COMP LabBook 2024 2 - E1

Lucas M. Schnorr

October 8, 2024

Contents

1 Logs de testes, tabelas de decisão e entradas de teste

Estes são os arquivos fornecidos no ZIP.

```
Archive:  e1_CSV_LOG.zip
 Length   Method    Size  Cmpr    Date    Time   CRC-32   Name
--------  ------  ------- ---- ---------- ----- --------  ----
  585608  Defl:N    41775  93% 2024-10-08 22:01 1998e3be  e1.log
   43400  Defl:N     5176  88% 2024-10-08 22:01 a3ae815d  e1.csv
     257  Defl:N      139  46% 2024-10-08 22:01 90b98a92  e1_output_grupos_por_erros.csv
     349  Defl:N      112  68% 2024-10-08 22:01 e2fb9406  e1_output_objetivo.csv
     162  Defl:N       82  49% 2024-10-08 22:01 e762ce02  e1_output_quais_os_testes_mais_errados.
    1156  Defl:N      212  82% 2024-10-08 22:01 5708a85f  e1_output_quem_errou_o_que.csv
--------          -------  ---                            -------
  630932            47496  93%                            6 files
```

- `e1.log`: o arquivo contém o log cru dos testes
- `e1.csv`: após parsing para obter apenas a saída, as colunas são as seguintes:
  - Grupo, Test, Command, Result
  - Command == 0 → programa funcionou sem problemas

  - Command != 0 → programa deu problema (tipicamente timeout, segfault, etc)
  - Result == 0 → ordem dos tokens de acordo com o esperado
  - Result != 0 → o teste é considerado falho (não passou no teste)

- `e1_output_objetivo.csv`
  - E1.O: nota final do teste objetivo (sobre 10)
- `e1_output_quais_os_testes_mais_errados.csv`
  - Estatística que ilustra os testes que mais deram errado
- `e1_output_quem_errou_o_que.csv`
  - Indica apenas os testes que falharam, para cada grupo
  - Grupo, Test, Command, Result
  - Para o teste funcionar, Command e Result tem que ter o valor TRUE
  - Por exemplo:
    * Temos a linha `GrupoT,95,TRUE,FALSE`. Neste caso, procure pelo arquivo `tests/e1/entrada_095`
    * Então, execute o comando:

      ```
      ./etapa1 < tests/e1/entrada_095
      ```

      `1`
    * Em seguida, compare a sua saída com aquela no arquivo `tests/e1/aval_095.tesh`, nas linhas que começam por >, ignorando a string "> " no início de cada uma dessas linhas. Essa é a saída esperada.
  - Para fazer essas verificações, consulte o TGZ (listagem abaixo).
- `e1_output_grupos_por_erros.csv`
  - Indica o teste e quais os grupos que falharam nele
  - Em ordem descrescente de quantidade de grupos que erraram o teste

Estes são os arquivos fornecidos no TGZ.

```
tar vftz e1/E1.tgz
```

2 Comentários

2.1 Comentários para todos os grupos

2.1.1 Importante! Nomes nos arquivos!

- Todos os arquivos devem conter a identificação do grupo e os seus membros

2.1.2 Sugestão

- Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)
- Adicionar um alvo `clean` que remove arquivos gerados/compilados

2.2 Quem errou qual teste?

A tabela abaixo apresenta a visão geral de quem errou cada teste.

Test	Quem
52	I,K,L
53	I,K,L
58	I,$P_{Z,ZZ}$
59	I,O,R_Z
69	K,$P_{Z,RZ}$
35	I,L
38	I,L
41	I,N_Z
50	I,L
51	I,L
54	I,K
60	O,R_Z
62	$P_{Z,RZ}$
66	I,N_Z

67	I,L
14	L
27	I
28	I
29	I
30	I
33	K
36	I
39	I
42	K
43	I
44	I
48	I
49	I
56	K
61	P_Z
70	K

2.3 Comentários para cada grupo

Os comentários abaixos focam na observação do código e a experiência do professor da execução dos testes com o código do grupo. Os comentários abaixo não mencionam explicitamente os erros detectados através dos testes automáticos. Para estes, a sugestão é olhar o ZIP e o TGZ e a introdução deste relatório para decodificar os ajustes que devem ser feitos. Caso houverem dúvidas, entre em contato.

2.3.1 GrupoA

- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal
- Quebra de linha é considerado espaço em branco (pode-se colocar na mesma regra)

2.3.2 GrupoB

- Usando Makefile como um "wrapper" do CMake (só para deixar anotado)
- Boa documentação do projeto com o arquivo `README.md`
- Espaços ignorados podem ser aglutinados em uma única regra
  - Quebras de linha são considerados espaços
- Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

2.3.3 GrupoC

- O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`
- O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.
  - O alvo `test` não funciona, provavelmente porque o conteúdo de `src/testing` foi removido do pacote
  - Pode-se criar uma arquivo `Makefile.alt` somente para testes
    - Removê-lo do tgz na hora de submeter

2.3.4 GrupoD

- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal
- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

2.3.5 GrupoE

- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal

2.3.6 GrupoF

- O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.
- Os espaços podem ser aglutinados em uma única regra
- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal

2.3.7 GrupoG

- Arquivos devem estar na raiz
- Normalmente evita-se de #include com um caminho relativo (ou absoluto)
  - No caso farias somente #include "tokens.h", instruindo o compilador (com -I) a procurar os cabeçalhos em determinado lugar
- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal
- A regra do barra-enne é redundante com a classe [:space:]
- Boa organização em subdiretórios, mas o Makefile precisa melhorar visto que não é uma boa prática em receitas misturar entradas .c e .o como no alvo $(ETAPA). Além disso, a compilação de .o pode ser via regra única por intermédio de wildcards, conforme visto no tutorial indicado.
- Arquivos `deliver.sh`, `tester.py`, `tests` podem ser omitidos do tgz

2.3.8 GrupoH

- O caractere barra-ene está incluso na classe `[:blank:]`, portanto temos uma regra redundante
- O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.

2.3.9 GrupoI

- Arquivos devem estar na raiz
- Deve-se evitar colocar a implementação de funções no cabeçalho do scanner, priorizando a última seção do arquivo ou em arquivos suplementos.
- Ao invés de implementar yywrap, pode-se usar a opção para desabilitar essa funcionalidade.
- Alvos e receitas do makefile são majoritariamente manuais, sem wildcards. O makefile pode ficar bem mais sucinto se empregar os conhecimentos do tutorial indicado.
  - Além disos, possui regras específicas da etapa1, mesmo sabendo que temos outras etapas por vir.

2.3.10 GrupoJ

- As ações associadas às regras estão com identação diferente, algumas alinhadas outras não, no arquivo scanner.l
- O makefile parece ter código boilerplate que não se aplica neste projeto, pois ele vê se existe um subdiretório src, adaptando-se em função. Se o grupo não tem a intenção de organizar em subdiretórios, sugiro simplificar o Makefile.
- Não se usa a filosofia de compilação parcial dos arquivos (-c), ou seja, sempre se compila tudo novamente.
- A variável tar file pode ser definida a partir do nome de binary.

2.3.11 GrupoK

- Não há necessidade de stdio.h no arquivo scanner.l
- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

2.3.12 GrupoL

- Normalmente o ; nos comandos em C ficam imediatamente após a último token do comando, sem espaços.
- Não há necessidade de incluir stdio.h. Além disso, cabeçalhos de bibliotecas de sistema são incluídas com <stdio.h> ao invés de "stdio.h".
- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.
- Evitar de empacotar o diretório `testes` (e os arquivos `quero.*\.sh`).

2.3.13 GrupoM

- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.
- Não há necessidade de incluir stdio.h.
- Os espaços ignorados podem ser aglutinados em uma única regra

2.3.14 GrupoN

- Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.
- Não temos aspas simples
- Não temos comentários multilinha portanto não há necessidade de %x

2.3.15 GrupoO

- Os espaços ignorados podem ser aglutinados em uma única regra
  - Inclusive o barra-enne poderia ser aglutinado visto que trata-se de um espaço também
- Comentários são legais, mas melhor se estiverem não todos em maiúscula
  - Para não confundir com constantes do código

2.3.16 GrupoP

- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

2.3.17 GrupoR

- Na função `get_line_number`, corrigir a identação.
- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.
  - Ter um alvo que empregue o parâmetro `-c`

2.3.18 GrupoS

- Ignorar também o caractere tab com barra-t
- Aglutinar caracteres ignorados (espaços) em uma única regra
- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

2.3.19 GrupoT

- O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`
- O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.
- Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal

2.3.20 GrupoZ

- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

3 Final

- **E1.P**: valor entre [0, 1] indicando o peso empregado
  - Tipicamente, temos 1 para entrega no prazo, 0.8 para atraso/recuperação e 0 para entrega não considerada por razões específicas
- **E1.O**: resultado dos testes objetivos
- **E1.S**: resultados da avaliação subjetiva
- **E1**: média entre E1.O e E1.S

Grupo	Etapa	E1.O	E1.S	E1.P	E1
GrupoB	E1	10	9.75	1	9.88
GrupoH	E1	10	9.75	1	9.88
GrupoK	E1	10	9.75	1	9.88
GrupoL	E1	10	9.75	1	9.88
GrupoM	E1	10	9.75	1	9.88
GrupoO	E1	10	9.75	1	9.88
GrupoS	E1	10	9.75	1	9.88
GrupoZ	E1	9.86	9.75	1	9.8
GrupoA	E1	10	9.5	1	9.75
GrupoC	E1	10	9.5	1	9.75
GrupoE	E1	10	9.5	1	9.75
GrupoF	E1	10	9.5	1	9.75
GrupoJ	E1	10	9.5	1	9.75
GrupoT	E1	10	9.5	1	9.75
GrupoD	E1	10	9.25	1	9.62
GrupoG	E1	10	9.25	1	9.62
GrupoI	E1	10	9.25	1	9.62
GrupoN	E1	9.71	9.5	1	9.61
GrupoP	E1	9.43	9.75	1	9.59
GrupoR	E1	9.43	9.75	1	9.59

4 Recuperação

Nenhum grupo em recuperação.