Ponteiros

Passando vetores para funções

Em C, existem fundamentalmente 4 modos adequados de se declarar a passagem de um vetor para uma função. Vejamos um exemplo:

```
void F1(int []); /*aqui é somente a declaração, por isso não precisa do nome do parâmetro*/
main()
{
    int v[] = {1, 2, 3, 4};
    F1(v);
}

/* modo 1 de implementar F1() */
void F1(int x[5]) /* declarar como se fosse um vetor, com a sua dimensão correta */
{}

/* modo 2 de implementar F1() */
void F1(int x[]) /*declarar como se fosse um vetor, sem dimensão, pois C necessita somente*/
{}         /*saber o que F1 estará recebendo, e não será alocado espaço de memória aqui. */
          /*  Por isso, não é necessário ter o índice no vetor */

/* modo 3 de implementar F1() */
void F1(int *x) /* um ponteiro para um inteiro, pois foi visto que um vetor nada mais é do
{}                 que um ponteiro. No caso, é um vetor de inteiros. Não é interessante
porque, embora seja de fato como C enxerga o parâmetro x, não permite a alguém que leia a
função saber que a função espera receber um vetor */

/* modo 4 de implementar F1() */
void F1(int x[32]) /*declarar como se fosse um vetor, com qualquer valor como dimensão. C
{}                 vai aceitar porque não aloca espaço aqui para um vetor de 32 elementos
                     somente serve para indicar que vai receber um ponteiro para um inteiro.
                     SÓ que NÃO deve declarar suas funções assim! */
```

Como no fundo, todos os casos são equivalentes, vamos analisar um deles, pois o raciocínio vale para todos os demais. Vejamos o caso da declaração "void F1(int *x) {}".

Na função "main()", um momento antes de ser executada a linha "F1(v);" a configuração da memória estaria do seguinte modo:

Assumindo que o endereço do 1° elemento de v seja 1000

Endereço	RAM
1000	1
1004	2
1008	3
1012	4

Ao entrar em "F1" (executar a linha "F1(v)"), o compilador C fará com que o parâmetro "x" receba o valor de "v", em outras palavras, fará "x = v". Só que um detalhe é importante: **C entende x como sendo um ponteiro para um tipo inteiro**. Ou seja, ao fazer "x = v", C executa de fato o seguinte na memória:

Endereço	RAM
1000	1
1004	2
1008	3
1012	4
1016	1000

Assumindo que o endereço de v seja 1000 e x seja alocada no endereço 1016.

Conclusão, dentro de **F1**, **x** deve ser tratado como se fosse um ponteiro para um inteiro.
Então, se quiséssemos acessar o valor do 1° elemento do vetor, podemos fazer:

```
printf(“%i”, x[0])
printf(“%i”, *x);
```