Unos programas en PERL

Dr. Luis Gerardo de la Fraga

Resumen

Se presentan tres programas de PERL en donde se visualiza el poder de este lenguaje como un medio para conectar varias aplicaciones.

1 Una programa que suma dos números

Este es un programa simple, le llamaremos `prog`:

```
#!/usr/bin/perl
#
# Fraga 23/03/1999
#
if ( $#ARGV == 0  ) {
    die "Args: a b\n";
}
elsif ( $#ARGV == 1 ) {
    $a = $ARGV[0];
    $b = $ARGV[1];
}
else {
    print "De el valor de a = ";
    $a = <STDIN>; chop $a;
    print "De el valor de b = ";
    $b = <STDIN>; chop $b;
}
$c = $a + $b;
print "$a + $b = $c\n";
```

Su lógica es simple: si se da un solo argumento, `$#ARGV == 0`, el programa termina. Si se dan dos argumentos al programa, `$#ARGV == 1`, estos son números que se van a sumar. Y si no se dan argumentos se pregunta por cada número. Al final imprime los dos números y su suma.

2 Llamadas repetidas al programa

¿Cómo llamar varias veces al programa? Hay dos soluciones. La primera es hacer otro programa en PERL que llame las veces necesarias al programa `prog`:

```
#!/usr/bin/perl

# Primera opcion para llamar
# al programa prog
‘prog  1 2 > sale‘;
‘prog  3 4 >>sale‘;
‘prog  5 6 >>sale‘;
‘prog  7 8 >>sale‘;
```

La salida de este programa se almacena en el archivo `sale` que se muestra a continuación.

```
1 + 2 = 3
3 + 4 = 7
5 + 6 = 11
7 + 8 = 15
```

3 Otra solución

Una solución más elegante para llamar repetidas veces al programa `prog` es crear un archivo con la lista de argumentos, en este caso serán pares de números que guardaremos en el archivo `data.txt`:

```
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
```

Y creamos un programa en PERL, supongamos que se llama `p1`, que recibe como argumento el nombre del archivo con los datos. El programa `p1` quedaría como:

```
#!/usr/bin/perl
# Segunda opcion para llamar
# al programa prog

die "Sin argumento me lamento\n" if ( $#ARGV < 0 );
```

```
open( INPUT, "$ARGV[0]" ) or die "Can't open $ARGV[0] file\n";

if ( -e "sale" ) {
    unlink "sale";
}
while ( <INPUT> ) {
    chop;
    `prog $_ >> sale`;
}
close INPUT;
```

Y la salida al ejecutar este programa como `p1 data.txt` queda guardada en el archivo `sale`. El contenido de `sale` es:

```
1 + 2 = 3
3 + 4 = 7
5 + 6 = 11
7 + 8 = 15
9 + 10 = 19
11 + 12 = 23
13 + 14 = 27
15 + 16 = 31
17 + 18 = 35
```

Y aquí vemos como PERL nos ayuda a pegar el programa `prog` con los datos en el archivo `data.txt`. Es la flexibilidad y el poder de la programación en un lenguaje de muy alto nivel.