universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Aula Prática Nº 5

Objetivos

Passagem de argumentos a programas

Bibliotecas de sistema[1]

Tratamento de erros

Acesso a variáveis de ambiente

Guião

1. Leia atentamente o código fonte `args1.c`.

a) Crie o ficheiro executável `args1` (`gcc -o args1 args1.c`), execute o programa passando-lhe vários argumentos e interprete o texto impresso pelo programa.

b) Altere o programa anterior de modo a garantir o seu funcionamento apenas quando o utilizador passa dois argumentos. Em caso de número incorreto de argumentos, o programa deve imprimir uma mensagem a indicar o número errado de argumentos e terminar, devolvendo um valor de erro.

c) Tendo como base o programa anterior e as rotinas de conversão numérica das bibliotecas de sistema (`atoi`, `atof`, etc.), implemente o programa `calculadora.c` que simula uma simples calculadora para números reais com cinco operações possíveis: (`'+', '-', 'x', '/', 'p'`). O programa recebe três argumentos como se ilustra no exemplo de utilização seguinte:

```
./calculadora 5.0 p 2.0.
```

Para este exemplo de utilização, o programa deve imprimir o valor `25.0`. Tenha em atenção a validação do número de argumentos.

d) Altere a calculadora de modo a utilizar a função `strtod` para a conversão de uma *string* para um valor real. Leia com atenção o manual da função e implemente uma forma de validar as potenciais situações de erro que possam ocorrer nessa conversão.

e) Explique porque razão o programa, se corretamente desenvolvido, considera o número de argumentos incorreto quando utilizado da seguinte forma (atenção que a letra para a multiplicação na nossa calculadora é `'x'` e por isso seria uma operação inválida):

```
./calculadora 5.0 * 2.0.
```

1 Pode consultar o Cap. 2 no endereço http://www.acm.uiuc.edu/webmonkeys/book/c_guide/ sobre as bibliotecas de sistema mais importantes para este guião.

2. Leia atentamente o código fonte `args2.c`.

a) Crie o ficheiro executável `args2` (`gcc -o args2 args2.c`), execute o programa passando-lhe várias palavras como argumentos e interprete o resultado do programa.

b) Crie uma nova variável de ambiente na *bash* designada por `NEWUSER` e altere o programa anterior de modo a que a informação sobre o utilizador seja dada usando o valor desta nova variável. Teste o programa com valores distintos da variável `NEWUSER`.

c) Implemente o programa `joinWords.c` que junta todos os argumentos numa única frase e a imprime no terminal. Apesar de poder resolver este problema de inúmeras formas, pretendemos que crie um `array` de carateres com todo o texto antes de o escrever no terminal, explorando as rotinas de manipulação de `strings` das bibliotecas de sistema (`string.h`).

d) Implemente uma nova versão do programa anterior, designado `joinWordsText.c`, de modo a que todos os argumentos que não comecem por uma letra sejam ignorados. Explore as rotinas de manipulação de `carateres` das bibliotecas de sistema (`ctype.h`).

3. Implemente o programa `altobaixo.c` que lhe permita jogar o "AltoBaixo". Este jogo consiste em tentar adivinhar um número inteiro dentro de um intervalo de valores. O programa escolhe um número aleatoriamente dentro de uma gama de valores inteiros que deverão ser passados como argumento ao programa.

Depois, o utilizador insere uma tentativa e o programa indica se é demasiado alta, ou demasiado baixa relativamente ao valor gerado internamente. Isto é repetido até o utilizador acertar no número. O jogo acaba indicando quantas tentativas foram feitas. Pode gerar o número secreto com base na utilização da função `rand()`.

4. Implemente o programa `sortWords.c` que ordena por ordem alfabética todos os argumentos do programa que comecem por uma letra e os imprima ordenados no final. A ordenação será crescente ou decrescente de acordo com o valor de uma variável de ambiente (`SORTORDER`) que terá que definir na *bash* e aceder dentro do programa.

a) Altere o programa anterior de modo a realizar a ordenação ignorando as diferenças entre letras maiúsculas e minúsculas.

b) Compare o funcionamento do seu programa com o programa `sort` (`man sort`) e implemente uma nova versão do programa (`sortWords2.c`) em que as palavras são pedidas ao utilizador e não argumentos do programa.