (PID=2945; return NOK)
END.
```

3 Compilação dos programas

A compilação e geração dos executáveis `CircuitRouter-SeqSolver` e `CircuitRouter-SimpleShell` deve ser automatizada com recurso a um (ou mais) Makefiles. Em ambos os casos, deve ser possível gerar os executáveis correndo apenas o comando `make` (sem argumentos). A submissão do código deve ser feita tal como descrito no enunciado geral do projeto.

3.1 CFLAGS

Todo o código deve ser compilado com o `gcc`, utilizando as flags `-Wall` e `-std=gnu99`:

```
CFLAGS = -Wall -std=gnu99
```

Caso seja necessário, pode ser incluída a flag `-g`.