Dicas sobre o uso da linguagem C/C++

Além dos recursos gerais da linguagem C/C++, pode ser necessário vez por outra utilizar alguns recursos especiais providos pelo ambiente de execução para obter resultados especiais como, por exemplo, para manipular arquivos, acessar a tela diretamente, ou gerar números aleatórios. Este documento apresenta uma breve descrição dos recursos mais comuns que podem ser úteis durante o desenvolvimento de programas.

Argumentos de linha de comando

Ao ser iniciado, um programa pode receber informações adicionais sobre como ele deve ser comportar durante a execução. Isso é feito através do que se denomina usualmente parâmetros (ou argumentos) de linha de comando. Por exemplo, quando se executa um comando como `diff arq1 arq2`, o comando `diff` recebe dois argumentos, o primeiro igual a `arq1` e o segundo igual a `arq2`. Assim sendo, um programa deve ser capaz de manipular os parâmetros de linha de comando com que for executado. Parâmetros são usualmente vistos como seqüências de caracteres separadas por caracteres em branco.

Em C/C++, parâmetros da linha de comando (qualquer coisa digitada após o nome do programa na linha de execução) são manipulados através de dois parâmetros definidos para a função `main`, que é na verdade definida como

```
int main( int argc, char* argv[] );
```

o primeiro parâmetro, `argc`, indica o número de argumentos do programa (incluindo seu próprio nome) e o segundo, `argv`, é um vetor com as cadeias de caracteres (strings) que correspondem a cada parâmetro (o nome do programa pode ser encontrado em `argv[0]`).

Um exemplo de programa que use esses recursos é apresentado a seguir:

```
#include <stdio.h>

main(int argc, char* argv[])
{
  int i;
  printf("argc=%d\n",argc);
  for (i=0;i<argc;++i)
    printf("arg.%d: \"%s\"\n",i,argv[i]);
}
```

Manipulação de arquivos

Em C, arquivos texto são manipulados de forma bastante semelhante à entrada e saída padrão, utilizando comandos de leitura e escrita (`fprintf`, `fscanf`). O que é preciso fazer para aplicar esses comandos a um arquivo é declarar uma variável de acesso ao arquivo, abri-lo e fechá-lo ao final. Isso é feito pelos comandos `fopen` e `fclose`.

Um pequeno exemplo de programa em C que manipula um arquivo de nome "exemplo.txt´´ é apresentado a seguir:

```
#include <stdio.h>
main()
{
  FILE* f;
  char  c;
  f = fopen("exemplo.txt","r"); // "r": leitura; "w": escrita
  printf("Os caracteres antes do primeiro espaco em exemplo.txt sao: ");
  while ( (fscanf(f,"%c",&c)>0)&&(c!=' ') )
    printf("%c",c);
```

```
  printf("\n");
  fclose(f);
}
```

Manipulação de entradas de dados como caracteres

Em certos casos, pode ser preciso tratar a entrada de dados caractere por caractere, como feito com o arquivo `exemplo.txt` no exemplo anterios. Para simplificar esse tipo de operação, C oferece funções específicas para tratar caracteres e a constante EOF que permite testar pelo fim do arquivo de entrada.

O exemplo a seguir ilustra o uso dessas funções.

```
main()
{
  char  c;
  while ((c=getchar())!=EOF) {
     while ((c!='\n')&&(c!=EOF)) {
       putchar(c);
       putchar('\n');
       c=getchar();
     }
     putchar('@'); // para marcar o fim de linha
  }
}
```

Geração de números aleatórios

Praticamente todas as linguagens oferecem alguma forma de geração de números aleatórios. A teoria por trás dessa geração foge ao escopo do curso, bastando que se entenda que a idéia é que programas devem ser capazes de gerar números seguindo uma sequência *aparentemente* aleatória. Isso não quer dizer que os números sejam realmente imprevisiveis, mas apenas que não haja uma regra simples que relacione um número na sequência com os anteriores e que a distribuição dos números ao longo do tempo seja uniforme dentro do intervalo definido.

Em particular, para simulações é às vezes importante que a sequência não seja tão aleatória que não seja possível se repetir um experimento exatamente. Para isso, todo gerador de números aleatórios possui a definição de um estado inicial denominado *semente*, que pode ser um único número ou um conjunto de números. Para os casos em que o usuário não deseja escolher esse valor, um procedimento permite ler o relógio do computador e gerar uma semente a partir do mesmo. Como o relógio muda continuamente, isso dá uma boa ilusão de que as sequências são realmente aleatórias.

A semente, em C, é armazenada através de uma chamada da função `srandom(seed)`; o procedimento `time(0)` pode ser usado para extrair um inteiro baseado no relógio que sirva como semente (com a resolução de segundos) e a função `random` é usada para gerar números entre zero e `RAND_MAX` na seqüência aleatória. Para gerar números aleatórios entre zero e um, basta dividir esse valor por `RAND_MAX`.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  unsigned int seed = 13131; // Primeiro, uma semente fixa.
  srandom(seed);
  printf("Semente fixa: %u\n",seed);
  printf("No. aletorio entre 0 e 13: %2ld\n",random()%14);
  printf("Outro:                     %2ld\n",random()%14);
```

```
  printf("No. aletorio entre 0 e 1: %f\n",((double)random())/RAND_MAX);
  printf("Outro:                    %f\n",((double)random())/RAND_MAX);

  printf("\n"); // Agora, trocamos a semente para outra derivada do relogio

  seed=time(0);
  srandom(seed);
  printf("Semente baseada no tempo: %u\n",seed);
  printf("No. aletorio entre 0 e 13: %2ld\n",random()%14);
  printf("Outro:                     %2ld\n",random()%14);
  printf("No. aletorio entre 0 e 1: %f\n",((double)random())/RAND_MAX);
  printf("Outro:                    %f\n",((double)random())/RAND_MAX);
}
```

Redirecinamento da entrada e saída padrão

Muitas vezes, os programas utilizam apenas rotinas de manipulação da entrada (e saída) padrão, como `getchar()`, `printf()`, etc. Isso não significa que os programa sejam limitados a operar com o teclado e a tela. Para execução com um arquivo fornecido e para coleta de resultados em um arquivo de saída para utilização em relatórios deve-se utilizar os comandos de redirecionamento de entrada e saída do DOS ou Unix, os sinais `<` (maior que) e `>` (menor que).

Por exemplo, se o programa foi compilado para gerar um executável com o nome `tp0` (ou `tp0.exe`, no caso de DOS/Windows) e um arquivo entrada.txt contém os dados de entrada, o comando

```
tp0 < entrada.txt > saida.txt
```

tem o efeito de redirecionar a entrada padrão do comando tp0 para o arquivo entrada.txt e a saída para o arquivo saida.txt (que será criado, caso ainda não exista). Pode-se também redirecionar apenas a entrada, ou apenas a saída, independentemente.

Última alteração: 25 de janeiro de 2005