

## A. TUJUAN

1. Menggunakan perintah awk untuk proses control program dengan perintah pengkondisian dan perulangan.
2. Menggunakan operator relasi, pembandingan untuk numeric maupun string serta operator logika dalam program awk
3. Menggunakan array, fungsi dan variabel built-in dalam program awk.

## B. DASAR TEORI

### Statement Control

Script awk telah dibahas pada praktikum sebelumnya dengan beberapa kelebihanannya untuk manipulasi file teks. Seperti bahasa pemrograman lainnya awk juga memiliki statement control diantaranya untuk pengkondisian menggunakan perintah *if*, perulangan dengan perintah *while* dan *for*. Selain itu pada program awk juga terdapat operator, operator yang digunakan untuk berbagai keperluan pemrograman yaitu untuk operasional relasi menggunakan simbol  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $==$ , dan  $!=$ . Serta operasional pembandingan baik untuk data numerik maupun data string dengan regular ekspresi:  $\sim$  dan  $!\sim$ . Operator  $\sim$  untuk sukses jika data string sesuai, operator  $!\sim$  untuk sukses jika data string tidak sesuai. Operator relasi diatas juga dapat dikombinasikan dengan menggunakan operator logika AND ( $\&\&$ ) dan OR ( $\|\|$ ).

Selain statement diatas, perintah *break* dan *continue* pada proses loop di awk juga bisa dipergunakan. Ketika sampai pada perintah break, proses loop menggunakan perintah for atau while segera dihentikan selanjutnya melakukan

control kembali untuk mengikuti statement loop berikutnya. Ketika ketemu perintah continue, segera memulai iterasi dari awk yaitu mengikuti alur control dari perintah next. Ketika next ditemukan, maka baris input dibaca dan dieksekusi setelah itu kembali lagi melakukan aksi yang pertama/awal selama inputnya masih cocok (tidak ada perubahan dengan sebelumnya).

## **Array**

Seperti pemrograman umumnya, awk juga memiliki array. Array adalah sebuah daftar/list yang mempunyai nama dan elemen individu yang dapat diakses dengan menggunakan indeks. Jika x adalah sebuah array dengan tiga elemen, maka indeks elemennya adalah x[0], x[1], dan x[2]. Array dalam awk adalah fleksibel sebab indeks array pada awk dapat berupa beberapa tipe data, termasuk data string. Dimisalkan array awk dengan nama "scores" mempunyai elemen-elemen string seperti berikut: scores["mary"], scores["rich"], scores["redteam"] dan scores["champ"].

Awk juga mempunyai bentuk khusus (special) untuk statement for yang digunakan untuk menguji semua elemen pada sebuah array. Untuk lebih jelasnya diperlihatkan pada sintak for dibawah ini.

Sintak:        ***for ( var in arrayname )***

Var digunakan untuk memberi nilai indeks didalam nama array. Contoh implementasi dari sintak diatas adalah sebagai berikut:

***for ( i in lines )***

## **Operator, Fungsi, dan Variabel built-in**

### **• Operator**

Seperti dijelaskan sebelumnya, bahwa awk mempunyai beberapa operator untuk operasi matematik, pembandingan dan logika yaitu:

++ --        Operator otomatis increment dan decrement x=x+1 atau x--

\* / %            Operator perkalian, pembagian dan sisa hasil bagi (modulus)  
+ -                Operator penjumlahan dan pengurangan  
> >= < <= == != ~ !~    Operator pembandingan untuk string atau numeric  
&& ||             Operator logika AND dan OR  
!                  Operator untuk komplemen

Format penulisan operator diperlihatkan pada tabel 12.1 dibawah ini.

Tabel 12.1. Penulisan Operator pada awk

| Assignment operator | Use for                             | Example           | Equivalent to           |
|---------------------|-------------------------------------|-------------------|-------------------------|
| +=                  | Assign the result of addition       | a += 10<br>d += c | a = a + 10<br>a = a + c |
| -=                  | Assign the result of subtraction    | a -= 10<br>d -= c | a = a - 10<br>a = a - c |
| *=                  | Assign the result of multiplication | a *= 10<br>d *= c | a = a * 10<br>a = a * c |
| %=                  | Assign the result of modulo         | a %= 10<br>d %= c | a = a % 10<br>a = a % c |

## • Variabel

Awk memiliki variabel yang sudah built-in. Beberapa variabel yang digunakan dalam awk ditunjukkan pada tabel dibawah ini.

Tabel 12.2. Variabel built-in pada awk

| awk Variable | Meaning                                                |
|--------------|--------------------------------------------------------|
| FILENAME     | Name of current input file                             |
| RS           | Input record separator character (Default is new line) |
| OFS          | Output field separator string (Blank is default)       |
| ORS          | Output record separator string (Default is new line)   |
| NF           | Number of input record                                 |
| NR           | Number of fields in input record                       |
| OFMT         | Output format of number                                |
| FS           | Field separator character (Blank & tab is default)     |

Contoh penggunaan variabel built-in pada awk:

```
FILENAME != prev {
    NR = 1
    Prev= FILENAME
}
```

- **Fungsi**

Awk berisi fungsi-fungsi untuk operasi aritmatika dan pengolahan string teks. Deskripsi argumen yang digunakan dalam fungsi awk yaitu menggunakan karakter *s* dan *n*. Dimana karakter *s* untuk argumen berupa data string sedangkan karakter *n* untuk argumen berupa data numerik. Adapun bentuk fungsi dalam awk adalah sebagai berikut:

*exp(n)* Untuk menghitung eksponensial dari suatu argumen, dan menggunakan huruf *e* dalam penulisannya.

*getline* untuk membaca baris berikutnya dari masukkan

*index(s1,s2)* fungsi indeks untuk mencari string s1, dan s2.

*Int(n)* fungsi integer dari sebuah argument

*Length(s)* fungsi untuk mengetahui panjang string s

*Log(n)* fungsi untuk logaritma natural dari suatu argument

*Split(s,array,sep)* fungsi pembagi string s ke dalam field, meletakkan elemen array dengan indeks 1,2 dan seterusnya. Parameter sep memberi nilai input field separator

## C. PROSEDUR

1. Masuk ke sistem operasi Linux.
2. Login sebagai stD3XXYYY.
3. Gunakan statement control awk pada pemrograman shell.
4. Catatlah hasil dari setiap program pada prosedur no.3 sebagai laporan sementara.

## D. PERCOBAAN

1. Menggunakan statement control dengan perintah pengkondisian (if).

```
$ vi tri.awk
awk '
{ if ($4 != "2")
    print $0
  else {
    print $1, $2+12, $3, "0"
    print $1+10, $2-6, $3, "1"
    print $1-10, $2-6, $3, "1"
  }
}
```

```

        print $1, $2+12, $3, "1"
        print $1, $2, $3, "0" }
}' $*

$ chmod u+x tri.awk
$ cat > sdata
200 200 18 0
200 -200 18 1
-100 -200 18 1
-100 200 18 1
200 200 18 1
50 0 18 2
^d

$ ./tri.awk sdata

```

## 2. Menggunakan variabel NF pada program awk untuk operasi aritmatika.

```

$ vi aveline

awk '
{ if (NF > 0) {
    sum = 0
    n = 1
    while (n <= NF) {
        sum = sum + $n
        n = n + 1
    }
    print sum/NF
}
else
    print
}' $*

$ chmod u+x aveline
$ cat > hits
11 3 44 10
13 -7
20 30

18 -18
^d

$ ./aveline hits

```

## 3. Menggunakan array dan variable built-in NR pada program awk

```

$ vi rmdups

awk '
{ found = 0
  for ( i=1; i<=NR; i++ )
    if (lines[i] == $0) {
        found = 1
        break
    }
}

```

```

        if ( found != 1 ) {
            lines[NR] = $0
            print
        }
    }' $*

$ chmod u+x rmdups
$ $ cat > dups

Now is
the
time for men and women of
the
Principle to unite.
^d

$ ./rmdups dups

```

4. Menggunakan statemen for dan array pada program awk untuk mengetahui user yang login.

```

$ vi nlogins
who | awk '{
    logins[$1] = logins[$1] + 1
}
END {
    for (i in logins)
        print logins[i], i
}'

$ chmod u+x nlogins
$ ./nlogins

```

5. Menggunakan array dua dimensi pada program awk untuk membalik posisi data dari sebuah file.

```

$ vi rev.awk
awk '{
    for(i=1; i<=NF; i++)
        x[NR "." i] = $i
    if (NFIELDS < NF)
        NFIELDS = NF
}
END {
    for (i=1; i<=NFIELDS; i++) {
        line = ""
        for(j=1; j<=NR; j++)
            line = line " " x[j "." i]
        print line
    }
}'

$ chmod u+x rev.awk

```

```
$ cat > data
92 18 17
28 09 13
47 43 18
^d

$ ./rev.awk < data
```

6. Menggunakan fungsi eksponensial pada program awk.

```
$ vi exp.awk

awk '
BEGIN {
    Print "i e**i"
    for(i=0;i<10;i++)
        print i, exp(i)
    exit
}'

$ chmod u+x exp.awk
$ ./exp.awk
```

7. Menggunakan fungsi index pada program awk

```
$ vi index.awk

awk 'BEGIN {
    print "substr", "sub", index("substr", "sub")
    print "sub", "substr", index("sub", "substr")
    print "substr", "str", index("substr", "str")
    exit
}'

$ chmod u+x index.awk
$ ./index.awk
```

8. Menggunakan fungsi split pada program awk

```
$ vi split.awk

awk 'BEGIN {
    v = "This:is:a split demo"
    n = split(v,word,":")
    for(i=1; i<=n; i++)
        print i, word[i]
    exit
}'

$ chmod u+x split.awk
$ ./split.awk
```

9. Menggunakan variabel built-in NF untuk menampilkan data string.

```
$ cat > for_loop

BEGIN {
    printf "To test for loop\n"
```

```

    printf "Press CTRL + C to stop\n"
}
{
    for(i=0;i<NF;i++)
    {
        printf "Welcome %s, %d times.\n" ,ENVIRON["USER"], i
    }
}
^d

```

```

$ awk -f for_loop
To test for loop
Press CTRL + C to Stop
ketik "Welcome to Linux!" dan catat hasilnya

```

10. Menggunakan perintah perulangan while pada program awk.

```

cat > while_loop
{
    no = $1
    remn = 0
    while ( no > 1 )
    {
        remn = no % 10
        no /= 10
        printf "%d" ,remn
    }
    printf "\nNext number please (CTRL+D to stop):";
}
^d

```

```

$ awk -f while_loop
5 (enter) catat hasilnya
65 (enter) catat hasilnya

```

11. Menampilkan informasi system menggunakan awk

```

$ cat > tryrmsys
{
    dcmd = "rm " $1
    if ( system(dcmd) != 0 )
        printf "rm command not successful\n"
    else
        printf "rm command is successful and %s file
is removed \n", $1
}
^d

```

```

$ awk -f tryrmsys
ketik nama file yang akan dihapus misal:coba.sh
untuk stop tekan ^c

```

12. Menggunakan fungsi getline untuk membaca input standard (keyboard).

```

$ cat > getlin

```

```

BEGIN {
    printf "Your name please:"
    getline na < "-"
    printf "%s your age please:",na
    getline age < "-"
    print "Hello " na, ", next year you will be " age + 1
}
^d

$ awk -f getlin

```

## E. Latihan

1. Buatlah program shell menggunakan awk jika hasil running programnya adalah sebagai berikut.

```

n nfact
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
10 3628800

```

2. Dengan menggunakan fungsi getline pada awk, buatlah program untuk menampilkan informasi tanggal, bulan, waktu (jam:mnt:dtk) dan tahun sekarang.
3. Buatlah program awk untuk mencopy file dengan ketentuan sebagai berikut:

- Jika program dijalankan dengan perintah

```

$ awk -f lat3.awk
nama_file_sumber nama_file_tujuan (coba.awk latih.awk)

```

Hasilnya:

```

Coping nama_file_sumber to nama_file_tujuan
(Coping coba.awk to latih.awk)

```

- Jika nama file sumber tidak ada pada direktori maka ditampilkan komentar  
*nama\_file\_sumber tidak ada dan tidak dapat ditampilkan*  
*(coba.awk tidak ada dan tidak dapat ditampilkan)*
- Jika nama file sumber yang dicopy sudah pernah disalin/copy akan  
 ditampilkan komentar

```
nama_file_sumber overwrite (y/N)?  
(coba.awk overwrite (y/N)?)
```

4. Buatlah program awk dengan nama *col.awk* dan file berisi data yang akan diakses oleh program awk dengan nama *data* . Adapun file *data* isinya adalah sebagai berikut:

```
1 2 3  
4 5 6  
7 8 9  
10 11 12
```

program awk diatas dapat dijalankan dengan perintah-perintah dan hasil eksekusi seperti dibawah ini

```
$ ./col.awk data  
$ ls data.?
```

Hasilnya :

```
data.1  data.2  data.3
```

```
$ cat data.1
```

Hasilnya :

```
1  
4  
7  
10
```

```
$ cat data.2
```

Hasilnya :

```
2  
5  
8  
11
```

```
$ cat data.3
```

Hasilnya :

```
3  
6  
9  
12
```