Trabalho de Projeto

1ª Fase

1. Objetivo

Desenhar e desenvolver um programa em *Python* que seja útil na prática e que envolva a criação de processos, assim como a comunicação e a sincronização entre os mesmos. Este documento descreve a primeira de duas fases de entrega do projeto.

2. Introdução

No âmbito da programação e automação de tarefas, a eficiência no processamento e na manipulação de dados é de extrema importância. Frequentemente, nos deparamos com a necessidade de contar palavras (e a ocorrência das mesmas) em grandes conjuntos de ficheiros de texto. É neste contexto que se insere o tema deste projeto.

Pretende-se desenvolver uma aplicação denominada `pwordcount` (*parallel word count*) que seja capaz de contar palavras em um ou mais ficheiros, proporcionando uma solução ágil e eficiente com paralelismo na sua execução para suportar grandes volumes de dados. Mais especificamente, o programa permitirá contar palavras em três modos distintos de operação:

- `t` – Conta o número total de palavras, incluindo palavras repetidas (*i.e.*, semelhante a contar o número de palavras com o comando `wc -w`);
- `u` – Conta o número total de palavras únicas/diferentes, sem contar as palavras repetidas (*i.e.*, semelhante a encontrar todas as palavras únicas/diferentes com o comando `uniq` e contá-las com o comando `wc -w`);
- `o` – Conta o número de ocorrências de cada uma das palavras únicas/diferentes (*i.e.*, semelhante ao comando `uniq -c` após o comando `sort`, assumindo que os ficheiros tenham uma palavra por linha).

3. Descrição do Trabalho

Pretende-se que os alunos implementem o comando `pwordcount` descrito de seguida:

NOME

`pwordcount` – calcula palavras em um ou mais ficheiros passados como argumento ao programa, escrevendo na saída a respetiva contagem pedida pelo utilizador com o nível de paralelização também escolhido pelo mesmo.

SINOPSE

```
./pwordcount [-m t|u|o] [-p n] ficheiros…
```

DESCRIÇÃO

`-m t|u|o`: opção que define o modo de contagem (apenas um destes modos por execução). O modo `t` conta o total de palavras, o modo `u` conta o total de palavras únicas/diferentes e o modo `o` conta o número de ocorrências de cada palavra nos ficheiros de entrada. Por omissão, o programa apenas conta o total de palavras (*i.e.*, modo `t`).

`-p n`: opção que define o nível de paralelização `n` do comando. Ou seja, o número de processos (filhos) que serão utilizados para efetuar as contagens. Por omissão, deve ser utilizado apenas um processo filho (*i.e.*, `n = 1`) para realizar as contagens.

`ficheiros`: podem ser passados ao programa um ou mais ficheiros, sobre os quais é efetuada a contagem.

a) Se forem passados ao programa mais do que um ficheiro, cada ficheiro é atribuído pelo processo pai a um único processo filho, não havendo assim divisão do conteúdo de um ficheiro por vários processos. Neste caso, se o valor de `n` for superior ao número de ficheiros, o processo redefine-o automaticamente para o número de ficheiros, ou seja, cria tantos processos quantos os ficheiros a pesquisar. Se existirem mais ficheiros do que `n` processos, o processo pai deverá decidir inicialmente como distribuir os ficheiros, tentando ser o mais equitativo possível (em termos do número de ficheiros atribuídos a cada processo);

b) Se for passado ao programa apenas um ficheiro, então a paralelização será feita pelo processo pai de forma que o conteúdo do ficheiro indicado seja dividido pelos `n` processos que estiverem disponíveis. Neste caso, o processo pai provavelmente terá de saber qual o tamanho do ficheiro ou quantas linhas tem o ficheiro para poder fazer uma divisão tão equitativa quanto possível.

Após validar as opções do comando, com base no número de ficheiros passados ao programa e o nível de paralelização `n`, o processo pai deve criar os processos filhos necessários. Nesta primeira fase do projeto, estes processos filhos contam as palavras nos ficheiros (no modo de contagem solicitado pelo utilizador) e escrevem os resultados diretamente na saída padrão (*stdout*). Como na primeira fase do projeto, não há comunicação nem sincronização entre os processos, o resultado impresso pelos processos filhos incluirá apenas a visão parcial e isolada de cada um, sem que o processo pai centralize e sumarize os resultados finais do programa (*e.g.*, a contagem do total de palavras deveria ser a soma da contagem individual de cada processo filho). Pelo mesmo motivo, a impressão dos resultados na saída padrão (*stdout*) por parte dos processos poderá ser intercalada (*i.e.*, os resultados de cada processo não necessariamente serão apresentados sequencialmente, sem serem intercalados com os resultados dos outros).

4. Desafios

- Como garantir que a contagem é efetuada em todos os ficheiros uma e apenas uma vez?
- Como concretizar a divisão de um ficheiro em várias partes, e como atribuir cada parte a um processo filho?
- Podem já começar a refletir sobre como garantir, na segunda fase do projeto, que os resultados enviados para a saída padrão (*stdout*) sejam escritos de forma não intercalada?
- Podem também já começar a refletir sobre a necessidade de, na segunda fase do projeto, os processos filhos retornarem algumas informações ao processo pai de modo a ser possível este calcular o resultado final do programa?

5. Ficheiros Iniciais de Teste

Juntamente com o enunciado do projeto, serão disponibilizados um ficheiro ZIP (`SO-TI-XX.zip`) com a estrutura inicial do projeto e alguns ficheiros de texto que servirão para os alunos testarem a solução do comando `pwordcount`. Os alunos terão de descarregar estes ficheiros de teste para a sua máquina. Não os deverão abrir no Moodle, pois alguns dos ficheiros podem ser bastante grandes (*e.g.*, centenas de megabytes).

6. Entrega

A entrega do trabalho é realizada da seguinte forma:

- Os grupos inscrevem-se atempadamente, de acordo com as regras afixadas para o efeito, no Moodle.
- O grupo coloca os ficheiros `pwordcount` e `pwordcount.py` do projeto numa diretoria cujo nome deve seguir exatamente o padrão `SO-TI-XX` (e.g., `SO-TI-01` ou `SO-TI-23`). Juntamente com os dois ficheiros, devem incluir um ficheiro de texto `README.txt` (não é `pdf`, `rtf`, `odt` nem `docx`) que deve conter:
    1. A identificação dos elementos do grupo;
    2. Exemplos de chamadas do comando `pwordcount`;
    3. As limitações da implementação;
    4. A abordagem usada para a divisão dos ficheiros pelos processos;
    5. Outras informações que acharem pertinente sobre a implementação do projeto.
- A diretoria será incluída num ficheiro ZIP cujo nome deve seguir exatamente o padrão grupo `SO-TI-XX.zip`, novamente onde `XX` corresponde ao número do vosso grupo. Este ficheiro deverá ser submetido no Moodle (apenas uma submissão por grupo).

De notar que o ficheiro ZIP entregue deve conter apenas a diretoria com os ficheiros (`pwordcount`, `pwordcount.py` e `README.txt`), pois qualquer outro ficheiro será ignorado.

Se não se verificar algum destes requisitos, o trabalho é considerado não entregue.
Não serão aceites trabalhos entregues por email nem por qualquer outro meio não definido nesta secção.

7. Prazo de Entrega

O trabalho deve ser entregue até o dia **05 de novembro de 2023 (domingo) às 23:59h**.

8. Avaliação dos Trabalhos

A avaliação do trabalho será realizada:

1. Pelos alunos, através do preenchimento do formulário de contribuição de cada aluno no desenvolvimento do projeto. O formulário será disponibilizado no Moodle e preenchido após a entrega do projeto.
2. Pelo corpo docente, sobre dois conjuntos de ficheiros de texto: os ficheiros de teste disponibilizados aos alunos e outros somente usados pelos docentes.

Para além dos testes a efetuar, os seguintes parâmetros serão avaliados: funcionalidade, estrutura, desempenho, algoritmia, comentários, clareza do código, validação dos parâmetros de entrada e tratamento de erros.

9. Plágios

Não é permitido aos alunos partilharem códigos com soluções, ainda que parciais, de nenhuma parte do projeto com outros alunos (nem através do Fórum da disciplina, nem por qualquer outro meio). Além disso, todos os códigos serão testados por um verificador de plágio. Caso alguma irregularidade seja encontrada, os projetos de todos os alunos envolvidos serão anulados e o caso será reportado aos órgãos responsáveis em Ciências@ULisboa.

Chamamos a atenção para o facto das plataformas generativas baseadas em Inteligência Artificial (*e.g.*, o *ChatGPT* e o *GitHub Co-Pilot*) gerarem um número limitado de soluções diferentes para o mesmo problema, as quais podem ainda incluir padrões característicos deste tipo de ferramenta e que podem ser detetáveis pelos verificadores de plágio. Desta forma, recomendamos fortemente que os alunos não submetam trechos de código gerados por este tipo de ferramenta a fim de evitar riscos desnecessários.

Por fim, é responsabilidade de cada aluno garantir que a sua *home*, as suas diretorias e os seus ficheiros de código estão protegidos contra a leitura de outras pessoas (que não o utilizador dono dos mesmos). Por exemplo, se os ficheiros estiverem gravados na sua área de aluno nos servidores de Ciências@ULisboa, então todos os itens mencionados anteriormente devem ter as permissões de acesso `700`. Se os ficheiros estiverem no *GitHub*, garantam que o conteúdo do vosso repositório não esteja visível publicamente.