INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados

2º semestre | 2013/2014

Enunciado do 2º Projecto

Data de entrega: Sexta-feira, 16 de Maio de 2014 (23h59)

1. Introdução

Neste projecto pretende-se desenvolver um programa em linguagem C que permita o processamento de mensagens entre um número fixo (mas arbitrário) de utilizadores. Cada mensagem tem um emissor e um receptor. Os receptores deverão processar (e eliminar) cada mensagem recebida por ordem de chegada (*First-In-First-Out*). O programa deverá permitir o registo de mensagens entre qualquer par de utilizadores, a eliminação/processamento de mensagens por parte do receptor, e a listagem ordenada das mensagens em espera para cada utilizador para serem processadas.

O programa deverá começar por ler o número total de utilizadores (**`N`**), processando em seguida um conjunto de linhas que começam com um dos comandos listados na secção seguinte e que indicam as operações a executar. Cada utilizador é identificado por um inteiro entre **`0`** e **`N-1`**.

2. Dados de entrada

O programa deverá ler os dados de entrada a partir do *standard input*. Os comandos são dados em linhas distintas e consecutivas, tendo a seguinte sintaxe:

- O comando **`send e r info`** permite registar a mensagem **`info`** enviada pelo utilizador **`e`** (emissor) para o utilizador **`r`** (receptor). A mensagem **`info`** consiste numa sequência de 0 a 500 caracteres (incluindo espaços). Tanto **`e`** como **`v`** são números inteiros entre **`0`** e **`N-1`** e podem ser iguais. A mesma convenção será usada nos próximos comandos. Este comando não tem qualquer output.
  Exemplo:
  ```
  send 30 12 aguia 1 para aguia 2 Pastel de nata para a aguia 3 over
  ```
  Dica: Para a leitura da mensagem, ver a função **`LeLinha`** dada nas teóricas ou a função **`fgets`**.

- O comando **`process u`** imprime a próxima mensagem destinada ao utilizador **`u`** (ou **`NULL`** caso não exista) e apaga esta mensagem dos registos do sistema. Se **`u`** não possuir qualquer mensagem em espera, o comando deverá escrever **`NULL`**. Ver detalhes do output na secção seguinte.
  Exemplo:
  ```
  process 4
  ```

- O comando **`list u`** lista todas as mensagens em espera para o utilizador **`u`** ordenadas por ordem de chegada. Se **`u`** não possuir qualquer mensagem em espera, o comando deverá escrever **`NULL`**. Ver detalhes do output na secção seguinte.
  Exemplo:
  ```
  list 0
  ```

- O comando **`listsorted u`** lista todas as mensagens em espera para o utilizador **`u`** ordenadas por ordem alfabética[1]. Caso **`u`** tenha duas mensagens iguais, a ordenação deverá seguir o índice do emissor das mensagens por ordem crescente. Se **`u`** não possuir qualquer mensagem em espera, o comando deverá escrever **`NULL`**. Ver detalhes do output na secção seguinte.

- Exemplo:
  ```
  listsorted 2
  ```

- O comando **`kill u`** apaga todas as mensagens em espera do utilizador **`u`**. Não tem output.
  Exemplo:
  ```
  kill 2
  ```

- O comando **`quit`**, apaga todas as mensagens em espera, saindo em seguida do programa. Não tem output.

3. Dados de saída

Todos os dados de saída deverão ser escritos no *standard output*. Com excepção dos comandos **`send`**, **`kill`** e **`quit`**, todos os outros comandos implicam a escrita da informação referente a uma ou mais mensagens. Se não existir qualquer mensagem a ser escrita, o programa deverá escrever a palavra **`NULL`** (seguida do carácter '**`\n`**') no *standard output*. Cada mensagem **`info`** recebida pelo utilizador **`r`** com origem no utilizador **`e`** deverá ser apresentada numa linha distinta sob a forma:

```
r e info
```

Exemplo:

```
1 2 I'm gonna make him an offer he can't refuse
1 3 I think this is the beginning of a beautiful friendship
1 0 Show me the money!
1 7 You've got to ask yourself one question: 'Do I feel lucky?' Well, do ya, punk?
1 0 I'll be back
```

[1] Dica: Utilize a função **`strcmp`** disponível em **`string.h`**

4. Exemplo 1[2]

4.1. Dados de entrada

```
2
send 0 1 If only you knew the power of the Dark Side.
send 0 1 Obi-Wan never told you what happened to your father.
process 1
process 1
send 1 0 He told me enough! He told me you killed him!
process 0
send 0 1 No, I am your father.
process 1
send 1 0 No. No! That's not true! That's impossible!
process 0
send 0 1 Search your feelings; you know it to be true!
process 1
send 1 0 NOOOOOOO! NOOOOOOOO!!!
process 0
send 0 1 Luke, you can destroy the Emperor.
send 0 1 He has foreseen this.
send 0 1 It is your destiny!
send 0 1 Join me, and together, we can rule the galaxy as father and son!
send 0 1 Come with me. It is the only way.
process 1
process 1
process 1
process 1
process 1
list 0
list 1
quit
```

4.2. Dados de saída

```
1 0 If only you knew the power of the Dark Side.
1 0 Obi-Wan never told you what happened to your father.
0 1 He told me enough! He told me you killed him!
1 0 No, I am your father.
0 1 No. No! That's not true! That's impossible!
1 0 Search your feelings; you know it to be true!
0 1 NOOOOOOO! NOOOOOOOO!!!
1 0 Luke, you can destroy the Emperor.
1 0 He has foreseen this.
1 0 It is your destiny!
1 0 Join me, and together, we can rule the galaxy as father and son!
1 0 Come with me. It is the only way.
NULL
NULL
```

[2] Ver ficheiros `exemplo1.in` e `exemplo1.out` disponíveis na página da disciplina.

5. Exemplo 2[3]

5.1. Dados de entrada

```
10
send 0 1 XXX1111
send 0 2 XXX1111
send 0 3 XXX1111
send 0 4 XXX1111
send 0 5 XXX1111
send 0 6 XXX1111
send 0 7 XXX1111
send 3 8 XXX1111
send 1 8 XXX1111
send 2 8 XXX1111
send 7 8 XXX1111
send 0 8 AXX1112
send 0 8 CXX1113
send 0 8 BXX1113
send 0 8 0XX1114
listsorted 8
kill 8
list 8
quit
```

5.2. Dados de saída

```
8 0 0XX1114
8 0 AXX1112
8 0 BXX1113
8 0 CXX1113
8 1 XXX1111
8 2 XXX1111
8 3 XXX1111
8 7 XXX1111
NULL
```

[3] Ver ficheiros `exemplo2.in` e `exemplo2.out` disponíveis na página da disciplina.

6. Compilação do Programa

O compilador a utilizar é o **gcc** com as seguintes opções de compilação:

```
-ansi -Wall -pedantic
```

Para compilar o programa deve executar o seguinte comando:

```
$ gcc -ansi -Wall -pedantic -o proj2 *.c
```

o qual deve ter como resultado a geração do ficheiro executável **proj2**, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal. Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de aviso (warnings).

7. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj2 < test1.in > test1.myout
```

Posteriormente poderá comparar o seu output com o output previsto usando o comando **diff**:

```
$ diff test1.out test1.myout
```

8. Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro arquivo com extensão **.zip** que inclua os ficheiros fonte (**.c**) e, se for o seu caso, os cabeçalhos (**.h**) que constituem o programa.
- Para criar um ficheiro arquivo com a extensão **.zip** deve executar o seguinte comando na directoria onde se encontram os ficheiros com extensão **.c**, criados durante o desenvolvimento do projecto:
  ```
  $ zip proj2.zip *.c
  ```
  ou
  ```
  $ zip proj2.zip *.c *.h
  ```
- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.
- O sistema não permite submissões com menos de 10 minutos de intervalo para o mesmo grupo. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: Sexta-feira, 16 de Maio de 2014 (23h59m). Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega

realizada corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

9. Avaliação do Projecto

- **Componentes da Avaliação**

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.

2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, indentação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto. Nesta componente será também utilizado o sistema **`valgrind`** por forma a detectar fugas de memória ("memory leaks") ou outras incorrecções no código, que serão penalizadas. Aconselha-se por isso que os alunos utilizem este sistema para fazer debugging do código e corrigir eventuais incorrecções antes da submissão do projecto.

3. Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

- **Atribuição Automática da Classificação**

- A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados num computador com o sistema operativo GNU/Linux. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 64 Mb, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

- Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

- Em caso algum será disponibilizada qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.