**UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES**

**PRÓ-REITORIA DE ENSINO, PESQUISA E PÓS-GRADUAÇÃO**

**CAMPUS DE ERECHIM**

**DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**TIAGO CARLOS AMRGINSKI**

**TIAGO FERRARESI BELLAVER**

**WILLIAM DE ALMEIDA**

**PROGRAMA PARA DOWNLOAD DE E-MAILS**

**ERECHIM – RS**

**2019**

1. **INTRODUÇÃO**

O trabalho foi realizado seguindo sugestões abordadas em aula pelo professor, utilizando a linguagem de programação python 3.7 e a IDE PyCharm. Nosso programa é capaz de realizar login em uma conta de e-mails e realizar download dos e-mails que não foram lidos para uma pasta dentro do projeto.

1. **OBJETIVO**

O principal objetivo do trabalho foi a realização de um programa capaz de realizar login em uma conta de e-mail criada para o trabalho e efetuar o download de e-mails não lidos através do protocolo IMAP (Internet Message Access Protocol) que é o protocolo de e-mail que acessa a caixa de e-mail e sincroniza todas as caixas, pastas e sub-pastas da conta de e-mail. A utilização do IMAP foi escolhida pois é possível executar rotinas de backup no servidor para recuperação da conta, diferente do POP3 que baixa e deleta do servidor cada e-mail recebido.

1. **MODELAGEM**

Nosso programa foi modelado seguindo a linguagem de programação aprendida durante a cadeira de Linguagem de Programação 4, o Python 3.7, para realizar a programação utilizamos a IDE PyCharm que possuí compilador para realizar os testes em nosso software.

Em nosso código buscamos deixar o mais simples possível, utilizando recursos de fácil entendimento.

1. **RAZÃO DA UTILIZAÇÃO DE THREADS**

Threads são componentes de um processo, que podem ser executados paralelamente. Pode haver vários encadeamentos em um processo, e eles compartilham o mesmo espaço de memória, ou seja, o espaço de memória do processo pai. Isso significaria que o código a ser executado, bem como todas as variáveis ​​declaradas no programa, seriam compartilhadas por todas threads.

* Threads executados no mesmo espaço de memória; processos têm memória separada.
* Seguindo o ponto anterior: o compartilhamento de objetos entre threads é mais fácil, mas por outro lado é necessário tomar medidas extras para sincronizar objetos, para garantir que duas threads não gravem no mesmo objeto ao mesmo tempo e que uma condição de corrida não ocorre.
* Devido às limitações estabelecidas pelo GIL no Python, os threads não podem alcançar um verdadeiro paralelismo utilizando vários núcleos de CPU. O multiprocessamento não possui essas restrições.
* O agendamento de processos é tratado pelo sistema operacional, enquanto o agendamento de threads é feito pelo intérprete Python.
* Os processos filhos são interruptivos e elimináveis, enquanto os segmentos filho não são. É necessário aguardar o término dos threads ou join.

1. **COMPARAÇÃO DE TEMPO**

Para realizar as comparações de tempo de execução entre os códigos fizemos dois testes no qual o primeiro executamos ambos códigos com apenas 2 e-mails na caixa de entrada. Já o segundo teste executamos novamente os dois códigos, mas dessa vez com 50 e-mails na caixa de entrada para serem baixados. Os tempos de velocidade encontram-se abaixo na tabela.

Foi possível observar que o código com thread é muito mais eficaz e veloz que o código sequencial para grandes quantidades de e-mails. Para baixar apenas 2 e-mails o código sequencial levou 17.03 segundos enquanto o código com thread levou 26.05 segundos. Para a quantidade 50 e-mails o código com Thread se mostrou mais rápido, levando apenas 52.57 segundos para baixar os e-mails enquanto o código sequencial levou 72.06 segundos para baixar.

1. **CÓDIGO DO PROGRAMA**

Em anexo segue cópia de nossos códigos com programação sequencial e com programação thread.



Figura 1 - Código Sequêncial



Figura 2 - Código com Thread