Processamento de Linguagens 2021/2022

Licenciatura em Ciências da Computação

Trabalho Prático nº 1: Processador de Utilizadores registados no sistema Clav

Grupo 3

Alexandre Rodrigues Balde (A70373)\ Marco Alexandre Félix de Lima (A86030)

Implementação e Relatório

Como diz no enunciado:

> N"[...]o texto 'clav-users.txt' em que campos de informação têm a seguinte ordem: nome, email, entidade, nível, número de chamadas ao backend

Assume-se que:

- cada linha do ficheiro está no seguinte formato: `{nome} :: {email} :: {entidade} :: {nivel} :: {backend_calls}`
- cada um dos campos acima é **não opcional** i.e. pode estar em branco mas têm todos que contabilizar `5`.
- entre cada um dos `5` campos há necessariamente a string `" :: "`, ou seja, há 4 ocorrências de `" :: "` em cada linha.

In [1]:
```
import re
import os
import icu
```

Para se obter o `path` do ficheiro `clav-users.txt` - que tem a informação que se pretende processar dos utilizadores do sistema `clav` - usam-se as funções do módulo `os`.

In [2]:
```
ficheiro = 'clav-users.txt'

def find_data_file(filename):
    datadir = os.path.join(os.getcwd())
    return os.path.join(datadir, filename)

clav_file = find_data_file(ficheiro)
```

PyICU

Algumas das questões pedem ordenação alfabética.\ Para tal, faz-se uso do módulo `PyICU` do Python para lidar com strings Unicode.\ Permite e.g. fazer `sort` de strings `UTF-8` sem ter que modificá-las para retirar diacríticos ou acentos.

In [3]:
```
collator = icu.Collator.createInstance(icu.Locale('pt_PT.UTF-8'))
```

Detalhes da Implementação

Considere-se uma linha qualquer do ficheiro Clav que pretendemos processar.

> `Madalena Ribeiro :: madalena.ribeiro07@gmail.com :: ent_DGLAB :: 4 :: 2`

Há alíneas diferentes que pedem para recolher listas com os mesmos elementos de cada utilizador, mas estruturadas de forma diferente. Como essas tarefas só diferem na parte da manipulação de dados e não na filtragem, para evitar repetição criou-se a ER abaixo.

A seguinte ER é usada para capturar nomes e entidades em cada uma das linhas do ficheiro lido.\ Só tem dois grupos - um para o nome, o outro para a entidade.

Ao longo do projeto usar-se-á `'[^:]+'` para capturar os campos em cada linha - é importante referir que só resultará se **nenhum** dos campos tiver o caracter `':'`, que é o que se assume nesta implementação.

In [4]:
```
nome_e_entidade = r'^([^:]+)[ ]::[ ][^:]*[ ]::[ ]([^:]*)[ ]::[ ]'
name_ent_regexep = re.compile(nome_e_entidade, re.MULTILINE)
```

Notas

1

Veja-se acima o uso da função `re.compile` para transformar uma expressão regular num Regular Expression Object.\ Isto tem vários benefícios, mas aquele que interessa neste caso é poder passar à função `re.compile` flags que regem o comportamento da expressão regular.\ Aqui, usa-se a flag `re.MULTILINE`.\ É então possível usar o símbolo `'^'` para dar "match" não só ao início da string - que seria o seu único uso sem esta flag - como também a cada linha (imediatamente após cada `'\n'`).

Sabemos que `clav-users.txt` tem um utilizador por linha, no formato descrito acima.\ A resolução deste projeto passa então por estruturar expressões regulares que trabalhem sobre uma única linha, e aplicá-las a todas as linhas através da técnica descrita.

2

Note-se que poderia ter-se utilizado uma ER como

`r'^([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)$'`

para capturar todos os campos de informação em grupos separados, mas isso significaria guardar forçosamente resultados de pesquisa irrelevantes para um dado propósito.\ Por exemplo, querendo-se o nome e a entidade como na primeira alínea, estar-se-ia a guardar todos os outros campos para todos os utilizadores.\ Neste exemplo simples de dimensões reduzidas não há problema, mas seria oneroso fazê-lo num ficheiro com $O(10^6)$ linhas.

Em alternativa, poder-se-ia fazer o processamento do ficheiro com esta ER uma só vez, mas novamente - haveria a necessidade de guardar toda\ a informação lida para uso posterior, o que não será comportável se o ficheiro for grande.

Assim sendo, decidiu-se usar um ER para cada caso do enunciado, havendo reutilização só quando ela não acarreta ERs que produzam grupos não diretamente relacionados com a pesquisa em causa.

Neste caso, há mais que uma alínea que pede **apenas** nomes e entidades de utilizador, mas organizados de forma diferente.\ Logo, esta ER pode ser usada nesses caso para filtrar nomes e entidades, sendo o procecssamento feito no resto da função, que já não pode ser comum.

A variável `read_data` terá o conteúdo do ficheiro `clav-users.txt` .\ Isto permite que o ficheiro seja lido uma só vez durante a execução completa do programa.\ Aqui tem-se em conta a dimensão reduzida do ficheiro, caso contrário seria melhor lê-lo por segmentos.

In [5]:
```
clav = open(clav_file, r'r', encoding="utf-8")
read_data = clav.read()
```

Alínea 1

A função abaixo produz uma lista de pares de strings.\ A primeira componente é o nome do utilizador no sistema, e a segunda é a sua entidade.\ Os pares estão ordenados alfabeticamente pela primeira componente, ou seja, pelo nome dos utilizadores.

In [6]:
```
def nome_e_ent():
    unicode_string_key = lambda tup: collator.getSortKey(tup[0])
    regexp_results = re.findall(name_ent_regexep, read_data)
    return sorted([(match[0], match[1]) for match in regexp_results], key = unicode_string

nome_e_ent()
```

Out[6]:
```
[('Alda do Carmo Namora Soares de Andrade', 'ent_FLUL'),
 ('Alexadre Teixeira', 'ent_KEEP'),
 ('Alexandra Lourenço', 'ent_DGLAB'),
 ('Alexandra Maria Alves Coutinho Rodrigues', 'ent_UTAD'),
 ('Alexandra Testes', 'ent_A3ES'),
 ('Alexandre Teixeira', 'ent_A3ES'),
 ('Aluno de DAW2020', 'ent_A3ES'),
 ('Ana Lúcia Cabrita Guerreiro', 'ent_CCDR-Alg'),
 ('Ana Maria Teixeira Gaspar', 'ent_SGMF'),
 ('António José Morim Brandão', 'ent_MdP'),
 ('Carlos Barbosa', 'ent_A3ES'),
 ('Carlos Matoso', 'ent_IEFP'),
 ('Cármen Isabel Amador Francisco', 'ent_CMSNS'),
 ('Cátia João Matias Trindade', 'ent_DGLAB'),
 ('Cátia Trindade', 'ent_DGLAB'),
 ('clara cristina rainho viegas', 'ent_DGLAB'),
 ('CLAV-migrator', 'ent_A3ES'),
 ('David Ferreira', 'ent_IEFP'),
 ('DAW2020-teste', 'ent_A3ES'),
 ('Design-DGLAB', 'ent_DGLAB'),
 ('Duarte Freitas', 'ent_A3ES'),
 ('Élia Cristina Viegas Pedro', 'ent_CCDR-Alg'),
 ('Fernando Manuel Antunes Marques da Silva', 'ent_STI-M'),
 ('Filipa Carvalho', 'ent_DGLAB'),
 ('Filipe Ferreira Cardoso Leitão', 'ent_CMSPS'),
 ('Formação DGLAB', 'ent_DGLAB'),
 ('Frederico Pinto', 'ent_ACSS'),
 ('jcm', 'ent_AAN'),
 ('jcr-rep-entidade', 'ent_A3ES'),
 ('Joana Braga', 'ent_IEFP'),
 ('João Paulo de Melo Esteves Pereira', 'ent_APA'),
 ('João Pimentel', 'ent_A3ES'),
 ('José Carlos Leite Ramalho', 'ent_A3ES'),
 ('José Carlos Leite Ramalho', 'ent_DGLAB'),
```

```
('José Carlos Martins', 'ent_A3ES'),
('Madalena Ribeiro', 'ent_DGLAB'),
('Madalena Ribeiro', 'ent_DGLAB'),
('Manuel Monteiro', 'ent_A3ES'),
('Maria Celeste Pereira', 'ent_DGLAB'),
('Maria José Maciel Chaves', 'ent_DGLAB'),
('Maria Leonor da Conceição Fresco Franco', 'ent_CCDR-LVT'),
('Maria Matos de Almeida Talhada Correia', 'ent_ICNF'),
('Maria Rita Gago', 'ent_DGLAB'),
('Miguel Ferreira', 'ent_KEEP'),
('Nuno Filipe Casas Novas', 'ent_CCDR-LVT'),
('octavio', 'ent_A3ES'),
('Paulo Lima', 'ent_KEEP'),
('Pedro Penteado', 'ent_DGLAB'),
('PRI2020-teste', 'ent_A3ES'),
('Regina Neves Lopes', 'ent_SGMF'),
('Ricardo Almeida', 'ent_DGEG'),
('Ricardo Canela', 'ent_BdP'),
('Rui Araújo', 'ent_II'),
('Rui Araújo Entidade', 'ent_AAN'),
('Rui Araújo Simples', 'ent_LNEC'),
('Sandra Cristina Patrício da Silva', 'ent_CMSNS'),
('Silvestre Lacerda', 'ent_DGLAB'),
('Sónia Isabel Ferreira Gonçalves Negrão', 'ent_CMABF'),
('Sónia Patrícia Pinheiro Reis', 'ent_ICNF'),
('Zélia Gomes', 'ent_DGLAB')]
```

Alínea 2

A função abaixo produz um dicionário cujas chaves são entidades presentes no ficheiro (ordenadas alfabeticamente) e cujos valores são o número de utilizadores registados em cada entidade.

In [7]:

```
def entidade_e_users():
    regexp = r'^[^:]+[ ]::[ ][^:]*[ ]::[ ]([^:]*)[ ]::[ ]'
    prog = re.compile(regexp, re.MULTILINE)
    dic = dict()
    for ent in sorted(re.findall(prog, read_data), key = collator.getSortKey):
        if ent not in dic.keys():
            dic[ent] = 1
        else:
            dic[ent] += 1
    return dic

entidade_e_users()
```

Out[7]:

```
{'ent_A3ES': 14,
 'ent_AAN': 2,
 'ent_ACSS': 1,
 'ent_APA': 1,
 'ent_BdP': 1,
 'ent_CCDR-Alg': 2,
 'ent_CCDR-LVT': 2,
 'ent_CMABF': 1,
 'ent_CMSNS': 2,
 'ent_CMSPS': 1,
 'ent_DGEG': 1,
 'ent_DGLAB': 16,
 'ent_FLUL': 1,
 'ent_ICNF': 2,
 'ent_IEFP': 3,
 'ent_II': 1,
 'ent_KEEP': 3,
 'ent_LNEC': 1,
```

```
'ent_MdP': 1,
'ent_SGMF': 2,
'ent_STI-M': 1,
'ent_UTAD': 1}
```

Alínea 3

A função abaixo produz um dicionário cujas chaves são os diferentes níveis de acesso presentes no ficheiro `clav-users.txt` e cujos valores são o número de utilizadores registados com esse nível.

A expressão regular usada para obter o nome do utilizadores é `r'^([^:]*[ ]::[ ]){3}([0-9\.]+)'` - vêm primeiro os 3 campos iniciais, e só o quarto será o nível desse utilizador.

Usar parênteses em (`r'^([^:]*[ ]::[ ]){3}`) faz com que a primeira aplicação da ER `r'^([^:]*[ ]::[ ])` seja guardada num grupo - será o nome do utilizador seguido de `' :: '`, neste caso.
Nesta alínea não se buscam os nomes dos utilizadores, mas é necessário rodear a ER por parênteses para poder aplicar o operador de repetição para se chegar ao campo que interessa: `'{3}'`.

In [8]:
```
def utilizadores_por_nivel():
    nivel_e_user = dict()
    regexp = r'^([^:]*[ ]::[ ]){3}([0-9\.]+)'
    prog = re.compile(regexp, re.MULTILINE)
    for _, num in sorted(re.findall(prog, read_data), key = lambda tup: float(tup[1])):
        if num not in nivel_e_user.keys():
            nivel_e_user[num] = 1
        else:
            nivel_e_user[num] += 1
    return nivel_e_user

utilizadores_por_nivel()
```

Out[8]:
```
{'1': 23, '2': 7, '3': 1, '3.5': 1, '4': 8, '5': 2, '6': 2, '7': 16}
```

Alínea 4

A função seguinte devolve, a partir dos dados lidos, um dicionário com entidades como chaves, e a lista de utilizadores a ela associados como valores.\ As chaves do dicionário, e as listas que são os seus valores estão ordenados alfabeticamente por ordem crescente.

In [9]:
```
def group_users_by_ent():
    unicode_string_tup_key = lambda tup: (collator.getSortKey(tup[0]), collator.getSortKey
    l = sorted([(m[1], m[0]) for m in re.findall(name_ent_regexep, read_data)], key = unic

    from itertools import groupby
    dic = {}
    for key, group in groupby(l,  lambda x: x[0]):
        dic[key] = []
        for _, name in group:
            dic[key].append(name)
    for k in dic.keys():
        print(k + ": ")
        for v in dic[k]:
            print("    " + v)

group_users_by_ent()
```

```
ent_A3ES:
    Alexandra Testes
```

```
    Alexandre Teixeira
    Aluno de DAW2020
    Carlos Barbosa
    CLAV-migrator
    DAW2020-teste
    Duarte Freitas
    jcr-rep-entidade
    João Pimentel
    José Carlos Leite Ramalho
    José Carlos Martins
    Manuel Monteiro
    octavio
    PRI2020-teste
ent_AAN:
    jcm
    Rui Araújo Entidade
ent_ACSS:
    Frederico Pinto
ent_APA:
    João Paulo de Melo Esteves Pereira
ent_BdP:
    Ricardo Canela
ent_CCDR-Alg:
    Ana Lúcia Cabrita Guerreiro
    Élia Cristina Viegas Pedro
ent_CCDR-LVT:
    Maria Leonor da Conceição Fresco Franco
    Nuno Filipe Casas Novas
ent_CMABF:
    Sónia Isabel Ferreira Gonçalves Negrão
ent_CMSNS:
    Cármen Isabel Amador Francisco
    Sandra Cristina Patrício da Silva
ent_CMSPS:
    Filipe Ferreira Cardoso Leitão
ent_DGEG:
    Ricardo Almeida
ent_DGLAB:
    Alexandra Lourenço
    Cátia João Matias Trindade
    Cátia Trindade
    clara cristina rainho viegas
    Design-DGLAB
    Filipa Carvalho
    Formação DGLAB
    José Carlos Leite Ramalho
    Madalena Ribeiro
    Madalena Ribeiro
    Maria Celeste Pereira
    Maria José Maciel Chaves
    Maria Rita Gago
    Pedro Penteado
    Silvestre Lacerda
    Zélia Gomes
ent_FLUL:
    Alda do Carmo Namora Soares de Andrade
ent_ICNF:
    Maria Matos de Almeida Talhada Correia
    Sónia Patrícia Pinheiro Reis
ent_IEFP:
    Carlos Matoso
    David Ferreira
    Joana Braga
ent_II:
    Rui Araújo
ent_KEEP:
```

```
    Alexadre Teixeira
    Miguel Ferreira
    Paulo Lima
ent_LNEC:
    Rui Araújo Simples
ent_MdP:
    António José Morim Brandão
ent_SGMF:
    Ana Maria Teixeira Gaspar
    Regina Neves Lopes
ent_STI-M:
    Fernando Manuel Antunes Marques da Silva
ent_UTAD:
    Alexandra Maria Alves Coutinho Rodrigues
```

Alínea 5

A seguinte função devolve os indicadores pedidos no ponto 5. do enunciado do projeto 3.\ São:

- O número de utilizadores do sistema
- O número de entidades no sistema
- A distribuição em número no sistema
- A distribuição em número por nível

Notas

- Reutilizam-se algumas das funções anteriores, e para fazer as contagens, `len` ao resultado de `re.findall`.
- Para contar elementos que se podem repetir, inserem-se os resultados de `re.findall` num `set`, ao qual se faz-se `len`.

In [10]:
```
def indicadores():
    regexp_usernum = r'^([^:]*)[ ]::[ ]'
    prog_users = re.compile(regexp_usernum, re.MULTILINE)
    quantos_users = len(re.findall(prog_users, read_data))
    regexp_ent_num = r'^([^:]*[ ]::[ ]){2}([^:]*)'
    prog_ents = re.compile(regexp_ent_num, re.MULTILINE)
    quantas_ents = len({ e for _, e in re.findall(prog_ents, read_data)})
    dic_eu = entidade_e_users()
    dic_nu = utilizadores_por_nivel()

    return quantos_users, quantas_ents, dic_eu, dic_nu

indicadores()
```

Out[10]:
```
(60,
 22,
 {'ent_A3ES': 14,
  'ent_AAN': 2,
  'ent_ACSS': 1,
  'ent_APA': 1,
  'ent_BdP': 1,
  'ent_CCDR-Alg': 2,
  'ent_CCDR-LVT': 2,
  'ent_CMABF': 1,
  'ent_CMSNS': 2,
  'ent_CMSPS': 1,
  'ent_DGEG': 1,
  'ent_DGLAB': 16,
  'ent_FLUL': 1,
  'ent_ICNF': 2,
```

```
 'ent_IEFP': 3,
 'ent_II': 1,
 'ent_KEEP': 3,
 'ent_LNEC': 1,
 'ent_MdP': 1,
 'ent_SGMF': 2,
 'ent_STI-M': 1,
 'ent_UTAD': 1},
{'1': 23, '2': 7, '3': 1, '3.5': 1, '4': 8, '5': 2, '6': 2, '7': 16})
```

Alínea 6

As funções abaixo são usadas para a última alínea do projeto 3, que é imprimir um número de registos do ficheiro Clav num novo ficheiro em formato JSON.

In [11]:
```
def dict_transform(attr_list):
    attr_name = ["nome", "email", "entidade", "nivel", "backend_calls"]
    attr_num = len(attr_name)
    attr_json = dict()
    for i in range(attr_num):
        attr_json[attr_name[i]] = attr_list[i]
    return attr_json
```

Note-se o uso do módulo `json` , e de `re.split` duas vezes:

- a primeira `"\n+"` para separar as linhas pretendidas
- a segunda com `r'[ ]?::[ ]?'` para separar cada linha nos campos que a compões

In [12]:
```
def vinte_JSON(n = 20, prnt = False):
    # quantos utilizadores escrever no ficheiro JSON
    entries = re.split("\n+", read_data, maxsplit=n)
    # Só queremos os 20 primeiros resultados.
    entries = entries[:n]
    split_entries = [re.split(r'[ ]?::[ ]?', entry) for entry in entries]

    json_data = [dict_transform(split_entry) for split_entry in split_entries]
    filename = 'data.json'
    pathname = os.path.join(os.path.join(os.getcwd()), filename)
    import json
    with open(pathname, 'w+', encoding='utf-8') as f:
        json.dump(json_data, f, ensure_ascii=False, indent=4)
    if prnt:
        print("Dados JSON escritos com sucesso no ficheiro.")
        print(json.dumps(json_data, indent=4))

vinte_JSON(prnt = True)
```

```
Dados JSON escritos com sucesso no ficheiro.
[
    {
        "nome": "\u00c9lia Cristina Viegas Pedro",
        "email": "epedro@ccdr-alg.pt",
        "entidade": "ent_CCDR-Alg",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Forma\u00e7\u00e3o DGLAB",
        "email": "lurdes.almeida@dglab.gov.pt",
        "entidade": "ent_DGLAB",
        "nivel": "3.5",
```

```
        "backend_calls": "0"
    },
    {
        "nome": "Nuno Filipe Casas Novas",
        "email": "nuno.novas@ccdr-lvt.pt",
        "entidade": "ent_CCDR-LVT",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "S\u00f3nia Patr\u00edcia Pinheiro Reis",
        "email": "sonia.reis@icnf.pt",
        "entidade": "ent_ICNF",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "S\u00f3nia Isabel Ferreira Gon\u00e7alves Negr\u00e3o",
        "email": "sonia.negrao@cm-albufeira.pt",
        "entidade": "ent_CMABF",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Filipe Ferreira Cardoso Leit\u00e3o",
        "email": "arquivo@cm-spsul.pt",
        "entidade": "ent_CMSPS",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Ana L\u00facia Cabrita Guerreiro",
        "email": "alucia@ccdr-alg.pt",
        "entidade": "ent_CCDR-Alg",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Alda do Carmo Namora Soares de Andrade",
        "email": "aandrade@letras.ulisboa.pt",
        "entidade": "ent_FLUL",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Ricardo Almeida",
        "email": "ricardo.almeida@dgeg.gov.pt",
        "entidade": "ent_DGEG",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Sandra Cristina Patr\u00edcio da Silva",
        "email": "spatricio@mun-sines.pt",
        "entidade": "ent_CMSNS",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "C\u00e1tia Jo\u00e3o Matias Trindade",
        "email": "catia.trindade@dglab.gov.pt",
        "entidade": "ent_DGLAB",
        "nivel": "4",
        "backend_calls": "0"
    },
    {
```

```
        "nome": "Ricardo Canela",
        "email": "tyty@tyty.pt",
        "entidade": "ent_BdP",
        "nivel": "3",
        "backend_calls": "0"
    },
    {
        "nome": "C\u00e1tia Trindade",
        "email": "matiasjcatia@gmail.com",
        "entidade": "ent_DGLAB",
        "nivel": "4",
        "backend_calls": "0"
    },
    {
        "nome": "Miguel Ferreira",
        "email": "mferreira@keep.pt",
        "entidade": "ent_KEEP",
        "nivel": "7",
        "backend_calls": "0"
    },
    {
        "nome": "Fernando Manuel Antunes Marques da Silva",
        "email": "fernando.marques.silva@marinha.pt",
        "entidade": "ent_STI-M",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Ana Maria Teixeira Gaspar",
        "email": "ana.gaspar@sgmf.gov.pt",
        "entidade": "ent_SGMF",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Maria Matos de Almeida Talhada Correia",
        "email": "MariaMatos.Correia@icnf.pt",
        "entidade": "ent_ICNF",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "C\u00e1rmen Isabel Amador Francisco",
        "email": "carmem@mun-sines.pt",
        "entidade": "ent_CMSNS",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Maria Leonor da Concei\u00e7\u00e3o Fresco Franco",
        "email": "leonor.mina@ccdr-lvt.pt",
        "entidade": "ent_CCDR-LVT",
        "nivel": "1",
        "backend_calls": "0"
    },
    {
        "nome": "Maria Rita Gago",
        "email": "m-rita.gago@dglab.gov.pt",
        "entidade": "ent_DGLAB",
        "nivel": "6",
        "backend_calls": "3"
    }
]
```

Execução do programa

In [13]:
```
prompt = '\nEscolha a opção que pretende:\n' \
    '    1: Imprimir listagem com nome e entidade, ordenada alfabeticamente por nome.\n' \
    '    2: Imprimir lista ordenada alfabeticamente das entidades referenciadas, indicando
    '    3: Imprimir distribuição de utilizadores por níveis de acesso.\n' \
    '    4: Imprimir listagem dos utilizadores, agrupados por entidade, ordenada primeiro
    '    5: Imprime:\n' \
    '         a. Quantos utilizadores há.\n' \
    '         b. Quantas entidades há.\n' \
    '         c. A distribuição em número por entidade.\n' \
    '         d. A distribuição em número por nível.\n' \
    '    6: Coloca as n primeiras entradas num ficheiro JSON.\n' \
    '    String vazia: sair do programa.\n'
```

In [14]:
```
def main():
    inputFromUser = input(prompt)
    def print_nomes_e_ents(list):
            for (name, ent) in list:
                print (name + " :: " + ent)
    def print_user_level_dict(dic):
        for num, users in dic.items():
            print(num + ": " + str(users) + " ocorrências.")
    def print_entity_user_dic(dic):
        for e, n in dic.items():
            print(e + ": " + str(n) + " ocorrências.")
    while inputFromUser != "":
        match inputFromUser:
            case '1':
                l = nome_e_ent()
                print_nomes_e_ents(l)
            case '2':
                dic = entidade_e_users()
                print_entity_user_dic(dic)
            case '3':
                nivel_e_user = utilizadores_por_nivel()
                print_user_level_dict(nivel_e_user)
            case '4':
                group_users_by_ent()
            case '5':
                quantos_users, quantas_ents, dic_eu, dic_nu = indicadores()
                print("O sistema tem {} utilizadores.".format(quantos_users))
                print("O sistema tem {} entidades.".format(quantas_ents))
                print("\n\nDistribuição em número de entidades:")
                print_entity_user_dic(dic_eu)
                print("\n\nDistribuição em número por nível.")
                print_user_level_dict(dic_nu)
            case '6':
                print("Insira o número de entradas a escrever em JSON.")
                n = int(input())
                vinte_JSON(n)
            case '':
                return
            case _:
                print ("\nTente novamente.\n")
        inputFromUser = input(prompt)
```

In [15]:
```
if __name__ == "__main__":
    main()
```

Filtros extra

Aqui recalculam-se os indicadores da Alínea 6, usando a ER `r'^([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)$'` que captura todos os grupos de um utilizador.

In [16]:

```
def indicadores_2():
    all_fields_regexp = r'^([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^:]+)[ ]::[ ]([^
    all_fields = re.compile(all_fields_regexp, re.MULTILINE)
    lst = re.findall(all_fields, read_data)
    quantos_users = len(lst)
    quantas_ents = len({ent for _, _, ent, _, _ in lst})

    dic_ent = dict()
    for ent in sorted([e for _, _, e, _, _ in lst], key = collator.getSortKey):
        if ent not in dic_ent.keys():
            dic_ent[ent] = 1
        else:
            dic_ent[ent] += 1

    nivel_e_user = dict()
    for _, num in sorted([(user, lvl) for user, _, _, lvl, _ in lst], key = lambda tup: fl
        if num not in nivel_e_user.keys():
            nivel_e_user[num] = 1
        else:
            nivel_e_user[num] += 1

    return quantos_users, quantas_ents, dic_ent, nivel_e_user

indicadores_2()
```

Out[16]:

```
(60,
 22,
 {'ent_A3ES': 14,
  'ent_AAN': 2,
  'ent_ACSS': 1,
  'ent_APA': 1,
  'ent_BdP': 1,
  'ent_CCDR-Alg': 2,
  'ent_CCDR-LVT': 2,
  'ent_CMABF': 1,
  'ent_CMSNS': 2,
  'ent_CMSPS': 1,
  'ent_DGEG': 1,
  'ent_DGLAB': 16,
  'ent_FLUL': 1,
  'ent_ICNF': 2,
  'ent_IEFP': 3,
  'ent_II': 1,
  'ent_KEEP': 3,
  'ent_LNEC': 1,
  'ent_MdP': 1,
  'ent_SGMF': 2,
  'ent_STI-M': 1,
  'ent_UTAD': 1},
 {'1': 23, '2': 7, '3': 1, '3.5': 1, '4': 8, '5': 2, '6': 2, '7': 16})
```