



Universidade de São Paulo

Instituto de Ciências Matemáticas e de
Computação

Trabalho Prático I

Organização de Arquivos – SCC0215

Docente: Cristina Dutra de Aguiar Ciferri

PAE: Lucas de Carvalho Scabora

Alunos (Turma B):	Aulos Plautius Martines Marino	7986409
	Paulo Guarnier De Mitri	3555888
	Werik Amaral Fachim	7656512
	Wesley Tiozzo	8077925

Índice

Introdução.....2

Seção 1:

Detalhe dos campos.....3

Seção 2:

Funcionamento e Funções.....4

Seção 3:

Arquivos.....8

Seção 4:

Makefile.....9

Introdução

Na disciplina de organização de arquivos aprendemos conceitos introdutórios a manipulação de registros num arquivo. Com base neles fizemos um gerenciador de tweets que conta com armazenamento, exibição, remoção e busca de entradas.

Codificado em linguagem C, nosso programa é capaz de processar uma entrada do usuário conforme os requisitos do trabalho. O código, apesar de modularizado, ficou extenso e complexo em algumas partes, porém eficiente. Notamos que esse trabalho poderia ser melhor abordado se usasse os conceitos de programação orientada a objetos. Se isso fosse feito, a parte de planejamento e engenharia do software em si ganhariam qualidade. Muitas vezes há repetição de procedimentos ou é necessário interações onde uma estrutura de classes seria bem aplicada.

Atingimos o objetivo do trabalho e nesse relatório descrevemos decisões e outras partes do nosso projeto.

Seção 1:

Detalhes dos Campos

Cada campo do registro tinha um valor estático e predefinido a nível de código. Como vemos a seguir, esse valor em geral era de um número N de casas mais um por conta do caracter nulo (\0).

Na imagem ao lado vemos a estrutura principal chamada "TWEET" e cada um de seus campos.

- "tweet": Contém a mensagem do usuário. Tem um tamanho de 141 chars. Inspirado pelo exemplo real em que um tweet tem no máximo 140 caracteres.
- "user": Contém o login ou como o usuário se identificou no sistema. O campo usuário tem 20 caracteres, não há motivos especiais para essa escolha.
- "coordinates": Contém as coordenadas que, numa aplicação real, seriam fornecidas por um navegador ou aplicativo, mas nesse caso era digitadas pelo usuário. O tamanho foi subjetivo. Poderia ter sido especificado como um par de doubles.
- "favorite_count": Contador de favoritos do tweet. Seguiu a especificação dada e não teve alterações.
- "language": Língua do tweet. Interessante para uma possível busca refinada.
- "view_count": Número de visualizações do tweet. Novamente seguiu a especificação dada.
- "retweet_count": Contador de retweets daquele tweet. Poderia ser um unsigned int, mas seguimos o que foi dado.

```
#define TWEET_TEXT_SIZE    141 //140 + \0
#define USER_SIZE          21
#define LANG_SIZE          21
#define COORD_SIZE         12

typedef struct tweet{

    char tweet[TWEET_TEXT_SIZE];
    char user[USER_SIZE];
    char coordinates[COORD_SIZE];
    int favorite_count;
    char language[LANG_SIZE];
    long int view_count;
    int retweet_count;

}TWEET;
```

Seção 2:

Funcionamento e funções

O programa possui um menu, onde há 5 opções de escolha. Em termos de código, esse menu é um "do while", onde, dependendo da requisição do usuário, é chamada uma função diferente.

Funções do Menu.h:

- void menu_add(): Adiciona um registro apos ser chamada por menu.
- void menu_exibir(): Exibe um registro conforme a escolha do usuario.
- void menu_exibir_all(): Exibe todos os registros contidos no arquivo de dados.
- void menu_buscar(): Busca um registro atraves de um campo especificado pelo usuario.
- void menu_remove(): Remove um registro por RRN.

```
MENU
1- Adicionar tweet
2- Exibir um tweet
3- Exibir todos os tweets
4- Buscar e exibir tweets pelo nome de usuario
5- Remover um tweet
0- Sair
>> 3

Você deseja exibir o RRN dos tweets? (S/N)S

Exibindo todos os tweets...

RRN: 0
User: paulo
Tweet: "bla bla bla"
Local: 22
Favorite: 44 || Retweet: 33
Total views: 2
Language: brasil

Para o próximo tweet pressione qualquer tecla...
```

```
Para o próximo tweet pressione qualquer tecla...
m
Fim dos tweets
Pressione qualquer tecla para voltar ao menu...

MENU
1- Adicionar tweet
2- Exibir um tweet
3- Exibir todos os tweets
4- Buscar e exibir tweets pelo nome de usuario
5- Remover um tweet
0- Sair
>> 4

Digite o nome do usuario: aulos

Você deseja exibir o RRN dos tweets? (S/N)S

Exibindo todos os tweets...

RRN: 2
User: aulos
Tweet: "hellow www"
Local: 33
Favorite: 5 || Retweet: 4
Total views: 2
Language: ingles

Para o próximo tweet pressione qualquer tecla...

```

```
Exibindo todos os tweets...

RRN: 0
User: spider man
Tweet: "ola ola"
Local: 23
Favorite: 3 || Retweet: 2
Total views: 1
Language: ingles

Para o próximo tweet pressione qualquer tecla...
f
RRN: 1
User: werik
Tweet: "ola mundo"
Local: 24
Favorite: 3 || Retweet: 3
Total views: 2
Language: ingles

Para o próximo tweet pressione qualquer tecla...
RRN: 2
User: aulos
Tweet: "hellow www"
Local: 33
Favorite: 5 || Retweet: 4
Total views: 2
Language: ingles

Para o próximo tweet pressione qualquer tecla...
f
RRN: 3
User: batman
Tweet: "vamos dar uma bat volta"
Local: 2
Favorite: 4 || Retweet: 3
Total views: 1
Language: brasil

```

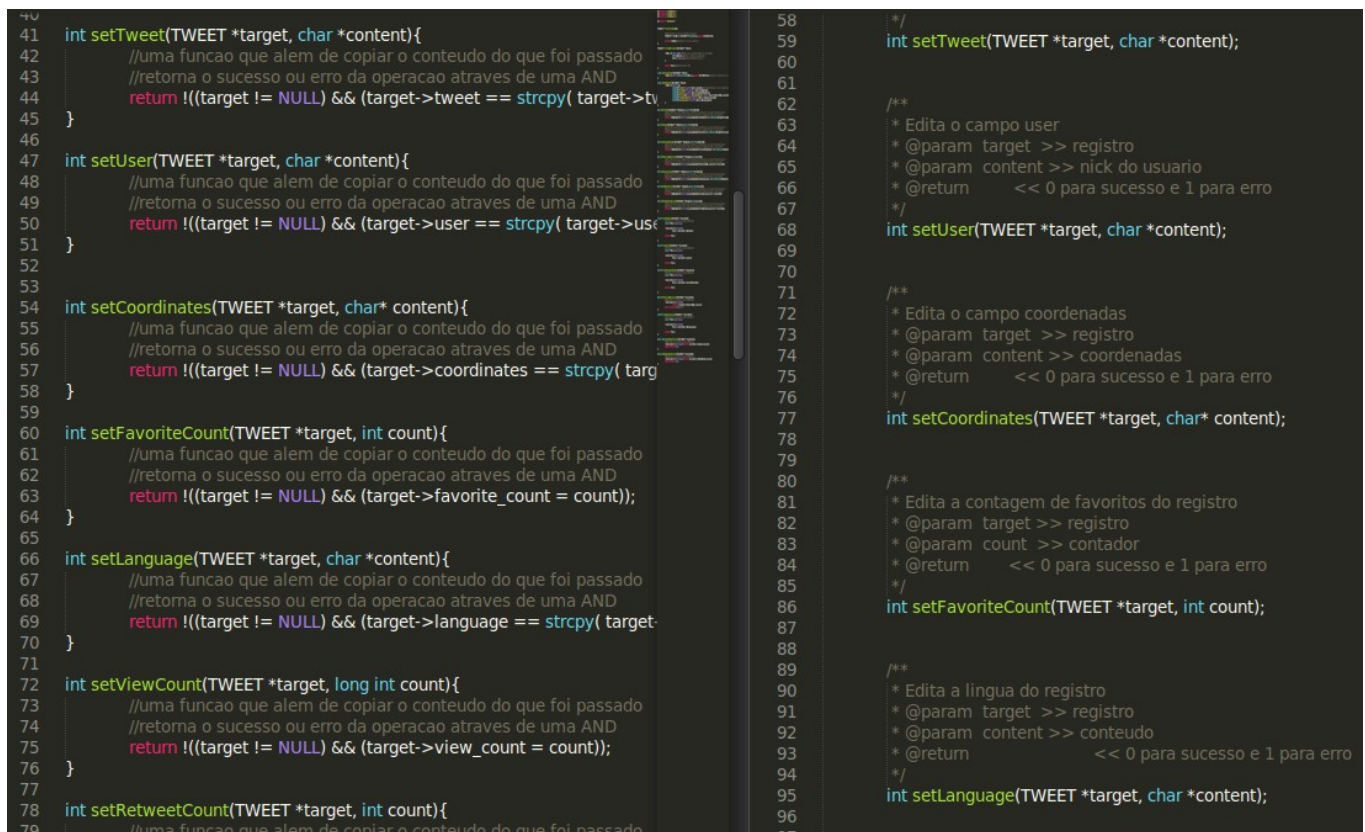
Funções data.h:

- int fileExist () : Verifica existência do arquivo de dados.
- void output(TWEET *, int *): Escreve no arquivo um tweet.
- TWEET* inputRRN (int, int*): Função para ler do arquivo um tweet.
- TWEET* inputALL(FILE* , int*): Ler todos os registro um de cada vez.
- void removeRRN(int, int*): Remove um tweet a partir de seu RRN.

Funções tweet.h:

- TWEET * newTweet (): Cria um novo tweet.
- TWEET * deleteTweet(TWEET *): Destrói um tweet.
- void renewTweet(TWEET *): Apaga todos os campos de um tweet.
- void printTweet (TWEET *): Imprime um tweet.
- int setTweet(TWEET *, char *): Edita o conteúdo do campo tweet no registro dado.
- int setUser(TWEET *, char *): Edita o campo USER.
- int setCoordinates(TWEET *,char *): Edita as coordenadas.
- int setFavoriteCount(TWEET *,int): Edita contagem de favoritos do registro.
- int setLanguage (TWEET *,char *): Edita a lingua do registro.
- int setViewCount(TWEET *, long int): Edita o campo view do registro.
- int setRetweetCount(TWEET *,int): Edita o campo retweet do registro.
- char *getTweet(TWEET *): Retorna o tweet.
- char *getUser(TWEET *): Retorna a string usuario.
- char *getCoordinates(TWEET *): Retorna as coordenadas.
- int getFavoritecount(TWEET *): Retorna String do campo contador de favoritos.
- char *getLanguage(TWEET *): Retorna a Linguagem do tweet.
- long int getViewcount(TWEET *): Retorna string contador de visualizações.
- int getRetweetCount(TWEET *): Retorna a string do campo contador de retweets.

Vale ressaltar que a descrição do funcionamento mais detalhado das funções está presente em comentários, no próprio código. Segue, abaixo, exemplos de imagens do código:



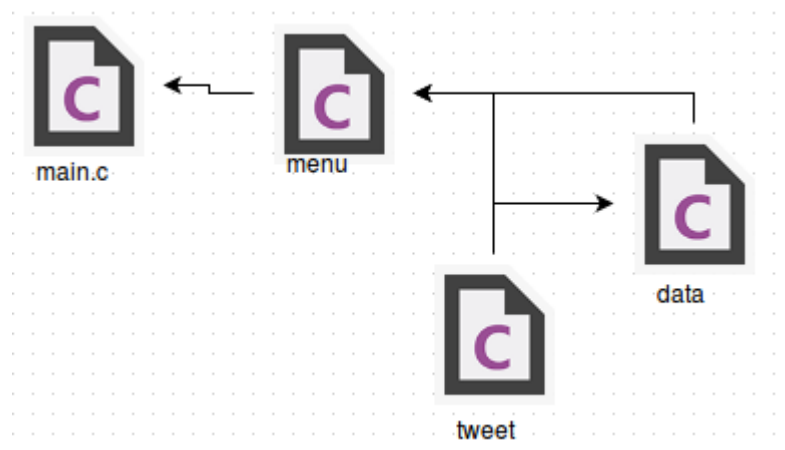
```
41 int setTweet(TWEET *target, char *content){
42     //uma funcao que alem de copiar o conteudo do que foi passado
43     //retorna o sucesso ou erro da operacao atraves de uma AND
44     return !((target != NULL) && (target->tweet == strcpy( target->t
45 })
46
47 int setUser(TWEET *target, char *content){
48     //uma funcao que alem de copiar o conteudo do que foi passado
49     //retorna o sucesso ou erro da operacao atraves de uma AND
50     return !((target != NULL) && (target->user == strcpy( target->us
51 })
52
53
54 int setCoordinates(TWEET *target, char* content){
55     //uma funcao que alem de copiar o conteudo do que foi passado
56     //retorna o sucesso ou erro da operacao atraves de uma AND
57     return !((target != NULL) && (target->coordinates == strcpy( targ
58 })
59
60 int setFavoriteCount(TWEET *target, int count){
61     //uma funcao que alem de copiar o conteudo do que foi passado
62     //retorna o sucesso ou erro da operacao atraves de uma AND
63     return !((target != NULL) && (target->favorite_count = count));
64 }
65
66 int setLanguage(TWEET *target, char *content){
67     //uma funcao que alem de copiar o conteudo do que foi passado
68     //retorna o sucesso ou erro da operacao atraves de uma AND
69     return !((target != NULL) && (target->language == strcpy( target-
70 })
71
72 int setViewCount(TWEET *target, long int count){
73     //uma funcao que alem de copiar o conteudo do que foi passado
74     //retorna o sucesso ou erro da operacao atraves de uma AND
75     return !((target != NULL) && (target->view_count = count));
76 }
77
78 int setRetweetCount(TWEET *target, int count){
79     //uma funcao que alem de copiar o conteudo do que foi passado
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```


Seção 3:

Arquivos

- `main.c` : Só inicializa o menu.
- `menu.h/.c`: Controla o menu do usuário e faz as chamadas apropriadas para os TADs (Tipo Abstrato de Dados) `tweet` e `data`.
- `data.h/.c`: Faz as operações de leitura e escrita no arquivo. Escrever, buscar, remover e ler são algumas operações embutidas nesse TAD.
- `tweet.h/.c`: Faz as operações referentes ao registro principal. Editar, imprimir e receber são algumas funções presentes nesse TAD.

O diagrama abaixo mostra o esquema de “includes”. “Main.c”, por exemplo, inclui `menu` que por sua vez inclui tanto `data` quanto `tweet`. Logo a seta indica a frase “é incluído por”.



Seção 4:

Makefile

O makefile que usamos é um genérico. Um dos integrantes do grupo tinha esse Makefile e portanto usamos ele no trabalho. Ele utiliza alguns conceitos de linux e ferramentas para que o próprio shell encontre os objetos desejados pelo usuário.

```
#Compilation variables. Compilation flags.  
#for special includes copy this -I/path/for/somehting  
CC=gcc  
FLAGS=-Wall -lm  
  
#File sources  
SOURCES=$(wildcard *.c)  
OBJECTS=$(SOURCES:.c=.o)  
TARGET=main  
ZIPNAME=T1_Ciferri  
  
#Main  
all: $(TARGET)  
  
#  
$(TARGET): $(OBJECTS)  
    @$(CC) -o $@ $^  
  
#ADT (TAD em portugues)  
%o: %.c %.h  
    @$(CC) $(FLAGS) -c $<  
  
#if no header  
%o: %.c  
    @$(CC) $(FLAGS) -c $<  
  
#clean .o and main  
clean:  
    @rm -f *.o $(TARGET)  
  
#Quick clean+all  
rebuild: clean all
```

run:

@./\$(TARGET)

#####

#####CUSTOM#####

#####

zip:

zip \$(ZIPNAME) *.h *.c Makefile