Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Sistemas Operativos

2017/2018

1º Trabalho

Objetivo

Realização de um trabalho de programação em Python envolvendo a criação de processos/*threads*, a comunicação entre processos/*threads* e a sincronização de processos/*threads*.

Introdução

Com este trabalho pretende-se desenvolver o comando pzip (paralell zip), o qual permite comprimir/descomprimir um conjunto de ficheiros em paralelo.

Descrição do trabalho

Pretende-se que os alunos concretizem o comando pzip descrito de seguida:

NOME

`pzip` – comprime/descomprime, em paralelo, vários ficheiros

SINOPSE

```
pzip –c|-d [-p n] [-t] {ficheiros}
```

DESCRIÇÃO

O comando pode ser utilizado em modo comprimir (opção -c) ou em modo descomprimir (opção -d).

A opção p é opcional e permite definir o nível de paralelização do comando (ou seja, o número de processos/*threads* que são utilizados para fazer a compressão). Inicialmente, o processo pai deve criar os processos/*threads* definidos pelo nível de paralelização do comando. Por omissão, deve ser utilizado apenas um processo.

Podem ser dados zero ou mais ficheiros, sobre os quais é feita a compressão. Caso não sejam dados ficheiros na linha de comandos, estes devem ser lidos de *stdin*.

Quando em modo de compressão, para cada ficheiro dado deverá ser criado um ficheiro comprimido, acrescentando ao nome do ficheiro a extensão **`zip`**. Por exemplo, para o ficheiro **`pzip.py`**, seria criado o ficheiro **`pzip.py.zip`**.

Quando em modo de descompressão, para cada ficheiro dado deverá ser recuperado o ficheiro original sem compressão.

Tanto para a opção de compressão como para a de descompressão, os processos/threads comprimem/descomprimem o conteúdo dos ficheiros com recurso à classe **`ZipFile`** do módulo **`zipfile`**.

Caso seja dada a opção -t, se um dos ficheiros não existir, o comando deve finalizar a operação em curso para os ficheiros que já iniciou sem dar início à compressão/descompressão de novos ficheiros. Caso não seja dada a opção -t o comando continua até ao fim mesmo que não exista algum dos ficheiros.

No final, o processo pai deve escrever para *stdout* o número total de ficheiros comprimidos/descomprimidos.

Os alunos devem concretizar duas soluções: uma com processos e outra com *threads*. Na ficha de entrega será pedido aos alunos que analisem de forma crítica os resultados que obtêm com cada uma das soluções.

Desafios

Como sincronizar os vários processos/*threads* de modo a garantir que a compressão/descompressão é feita em todos os ficheiros uma e apenas uma vez?

Como sincronizar os vários processos de modo a garantir que, caso seja dada a opção -t e um dos ficheiros não exista, a operação não é efetuada para os ficheiros seguintes?

Como passar a informação necessária ao processo pai de modo a ser possível calcular o número total de ficheiros comprimidos/descomprimidos?

Exemplos de utilização

```
so000@kali:~/so$ python pzip.py -c exemplo

so000@kali:~/so$ ls
a1      a2      a4      exemplo         exemplo.zip     pzip.py

so000@kali:~/so$ python pzip.py -c -t a1 a3 a4
O ficheiro a3 não existe.

so000@kali:~/so$ ls
a1      a1.zip          a2      a4      exemplo         exemplo.zip     pzip.py
```

Entrega

A entrega do trabalho é realizada da seguinte forma:

- Os grupos devem inscrever-se atempadamente, de acordo com as regras afixadas para o efeito, no moodle.
- Submeter o ficheiro `pzip.py` no moodle (um por grupo).
- Submeter o ficheiro `pzip_threads.py` no moodle (um por grupo).
- Preencher a ficha de entrega no moodle (todos os elementos do grupo).

Prazo de entrega

O trabalho deve ser entregue até dia 12 de novembro de 2017 (domingo) às 20.00h.

Avaliação dos Trabalhos

As avaliações dos trabalhos serão realizadas na última semana de aulas. Todos os elementos do grupo terão de comparecer à avaliação e a avaliação é feita **individualmente**. Deste modo, cada elemento do grupo deve estar preparado para responder a qualquer questão relacionada com os trabalhos e com a matéria das aulas teórico-práticas.

Alguns parâmetros de avaliação

Funcionalidade, Estrutura, Desempenho, Algoritmia, Comentários, Clareza do código.