Programação II

Capítulo 13

O gato latiu na porta da toca do rato. O rato pensou "se o cachorro está ai, estou seguro." E saiu, e foi comido. Enquanto comia, o gato pensava: "É... hoje quem não fala mais de uma língua está perdido..."[1]

Piadinha circulante na Internet

Neste capítulo, iremos abordar as linguagens de programação compiladas mais famosas e utilizadas por hackers e desenvolvedores no mundo. Algumas são consideradas de baixo nível, o que não quer dizer que sejam de "má qualidade" ou deficientes, mas elas trabalham muito próximas da linguagem da própria máquina, como o assembler.

A programação exerce uma grande influência na "carreira" dos hackers, pois é esse conhecimento que vai proporcionar os subsídios para a criação de exploits, trojans, scanners e uma série de outros programas essências para suas atividades. O empenho é fundamental, pois dominar códigos prontos é muito fácil, e geralmente os que fazem isso não podem ser chamados de hacker, pois esse procura se superar sempre, descobrindo novas fórmulas para exercer sua influência.

Nosso foco é dar a noção básica de alguns componentes pertencentes a cada linguagem, pois para explicar os aspectos de cada uma delas, seria necessário montar mais um livro para cada livro específico sobre programação, e mesmo assim não seria possível abordar todas. A intenção é tornar esse capitulo o trampolim para que o leitor procure explorar o universo da programação e perder a dependência de scripts alheios.

Linguagem C

Agora, estudaremos as características básicas da formulação de programas na linguagem C, muito usada na criação de exploits. A linguagem C é uma linguagem de alto nível, genérica. Foi desenvolvida por programadores para programadores, tendo como meta características de flexibilidade e portabilidade.

O C é uma linguagem que nasceu juntamente com o advento da teoria de linguagem estruturada e do computador pessoal. Assim, tornou-se rapidamente uma linguagem "popular" entre os programadores. O C foi usado para desenvolver o sistema operacional Unix, e hoje está sendo usada para desenvolver novas linguagens, entre elas a linguagem C++ e Java.

Tipos básicos

Aqui temos uma lista dos tipos básicos de dados com os quais podemos trabalhar na linguagem C. Devemos lembrar que os tipos char e int são de valor inteiro, já float e doublé possuem valor de ponto flutuante.

Tipo	Tamanho	Intervalo	Uso
char	1 byte	-128 a 127	número muito pequeno e caractere ASCII
int	2 bytes	-32768 a 32767	contador, controle de laço
float	4 bytes	3.4e-38 a 3.4e38	real (precisão de 7 dígitos)
double	8 bytes	1.7e-308 a 1.7e308	científico (precisão de 15 dígitos)

Declaração de variáveis

Para que possamos usar variáveis em um programa em C, é necessário declará-las para que o processador seja avisado sobre quais os valores e os tipos. Dessa forma, ele poderá alocar espaço suficiente para as mesmas na memória. Há ainda a possibilidade de declarar mais de uma variável ao mesmo tempo, bastando apenas separá-las por uma ",".

A sintaxe para declaração de variáveis é a seguinte:
tipo variável , variável, ... **;**

Onde *tipo* é o tipo de dado e *variável* é o nome da variável a ser declarada. Se houver mais de uma variável, seus nomes são separados por vírgulas.

Declaração das variáveis:

```
int i;
int x,y,z;
char letra;
float nota_1,nota_2,media;
double num;
```

Bibliotecas

A biblioteca é simplesmente coleção de funções utilizadas na linguagem de programação. Ela armazena o nome de cada função, o código-objeto da função e as informações necessárias ao processo de vinculação. Para incluir uma biblioteca, é necessário incluir o seguinte código no início do programa:

```
#include <biblioteca>
```

Estrutura básica

Como exemplo de estrutura básica, temos o seguinte programa, cuja função é imprimir uma mensagem de texto na tela do computador:

```
main()
{
 printf("Hello world!\n");
 exit(0);
}
```

Note a presença de duas funções nesse programa: printf e exit. Essas são funções existentes na biblioteca denominada "standard" do C, o que significa que estas funções já vêm como padrão da linguagem.

Compilação

A compilação de um programa C depende do ambiente no qual o mesmo for criado (Windows ou Unix). Após concluir o código-fonte e salvar o programa, é necessário salvá-lo com a extensão ".c", como "programa.c". Então, é só rodar o compilador seguido do nome do programa, como:

```
cc programa.c
```

Onde "cc" é o nome do compilador a ser utilizado e "programa.c" é o nome do arquivo onde o código-fonte foi previamente escrito. A execução do programa é feita como qualquer arquivo executável pelo sistema, apenas digitando seu nome no prompt e teclando enter/return.

Exemplo

O seguinte programa faz a conversão de uma certa quantidade de dias para anos:

```
#include <stdio.h>
void main()
{int Dias;
float Anos;
printf ("Número de dias: ");
scanf ("%d", &Dias);
Anos=Dias/365.25;
Printf ("\n\n%d dias equivalem a %f anos.\n, Dias, Anos);
}
```

Java

A linguagem Java começou a ser desenvolvida em 1991 pela SUN, com o objetivo de programar dispositivos eletrônicos. Seu projeto foi baseado na linguagem C (estruturada) e C++ (orientada a objetos). Como o mercado para dispositivos eletrônicos inteligentes não prosperou rapidamente, seu potencial foi desviado para o desenvolvimento de páginas para Web com "conteúdo dinâmico".

Variáveis e Tempo de vida

Você tem dois meios para descrever variáveis: usando o tipo simples de ligação int e float ou usando tipos de classes definidas pelo programa. Você pode declarar as variáveis de duas formas, uma dentro de um método e a outra dentro da classe na qual este método está incluído.

Inicialização de variáveis

No Java não é permitido o uso de variáveis indefinidas. Variáveis definidas dentro do método são chamadas de variáveis automáticas, locais, temporárias ou estáticas e devem ser inicializadas antes do uso. Quando um objeto é criado, as variáveis-membro são inicializadas com os seguintes valores em tempo de alocação:

Tipo de variável	Valor inicial	Tamanho
Byte	0	8 bits
Short	0	16 bits
Int	0	32 bits
Long	0L	64 bits
Float	0.0f	32 bits
Double	0.0d	64 bits
Char	'\u0000' (Null)	64 bits
Boolean	False	

Este programa que estudaremos agora é um aplicativo *standalone* (não é uma applet) que roda através da linha de comando (MS-DOS Prompt, Unix Shell) e calcula uma lista de fatoriais. É um programa interessante para ilustrar a estrutura da linguagem Java, porque usa uma série de recursos da linguagem, que explicaremos neste capítulo e no próximo, tais como:

- os três formatos de comentários usados em Java;
- declaração de uma classe;
- declaração e definição de variáveis e constantes;
- atribuições, sintaxe de expressões;
- captura e tratamento de condições excepcionais;
- expressões condicionais e de controle de fluxo;
- operadores e precedência, separadores, literais;
- tipos de dados, palavras reservadas;
- invocação de métodos;
- impressão na saída-padrão;
- argumentos de linha de comando;
- vetores e cadeias de caracteres.

Exemplo:

```
/**
* lista de fatoriais
*/
class Fatorial {
     /* Maior número a ter seu fatorial calculado */
     static final int MAX_VAL = 15;
     /* método main */
     public static void main(String[] args) {
        int valor = 0;  // valor inicial igual a zero
        int maxValor;   // valor máximo é inteiro de 32 bits
        long fatorial;  // valor fatorial é inteiro de 64 bits
        try {
             maxValor = Integer.parseInt(args[0]);
               // 1. Exceção se args=null ou se args != inteiro
            if (maxValor > MAX_VAL) {
                throw new Exception();
              // 2. Exceção se maxValor > MAX_VAL
            }
        } catch (Exception e) {
            maxValor = MAX_VAL;
             // redefine maxValor caso haja exceção
        }
         System.out.println("- Fatoriais de números até " +
                        maxValor + " -\n");
         System.out.println("Valor" + "\t" +
                        "Fatorial\n——\t———\n");
         while(valor <= maxValor) {
              if (valor == 0) {
                        fatorial = 1;
              } else {
                        fatorial = valor;
                        for (int parte = valor; parte > 1;) {
            fatorial = fatorial * —parte;
                       // decrementa antes do uso
              }
           }
            System.out.println(valor + "\t" + fatorial);
           valor++;// valor = valor + 1
         } // fim do while
       } // fim do main
}
```


Compilador

Para compilar um programa em Java, é necessário o JDK (Java Developer Kit). E utilizar os seguintes comandos, lembrando de salvar os arquivos com a extensão ".java" antes de compilá-lo:

C:\.....\Javac Progr0101.java

E para rodar:

C:\.....\ Java Prog0101

Pascal

Um programa em Pascal consiste em um cabeçalho seguido de uma seleção de declarações em que todos os objetos devem ser definidos, e logo após o corpo, serão encontrados todas as ações e comandos a serem executados. Como podemos ver no seguinde esquema:

PROGRAM Nome_Do_Programa; { Cabeçalho }

[declaração de units]
[declaração de rótulos]
[declaração de constantes] { Seção de Declarações }
[declaração de tipos]
[declaração de variáveis]
[declaração de subprogramas]
begin
comando [; comando] ... { Corpo do Programa }
end.

Tipos de dados:

Algoritmo	Pascal	Descrição
a) Inteiro	a) INTEGER:	Representa números entre -32768 até +32767. Ocupa 2 bytes na memória.
b) Real	b) REAL:	Representa os números entre 2.9 x 10^{-39} até 1.7 x 10^{38} . Ocupa 6 bytes na memória.
e) Caractere	e) CHAR:	Representa um dos caracteres, da tabela ASCII. Ocupa 1 byte na memória.
d) Cadeia	f) STRING:	Conjunto de caracteres (CHAR). Ocupa de 1 a 255 bytes na memória.

g) Lógica	g) BOOLEAN:	Valor lógico. Assume somente dois valores: TRUE (Verdade) ou FALSE (Falso). Ocupa 1 byte na memória.
	c) WORD:	Números de 0 até 65535. Ocupa 2 bytes na memória.
	d) BYTE:	Números de 0 até 255. Ocupa 1 byte na memória.
	h) Shor Int	Representa os números entre -128 até 128. Ocupa 1 bytes na memória.
	i) LongInt	Representa os números entre -2.147.483.648 até 2.147.483.648 . Ocupa 4 bytes na memória.
	j) Single	Representa os números entre 1.5 x 10^{-45} até 3.4 x 10^{38} . Ocupa 4 bytes na memória.
	l) Double	Representa os números entre 5 x 10^{-324} até 1.7 x 10^{308} . Ocupa 8 bytes na memória.

Cabeçalho

O cabeçalho consiste da palavra reservada Program seguida de um identificador que corresponde ao nome do programa, terminando com um ponto e vírgula.

Exemplos:

program Lista;

program Relatório;

Seção de Declarações

O Pascal requer que todos os identificadores sejam predefinidos, ou que seja declarados antes que possam ser usados. Essa declaração é feita na área de declaração onde são associados os nomes dos objetos que farão parte do programa (dados, variáveis, etc.).

Corpo do Programa

Aqui são especificados os comandos e ações que vão compor o programa, ou seja, sua parte lógica. Por exemplo:

```
program Exemplo;
var
I : integer;
begin
for I := 1 to 500 do
if I mod 5 = 0 then
writeln( I:5, ´ é divisível por 5´);
end.
```


Exemplo de Programa

Os comandos do Pascal são simples, pois eles partem do princípio dos algoritmos vistos no capítulo de programação I. Vemos no exemplo a conversão de um algoritmo estruturado para o Pascal:

Em Algoritmo:

```
Algoritmo Triangulo
Variáveis:
  base, altura, area : real;
Inicio
   Leia(Base)
   Leia(Altura)
   Area ? (Base * Altura)/2
   Escreva (Area)
Fim
```

Em Pascal:

```
program triangulo;
var
  area, base, altura: real;
begin
  { Entrada }
  write (‘Digite a base:‘);
  readln (base);
  write (‘Digite a altura:‘);
  readln (altura);

  { Calculos }
  area:= (base*altura)/2;
  { Saida}
  writeln (‘A area do triangulo e:‘,area:10:2);
end.
```

Compilação

O próprio Pascal já vem com um compilador, o que torna o serviço bem mais fácil para o programador.

Assembler

A linguagem Assembler proporcina ao programador a experiência de total controle do equipamento, além da criação de programas muito mais consistentes e com maior velocidade. Como ferramenta para a criação de programas em Assembler, usaremos o Debugger, presente no prompt de comando do no DOS ou Windows. A limitação é que ele pode criar arquivos apenas do tipo ".com" com 64 KB de tamanho no máximo.

A seguir veremos uma lista de comandos do debug:

A - Montar instruções simbólicas em código de máquina
D - Mostrar o conteúdo de uma área da memória
E - Entrar dados na memória, iniciando num endereço específico
G - Rodar um programa executável na memória
N - Dar nome a um programa
P - Proceder ou executar um conjunto de instruções relacionadas
Q - Sair do programa debug
R - Mostrar o conteúdo de um ou mais registradores
T - Executar passo a passo as instruções
U - Desmontar o código de máquina em instruções simbólicas
W- Gravar um programa em disco

Estrutura

No código da linguagem, as linhas são divididas em duas partes, a instrução e os parâmetros de comando, como em:

Mov al, 25

Onde "mov" é o comando e "al" e "25" são os parâmetros. No exemplo, estamos usando a instrução "mov", que significa mover o valor 25 para o registrador "al". Onde as instruções são chamadas de códigos de operação.

Exemplo de programação

Para iniciar o exemplo, primeiro é necessário entrar no debug (digitar "debug" no DOS ou linha de comando). Para montar o programa, devemos usar o comando "a", que significa assemble, e especificar o registrador em 0100h, onde devem ser iniciados os programas ".com". Sendo assim, o código ficaria da seguinte forma:

```
a 100
mov ax,0005
mov bx,0006
add ax,bx
nop
```


No final, deve-se pressionar a tecla enter/return duas vezes. Esse programa move o valor 0005 para o registrador ax, move o valor 0006 para o registrador bx e adiciona o conteúdo dos dois registradores, armazenando o resultado em ax. Finalmente a instrução nop (no operation) finaliza o programa. No programa debug, a tela se parecerá com:

```
C:\>debug
-a 100
0D62:0100 mov ax,0005
0D62:0103 mov bx,0006
0D62:0106 add ax,bx
0D62:0108 nop
0D62:0109
```

Entramos com o comando "t" para executar passo a passo as instruções:

```
-t

AX=0005  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000
DS=0D62  ES=0D62  SS=0D62  CS=0D62  IP=0103   NV UP EI PL NZ NA
PO NC
0D62:0103 BB0400        MOV     BX,0004
```

Vemos o valor 0005 no registrador AX. Teclamos "t" para executar a segunda instrução:

```
-t

AX=0005  BX=0006  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000
DS=0D62  ES=0D62  SS=0D62  CS=0D62  IP=0106   NV UP EI PL NZ NA
PO NC
0D62:0106 01D8          ADD     AX,BX
```

Teclando "t" novamente para ver o resultado da instrução add:

```
-t

AX=0011  BX=0005  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000
DS=0D62  ES=0D62  SS=0D62  CS=0D62  IP=0108   NV UP EI PL NZ NA
PE NC
0D62:0108 90            NOP
```

A possibilidade de os registradores conterem valores diferentes existe, mas AX e BX devem conter os mesmos valores acima descritos. Para sair do Debug ,usamos o comando “q” (quit). Assim, você pode guardar comandos diretamente nos registradores da CPU. Comandos como o “format”, “copy”, entre outros.

Conclusão

Como foi dito, demonstramos aqui o básico para a criação de programas. A programação é algo que requer prática e pesquisa, por isso, muitas pessoas acabam escolhendo apenas uma linguagem para se especializar, mas obviamente sem deixar de adquirir conhecimentos sobre as outras linguagens. Afinal, quem fala apenas uma língua não está pronto para encarar o mundo de frente!