

CURSO DE PERL/Básico

 Erro de leitura

Autor: Ricardo Filipo
rfilipo@cipsga.org.br

Junho de 2000

Curso de PERL

Comite de Incentivo a Produção
do Software Gratuito e Alternativo
CIPSGA

Autor:

Ricardo Filipo
rfilipo@cipsga.org.br
r@w3c.com.br

Junho de 2000

Arte Final:

Djalma Valois Filho
dvalois@cxpostal.com

Copyright (c) 2000, Ricardo Filipo.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

A copy of the license is included in the section entitled "GNU Free Documentation License".

Copyright (c) 2000, Ricardo Filipo

E garantida a permissão para copiar, distribuir e/ou modificar este documento sob os termos da GNU Free Documentation License, versão 1.1 ou qualquer outra versão posterior publicada pela Free Software Foundation; sem obrigatoriedade de Seções Invariantes na abertura e ao final dos textos.

Uma cópia da licença deve ser incluída na seção intitulada GNU Free Documentation License.

Índice

PARA OS APRESSADOS	4
USOS DO PERL	5
COMO OBTER O PERL	6
COMO INSTALAR O PERL	6
<i>Linux</i>	6
<i>Instalando módulos com o CPAN.</i>	6
<i>make, test, install, clean, módulos ou distribuições</i>	7
COMO EXECUTAR OS SCRIPTS PERL	7
<i>UNIX</i>	7
<i>Opções na linha de comando</i>	8
FUNDAMENTOS DA LINGUAGEM	8
<i>Notação</i>	8
<i>Blocos</i>	9
<i>Tipos</i>	9
<i>Controle de fluxo</i>	12
<i>Variáveis especiais</i>	14
<i>Arquivos – leitura e escrita</i>	14
<i>Subrotinas</i>	15
<i>Bibliotecas</i>	15
USANDO O PERL NA INTERNET	16
<i>Cabeçalho do HTTP 1.0</i>	16
<i>Variáveis de ambiente</i>	16
<i>Lendo um <form></i>	17
<i>Módulo CGI</i>	18
ADMINISTRANDO BANCOS DE DADOS	20
<i>Módulo DB_File</i>	20
INTERFACE GRÁFICA	21
<i>Módulo perlTk</i>	21
SOBRE O AUTOR:	22
CONTROLE DE VERSÕES	24
GNU FREE DOCUMENTATION LICENSE	25

Para os apressados

Com o desenvolvimento das facilidades de programação para a Internet, como PHP, ASP e outras coisas, várias pessoas tem me perguntado: Para que serve o Perl, afinal? Tenho escutado comentários sobre a obsolescência da linguagem frente a estas outras, muito mais modernas (será mesmo?).

Talvez seja verdade que para algumas aplicações não seja preciso usar o Perl. Mas sem dúvida todas as coisas que se faz com outras linguagens poderiam ser feitas usando Perl, talvez com mais poder, versatilidade e facilidade. No lado do servidor, códigos incrustados no documento em HTML, tipo PHP ou ASP podem ser implementados com o módulo do Apache `mod_perl`. No lado do cliente, com funções muito parecidas com o javascript o vbscript há o `perlscript`, um plugin para navegadores como Netscape e Explorer. E como o Perl é uma linguagem de uso geral, aplicações para desktop, softwares de produtividade, gerenciadores de bancos de dados, interfaces gráficas, programação de sockets, clientes e servidores para Internet, CGI, etc. podem ser desenvolvidos rapidamente..

Nesta apostila procuramos apresentar o Perl de uma forma bem abrangente, permitindo uma primeira experiência com o ambiente em suas principais e mais comuns aplicações.

Para um aprofundamento dos conceitos aqui apresentados podem ser usados os comandos

`/> man perl`

Deve ser lido ainda:

`/> man perlfaq`

Para ler a documentação de um módulo:

`/> perldoc nome-do-modulo`

Usos do Perl

Internet

Administração de servidores

Software em geral

Bancos de Dados

Perl (Practical Extraction and Report Language) é um ambiente de programação de uso geral, de alto nível e fácil aprendizagem, implementável nas principais plataformas (quase todas as distribuições de UNIX trazem o Perl como padrão do sistema. As versões para Windows e Mac também são perfeitas, aproveitando as características destas plataformas). Possui recursos poderosos para processamento de strings, interação com o sistema operacional e com a rede, além de facilidades como gerenciamento automático de memória e casting inteligente e automático de tipos.

Apesar de se apresentar como uma linguagem de scripts, seus executáveis funcionam compilados em tempo real, sendo rápidos e poderosos. Esta característica facilita muito o trabalho do programador, pois os programas podem ser desenvolvidos, modificados e testados muito facilmente e sem a perda de tempo inerente aos procedimentos de compilação de executáveis. Outra grande vantagem é o Perl possuir, embutido no compilador, um depurador de excelente funcionamento.

Estes motivos tornaram o Perl a linguagem preferida para a programação de aplicações que rodam em servidores da Internet. Os administradores de sistemas UNIX, em especial, precisam dominar o ambiente, já que muitos utilitários do sistemas são baseados em Perl ou o usam ostensivamente.

Mas a característica mais importante do Perl é sua condição de "Software Livre". Os códigos, por definição, sendo uma linguagem de scripts, ficam "abertos". Desenvolvido inicialmente para implementar um ambiente de catalogação de erros, os códigos do primeiro Perl foram disponibilizados na Internet por seu criador, Larry Wall. Uma multidão de programadores (os Perl Porters) passaram a incorporar melhorias à linguagem, tornando-a completa.

Como obter o Perl

O Perl pode ser obtido em <http://www.cpan.org> ou <http://www.perl.com>

Utilize os seguintes comandos para baixa-lo:

```
/>wget http://www.perl.com
```

```
/>cd www.perl.com
```

Como instalar o Perl

Linux

Descompacte o Perl em qualquer diretório, rode o utilitário de configuração, compile e instale. Os comandos para obter, descompactar e instalar o Perl, num Linux "bash" seriam:

```
/>tar xzvf perl.tgz  
cd perl  
make dep  
make install
```

Instalando módulos com o CPAN.

Um ou dois módulos podem ser facilmente instalados manualmente, mas para fazer a administração eficiente do sistema perl residente em uma máquina será preciso usar o módulo CPAN (Comprehensive Perl Archive Network).

O módulo CPAN pode buscar módulos nos "site mirror" do CPAN e desempacotá-los em um diretório dedicado.

O módulo de CPAN também implementa o conceito de nome ou versão de módulos empacotados. Pacotes simplificam o gerenciamento e manutenção de sistemas de módulos relacionados..

Digite na linha de comando:

```
/> perl -MCPAN -e shell
```

Isto carregará o shell do CPAN, permitindo interações. O Modo interativo do CPAN permite a fácil realização de algumas tarefas:

Para ajuda digite **h** ou **?**.

Procurando os autores, pacotes, distribuição arquivos e módulos:

Existe um comando de uma única letra: 'a', 'b' (de "bundle"), 'd', e 'm', para cada uma das quatro categorias e outro, 'i' para quaisquer destas.

Podem ser usados como argumentos qualquer texto ou expressão regular (estas entre "/" e "/")

Se desejar ler o arquivo README sobre algum modulo que contenha o texto "xml", digite:

```
cpan> readme /xml/
```

make, test, install, clean, módulos ou distribuições

Estes comandos levam qualquer número de argumentos e investigam o que é necessário para executar a ação. Se o argumento é o nome de um arquivo de distribuição, este é processado. Se é um módulo, CPAN determina o arquivo de distribuição no qual este módulo é incluído e o processa. Se existirem dependências, estas são resolvidas.

Exemplo:

```
cpan> install OpenGL
OpenGL is up to date.
cpan> force install OpenGL
Running make
OpenGL-0.4/
OpenGL-0.4/COPYRIGHT
[...]
```

Como executar os scripts Perl

UNIX

Para rodar o script "oi_turma", que está no diretório atual digite no shell:

```
>perl ./oi_turma
```

É possível fazer do script Perl um executável. Basta seguir os passos:

1 – Acrescentar o comando abaixo na primeira linha do script:

```
#!/usr/bin/perl
```

Substitua o caminho por outro que indique a localização do executável Perl em sua plataforma, ou crie um alias para o mesmo para o caminho indicado no comando acima.

2 – Mude a permissão do script para que possa ser executado pelo usuário corrente:

```
>chmod +x ./oi_turma
```

Opções na linha de comando

Os argumentos passados pela linha de comando que contem só uma letra podem ser agrupados

Exemplo:

```
#!/usr/bin/perl -spi.bak
```

é igual a:

```
#!/usr/bin/perl -s -p -i.bak
```

As opções mais comuns são:

- d** carrega o depurador Perl
- w** o Perl imprimirá em STDERR todos os avisos (sintaxe, variáveis não inicializadas, etc).
- e** executa o proximo argumento (entre aspas).

Um uso muito útil para o perl seria trocar todas as ocorrências de um texto por outro, em determinados arquivos, salvando um backup destes.

```
/> perl -p -i.bak -e "s/texto antigo/novo texto/g" *.html
```

Isto pode ser uma solução fantástica se você precisar consertar um link quebrado, presente em centenas de arquivos.

Fundamentos da linguagem

O Perl é um ambiente de programação muito poderoso, mas a linguagem é de fácil aprendizagem. Apresenta características muito parecidas com C, sed, awk, basic e shell do UNIX.

Suas melhores características estão relacionadas ao tratamento automático da memória, conversão automática de tipos e à grande facilidade no processamento de texto.

Entretanto quem não tem experiência em UNIX ou sed, awk e shell poderá encontrar alguma dificuldade com a assimilação de alguns poderosos recursos.

Notação

O Perl possui um esquema de notação que se aproxima muito do usado em "C", possuindo ainda diversas características do Basic, awk e sed. A estrutura de inicialização do programa assume algo muito próximo do Pascal. Com o uso dos módulos, entretanto, o Perl pode assumir a feição de outras linguagens, como Tk ou Lisp.

Como em C, no Perl os comandos precisam ser terminados com um ";" (ponto e vírgula).

Comentários precisam ser precedidos por "#" (sustenido). Exemplo:

```
# O comando abaixo imprime uma saudação
```

```
print "Oi, turma!/n";
```


Como você pode observar, o texto a ser impresso deve estar envolvido pelas aspas (simples ou duplas). O caracter de nova linha é representado pelo escape: `"/n"`.

Não existe necessidade de se manter o comando em uma única linha. O escape `"/n"` poderia ser substituído pelo caracter literal de nova linha, dentro das aspas. O comando abaixo teria efeito idêntico:

```
print
    " Oi, turma!
"
```

Blocos

Os trechos do programa podem ser dispostos em blocos:

```
do {
    print "Oi, turma!\n";
}
print "Oi, de novo";
```

Tipos

O Perl define 3 tipos de dados: escalares, arrays e hashes.

Escalares:

Precedidos por um cifrão (\$), exemplo: `$escalar`.

Os escalares podem ser números, strings e referências. O compilador saberá diferenciar que operação realizar, em cada caso. Na dúvida o valor será considerado um string, ou este "casting" se processará automaticamente. Da mesma forma um escalar número interpolado em um string assumirá o tipo string Exemplo:

```
$numero = 5;
print "Digite um numero:"
$input=getc();
$soma = $numero + $input;
$string = "Valores:\n numero digitado: $input\n a somar com:$numero \nsoma:$soma\n";
```

Números

Inteiros, precisão dupla, com ou sem sinal, notação científica, decimal, hexa, octal, binário.

Strings

Sequências de caracteres. Em sua forma literal devem ser delimitadas por aspas simples (') ou duplas ("). Strings delimitadas por aspas duplas podem conter variáveis interpoladas ou caracteres especiais, usando a barra de escape.

Exemplo:

```
$variavel= "Cyber";  
$aspas_simples = 'Interatividade?\nChame o $variavel.';  
$aspas_duplas = "Interatividade?\nChame o $variavel.";  
print $aspas_simples;  
print "\n-----\n";  
print $aspas_duplas, "\n";
```

Isto imprimirá na tela:

Interatividade?\nChame o \$variavel.

Interatividade?
Chame o Cyber

Repare que o último comando "print" usa uma vírgula para separar os seus dois argumentos: o escalar \$aspas_duplas e o string literal "\n". Se for preciso, entretanto, colocar uma aspa simples numa string delimitada por aspas simples, será necessário usar um escape antes desta (\').

Referências

O equivalente aos ponteiros. Podem ser referenciados escalares, arrays, hashes, funções, classes, objetos, etc. Mais à frente nesta apostila este assunto será tratado com mais profundidade.

Arrays

Precedidos por um arroba (@). Exemplo: **@Array**

Armazena listas de escalares. Cada elemento da lista pode ser lido usando-se o seu índice numérico entre colchetes. O índice do primeiro elemento é 0.

Listas

São coleções de escalares. Listas de literais ou variáveis podem ser construídas com os elementos separados por vírgulas e envolvidos por parênteses. Listas podem ser atribuídas a arrays ou a listas de variáveis.

Exemplo:

```
@Array = ('texto', $foo, 8, "var interpolada: $foo");
```

Uma lista de literais pode ser atribuída a uma lista de variáveis:

```
($ouro, $prata, $bronze) = ("dourado", "prateado", "bronzado");
```

Podemos atribuir um array a uma lista de variáveis escalares:

```
@Agora = gmtime;  
($dia,$mes,$ano..)=@Agora;
```

Acessando os elementos do array usando o índice numérico:

```
print $Agora(0);
```

Acessando os elementos do array usando o comando "foreach":

```
foreach $elemento (@Agora){  
    print $elemento;  
}
```

Atribuindo-se um array a uma variável escalar, estaremos passando o tamanho do array, um inteiro.

```
$tamanho=@Agora;  
print $tamanho;
```

Isto imprimirá na tela:

3

Hashs

São coleções de escalares referenciados por uma chave definida como outro escalar (string). São precedidos pelo sinal de percentagem (%).

Exemplo: **%Hash**

Podemos acessar e modificar cada par chave e valor usando as palavras reservadas key e value ou as outras notações alternativas.

```
keys %Hash=(fruta,masa,liquido);  
values %Hash=("pera", "lasanha", "vinho");
```

O comando abaixo produziria o mesmo efeito:

```
%Hash=(  
    fruta=>pera,  
    masa=>lasanha,  
    liquido=>vinho  
);
```

Ou ainda:

```
%Hash=(fruta,pera,masa,lasanha,liquido,vinho);
```

Para acessar os dados de um hash precisamos usar a palavra-chave entre chaves ("{" e "}")

Exemplo:

```
print $Hash{"fruta"};
```

Este comando imprimirá:

pera/>_

O prompt aparecerá logo após a palavra pera porque não incluímos o caracter "\n".

Podemos imprimir todo o Hash com os comandos:

```
foreach $chave (keys %Hash) {
    print "$chave : $Hash{$chave}\n";
}
```

Controle de fluxo

O Perl usa as mesmas palavras reservadas que o C para o controle dos blocos de programa e algumas próprias, como foreach.

if elif

Exemplo:

[illegible]

```
    print("$foo é menor que 10\n");
    $foo++;
} else {
    print "Ops! $foo é igual a 10!!!\n";
}
```

while

Exemplo:

```
$foo=0;
while ( $foo < 10 ) {
    print("$foo é menor que 10\n");
    $foo++;
}
print "Ops! $foo é igual a 10!!!\n";
```

for

Exemplo:

```
for ( $foo=0;$foo < 10 ;$foo++;) {
    print("$foo é menor que 10\n");
}
print "Ops! $foo é igual a 10!!!\n";
```

foreach

Exemplo:

```
@foo=(0,1,2,3,4,5,6,7,8,9);
foreach $foo(@foo) {
    print("$foo é menor que 10\n");
}
$foo=10;
print "Ops! $foo é igual a 10!!!\n";
```

goto label

Exemplo:

```
print "Isto será impresso\n";
goto Fim;
print "Isto nunca será impresso";
```

```
:Fim  
print "Isto também será impresso\n";
```

Variáveis especiais

O perl usa uma série de variáveis especiais que facilitam e otimizam o código. As mais importantes são \$_ e @_ que sempre se referem ao último valor processado.

Exemplo:

```
$foo="Olha, cara!";  
print;
```

Imprimira o valor de \$foo.

Arquivos – leitura e escrita

Handle de arquivo é um container que guarda o arquivo, geralmente em forma de array

Exemplo: abrindo um arquivo para escrita:

```
open FILE, ">arquivo.txt";
```

Exemplo: abrindo um arquivo para leitura e atribuindo seu conteúdo a um array:

```
open FILE, "arquivo.txt";  
@file=<FILE>;
```

O sinal <FOO> refere-se ao conteúdo do arquivo cujo handle é FOO. Se for usado <> será acessado o conteúdo de STDIN.

O algoritmo abaixo lê as entradas do console, até encontrar um final de arquivo (control D) acumulando no string foo:

```
while (<>) {  
    $foo .=$_;  
}
```

Para assuntos mais aprofundados sobre abertura e fechamento de arquivos, é bom ler o tutorial sobre arquivos, usando o man, do GNU/Linux.

```
/>man perlperltut
```

Subrotinas

Um bloco de comandos nomeado com o comando "sub" define uma subrotina ou função. Para executar a subrotina basta preceder seu nome pelo "&";

Exemplo:

O comando abaixo imprimirá "Oi, Turma!!"

&oiturma;

```
sub oiturma {  
    print "Oi, Turma!!\n\n";  
}
```

A subrotina retornará o valor de \$_ ou @_, o último a ser alterado, a menos que exista o comando "return";

print \$oi,&oiturma;

```
sub oiturma {  
    $oi="Oi, ";  
    return "Turma!!");  
}
```

Bibliotecas

Podem ser carregadas bibliotecas com o comando "require". Bibliotecas são arquivos contendo subrotinas. O valor de retorno de um script biblioteca precisa ser 1.

Exemplo:

Arquivo executável

```
-----  
#!/usr/bin/perl  
# printwindow.pl  
# incluindo o arquivo que contem a rotina "janela"  
require "/home/perl/rotinas/janela.pl"  
# imprimindo  
print &janela;  
-----
```

Biblioteca

```
-----  
#!/usr/bin/perl  
# janela.pl  
# define palavra  
sub janela{  
    $palavra="Window";  
}  
return -1;  
-----
```

Podemos ainda carregar módulos usando o comando "use".

Exemplo

use CGI;

Usando o Perl na Internet

Devido às suas características que facilitam o processamento de strings, à versatilidade e simplicidade da linguagem, a facilidade de manutenção do programa e sua velocidade de execução, o Perl tornou-se o ambiente preferido para o desenvolvimento de scripts para a CGI. Em 1996 Steven E. Brenner criou a biblioteca cgi-lib.pl, que tornou-se uma espécie de padrão no Perl 4. Atualmente, o Perl 5 oferece uma série de módulos que facilitam muito o processamento de html, forms, email, ftp, newsgroups, XML WML entre outros.

Cabeçalho do HTTP 1.0

Quando um arquivo é enviado para um cliente Internet (como o Netscape, por exemplo) pelo servidor, este deve conter um cabeçalho na forma mínima:

Content-type: text/plain

Seguido de 2 caracteres de nova linha. Isto indica o tipo MIME do documento, orientando o navegador de como este deve processado.

Um exemplo usando o cabeçalho completo seria:

```
HTTP/1.0 200 OK  
Date: Thursday, 28-June-96 11:12:21 GMT  
Server: NCSA/1.4.2  
Content-type: text/html  
Content-length: 2041
```

Variáveis de ambiente

A CGI usa as variáveis de ambiente na transação de pedido e envio de arquivos para controlar e processar algoritmos.

Temos acesso a dados do cliente que fez o pedido e a detalhes deste pedido, pode-se lidar com os protocolos de autorização, "Mime Types", cookies. Não pretendemos aqui neste trabalho aprofundar o estudo da CGI, apenas pretendemos exemplificar o uso do Perl.

Para Ter acesso às variáveis do CGI, use o hash %ENV:

```
#!/usr/bin/perl
```

```
print "Content-type: text/plain\n\n";  
foreach $var (keys %ENV) {  
    print "<p>$var: $ENV{$var}\n";  
}
```

Este programa gerou na tela no meu navegador da Internet:

```
SERVER_SOFTWARE: Apache/1.2.4  
GATEWAY_INTERFACE: CGI/1.1  
DOCUMENT_ROOT: /home/httpd/html/gallerie/html/
```



```
REMOTE_ADDR: 172.16.1.1
SERVER_PROTOCOL: HTTP/1.1
REQUEST_METHOD: GET
REMOTE_HOST: guitar.escriptorio.casa
QUERY_STRING:
HTTP_USER_AGENT: Mozilla/4.0 (LINUX)
PATH: /bin:/sbin:/usr/bin:/usr/sbin
HTTP_ACCEPT: */*
HTTP_CONNECTION: Keep-Alive
REMOTE_PORT: 2057
HTTP_ACCEPT_LANGUAGE: pt-br
SCRIPT_NAME: /cgi-bin/getenv
HTTP_ACCEPT_ENCODING: gzip, deflate
SCRIPT_FILENAME: /home/httpd/html/gallerie/cgi-bin/getenv
SERVER_NAME: gallerie.escriptorio.casa
REQUEST_URI: /cgi-bin/getenv
SERVER_PORT: 80
HTTP_HOST: gallerie.escriptorio.casa
SERVER_ADMIN: ricardo@gallerie.escriptorio.casa
```

Lendo um <form>

Podemos chamar um script Perl de dentro de uma página HTML que contém um <form>. O meu Apache está configurado para rodar scripts no diretório /cgi-bin/ que aponta para o diretório real, na máquina: /home/httpd/cgi-bin/

O Formulário teria a forma:

```
<html><body>
<form action=http://gallerie.escriptorio.casa/cgi-bin/getenv method=post>
<input name=palavra>
<input type=hidden name=escondido value="Isto já veio escrito...">
</form>
</body></html>
```

O script abaixo deve ser salvo no diretório /home/httpd/cgi-bin/ e deve Ter permissão para executar:

```
/> chmod 744 ./getenv
```

Note que foi usada a biblioteca cgi-lib.pl e sua rotina &ReadParse, que coloca a string "Query" (aqui escondida pelo método post) no array @in:

```
#!/usr/bin/perl
# getenv
require "cgi-lib.pl";
&ReadParse;

print "Content-type: text/plain\n\n";
print "Variaveis:\n";
foreach $vari(@in){
print "$vari\n";
}
}
```

```
foreach $var (keys %ENV) {  
    print "<p>$var: $ENV{$var}\n";  
}
```

Gerando a tela:

Variaveis:

```
palavra=teste  
escondido=Isto j%E1 veio escrito...  
SERVER_SOFTWARE: Apache/1.2.4  
GATEWAY_INTERFACE: CGI/1.1  
DOCUMENT_ROOT: /home/httpd/html/gallerie/html/  
REMOTE_ADDR: 172.16.1.1  
SERVER_PROTOCOL: HTTP/1.1  
REQUEST_METHOD: POST  
REMOTE_HOST: guitar.escriptorio.casa  
QUERY_STRING:  
HTTP_USER_AGENT: Mozilla/4.0 (LINUX)  
PATH: /bin:/sbin:/usr/bin:/usr/sbin  
HTTP_ACCEPT: */*  
HTTP_CONNECTION: Keep-Alive  
REMOTE_PORT: 2059  
HTTP_ACCEPT_LANGUAGE: pt-br  
SCRIPT_NAME: /cgi-bin/getenv  
HTTP_ACCEPT_ENCODING: gzip, deflate  
SCRIPT_FILENAME: /home/httpd/html/gallerie/cgi-bin/getenv  
SERVER_NAME: gallerie.escriptorio.casa  
REQUEST_URI: /cgi-bin/getenv  
SERVER_PORT: 80  
CONTENT_LENGTH: 49  
CONTENT_TYPE: application/x-www-form-urlencoded  
HTTP_HOST: gallerie.escriptorio.casa  
SERVER_ADMIN: ricardo@gallerie.escriptorio.casa
```

Módulo CGI

Poderíamos escrever o mesmo programa com o módulo CGI, gerando a saída em html. Para um aprofundamento na questão digite:

```
/>perldoc CGI
```

Abaixo o novo script getenv:

```
#!/usr/bin/perl  
# getenv  
use CGI;  
$tela=CGI::new();  
print (  
    $tela->header,  
        $tela->start_html ( -title=>"Variaveis de Ambiente"),  
    $tela->p("Variaveis:");  
    $tela->dump  
);
```

```
foreach $var (keys %ENV) {  
    print "<p>$var: $ENV{$var}\n";  
}  
print $tela->end_html;
```

o arquivo de retorno em meu navegador foi:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">  
<HTML><HEAD><TITLE> Variaveis de Ambiente </TITLE>  
</HEAD><BODY><P>Variaveis:</P><p>SERVER_SOFTWARE: Apache/1.2.4  
<p>GATEWAY_INTERFACE: CGI/1.1  
<p>DOCUMENT_ROOT: /home/httpd/html/gallerie/html/  
<p>REMOTE_ADDR: 172.16.1.1  
<p>SERVER_PROTOCOL: HTTP/1.1  
<p>REQUEST_METHOD: POST  
<p>REMOTE_HOST: guitar.escriptorio.casa  
<p>QUERY_STRING:  
<p>HTTP_USER_AGENT: Mozilla/4.0 (LINUX)  
<p>PATH: /bin:/sbin:/usr/bin:/usr/sbin  
<p>HTTP_ACCEPT: */*  
<p>HTTP_CONNECTION: Keep-Alive  
<p>REMOTE_PORT: 2063  
<p>HTTP_ACCEPT_LANGUAGE: pt-br  
<p>SCRIPT_NAME: /cgi-bin/getenv  
<p>HTTP_ACCEPT_ENCODING: gzip, deflate  
<p>SCRIPT_FILENAME: /home/httpd/html/gallerie/cgi-bin/getenv  
<p>SERVER_NAME: gallerie.escriptorio.casa  
<p>REQUEST_URI: /cgi-bin/getenv  
<p>SERVER_PORT: 80  
<p>CONTENT_LENGTH: 49  
<p>CONTENT_TYPE: application/x-www-form-urlencoded  
<p>HTTP_HOST: gallerie.escriptorio.casa  
<p>SERVER_ADMIN: ricardo@gallerie.escriptorio.casa  
</BODY></HTML>
```

O módulo CGI facilita muito a formatação da página, leitura e processamento de cookies, escrita de tabelas, forms, etc.

Administrando bancos de dados

Módulo DB_File

O módulo DB_File e a distribuição Berkeley DB em geral está presente nos sistemas GNU/Linux, mas pode ser obtida em <http://www.sleepycat.com/>

Podemos ligar um arquivo com dados do tipo nome:valor a um arquivo Hash. Bancos de dados simples podem ser desenvolvidos desta forma.

Devemos usar o comando "tie" para fazer a ligação. Exemplo:

```
#!/usr/bin/perl  
# telefones
```

```

use DB_File;
$caderno= tie %TEL, "DB_File", "/home/ricardo/telefones.txt";
                                while (TRUE){ # criando um loop:

# Menu
system("clear");
print "----- Agenda de Telefones -----\\n";
$lido=&Digitado("Entre com o comando: [A] Incluir [M] Mostrar Lista [S] Sair:", c);
if ($lido eq "A"){&Incluir;}
elsif ($lido eq "M"){&Mostrar;}
elsif ($lido eq "S"){&Sair;}                } # fechando o loop

sub Digitado {
my $xx, $tx;
$tx="";
print $_[0], "\\n";
while ($lendo ne 1){$xx=getc; if ($xx eq "\\n"){&lendo=1;}else{$tx.=$xx;} }
&lendo=0;
print "-----> Foi lido: $tx\\n";
return $tx;}

sub Incluir {
    $nome=&Digitado ("Nome a incluir: ");
    $tel=&Digitado("Telefone: ");
$caderno->put($nome, $tel);}
sub Mostrar {
    foreach $nome(keys %TEL){
        print "Nome: $nome\\n";
        print "Telefone: $TEL{$nome}\\n-----\\n";
        &Digitado("Tecle enter para continuar");
    }
}
sub Sair {exit(0);}

```

Para maiores detalhes sobre o módulo, digite:

```
/>perldoc DB_File
```

Bancos de dados mais complexos devem usar o módulo DBI, que servirá de interface com servidores SQL, como mSQL, MySQL, Oracle, postgres, etc.

Interface gráfica

Módulo perlTk

Pré-requisito: Tk 4.0 ou superior instalado.

O Nosso objetivo aqui é apenas apresentar o Perl/Tk. Estudos mais aprofundados dos comandos do Tk podem ser feitos digitando:

```
/>perldoc Tk
```

O Perl/Tk é usado para criar uma interface gráfica para os programas em Perl. Com este módulo torna-se muito fácil a criação de janelas, botões, scrolls, canvases, caixas de texto, etc.

Olha o programa "oiturma" com uma interface gráfica:

```
#!/usr/bin/perl  
use Tk;  
$janela=MainWindow->new;  
$janela->Button(  
    -text=>"Oi, Turma!!",  
    -command=>sub{exit})->pack;  
MainLoop;
```

Aqui é importante lembrar que a interface só será desenhada na tela depois do comando "MainLoop". O método "pack" é responsável pela formatação e posicionamento dos dispositivos na janela. Existem ainda mais dois gerenciadores de geometria: grid e place.

A chamada de programa quando o botão é clicado pode usar uma referência:

Exemplo:

```
#!/usr/bin/perl  
use Tk;  
$janela=MainWindow->new;  
$janela->Button(  
    -text=>"Oi, Turma!!",  
    -command=>\&Sair)->pack;  
MainLoop;  
  
sub Sair {  
    exit;  
}
```

Nota sobre o autor

Ricardo Luiz Fillipo

Formação Acadêmica:

- 1999– Mestrado em Musicologia concluído no Conservatório Brasileiro de Música
Tema da tese: Expressão Musical na Internet.
- 1989– Curso Técnico de Análise de Sistemas concluído no FESP–RJ
- 1985– Bacharelado em Música concluído na UFRJ.
- 1980– Curso de Engenharia (Fundação Valeparaibana de Ensino–São José dos Campos–São Paulo) até o quinto período.
- 1977– Curso de Segundo Grau concluído no Instituto Mackenzie – S.Paulo
- 1974– Primeiro Grau concluído no Colégio Santo Agostinho – S.Paulo.

Experiência Profissional **Informática**

- Diretor da empresa Gallerie (GWebCultural Ltda.), especializada em consultorias para negócios na Internet.
- Consultor desde 1989.
- Webmaster desde 1996.
- Professor na graduação (Universidade Estácio de Sá).
- Professor na pós-graduação – Projeto e Implementação na WEB (Curso Análise, Projeto e Gerência de Sistemas– Universidade Estácio de Sá)
- Professor membro da CIPSGA, braço brasileiro da "Free Software Foundation" – Especialidade: CGI e Perl.
- Membro da IWA (International Webmasters Association).

* Especialidades na Informática: conceito, desenvolvimento e design de Websites, programação de CGI (Internet e intranet), programação em ambientes UNIX, programação em Perl.

* Ambientes operacionais: UNIX, Windows, MacOs.

* Linguagens que domina: perl, pascal, C, C++, VB, Basic, shell do UNIX (bash, tcsh), shell do DOS (arquivos de lote).

Música

- * Concertista desde 1979
- * Arranjador, Maestro desde 1986
- * Professor desde 1982 (Conservatório Brasileiro de Música).
- * Especialidades: instrumento (violão), arranjos orquestrais e de camera, regência orquestral.

Produção **Informática:**

Atualmente mantém os seguintes sites na Internet:

<http://www.uol.com.br/ruthrocha>

<http://www.gallerie.com.br/rc-botafogo>

<http://www.gallerie.com.br/pdominguez>

<http://www.gallerie.com.br>

<http://www.bga.com.br>
<http://www.cdclassic.com.br>
<http://www.cbm-musica.org.br>
<http://www.arcadoce.com.br>
<http://www.cliquesolidariedade.com.br>
<http://www.novorio.com.br/hoteisepousadas>
<http://www.e-jornal.com.br> –Site do EnseadaJornal
<http://www.enseadajornal.com.br> –Site do EnseadaJornal
<http://violao.w3.to>
<http://www.gwebcultural.com.br>
<http://rionet.com.br/~diogenes>

Música

Ricardo é professor de música no "Conservatório Brasileiro de Música", onde leciona nas cadeiras de Violão, Contrabaixo, Guitarra Elétrica e Harmonia Funcional, desde 1982. Tem apresentado Workshop sobre Informática e Música abrangendo o uso de aplicativos sintetizadores de som, editores de partituras e de arquivos sonoros e uso de música na Internet e Multimídia em Geral.

Está atualmente implementando um projeto de desenvolvimento de software musical em ambiente LINUX, no Laboratório de Informática e Música do Conservatório Brasileiro de Música.

É concertista, compositor e maestro, tendo se apresentado em centenas de ocasiões, em programas solo, à frente de orquestras e em formações camerísticas, em diversas cidades do país. Ricardo iniciou seus estudos de música aos cinco anos de idade.

Gravou 3 discos:

O primeiro com Turíbio Santos e Orquestra de Violões do Rio de Janeiro, em 1984.

O Segundo em 1989 com a Orquestra Rio de Violões a qual dirigiu.

O terceiro, "Carinhoso" em 1996, juntamente com Duda Anízio, lançado pela gravadora L'ART.

Ricardo Filipo
rfelipo@cipsga.org.br
r@w3c.com.br

Gallerie
<http://www.gallerie.com.br>

Controle de versões

<i>Data</i>	<i>Autor</i>	<i>Observações</i>
01/11/1999	Ricardo Filipo	Texto inicial
11/06/00	Ricardo Filipo	Apostila disponibilizada sobre a FDL para o CIPSGA.

GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for

which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page.

For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.
- O. If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections

of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document. If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires especial permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents.

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1

or any later version published by the Free Software Foundation;

with the Invariant Sections being LIST THEIR TITLES, with the

Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Comitê de Incentivo a Produção do Software Gratuito e Alternativo



Fundado em 29 de janeiro de 1999.

1ª Diretoria

Djalma Valois Filho

Diretor Executivo

dvalois@cxpostal.com

José Luiz Nunes Poyares
Diretor Administrativo

Paulo Roberto Ribeiro Guimarães
Diretor Institucional

CIPSGA
Rua Professora Ester de Melo, numero 202,
Parte, Benfica, Rio de Janeiro, RJ, CEP. 20930-010;
Telefone (Fax/Dados): 021-5564201;
e-mail: administracao@cipsga.org.br
CNPJ: 03179614-0001/70