Relatório Laboratórios de Informática 3 Grupo 58

Nuno Miguel Leite da Costa A96897
Daniel Jose Silva Furtado A97327
Nuno Miguel Sanches Rocha Peixoto A93244

1 Resumo

Este relatório tem como objetivo documentar as estratégias que utilizamos e as limitacações que encontramos com o primeiro guião da disciplina de Laboratórios de Informática 3 sobre parsing de dados e gestão de memória em C.

Primeiro será feita uma introdução ao projeto com uma descrição do que foi proposto. Depois será apresentada a implementação das tecnicas utlizadas para resolver os exercicios e serão discutidos alguns problemas na implementação .

2 Introdução

No âmbito de UC de Laboratórios de Informática 3 foi-nos pedido desenvolver dois exercicios, o primeiro que descartava os registos incorretos , enquanto que o segundo baseiava-se na filtragem do cruzamento dos ficheiros do exercicio anterior . Para tal foram desenvolvidos várias funções para poder realizar ambos exercicios. Para a realização deste trabalho foi necessário aplicar os conhecimentos que estudamos no ano passado na cadeira de Programação Imperativa, bem como familiarizarmo-nos com a manipulação de ficheiros. Iremos apresentar toda a linha de raciocino para a concretização do guião 1.

3 Exercicio 1

3.1 Descrição informal do exercicio

Este projeto como foi referido anteriormente foca-se na remoção de registos que se encontrem inválidos e escreve num ficheiro aqueles que se encontrem válidos. Adjacente a este exercicio, obtivemos 3 ficheiros csv responsáveis pelos formatos para cada respetivo campo.

Ficheiros

users.csv
commits.csv
repos.csv

3.2 Estratégia utilizada

A solução que escolhemos para resolução deste exercicio começou por criar uma função que ao ler uma linha em cada ficheiro separava as informações . Após seperarmos as informações analisamos indivudualmente cada uma e

verificamos se ela era válida, ou seja, correspondia a todos os aspetos pretendidos. Com a ajuda de uma variável atribuímos o valor 0 caso fosse válida e 1 caso contrário. No final somavamos todos os valores obtidos ao analisar cada informação da linha e caso fosse 0 então a linha era válida, se fosse 1 a linha era inválida.

```
int structdate(char *string){
    char *date= strdup(string);
    int len = strlen(date);
    if (len != 19) return 1;
    char *t1 = &date[4]; // posição onde se econtra o '-' da data;
    char *t2 = &date[7]; // posição onde se econtra o '-' da data;
    char *p1 = &date[13]; // posição onde se econtra o ':' da data;
    char *p2 = &date[16]; // posição onde se econtra o ':' da data;

    if ((*t1 == '-') && (*t2 == '-')  && (*p1 == ':') && (*p2 == ':')) return 0;
    else return 1;
}


int numdate(char *string){
    int ano, mes, dia, hora, min, seg;
    ano = mes = dia = hora = min = seg = 100;
    char *date = strdup(string);
    char *p = &date[18];
    int v = numvalid(p);
    if (v != 0) return 1;
    int k = structdate(date);
    if (k != 0) return 1;
    sscanf(date, "%d-%d-%d %d:%d:%d", &ano, &mes, &dia, &hora, &min, &seg);
    if ((ano == 100) || (mes == 100) || (dia == 100) || (hora == 100) || (min == 100) || (seg == 100)) return 1;
    else return 0;
}


// Função que testa se a data é válida
int datevalid(char *date){
    int k = numdate(date);
    if (k != 0) return 1;

    char *fstdate = "2005-04-07 00:00:00";

    struct tm fst = {0};
    struct tm t = {0};
    time_t currdate;
    time (&currdate);

    strptime(date, "%Y-%m-%d %H:%M:%S", &t);
    time_t stringt = mktime(&t);

    strptime(fstdate, "%Y-%m-%d %H:%M:%S", &fst);
    time_t fsttime = mktime(&fst);

    int diff1 = difftime (currdate,stringt);
    int diff2 = difftime (stringt, fsttime);

    if ((diff1 > 0) && (diff2 > 0)) return 0;
    else return 1;
}
```

1. Funções que demonstram a estratégia utilizada com a data

3.2.1 Limitações

As principais limitações que encontramos inicialmente foi conseguir construir um codigo que permitisse a resolução do exercicio e que fosse ao mesmo tempo eficaz. Por fim, outra limitação baseia-se com o uso de memória. Tivemos de gerir melhor a memória, uma vez que, existem ficheiros com um milhão de usuários ou casos que o próprio usuário apresentava muitos caracteres. Desta forma, este problema inclinou-nos para a memória dinâmica em vez da memória estática.