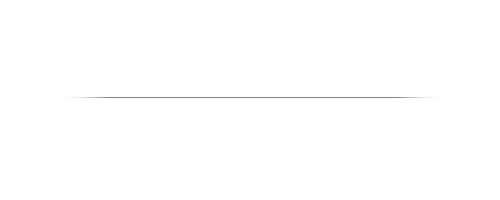
Trabalho Prático de Estruturas de Dados

*Mensagens Codificadas: Sequências Equidistantes de Símbolos*

Relatório



João Pedro da Cruz Brito

a37880

Índice

* **Objetivo;** *(pág. 3)*
* **Evolução do trabalho;** *(pág. 4 e 5)*
* ***Preview* dos menus e funcionamento;** *(pág. 6,7 e 8)*
* **Ferramentas/conhecimentos aplicados;** *(pág. 9)*
* **Principais dificuldades encontradas;** *(pág. 10)*
* **Conclusão.** *(pág. 11)*

Objetivo

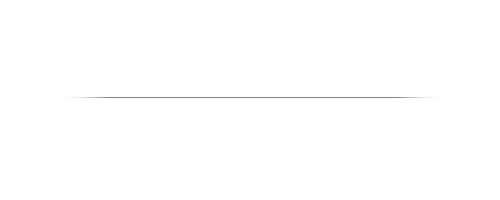
Este trabalho tem como objetivo **procurar, numa base de dados de 100 mensagens trocadas por terroristas** (cada uma com mais de 1 milhão de carateres), **palavras-chave** previamente definidas como sendo perigosas.

Contudo, estas ***keywords*** encontram-se codificadas, por um processo chamado **“Sequências Equidistantes de Símbolos”**, no qual as letras de cada ***keyword* são separadas**, colocando-se carateres aleatórios pelo meio. O espaçamento é constante e definido essencialmente pelas 2 primeiras letras.

***(exemplo)***

*(…)* t u **b** i p l e e **o** b g n r u **m** h h n o o **b** q s l e **a** i w f *(…)*

***espaçamento 6\****

Neste exemplo, ter-se-ia a *keyword* “bomba”, se a mensagem fosse lida com espaçamento 6

*\*pode-se considerar espaçamento 5, mas eu* ***defini o espaçamento tal como o exemplo mostra***

Evolução do trabalho

O trabalho em questão passou por várias revisões e mudanças, principalmente na forma como a pesquisa é feita.

Há 2 etapas cruciais para o funcionamento deste programa: a **forma como os dados são guardados** (carateres das mensagens) e a **pesquisa das *keywords***.

**Forma de guardar os carateres**

(Árvore Binária)

Vetor com as posições de cada carater numa dada mensagem\*

**m**

5 7 13 21 22 26 30 (…)

**o**

**a**

**b**

**e**

**d**

**c**

**j**

**g**

**l**

**k**

**i**

**u**

**x**

**z**

**y**

**v**

**w**

**s**

**q**

**r**

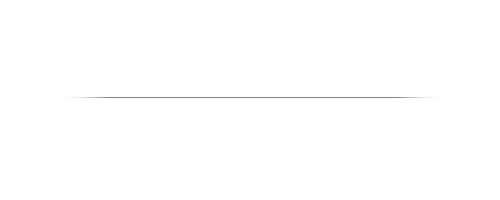
**p**

**t**

**f**

**n**

**h**



*\*o mesmo* ***acontece para cada carater na árvore****; os vetores* ***mudam de mensagem para mensagem***

Quanto à pesquisa das *keywords*, ou melhor, dos seus carateres, vendo sempre se estavam igualmente espaçados, houve 2 versões:

**Versão 1.0**

Chamava repetidamente funções (a maioria recursivas) para **obter os vetores das posições dos carateres e comparava as posições**. Se tudo corresse bem, prosseguia para a próxima *keyword*. Se não, avançava com a primeira letra (em termos de posição) e refazia tudo outra vez. Contudo, a **pesquisa era extremamente lenta** e com **muitos passos desnecessários** e chamadas a funções. Assim, era preciso uma versão 2.0.

**Versão 2.0**

Será mais fácil uma representação visual do que meramente textual:

**pointer**

7 50 52 69 70 71 80 (…)

\_\_ \_\_ \_\_ \_\_ \_\_

5 7 13 21 22 26 30 (…)

3 4 10 34 37 40 44 (…)

1 6 14 15 16 23 24 (…)

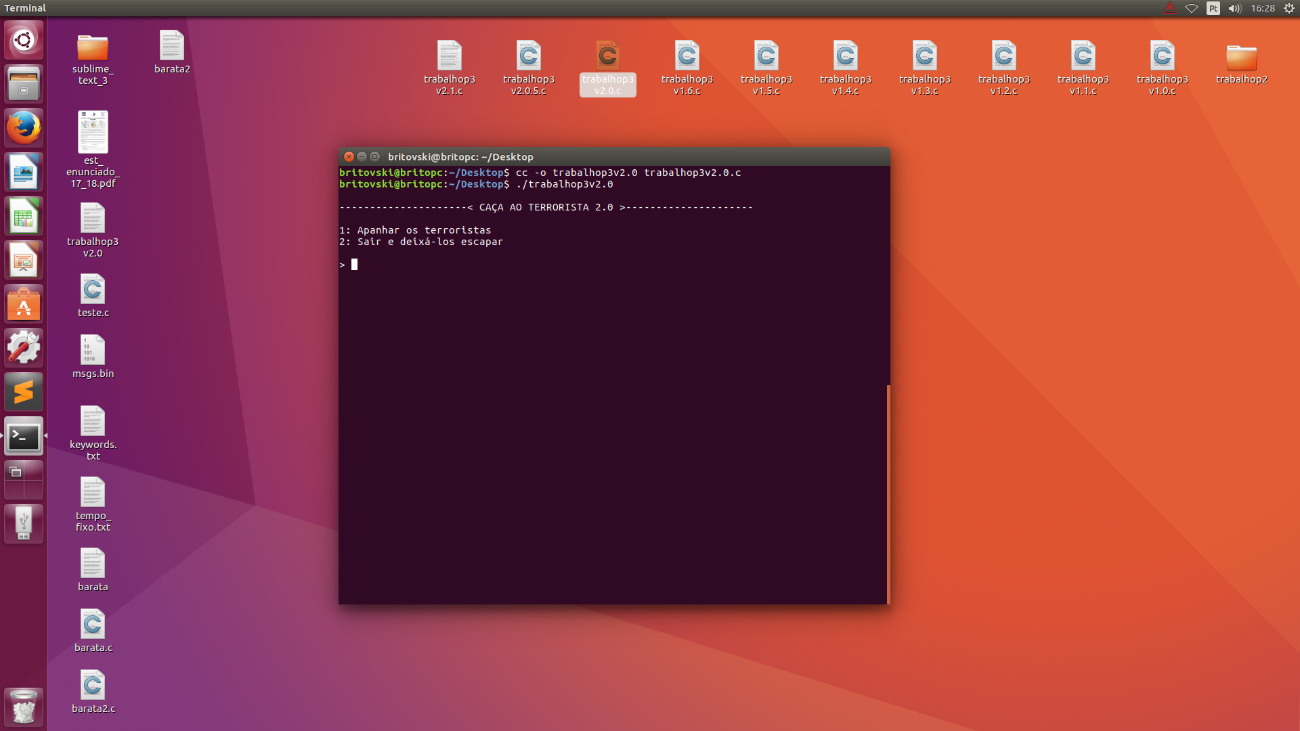
2 9 11 19 20 29 33 (…)

Imagine-se uma *slot machine*, em que as posições são “vetores verticais”, mexendo-se para baixo ou para cima. Neste programa, se as atuais posições forem ideais (igualmente espaçadas), encontramos uma *keyword*.

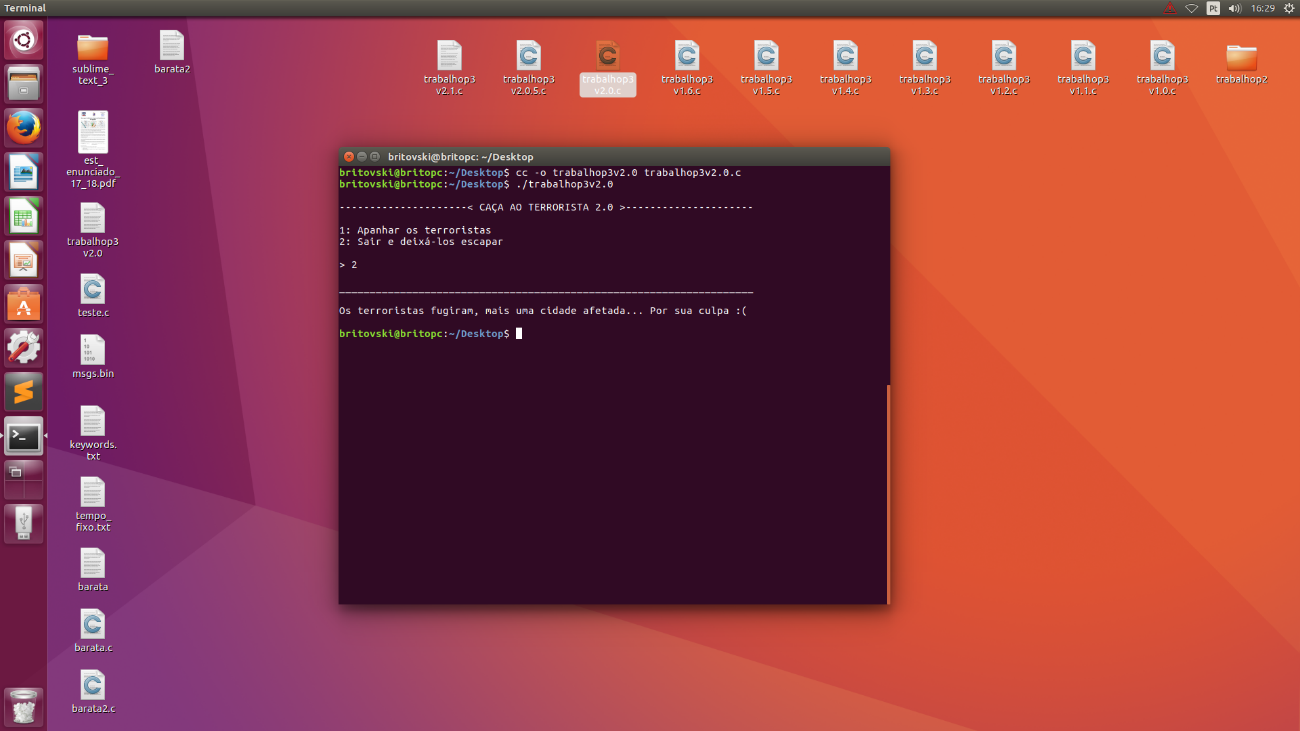
Se não, **mexem-se os vetores** e verifica-se outra vez. Note-se que o programa tem atenção ao tamanho dos vetores, nunca deixando nenhum “rebentar”.

*Preview* dos menus e funcionamento

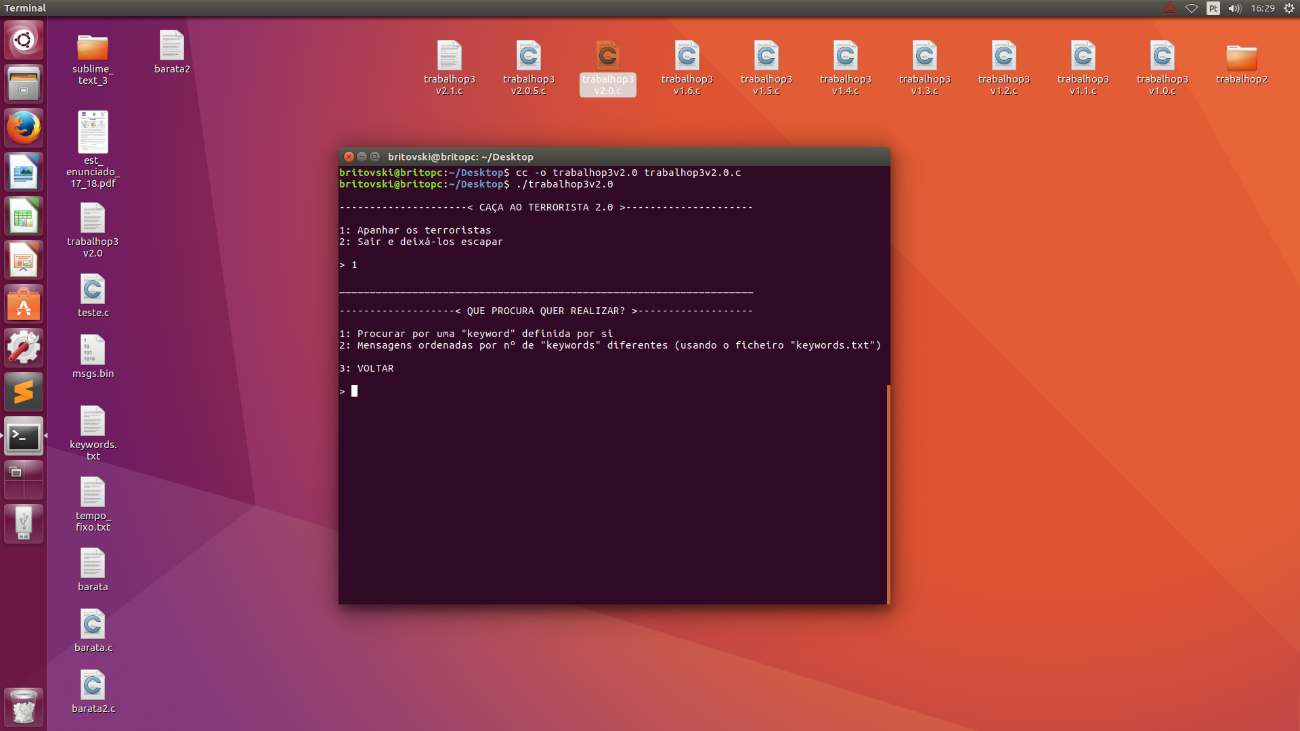
Breve visão geral dos menus e estrutura do programa:



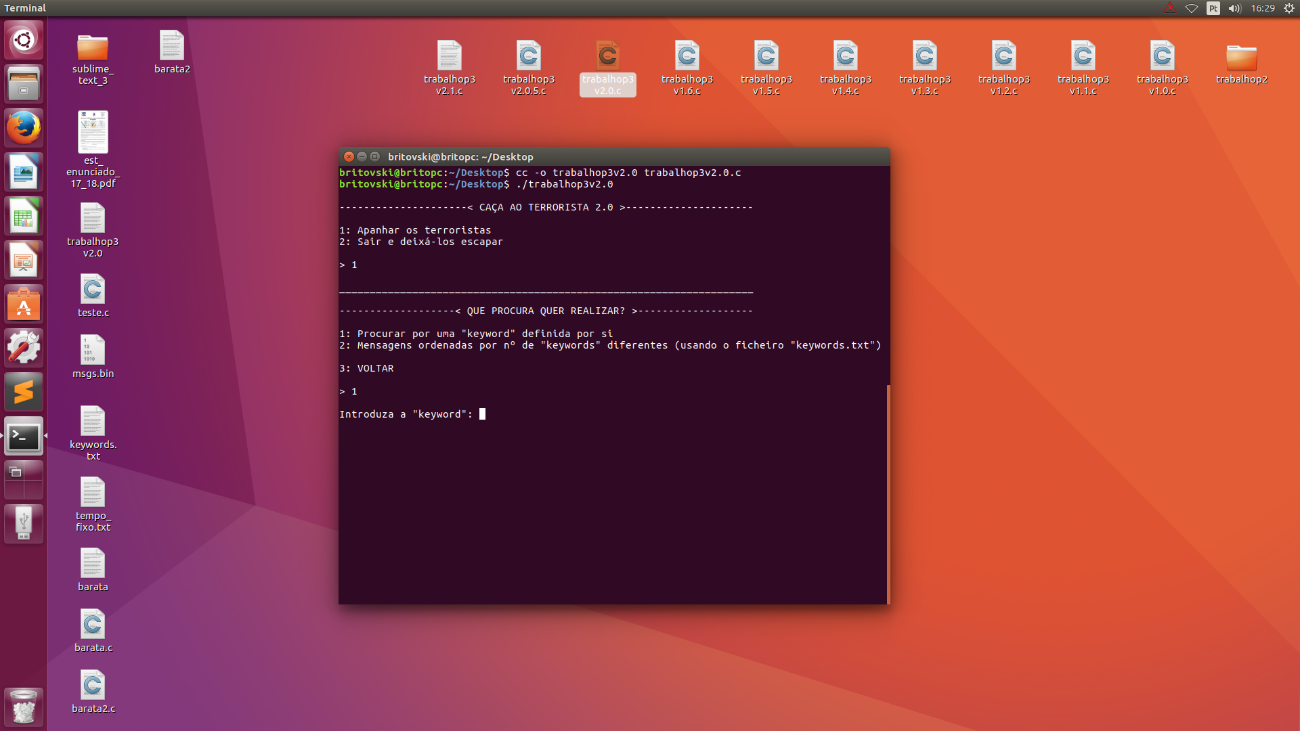
**Menu Principal**



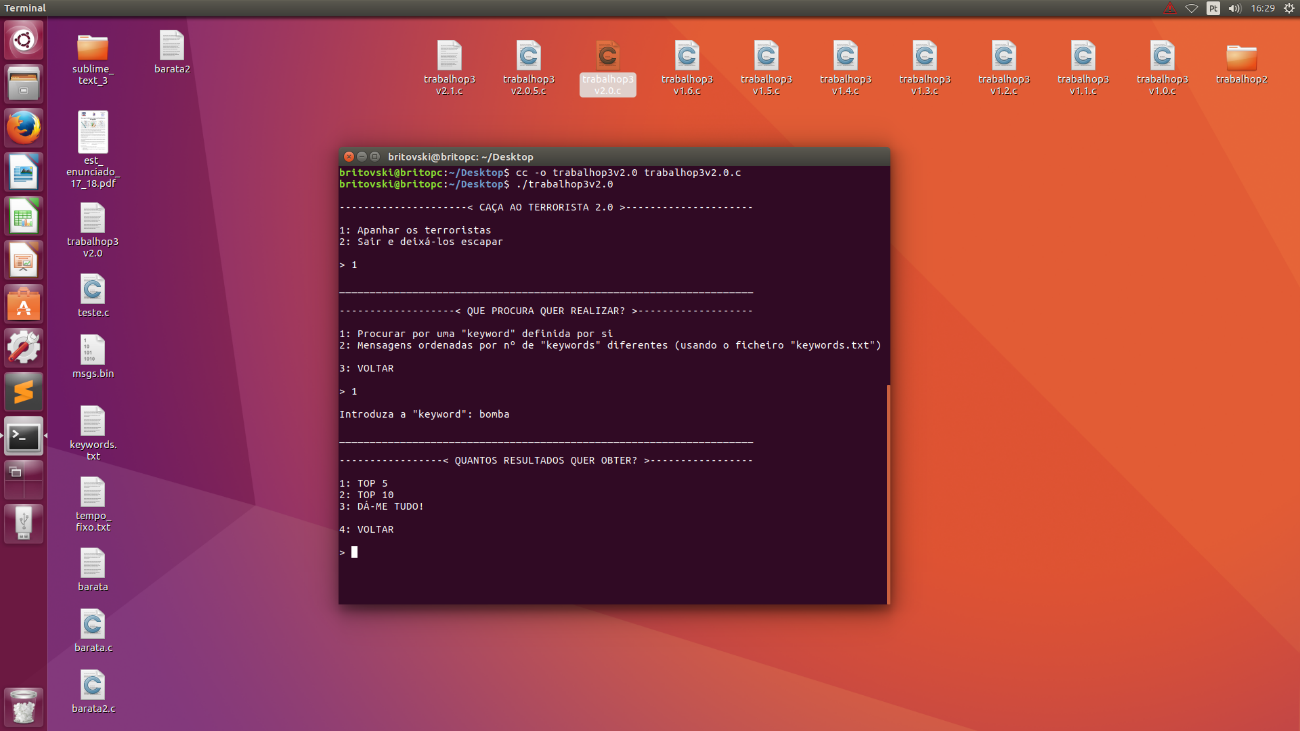
**Sair do Programa**



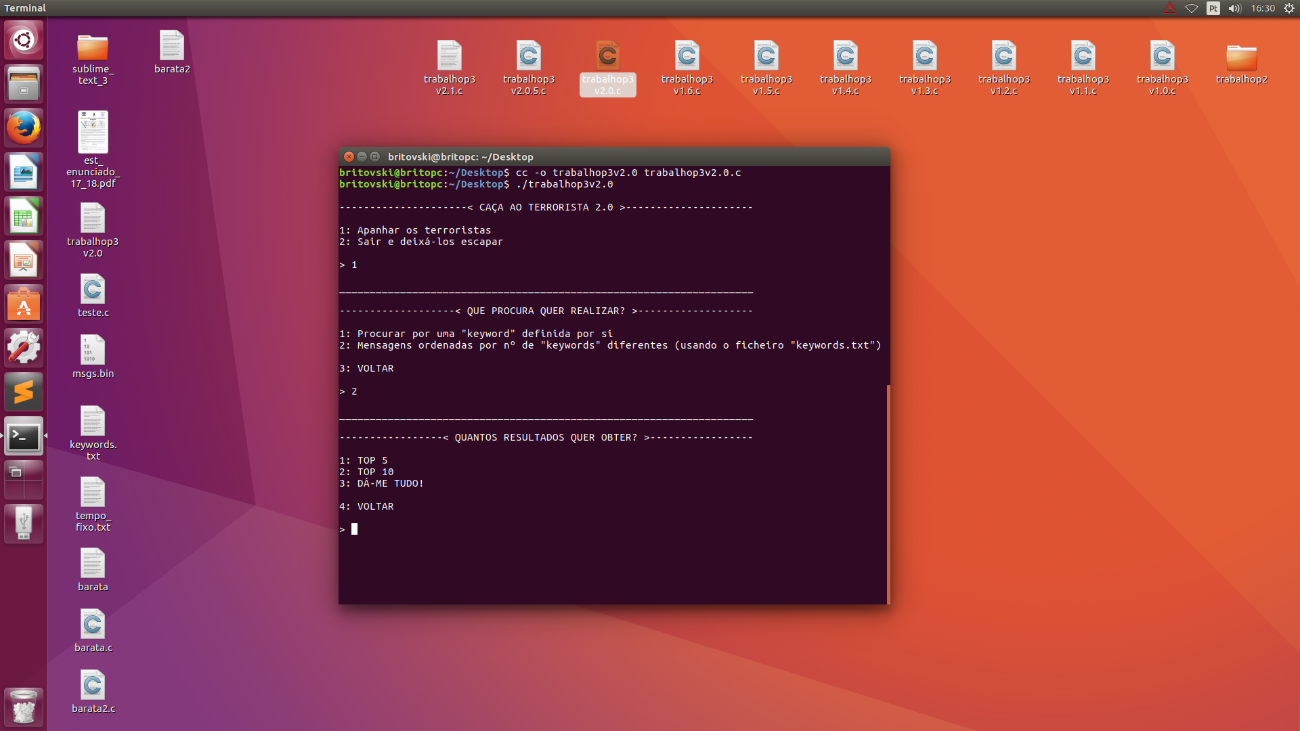
**Escolha entre a procura de uma *keyword* que o utilizador poderá introduzir ou procura nas mensagens, das *keywords* (no ficheiro “keywords.txt”), e ordená-las consoante o total de *matches***



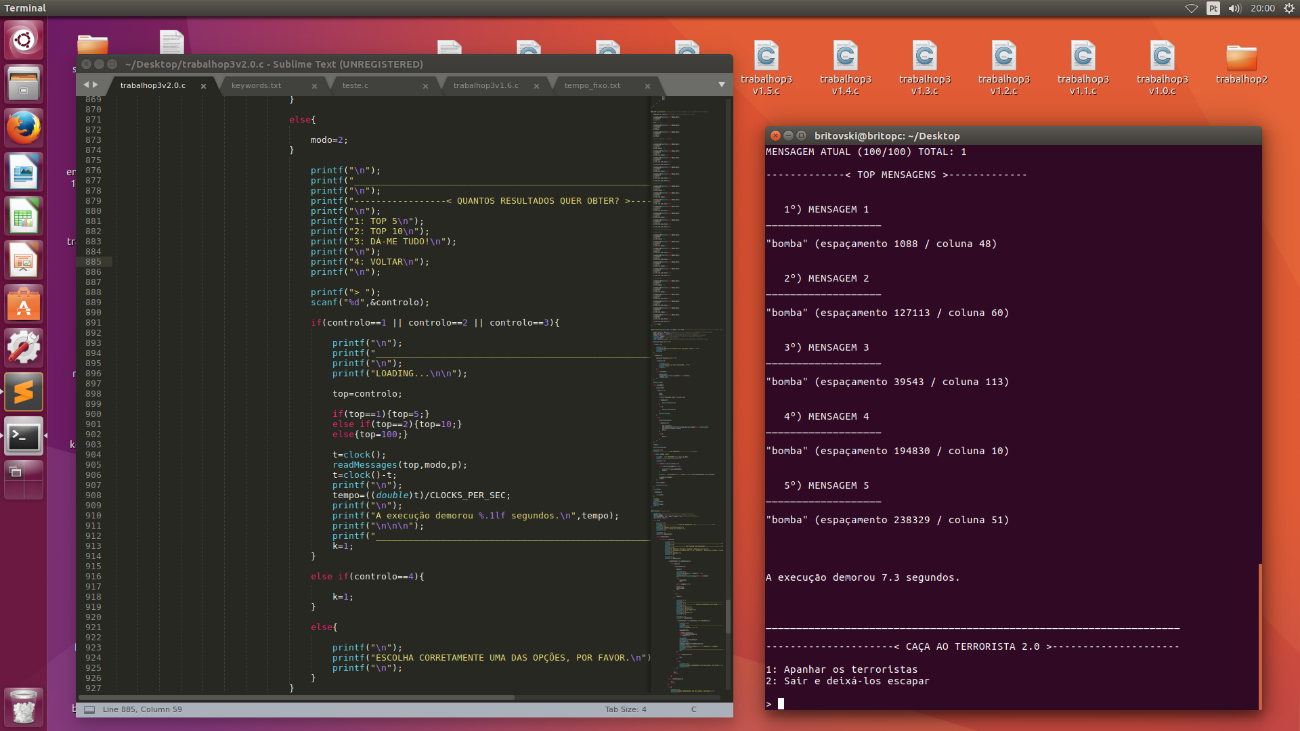
**O utilizador pode introduzir a sua *keyword***



**Introduzida a *keyword*, o utilizador pode escolher quantos resultados pretende**



**Se escolheu a pesquisa com o ficheiro “keywords.txt”, o utilizador pode agora escolher quantos resultados pretende**



**Por fim, o programa mostra os resultados, de acordo com a(s) *keyword(s)* a procurar e com o número de resultados que o utilizador pediu**

Ferramentas/conhecimentos aplicados

Neste trabalho foram aplicados os seguintes conhecimentos (da UC de Estruturas de Dados e anteriores):

* **Árvores Binárias** de Pesquisa;
* **Listas** Duplamente Ligadas;
* **Recursividade**;
* **Manipulação de um elevado volume de dados**;
* Alocação Dinâmica de Memória;
* Otimização de funções e preocupação com o tempo de resposta;
* (…)

Principais dificuldades encontradas

Os grandes obstáculos deste desafio/projeto prendem-se com a maneira como se devem tratar os dados, principalmente num volume grande.

A pesquisa foi de facto uma “pequena dor de cabeça”, pois houve necessidade de pensar numa **solução exequível e que desse resultados num tempo aceitável**.

O armazenamento em memória dos carateres, para depois serem processados, foi outra das etapas importantes, mas como já vimos ao longo deste relatório, foi ultrapassada com relativa destreza.

Conclusão

Finalizado o trabalho e o presente relatório, resta dizer que este trabalho serviu para consolidar conhecimentos e ganhar prática com as ferramentas aprendidas na UC.

Numa nota pessoal, sinto que este trabalho (e até o relatório) demonstram uma **evolução no aproveitamento e capacidade de trabalho** em relação à anterior cadeira de Programação (II).