Praktikum 8

Array di AWK

Tujuan Pembelajaran

Mahasiswa dapat memahami dan menggunakan array dalam bahasa pemrograman awk.

Dasar Teori

Array merupakan kumpulan nilai-nilai yang disebut dengan element. Elemen dalam array dipisahkan sesuai dengan indeksnya. Indeks dapat berupa string ataupun angka. Bab ini akan menjelaskan bagaimana array bekerja pada awk, bagaimana menggunakan elemen dalam array, bagaimana membaca setiap elemen array, serta bagaimana untuk menghapusnya. Selain itu kita juga akan mempelajari bagaimana awk mensimulasikan array multi dimensi, serta beberapa kelemahan array yang bisa kita lihat. Kemudian kita akan membahas fasilitas gawk untuk mengurutkan array dan kemampuan awk untuk menangani array multi-dimensi. Awk hanya mengenali satu nama untuk penamaan variable, array dan fungsi. Sehingga anda tidak bisa menggunakan variable dan array dengan nama yang sama dala satu program awk.

Bahasa awk menyediakan array satu dimensi untuk menyimpan sekelompok string atau angka yang berhubungan. Setiap array dalam awk harus memiliki nama. Penamaan array memiliki sintaks yang sama dengan penamaan variable, nama apapun yang bisa digunakan untuk variable juga bisa digunakan untuk penamaan array. Tapi satu nama tidak bisa digunakan untuk keduanya (array dan variable) dalam satu program awk.

Array dalam awk mirip dengan array pada bahasa pemrograman lainnya, namun terdapat beberapa perbedaan mendasar. Dalam awk anda tidak perlu menyebutkan ukuran array sebelum mulai menggunakannya. Terlebih lagi kita

bisa menggunakan angka atau string apapun sebagai indeks array di awk tidak hanya angka yang berurutan.

Di kebanyakan bahasa lainnya, array harus dideklarasikan sebelum digunakan, termasuk jumlah elemen didalamnya. Dalam bahasa pemrograman seperti ini, deklarasi akan menyebabkan alokasi memory untuk sejumlah elemen tersebut. Biasanya indeks dalam array haruslah angka positif. Sebagai contoh, indeks ke-0 digunakan sebagai indeks elemen pertama dalam array yang pada kenyataannya akan disimpan pada awal blok memory. Indeks ke-1 akan disimpan pada memory setelah elemen pertama dan seterusnya. Anda tidak mungkin menambahakan elemen yang melebihi jumlah yang dideklarasikan, arena program hanya menyediakan ruang sejumlah elemen tersebut.

Array dalam awk dapat menyimpan nilai-nilai pada setiap element array. Nilai yang dapat disimpan dalam array awk dapat berupa numerik, karakter dan string. Array pada awk hampir sama dengan array dalam bahasa pemrograman lainnya, perbedaannya pada indeks array. Indeks array awk selalu berupa string. Multidimensional array dalam awk, menggunakan penggabungan indeks array. Array tetap berupa single-dimensional array, hanya saja cara aksesnya yang bisa multidimensional. Pengurutan array dapat dilakukan pada nilai element array dan indeks array. Fungsi asort() digunakan untuk mengurutkan nilai element array, dan asorti() digunakan untuk mengurutkan indeks array.

Percobaan 1: Pengenalan Array

Di bagian ini kita akan mempelajari dasar array satu dimensi, bekerja dengan elemen array dalam satu waktu dan membaca setiap elemen dalam array. Sebuah array yang terdiri dari empat elemen bisa terlihat seperti ini (sebegai contoh nilai elemnnya adalah 8, “foo”, “” dan 30) :

8	"foo"	""	30	Value
0	1	2	3	Index

Hanya nilainya yang disimpan, indeksnya tersirat dari urutan nilainya. Disini 8 merupakan nilai pada indeks ke-0, karena 8 muncul di posisi dimana belum ada elemen sebelumnya. Berikut ini ada program “myarray.awk” sebagai representasi bentuk array di atas.

```
BEGIN {
      myarray[3]=30
      myarray[1]="foo"
      myarray[0]=8
      myarray[2]=""

      print myarray[3]
      print myarray[1]
      print myarray[0]
      print myarray[2]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f myarray.awk
```

Awk mendukung satu-dimensi array untuk menyimpan kumpulan data nilai atau string yang berhubungan. Array dalam awk didefinisikan dengan nama yang pemberian namanya seperti nama variabel yang valid. Setiap array merupakan kumpulan dari pasangan indeks dengan nilai elemen yang berhubungan. Pemberian nilai elemen array dapat dilakukan dengan assigment. Nilai array dimasukkan dalam array pada indeks tertentu yang akan menjadi nilai dari elemen array pada indeks tersebut. Nilai elemen array diakses dengan mengakses array pada indeks yang dikehendaki.

Array dalam awk merupakan array yang bersifat asosiatif, sehingga indeks-nya tidak harus berupa bilangan bulat positif. Sembarang angka, bahkan string dapat menjadi indeks. Percobaan ini menunjukkan bahwa suatu array dapat memiliki indeks berupa string maupun angka. Namun pada dasarnya,

subscript array selalu berupa string. Modifikasi program “myarray.awk” dengan bentuk berikut:

```
BEGIN {
      myarray["dog"]="chien"
      myarray["cat"]="chat"
      myarray["one"]="un"
      myarray[1]="un"

      print myarray["dog"]
      print myarray["cat"]
      print myarray["one"]
      print myarray[1]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f myarray.awk
```

Percobaan 2: Referring to an Array Element

Untuk mengetahui apakan sebuah elemen ada dalam array pada indeks tertentu, gunakan:

```
ind in array
```

```
BEGIN {
      myarray["dog"]="chien"
      myarray["cat"]="chat"
      myarray["one"]="uns"
      myarray[1]="un"

      if("dog" in myarray)
            print "Ada indeks \"dog\" berisi " myarray["dog"]
      else
            print "Tidak ada indeks \"dog\"" if(5 in myarray)
            print "Ada indeks 5 berisi " myarray[5]
      else
            print "Tidak ada indeks 5"
```

```
}
```

Referensi array terdiri dari nama array dan ekspresi indeks. Nilai dari referensi array adalah nilai dari elemen. Misal: `myarray["dog"]` adalah referensi dari array myarray pada indeks "dog". Ekpresi `"dog" in myarray` akan menghasilkan nilai true jika indeks "dog" terdapat dalam myarray.

Percobaan 3: Assigning Array Elements

Element array dapat diberi nilai dengan assignement sebagai berikut:

`array[indeks-ekspression] = value`

```
$ awk 'BEGIN {
      myarray[1]="foo"
      myarray["cat"]=25
      myarray["un"]=""

      print myarray[1]
      print myarray["cat"]
      print myarray["un"]
}'
```

Cara pemberian nilai elemen array sama seperti pemberian nilai pada variabel. Ekspresi dari assigning array elemen terdiri dari nama array, index-expression, menggunakan operator assignement diikuti nilai yang diberikan. Pada percobaan di atas, `myarray[1]="foo"` akan memberi nilai elemen array myarray pada indeks 1 dengan nilai "foo". Nilai yang dapat diberikan pada elemen array dapat berupa angka string, maupun kosong (""). Pada percobaan di atas elemen myarray*"un"+ diberi nilai kosong ("").

Percobaan 4 : Basic Array Example

Program berikut ini mengambil sebuah daftar dari baris, yang masing-masing diawali dengan angka, dan mencetaknya dalam urutan berdasarkan

nomor baris. Saat pertama dibaca, nomor baris masih acak. Program tersebut mengurutkan baris dengan membentuk array yang indeksnya diambil dari nomor baris. Kemudian mencetak baris tersebut terurut sesuai nomor bari dengan menggunakan looping for.

File : “array_data1”

```
5 I am the Five man
2 Who are you? The new number two!
4 . . . And four on the floor
1 Who is number one?
3 I three you.
```

Program : “basic_array.awk”

```
{
     if ($1 > max)
     max = $1
     arr[$1] = $0
}
END {
     for(x = 1; x <= max; x++)
     print arr[x]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f basic_array.awk array_data1
```

Coba modifikasi program diatas, dilakukan dengan menambhakan kontrol kondisi if dengan ekpresi “`x in arr`” yang mengecek bilamana suatu index terdapat dalam array, seperti pada program berikut ini :

Program : “basic_array.awk”

```
{
     if ($1 > max)
     max = $1
```

```
     arr[$1] = $0
}
END {
     for(x = 1; x <= max; x++)
     if (x in arr)
          print arr[x]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f basic_array.awk array_data1
```

Perhatikan apa yang terjadi dan berikan analisa Anda.

Percobaan 5 : Scanning All Elements of an Array

Percobaan berikut ini menggunakan bentuk pernyataan **for**. Aturan pertama memindai record input dan mencatat kata mana yang muncul (paling sedikit sekali) dalam input, dengan menyimpannya ke dalam array **used** dengan kata tersebut sebagai indeksnya. Aturan ke-2 memindai elemen dari array **used** untuk mencari semua kata yang terdapat dalam input. Kemudian mencetak setiap kata yang panjangnya lebih dari 10 karakter dan mencetak jumlah dari kata yang panjangnya lebih dari 10 karakter tersebut.

File : “array_data2”

```
The following program uses this form of the for statement. The
first rule scans the input records and notes which words appear
(at least once) in the input, by storing a one into the array used
with the word as index. The second rule scans the elements of used
to find all the distinct words that appear in the input. It prints
each word that is more than 10 characters long and also prints the
number of such words. See Section 8.1.3 String Manipulation
Functions, page 132, for more information on the built-in function
length. The order in which elements of the array are accessed by
this statement is determined by the internal arrangement of the
array elements within awk and cannot be controlled or changed.
```

```
This can lead to problems if new elements are added to array by
statements in the loop body; it is not predictable whether the for
loop will reach them. Similarly, changing var inside the loop may
produce strange results. It is best to avoid such things.
```

Program : “scanning_array.awk”

```
# Record a 1 for each word that is used at least once
{
     for (i = 1; i <= NF; i++)
     used[$i] = 1
}

# Find number of distinct words more than 10 characters long
END {
     for (x in used) {
      if (length(x) > 10) {
            ++num_long_words
            print x
      }
     }
     print num_long_words, "words longer than 10 characters"
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f scanning_array.awk array_data2
```

Perhatikan apa yang terjadi dan berikan analisa Anda.

Percobaan 6 : Delete Elements of an Array

Untuk menghapus satu elemen tunggal dalam array menggunakan statement “delete”, seperti berikut ini :

```
delete array[index-expression]
```

Pada percobaan berikut ini: elemen array dengan indeks “dog” dihapus dengan menggunakan statement: delete myarray["dog"] Jika suatu elemen array

dihapus dengan statement **delete**, maka nilai elemen tersebut hilang dari array. Subscript elemen tersebut juga hilang sehingga tidak lagi dapat diakses. Sedangakan jika element array diberi nilai kosong: myarray[1]="" elemen array tersebut tidak hilang dari array, dan nilai nya masih dapat diakses hanya saja nilainya kosong.

Program: “delete_array1.awk”

```
{
     myarray["dog"]="chien"
     myarray["cat"]="chat"
     myarray["one"]="un"
     myarray[1]="un"

     for(x in myarray)
      print x, ":", myarray[x]

     delete myarray["dog"]
     myarray[1]=""
     print

     for(x in myarray)
      print x, ":", myarray[x]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f delete_array1.awk
```

Perhatikan apa yang terjadi dan berikan analisa Anda.

Menghapus beberapa elemen array dapat dilakukan di awk, dengan menggunakan statement **delete** dengan perulangan. Program diatas menghapus elemen array yang indeksnya berupa bilangan genap. Untuk memindai semua elemen array digunakan bentuk for khusus:

```
for (x in myarray)
```

Elemen array yang dihapus tersebut tidak lagi terdapat dalam array myarray.

Program: “delete_array2.awk”

```
BEGIN {
     for(i=1;i<=10;i++)
      myarray[i]=i;
      print "myarray:"
      for(i=1;i<=10;i++)
            print myarray[i]

     # Menghapus elemen bernilai genap
     for(x in myarray)
      if(myarray[x]%2==0)
            delete myarray[x]

     print "beberapa elemen dihapus:"
      for(i=1;i<=10;i++)
      if(i in myarray)
            print myarray[i]
}
```

Untuk menghapus semua elemen array dapat dilakukan dengan menggunakan statement **delete** tanpa subscript, sebagai berikut:

```
delete myarray
```

Maka semua elemen array akan hilang. Cara ini jauh lebih efisien daripada menggunakan statement looping yang memindai dan satu-persatu elemen array.

Program: “delete_array3.awk”

```
BEGIN {
     myarray["dog"]="chien"
     myarray["cat"]="chat"
     myarray["one"]="un"
     myarray[1]="un"

     delete myarray
     for(x in myarray)
```

```
        print myarray[x]
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f delete_array3.awk
```

Perhatikan apa yang terjadi dan berikan analisa Anda.

Percobaan 7 : Using Number to Subscript Arrays

Subscript array selalu berupa string. Maka, apabila nilai angka digunakan sebagai array subscript, akan diubah menjadi string sebelum digunakan subscript. Yang berarti bahwa nilai variabel CONVFMT dapat mempengaruhi akses elemen array. Karena variabel CONVFMT adalah variabel convertion format mempengaruhi bagaimana awk mengkonversi angka ke dalam array.

Program : “number_subcript.awk”

```
BEGIN {
      xyz = 12.153
      data[xyz] = 1
      CONVFMT = "%2.2f"

      if (xyz in data)
            printf "%s is in data\n", xyz
      else
            printf "%s is not in data\n", xyz
}
```

Kemudian jalankan perintah berikut:

```
$ awk -f number_subcript.awk
```

Hasil percobaan menunjukkan ’12.15 is not in data’. Statement pertama memberikan xyz sebuah nilai angka. Kemudian memberi nilai data[xyz]= 1. Variabel xyz bernilai 12.153 dikonversi menggunakan format konversi default

yaitu “%.6g”, sehingga menjadi “12.153” yang berarti bahwa nilai 1 masuk pada elemen data*“12.153”+. Selanjutnya variabel CONVFMT diubah menjadi “%2.2f” sehingga selanjutnya konversi dari nilai variabel xyz menjadi string menggunakan format baru tersebut. Maka pada ekspresi xyz in data, nilai 12.153 diubah ke string menjadi “12.15”, yang mana indeks tersebut tidak terdapat dalam array data.

Percobaan 8 : Using Unitialized Variables as Subscripts

Misalkan diperlukan untuk menuliskan sebuah program untuk mencetak data masukan dalam urutan terbalik. Cara yang biasa untuk melakukannya (dengan berapa data uji) akan terlihat seperti perintah dibawah ini :

```
$ echo 'line 1
     line 2
     line 3' | awk '{ l[lines] = $0; ++lines }
     END {
          for(i = lines-1; i >= 0; --i)
          print l[i]
}'
```

Tampak pada hasil percobaan, baris pertama dari input tidak tercetak pada output. Variabel **lines** yang digunakan sebagai subscript, merupakan unitialized variable. Unitialized variable memiliki nilai awal 0. Maka seharusnya awk mencetak nilai dari `l[0]`. Namun, karena subscript array selalu berupa string, maka unitialized variable dikonversi menjadi nilai string “”, bukan nol. Oleh karena itu **line 1** tersimpan dalam `l[“”]`, sehingga `l[0]` tidak ada.

```
$ echo 'line 1
     line 2
     line 3' | awk '{ l[lines++] = $0 }
     END {
          for(i = lines-1; i >= 0; --i)
          print l[i]
```

```
}'
```

Pada program tersebut, operator '++' menyebabkan nilai **lines** menjadi numerik, yang berarti bahwa nilai awalnya adalah nol, sehingga ketika dipakai sebagai subscript, nilainya dikonversi menjadi "0". Sehingga **line 1** tersimpan pada `l[0]`.

Percobaan 9 : Multidimensional Array

Multidimensional array adalah array yang elemennya diidentifikasi dengan serangkaian indeks. Sehingga multidimensional array membutuhkan dua indeks. Dalam program di atas array dua-dimensi denyatakan dengan :

```
myarray[2,5]
```

Multidimensional array dibentuk dengan penggabungan dua indeks tersebut. Indeks dikonversi menjadi string dan digabung dengan menambahkan pemisah diantaranya. String yang telah dikombinasikan tersebut menjadi satu indeks yang me-referensi satu elemen array.

Cobalah perintah dibawah ini :

```
$ awk 'BEGIN {
     myarray[2,5]="Smithy"
     for(x in myarray)
      print myarray[x]
     print myarray["2"SUBSEP"5"]
}'
```

Subscript dari multidimensional array dalam awk merupakan gabungan dari dua indeks yang di antaranya dipisahkan dengan separator. Separator tersebut didefinisikan dalam built-in variable SUBSEP.

Cobalah perintah dibawah ini :

```
$ awk 'BEGIN {
```

```
     SUBSEP="@"
     myarray[2,5]="Smithy"
     for(x in myarray)
          print myarray[x]
     print myarray["2@5"]
}'
```

Pada program diatas, nilai SUBSEB didefinisikan dengan nilai karakter “@”. Maka subscript myarray[2,5], Angka 2 dan 5 akan diubah menjadi string dan digabungkan dengan memberi pemisah di antaranya “@”, sehingga manjadi “2@5”. Tampak ekspresi myarray[“2@5”] mereferensi nilai dari elemen myarray[2,5].

Percobaan 10 : Scanning Mutildimensional Arrays

Bagaimana memindai multidimensional array sama dengan memindai array biasa. Pada dasarnya, tidak ada array multidimensi, hanya saja cara aksesnya yang multi-dimensi. Untuk mendapatkan kembali subscript array multi-dimensi dapat digunakan kombinasi dari statement **for** dan fungsi **split**. Cobalah program berikut ini :

```
$ awk 'BEGIN {
      myarray[1, "foo"]
      for(combined in myarray)
            split(combined, separate, SUBSEP)
      print separate[1]
      print separate[2]
}'
```

Program di atas memberi nilai variabel **combined** untuk setiap gabungan indeks dalam array, dan memecahnya kedalam indeks tunggal dengan memisahnya berdasarkan nilai dari SUBSEP yang nilai defaultnya adalah karakter dengan kode 034. Indeks tunggal tersebut disimpan dalam arrat **separate**. Pada program di atas ini jika suatu nilai disimpan pada array:

```
myarray[1, "foo"]
```

maka elemen dengan indeks “1\034foo”. Kemudian indeks tersebut dipecah menjadi “1” dan “foo” yang masing-masing disimpan dalam separate[0] dan separate[1].

Percobaan 11: Sorting Array Values and Indices with gawk

Percobaan berikut ini menggunakan fungsi **asort** untuk mengurutkan array myarray. Dengan pemanggilan fungsi asort:

```
asort(myarray)
```

array myarray di-indeks kan dari 1 sampai n, dengan n adalah jumlah elemen dalam myarray. Fungsi tersebut mengurutkan dengan membandingkan nilai elemen dengan menggunakan metode pembandingan yang biasa digunakan. Maka hasil array yang telah diurutkan dapat dicetak dengan menggunakan statement for mulai dari myarray index 1 sampai n.

Program: “sorting_array1.awk”

```
{ myarray[++line] = $1 }
END {
      n = asort(myarray)
      for(i=1; i<=n; i++)
      print myarray[i]
}
```

Kemudian jalankan perintah berikut:

```
$ awk –f sorting_array1.awk BBS-list
```

Coba modifikasi program diatas seperti program berikut ini:

Program: “sorting_array2.awk”

```
{ myarray[++line] = $1 }
END {
      n = asort(myarray, dest)
      for(x in myarray)
```

```
            print myarray[x]
            print ""
            for(i=1; i<=n; i++)
      print dest[i]
}
```

Kemudian jalankan perintah berikut:

```
$ awk –f sorting_array2.awk BBS-list
```

Program di atas menggunakan fungsi asort dengan parameter tambahan dest: **asort(myarray, dest).** Pemanggilan fungsi asort pada percobaan sebelumnya mengakibatkan indeks dari array awal hilang dan diganti menjadi indeks angka dari 1 sampai n, dengan n adalah jumlag data. Pada percobaan ini, pemanggilan fungsi asort menambahkan parameter ke-dua yaitu dest, di mana dest adalah array yang digunakan sebagai tampungan dari array myarray yang telah diurutkan. Dalam kasus ini, awk akan mengcopy myarray ke dalam array dest dan mengurutkan array dest, dan mengubah indeksnya. Bagaimanapun juga, array myarray tidak mengalami perubahan. Tampak pada hasil percobaa, indeks myarray masih tetap, dan dest menyimpan data yang telah terurut.

Coba modifikasi program diatas seperti program berikut ini:

Program: “sorting_array3.awk”

```
{ myarray[++line] = $1 }
END {
      n = asorti(myarray, dest)
      for(i=1; i<=n; i++)
      printf "%10s %10s\n", dest[i], myarray[dest[i]]
}
```

Kemudian jalankan perintah berikut:

```
$ awk –f sorting_array3.awk BBS-list
```

Program di atas menunjukkan pengurutan indeks array menggunakan fungsi **asorti**: `asorti(myarray, dest)`. Fungsi **asorti** adalah fungsi untuk mengurutkan indeks array, bukan elemen array. Fungsi asorti pada program di atas menerima dua parameter, yaitu myarray dan dest. dest adalah sebagai array yang menyimpan urutan indeks dari myarray, dengan kata lain, indeks dari myarray yang telah diurutkan menjadi elemen dari dest. Pada hasil percobaan, kolom pertama adalah output dari dest, yang berisi indeks myarray yang telah terurut. Kolom kedua adalah output dari pencetakan elemen myarray dengan urutan indeks berdasarkan nilai dari elemen dest. Indeks array adalah berupa string, maka pengurutannya berdasarkan pembandingan string. Tampak urutan indeks yang dihasilkan adalah “1”, “10”, “11”,”2”, …, bukan 1,2,3,4, …, 9, 10, 11.