Timed mini-shell

Iuri, um jovem rapaz fissurado pelo tempo das coisas, está muito curioso para saber quanto tempo demora cada um dos seus programas preferidos.

Como Iuri é muito ocupado ele pediu a sua ajuda para escrever um *mini shell* capaz de executar os comandos instalados em seu computador.

O *mini shell* deve ser capaz de ler um conjunto de comandos, pela entrada padrão, executar o comando e ao terminar a execução do comando ele deverá imprimir o tempo (em segundos) de execução e o código de retorno do programa.

Entrada

A entrada é composta por um único caso de testes possuindo diversas linhas. Cada linha é composta por duas strings (sem espaço) de até 255 caracteres cada. A primeira linha contém o `PATH` completo de um binário e a segunda string possui um argumento a ser passado para o programa que o *mini shell* deve executar.

Saída

A saída é composta por diversas linhas. Para cada comando executado o seu programa deve imprimir uma única linha contendo: `> Demorou %X segundos, retornou %Y`, trocando `%X` pelo tempo, com precisão de 1 casa decimal, em segundos e `%Y` pelo código de retorno.

Caso o comando passado não exista, ou não possa ser executado, o seu programa deve imprimir a mensagem, em inglês, do código de erro configurado na variável `errno` modificado pela função `execl(3)`. Com a mensagem `> Erro: %s` , sendo `%s` substituída pela mensagem de erro gerada pela função `strerror(3)`

No término da execução deve imprimir quantos segundos foram utilizados pela execução completa dos comandos, com a string `>> O tempo total foi de %X segundos`.

O que fazer quando o tempo for quebrado?

O seu programa deve imprimir o tempo com a precisão de 1 casa decimal, ou seja, você vai precisar olhar para além do tempo em segundos e precisará verificar a diferença na casa dos milisegundos. A função `gettimeofday(2)` vai ajudá-lo a fazer essa comparação.

No primeiro exemplo há um `sleep 1.1` que geralmente demora um pouco mais que `1.1` segundos, veja o exemplo abaixo:

```
ribas@charge:~$ time sleep 1.1

real    0m1.101s
user    0m0.001s
sys 0m0.000s
```

Para garantir a simplificidade deste exercício nos interessamos somente com 1 casa de precisão.

Exemplos

Exemplo de entrada

```
/bin/sleep 1
/bin/sleep 1.1
/bin/eunaoexisto 123
```

Saída para o exemplo de entrada

```
> Demorou 1.0 segundos, retornou 0
> Demorou 1.1 segundos, retornou 0
```

```
> Erro: No such file or directory
> Demorou 0.0 segundos, retornou 2
>> O tempo total foi de 2.1 segundos
```

Exemplo de entrada

```
/bin/sleep 1.5
/bin/true ign
/bin/false 222
/bin/uname -s
/bin/sleep 1
```

Saída para o exemplo de entrada

```
> Demorou 1.5 segundos, retornou 0
> Demorou 0.0 segundos, retornou 0
> Demorou 0.0 segundos, retornou 1
Linux
> Demorou 0.0 segundos, retornou 0
> Demorou 1.0 segundos, retornou 0
>> O tempo total foi de 2.5 segundos
```

Exemplo de entrada

```
/bin/false oi
/bin/false tchau
/bin/uname -m
/bin/apt-get moo
/bin/sleep 0.3
```

Saída para o exemplo de entrada

```
> Demorou 0.0 segundos, retornou 1
> Demorou 0.0 segundos, retornou 1
x86_64
> Demorou 0.0 segundos, retornou 0
                 (__)
                 (oo)
           /------\/
          / |    ||
         *  /\---/\
            ~~   ~~
..."Have you mooed today?"...
> Demorou 0.0 segundos, retornou 0
> Demorou 0.3 segundos, retornou 0
>> O tempo total foi de 0.3 segundos
```

DICAS E DISCUSSÃO

- Você pode executar o seu mini shell de maneira interativa para ver o resultado aparecendo, e finaliza a sua execução com o fim de linha `EOF`, sendo enviada ao pressionar a combinação `C-d` (*Control+d*) no teclado.
- Leia o manual das seguintes funções:
  - `fork(2)`
  - `wait(2)`
  - ~~`time(2)`~~
  - `gettimeofday(2)`
  - `strerror(3)`

    O número entre parênteses (`fork(2)`) representa a página do manual que deve ser lida, no caso do `fork` pode chamar como `man 2 fork`, já no caso da `strerror` como `man 3 strerror`. Isso serve para evitar ambiguidade, por exemplo a `printf(3)`, por padrão `man printf` abre o manual da `printf(1)` que é o comando shell, enquanto `man 3 printf` abre o manual da função em `C`.

Author: Bruno Ribas