TRABALHO 1

Este primeiro trabalho visa desenvolver a habilidade de manipular cadeias de caracteres e arquivos-texto.

Basicamente, o trabalho consiste em processar um arquivo-texto contendo um conjunto de instruções para a criação de outros arquivos.

A ENTRADA DE DADOS

O programa processará um arquivo-texto contendo os seguintes comandos, descritos na tabela abaixo. Cada comando pode ter alguns parâmetro. O arquivo-texto contém exatamente um comando por linha. O comando está separado de seu parâmetro por, exatamente, um espaço em branco. Os parâmetros podem ser:

- **string**: uma sequência de caracteres delimitado por aspas, não as incluindo. Exemplo: "abacaxi".
- **path**: é uma sequência de nomes de diretórios (pastas), separados por "/". Um path pode ser ou não terminado por um "/". Por exemplo, "/a/b/c" e "/a/b/c/" são exatamente o mesmo path.
- **texto**: todos os caracteres, incluindo brancos, até o final da linha, marcado pelo caractere $
- **#v:** significa: o conteúdo da variável v (v pode ser x, y ou z)
- @1, @2 e @3: parâmetros usados na invocação do programa t1 (veja mais a frente)
- **id**: um número inteiro [0..9]
- **n**: número inteiro positivo (sem sinal)

comando	parâmetros	descrição
+x	(string | @1 | @2 | @3)	*concatena string ou os parametros 1, 2 ou 3 ao conteúdo de x*
+y	(string | @1 | @2 | @3)	*Idem para a variável y*
+z	(string | *n | **)	*Semelhante aos comandos +x e +y, mas não aceita os parâmetros @1,...,@3. O parâmetro n indica quantas vezes a variável z deverá se auto-concatenar. O parâmetro ** informa que o número de vezes que z deve se auto-concatenar é dado pelo valor de @1*
x		*limpa o conteúdo de x*
y		*idem para y*
z		*idem para z*
d	(path | #x | #y | #z)	*Define o diretório corrente, isto é, o diretório corrente é aquele informado pelo parâmetro path ou pelas variáveis x, y ou x*
a	(string | #x | #y | #z)	*Define o nome do arquivo*
e	(string | #x | #y | #z)	*Define a extensão do nome do arquivo*

c	id	*Cria o arquivo com nome extensão definidos pelos comandos a e e, dentro do diretório definido pelo comando d. O arquivo aberto é identificado por id*
o	id	*Semelhante ao comando c, mas abre o arquivo para inclusão (append). Se o arquivo não existir, primeiro cria.*
l	id	*Semelhante ao comando c, mas abre o arquivo para leitura.*
w	id (texto | #x| #y| #z)	*Grava uma linha no aquivo correntemente aberto. Caso não exista nenhum arquivo aberto, deve escrever na saída padrão uma mensagem de erro.*
r	id (#x | #y | #z)	*Lê linha do arquivo id e a armazena na variável indicada*
f	id	*fecha arquivo*

Exemplo 1:

+x "abcd"	*conteúdo de x: abcd*
+x "efg"	*conteúdo de x: abcdefg*
+y @2	
d ./saida/	
a "arq"	
e "txt"	
c 5	*cria o arquivo ./saida/arq.txt*
w 5 #x	*escreve em uma linha o conteúdo de x*
w 5 #y	
f 5	*fecha o arquivo criado por c*
exemplo01.txt	

Exemplo 2:

`+x "/home/eu"`	
`+x "/docs/"`	
`+x "d1"`	*conteúdo de x: /home/eu/docs/d1*
`+y "a1"`	*conteúdo de y: a1*
`d #x`	*diretório do arquivo: /home/eu/docs/d1*
`a #y`	*nome do arquivo: a1*
`e "txt"`	*extensão do arquivo: txt*
`c 2`	*cria arquivo: /home/eu/docs/d1/a1.txt*
`x`	*limpa o conteúdo da variável x*
`+x "Ola, galaxia"`	
`w 2 #x`	
`+z "aB"`	
`+z *2`	*conteúdo de z: aB**aBaB**
`+z *3`	*conteúdo de z: aBaB**aBaB**aBaBaB**aBaB**aBaB*
`+z *25`	
`w 2 #z`	
`f 2`	

exemplo02.txt

O Programa

O nome do programa deve ser `t1` e aceitar obrigatoriamente o parâmetro -f. Outros parâmetros são opcionais

`t1 -f` *`arquivo`* `[-o` *`dir`*`]` *`[par1]`* *`[par2]`* *`[par3]`*

O parâmetro **`-f`** especifica o nome do arquivo de entrada; o parâmetro **`-o`** especfica um valor default para o comando **`d;`** os paramêtros **`par1, par2 e par3`** são opcionais e e atribuem valores, respectivamente, às variáveis **`@1, @2, @3.`** Note que todos eles podem ser totalmente ou parcialmente omitidos. Assim, as seguintes invocações são possíveis:

- t1 -f *arquivo* -o *dir par1 par2 par3*
- t1 -f *arquivo* -o *dir par1 par2*
- t1 -f *arquivo* -o *dir par1*
- t1 -f *arquivo* -o *dir*
- *t1 -f arquivo par1 par2*

O valor das variáveis `@1`, `@2` e `@3` terão valor indefinido caso o respectivo parâmetro tenha sido omitido. Abaixo, alguns exemplos de invocações:

- `t1 -f a1.txt -o ./saidas abobora abacate banana`
- `t1 -f /home/ze/testes/a2.txt -o ./saidas/`
- `t1 -f ./testes chuchu repolho jilo`
- `t1 -f t005.txt -o ./a 10 Abacaxi abacaxi10`

A Implementação

Espera-se que o programa esteja bem dividido em procedimentos curtos. Por exemplo, poderiam ser implementadas funções que:

- leia e retorne uma linha de um arquivo-texto
- dado um path, retorne o diretório. Ex. f("`/a/b/c/d/e.txt`"), produz "`a/b/c/d`".
- dado um path, retorne o nome do arquivo. Ex. f("`/a/b/c/d/e.txt`"), produz "`e`".
- dado um path, retorne a extensão do arquivo. Ex. f("`/a/b/c/d/e.txt`"), produz "`txt`".
- entre outros.

O Que Entregar

Submeter a sala Moodle o arquivo ***nome*.zip**,[1] que deve conter um diretório *src*, com os fontes do programa e um makefile. Este makefile deve ter (pelo menos) o target `t1` que deve produzir o arquivo executável (dentro do diretório src).

Note que ***nome*** é alguma abreviatura do nome do aluno. A abreviatura só deve conter letras (por exemplo, **`jlsilva`**, poderia se referir ao aluno José Luís da Silva).

A Avaliação

Espera-se uma atitude pró-ativa para a aquisição dos conhecimentos (i.e., estudo) para resolver o problema proposto.

A avaliação se baseará em dois critérios: **(a)** compilação e testes do executável; **(b)** inspecção do código-fonte. O professor fará os testes em uma máquina virtual rodando o SO Linux.

A nota será proporcional ao número de testes executados com sucesso. Portanto, caso o programa não compile ou não execute corretamente, a nota poderá ser muito baixa!

1 ATENÇÃO: somente usar a extensão .zip. Não usar .rar, .7z etc.