

MENGENAL PERL

Tentang PERL

PERL adalah bahasa pemrograman yang menggunakan tipe data dinamis, program PERL dapat langsung dieksekusi tanpa harus melalui proses kompilasi tersendiri ke format binary yang dapat dieksekusi. PERL banyak digunakan pada aplikasi manajemen sistem dan jaringan, pemrograman web, manipulasi teks, akses ke database. Berikut ini beberapa aplikasi yang menggunakan PERL yang cukup dikenal: mrtg, SQL Ledger (aplikasi akunting), Webmin (aplikasi admin), spam assassin (anti spam).

Menjalankan Program PERL

Eksekusi dari file

Berikut ini sebuah program PERL salam.pl, yang mencetak salam universal di konsol, jika ada argumen pada command line maka ditampilkan setelah salam.

```
#!/usr/local/bin/perl
if ($#ARGV > 1)
{
print "hello $ARGV[0]\n";
} else {
print "hello\n";
}
```

Eksekusi di shell menggunakan PERL interpreter, sbb:

```
bash% perl salam.pl Bejo
helo Bejo
```

Agar dapat dieksekusi tanpa mengetikkan interpreter perl di command line, file salam.pl perlu diset permissionnya. Misalkan dengan perintah :

```
bash% chmod 755 salam.pl
bash% ./salam.pl Adhie
hello Adhie
```

Pastikan program PERL diawali dengan path lengkap ke interpreter perl yang terinstal di sistem operasi anda, untuk sistem berbasis Linux dan Unix, path tersebut harus diawali dengan karakter '#!' , seperti contoh:

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

Eksekusi dari command line

PERL dapat mengeksekusi perintah secara langsung dari command line, tanpa perlu membuat file program, menggunakan options e di interpreter perl, contoh:

```
bash% perl e
'print "hello\n"'
```

Dasar-dasar PERL

Program PERL

Program PERL atau skrip PERL, biasa disimpan dalam file teks berakhiran .pl.

Walaupun

demikian perl interpreter tetap akan mengeksekusi program PERL yang tidak

menggunakan

ekstensi .pl.

Sebuah program PERL yang executable dapat menggunakan hanya statement, tanpa perlu mendefinisikan fungsi khusus seperti main() dalam program bahasa C.

Program PERL dapat terdiri atas statemen, deklarasi subroutine, konstruksi kondisional, konstruksi loop, konstruksi block.

Statemen

Statemen diakhiri dengan tanda titik coma. Statemen dapat berbentuk deklarasi variabel, assignment variabel dan pemanggilan fungsi. White space (spasi, baris baru) diabaikan.

```
print "open source software ?!",  
" it's a miracle",  
", do you believe miracle?\n";
```

Komentar

Sebagaimana bahasa pemrograman lainnya, programmer PERL dapat menuliskan komentar

dalam kode program, dengan menempatkan karakter '#' sebelum baris komentar, contoh:

```
# menghitung hitung luas
```

```
$area = $pi * $radius * $radius;
```

Atau dapat juga setelah bagian kode:

```
$area = $pi * $radius * $radius; # menghitung hitung luas
```

Komentar diperlakukan sebagaimana white space.

Variabel dan Tipe data

Variabel digunakan untuk menyimpan data. PERL memiliki 3 tipe variabel utama: skalar, array dan hash. Nama-nama variabel, dan identifier lainnya dalam PERL adalah casesensitive.

Skalar

Variabel skalar menyimpan data bernilai tunggal, tipe data dapat berupa string atau bilangan. Tipe data secara otomatis ditentukan pada saat assignment variabel. tidak perlu mendeklarasikan tipe data sebelum assignment. Nama variabel skalar dimulai dengan karakter '\$'.

Contoh:

```
# string
```

```
$file = '/etc/profile';
```

```
$kota = "depok";
```

```
# integer
$_num = 255;
$byte_ = 0x00ff;
# float
$konstanta_pi = 22/7;
```

String petik tunggal (single quote) dan petik ganda (double quote)

String dapat dinyatakan dalam petik tunggal (') dan petik ganda (").

String dengan petik tunggal menyatakan data adalah seperti yang tertulis, tidak ada interpretasi terhadap karakter-karakter pada string. String dengan petik ganda menyatakan bahwa interpreter PERL harus melakukan interpretasi terhadap karakter-karakter pada string tersebut.

Contoh:

```
# dicetak dalam satu baris, karakter \n tidak diinterpretasikan
$str1 = '\niman \nilmu \namal \n';
print $str1;
# dicetak dalam 3 baris diikuti baris baru
$str2 = "\niman \nilmu\namal \n";
print $str2;
# mencetak 'honour the adzan' diikuti baris baru
$str3 = 'honour';
$str4 = "$str3 the adzan\n";
```

Variabel Khusus

PERL juga mengenal beberapa variabel skalar khusus, seperti \$ARG, \$_, \$!, \$1, \$2, \$3, dan sebagainya yang dijelaskan di man perlvar

Array

Array menyimpan sekumpulan nilai secara berurutan (sekuensial), yang biasa disebut list, nilai nilai yang disimpan dapat berbeda-beda tipe datanya. Untuk membaca nilai anggota array digunakan nomor indeks integer dimulai dari nol. Variabel array dideklarasikan menggunakan karakter @ di depan nama variabel, data anggota array ditempatkan dalam kurung (), nilai-nilai anggota dipisahkan oleh koma.

Nilai anggota array dibaca menggunakan operator [] dan nomor indeks.

Contoh:

```
@prima1 = (2,3,5,7,11,13,17);
# mencetak '5'
print $prima1[2];
```

Untuk mendapatkan jumlah anggota array digunakan variabel \$# diikuti nama variabel, tapi nilainya adalah jumlah anggota array dikurangi satu. Nilai ini juga dapat digunakan untuk membaca elemen terakhir.

Contoh :

```
@hari2 = (1, "senin", 2, "selasa", 3, "rabu", 4, "kamis", 5, "jumat");
```

```
# mencetak 'jumlah anggota : 9'
print "jumlah anggota : $#hari2 \n";
# mencetak 'elemen terakhir : jumat'
print "elemen terakhir : $hari2[$#hari2] \n";
```

Jika sebuah program mencoba menulis anggota array pada posisi indeks yang lebih besar dari ukuran array, maka PERL secara otomatis menambah ukuran array sampai posisi baru tersebut.

Contoh:

```
@bil3 = (10, 20, 30, 40);
# mencetak 'ukuran array : 3'
print "ukuran array : $#bil3 \n";
$bil3[110] = 1234;
# mencetak 'ukuran array : 110'
print "ukuran array : $#bil3 \n";
```

Membaca dari array dengan indeks yang lebih besar tidak akan menyebabkan error dan tidak akan menambah ukuran array.

Contoh:

```
@bil4 = (10,20,30,40);
$t1 = $bil4[89];
# mencetak 'ukuran array : 3'
print "ukuran array : $#bil4 \n";
Jika variabel array dievaluasi dalam konteks nilai skalar (scalar context) maka variabel tersebut dievaluasi sebagai jumlah anggota array,
```

contoh:

```
# dievaluasi dalam scalar context
@arr = (10,20,6,19,11,22);
if(@arr == 6) {
print "jumlah anggota array ada 6\n";
}
```

Array dapat disalin sebagian anggotanya ke array lain dengan mudah (array slice), menggunakan operator [],

Contoh:

```
@arr1 = ('semar', 'gareng', 'petruk', 'bagong', 'arjuna',
'srikandi');
@arr2 = @arr1[2,4] # 'petruk' dan 'arjuna'
@arr3 = @arr1[1..3] # 'gareng', 'petruk', 'bagong'
@arr4 = @arr1[1..$#arr1] # semua kecuali 'semar'
```

3.2.3.Hash

Hash menyimpan sekumpulan nilai yang menggunakan pasangan nama kunci dan nilai.

Nama kunci dalam sebuah variabel hash haruslah unik, tidak boleh ada 2 pasangan yang menggunakan nama kunci yang sama.

Variabel hash didefinisikan menggunakan karakter % diikuti nama variabel.

Nilai-nilai

anggota hash terdiri atas pasangan nama kunci dan nilai yang dipisahkan oleh koma. Nama kunci dan pasangan dituliskan berurutan, dapat dipisahkan tanda koma atau dapat juga dipisahkan oleh karakter '=>'. Untuk membaca nilai anggota hash digunakan operator kurawal {} dan nama kunci.

Contoh:

```
%rasa_buah1 = ('asam', 'asem djawa', 'manis', 'manggis', 'pahit', 'pare');  
# mencetak 'yang pahit = pare'  
print "yang pahit = $rasa_buah1{'pahit'} \n";# dapat juga ditulis seperti ini  
%rasa_buah2 = ('asam' => 'asem djawa',  
'manis' => 'manggis',  
'pahit' => 'pare');  
$rasa = 'manis';  
# mencetak 'yang manis = manggis'  
print "yang $rasa = $rasa_buah2{$rasa} \n";
```

Operator

PERL memiliki banyak operator dan fungsi yang built-in, berikut ini beberapa operator yang umum digunakan, selengkapnya dapat dilihat dengan man perlop.

Manipulasi String

. Concat (penggabungan)

Aritmetika

+ Penjumlahan

Pengurangan

* Perkalian

/ Pembagian

Perbandingan Bilangan

'==' kesamaan

'!=' Ketidaksamaan

< Kurang dari

> Lebih dari

<= Kurang dari atau sama dengan

>= Lebih dari atau sama dengan

Perbandingan String

eq kesamaan

ne Ketidak samaan

lt Kurang dari

gt Lebih dari

le Kurang dari atau sama dengan

ge Lebih dari atau sama dengan

Perhatikan bahwa PERL menyediakan operasi pembandingan string dan pembandingan bilangan.

String dibandingkan secara alpabetis. Secara alpabetis, string '100' lebih kecil dari '8'.

Contoh:

```
$a = 100; $b = 8
```

```
if ( $a > $b ) {
```

```
print "Bilangan $a lebih besar dari $b\n";
```

```
} else {
```

```
print "Bilangan $a tidak lebih besar dari $b\n";
```

```
}
```

```
if ( $a gt $b ) {
```

```
print "String $a secara alpabetis lebih besar dari $b";
```

```
} else {
```

```
print "String $a secara alpabetis tidak lebih besar dari $b";
```

```
}
```

Operasi perbandingan menghasilkan nilai benar (true) atau salah (false).

TRUE dan FALSE dalam PERL

PERL tidak mengenal tipe data khusus untuk mewakili TRUE dan FALSE.

Dalam PERL nilai berikut ini adalah FALSE: bilangan 0, string '0', list kosong (), dan undef, selain nilai-nilai tersebut adalah TRUE.

Fungsi

PERL memiliki fungsi built-in yang kaya, salah satu yang sudah diperkenalkan adalah print. Fungsi dapat menerima nol atau lebih argumen. Argumen sebuah fungsi dapat berupa skalar, list, atau keduanya. Pemanggilan fungsi dapat menggunakan tanda kurung () diantara argumen, dapat juga tidak. Fungsi print adalah fungsi yang menerima argumen list.

Contoh:

```
print "saya", "belajar", "PERL";
```

```
# sama dengan
```

```
print ("saya", "belajar", "PERL");
```

Fungsi print juga dapat menerima argumen skalar berupa filehandle dan list. Tentang filehandle akan dijelaskan dibagian File Input dan Output. Fungsi dapat mengembalikan nilai skalar atau list. Berikut ini fungsi localtime, yang mengembalikan list berisi data waktu, dipanggil tanpa argumen:

```
@bulan = ('jan', 'feb', 'mar', 'apr', 'mei', 'jun', 'jul', 'ags', 'sep', 'okt', 'nop', 'des');
```

```
@hari = ('minggu', 'senin', 'selasa', 'rabu', 'kamis', 'jumat', 'sabtu');
```

```
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) = localtime;
```

```
print "Bulan $bulan[$mon] hari $hari[$wday] tgl $mday tahun ", 1900+$year, "\n";
```

Perlu diingat bahwa PERL pertamakali dikembangkan di lingkungan Unix, jadi akan banyak ditemukan fungsi-fungsi yang familiar dengan sistem operasi ini, misalnya yang berhubungan dengan file system, socket, network, interprocess communication, dan

process control. Daftar fungsi builtin secara lengkap dapat dibaca dengan man perlfunc. Penjelasan untuk fungsi tertentu dapat dicari dengan perldoc -f .

Contoh:

```
bash% perldoc -f localtime
```

Konstruksi Kondisional

Berikut ini konstruksi kondisional yang disediakan PERL

if

Mengeksekusi statemen jika kondisi bernilai benar. Untuk percabangan menggunakan elsif dan else.

Bentuknya :

```
if ( kondisi1 ) { # statemenstatemen }
elsif ( kondisi2 ) {
# statemenstatemen
} elsif ( kondisi3 ) {
# statemenstatemen
} else {
# statemenstatemen
}
```

unless

Adalah kebalikan dari if.

Konstruksi Loop

PERL mengenal beberapa jenis loop:

for

Digunakan untuk melakukan pengulangan berdasarkan 3 ekspresi yang masing-masing untuk

: memulai loop, menentukan apakah melanjutkan eksekusi loop, dieksekusi pada akhir loop supaya kondisi untuk mengakhiri loop dapat dicapai.

Bentuknya:

```
for ( ; ; ) { # statemenstatemen }
```

Ekspresi yang umum digunakan adalah sebagai berikut:

```
for ( $i = 0; $i < $jumlah_pengulangan; $i++ ) { # statemen }
```

foreach

Digunakan untuk melakukan iterasi berdasarkan anggota sebuah array atau list.

Bentuknya:

```
foreach () { }
```

Contoh:

```
# iterasi atas array
```

```
@hari = ('senin','selasa','rabu','kamis','jumat','sabtu','minggu');
```

```
foreach $h (@hari) {
```

```

print "hari $h\n";
}
# terhadap list
$p = 1;
foreach $q (2..6) {
    $p *= $q;
}

print "faktorial 6 = $p\n";
# atau terhadap array slice
foreach $q (@hari[2..5]) {
    print "$q\n";
}

```

while

Loop dieksekusi selama kondisi adalah TRUE. Untuk keluar dari loop harus dengan mengubah kondisi menjadi false atau menggunakan statement last;

Bentuknya :

```
while(kondisi) {# statemen }
```

Input dan Output ke file

Membaca dari file teks

Sebelum file dapat dibaca maka harus file dibuka dengan fungsi open, dengan mode “baca”,

contoh:

```
open($fh, '/etc/passwd');
```

Variable \$fh adalah filehandle yang diperlukan untuk membaca dan menutup file.

Selanjutnya setiap baris dibaca dengan operator <>. sebagai berikut:

```
while($line = <$fh>) {
# memproses $line
}

```

Setelah selesai file ditutup dengan fungsi close, contoh:

```
close($fh)
```

Menulis ke file teks

Perintah yang sama dengan program di atas dapat digunakan untuk menulis ke file.

Pertama, file dibuka dengan mode 'tulis':

```
open($fh, "> /path/ke/file");
```

selanjutnya dapat ditulis dengan perintah print, menggunakan file handle, sbb:

```
print $fh "baris teks yang ditulis ...\n";
```

Seperti halnya dalam hal membaca, maka filehandle harus ditutup dengan close.

Berikut ini contoh membaca dari file dan menulis ke file baru, atau mengcopy file:

```
# mengcopy /etc/profile
$fnam = '/etc/profile';
$fbaru = '/home/didi/copyprofile';
open($fh1, $fnam) or die "gagal baca $fnam, $!";
```

```

open($fh2, ">$fbaru") or die "gagal menulis $fbaru, $!";
while($line = <$fh1>) {
print $fh2 $line;
}
close($fh1);
close($fh2);

```

Ekspresi “or die ...” digunakan sebagai error handler jika file tidak dapat dibuka. Variabel khusus \$! digunakan untuk menampilkan pesan error dari sistem.

Subrutin

Pada program aplikasi umumnya, kumpulan statemen yang mengimplementasikan sebuah

operasi tertentu didefinisikan dalam sebuah subrutin. Beberapa subrutin biasanya didefinisikan dalam sebuah file tersendiri, sehingga dapat digunakan oleh berbagai program pemanggil.

Berikut ini bentuk pendefinisian subrutin :

```
sub {}
```

Subrutin dapat menerima parameter, tapi PERL tidak mengenal named formal parameter, parameter yang dikirimkan ke subrutin dapat diakses dalam subrutin melalui array khusus @_

,berikut ini contoh subrutin sederhana yang menghitung nilai faktorial :

```

sub faktorial {
my ($num) = @_;
my $res = 1;
foreach my $a (1..$num) {
$res *= $a; # atau $res = $res * a
}
return $res;
}

```

```
# return TRUE value 1;
```

Perhatikan bahwa jika subrutin didefinisikan di file tersendiri, maka file tersebut harus mengembalikan nilai TRUE, dalam program diatas digunakan bilangan 1.

Jika subrutin tersebut disimpan di file /home/didi/mylib/math.pl, maka dapat dipanggil dari program lainnya sebagai berikut:

```

req uire "/home/didi/mylib/math.pl"
print "Faktorial 6 = ", faktorial(6), "\n";

```

Penutup

PERL adalah bahasa pemrograman yang berdaya guna tinggi, yang terus digunakan dan dikembangkan oleh komunitas open source. Seperti halnya bahasa open source lain, PERL memiliki sistem dokumentasi yang sangat lengkap, pembaca sangat dianjurkan menelaah manual PERL yang terdistribusi bersama source code PERL.