Praktikum 1
Berkenalan dengan “awk”

Tujuan Pembelajaran

Mahasiswa dapat mengenal, memahami dan menggunakan bahasa pemrograman awk sebagai *text-processing language*.

Dasar Teori

Awk adalah sebuah bahasa pemrograman seperti pada shell atau C yang memiliki karakteristik yaitu sebagai tool yang cocok untuk filter dan manipulasi file teks. Awk adalah penggabungan dari nama lengkap sang author, yaitu : Alfred V. Aho, Peter J. Weinberger dan Brian W. Kernighan. Awk atau juga disebut Gawk (GNU awk), yaitu bahasa pemrograman umum dan utility standard POSIX 1003.2. (Portable Operating System Interface for UNIX). Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai. Di Linux, Awk dan shell biasa dipakai untuk aplikasi yang berbeda. Awk sangat baik untuk manipulasi file teks, sedangkan shell sangat baik untuk pelaksana perintah UNIX.

Secara umum bahasa pemrograman awk dapat digunakan untuk :
- Mengelola database sederhana.
- Membuat report.
- Memvalidasi data.
- Menghasilkan index and menampilkan dokumen.
- Membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya.

Dengan kata lain awk menyediakan fasilitas yang dapat memudahkan untuk:
- Memecah bagian data untuk proses selanjutnya.
- Mengurutkan data.
- Menampilkankomunikasi jaringan yang sederhana.

Fungsi dasar awk adalah untuk mencari pola pada file per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut. Awk tetap memproses baris input sedemikian hingga mencapai akhir baris input.

Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat “*data-driven”*; yang mana Anda diminta untuk mendeskripsikan data yang dikehendaki untuk bekerja dan kemudian apa yang

akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat "*procedural*"; Anda harus mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca.

Saat Anda menjalankan program awk, Anda menentukan program awk yang memerintahkan awk melakukan suatu pekerjaan. Program berisi serangkaian instruksi (boleh juga berisi definisi fungsi). Setiap instruksi menentukan sebuah pola untuk dicari dan sebuah aksi dijalankan saat pola tersebut ditemukan.

Secara sintaksis, sebuah instruksi memuat pola yang diikuti oleh aksi. Aksi tersebut diapit oleh kurung kurawal untuk memisahkannya dari pola. Akhir baris biasanya memisah instruksi. Untuk itu, program awk terlihat seperti ini:

```
pattern { action }

pattern { action }

…
```

Percobaan 1: Menjalankan program awk yang pendek

Terdapat beberapa cara untuk menjalankan program awk. Jika programnya pendek, dapat dengan mudah dimasukkan dalam perintah yang menjalankan awk, seperti ini:

```
awk 'program' input-file1 input-file2 ...
```

Untuk menjalankan awk tanpa file input dapat dilakukan dengan tipe perintah sebagai berikut:

```
awk 'program'
```

Berikut ini contoh program untuk menampilkan string.

```
$ awk "BEGIN { print \"Konnichiwa\" }"
= Konnichiwa
```

Pada bagian action sebelum tanda kutip (") diperlukan tanda escape (\) karena dalam aturan tanda kutip shell tanda kutip yang berada didalam quote maka harus diberi karakter escape agar dikenali sebagai tanda kutip (double quote). Jika tidak maka tanda kutip tersebut dapat dikenali debagai akhir program yang kemungkinan dapat menyebabkan error.

Program berikut ini menunjukkan fungsi awk seperti pada fungsi “cat” pada shell-programming.

```
$ awk '{ print }'
Now is the time for all good men
= Now is the time for all good men
to come to the aid of their country.
= to come to the aid of their country.
Four score and seven years ago, ...
= Four score and seven years ago, ...
What, me worry?
= What, me worry?
Ctrl+d
```

Perintah “print” pada program awk di atas mensimulasikan bagaimana utilitas “cat”. Perintah tersebut akan menampilkan input dari keyboard segera setelah ditekan tombol Enter dan keluar ketika di tekan tombol “Ctrl+D”.

Percobaan 2: Menjalankan program awk yang panjang

Untuk menjalankan program awk yang panjang, maka akan lebih baik untuk menyimpan program tersebut pada sebuah file, kemudian menjalankannya dengan perintah sebagai berikut:

```
awk –f program-file input-file1 input-file2 ...
```

Berikut ini adalah contoh source-file untuk program, ketikkan contoh program dibawah ini dan simpan pada direktori home dengan nama “program”.

```
BEGIN { print \"I wonder this is a long program\" }
```

Program dari source-file tersebut dapat dijalankan dengan perintah:

```
$ awk –f program
= I wonder this is a long program
$ awk BEGIN { print \"I wonder this is a long program\" }
= I wonder this is a long program
```

Pada bagian argumen source-file tidak perlu diberi tanda quote karena bukan string yang akan diproses sebagai program melainkan nama dari source file. Untuk itu sebelum argumen source-file diperlukan argumen ‘`–f`’ karena kebanyakan nama file tidak mengandung karakter khusus shell. Tampak juga bahwa di dalam source file tanda kutip (“) pada bagian action tidak perlu diberi

tanda escape (/) karena program dalam source-file tidak diapit tanda kutip (quote). Hal ini dapat mempermudah pembuatan program awk yang cukup kompleks.

Percobaan 3: Membuat program awk yang "executable"

Program awk dapat diubah menjadi executable dengan menambahkan:

```
#! /usr/bin/awk -f
```

Baris program tersebut diletakkan paling atas source-file program awk. Berikut ini adalah source-file program awk:

```
#! /usr/bin/awk -f

BEGIN { print "I wonder this is a long program" }
```

Source-file program awk tersebut dijadikan executable dengan menggunakankan utilitas ***chmod***. Dengan perintah itu, ketika file tersebut dipanggil maka bagian ***awk -f 'program'*** dari source-file akan sama seperti menjalankan ***awk -f 'program'*** pada terminal.

```
$ chmod +x program
$ ./program
= I wonder this is a long program
```

Percobaan 4: Menulis komentar pada program awk

Comment adalah teks yang disertakan pada program untuk memudahkan pembaca. Comment adalah bagian dari program yang tidak dijalankan. Comment dapat menjelaskan apa yang dilakukan program bagaimana program tersebut bekerja. Pada awk comment dimulai dengan karakter pagar (#) sampai akhir baris.

```
#! /usr/bin/awk -f

# This program prints a nice friendly message.
# It helps keep novice user from miss understanding.

BEGIN { print "This program contains comments" }
```

Kemudian jalankan peritnah ini di terminal.

```
$ chmod +x program
```

```
$ ./program
= This program contains comments
```

Hasilnya akan tampak bagian program setelah tanda pagar (#) hingga akhir baris tidak dijalankan. Atau silahkan coba di terminal perintah berikut ini.

```
$ awk '{ print "Hello" } # lets be cute'
= Hello
```

Percobaan 5: Tentang "Shell-Quoting"

Shell-Quoting adalah mekanisme dalam pemrograman shell untuk melindungi meta-karakter dari intrepretasi sebagai sebuah simbol agar tetap diintrepretasi sebagai karakter biasa.
Terdapat beberapa aturan terkait pemakaian tanda kutip (quote) dalam program awk, sehingga program tersebut dapat di-compile dengan benar. Kesalahan pemakaian terkait letak, escaping, dll dapat mengakibatkan error pada program. Aturan berikut ini digunakan hanya pada *POSIX-compliant*, *Bourne-style shells* (contohnya: Bash, GNU Bourne-Again Shell).

- Karakter kutip (*quoted items*) dapat digabungkan dengan katakter yang bukan karakter kutip (*nonquoted item*) termasuk juga dengan karakter kutip lainnya.
- Mendahului setiap karakter tunggal dengan garis miring terbalik atau *backslash* ('\') untuk mengutip karakter. Shell akan menghilangkan backslash dan melewatkan karakter yang dikutip (*quoted character*) yang ada pada perintah.
- Tanda kutip tunggal atau *single quote* (') melindungi semua karakter yang ada diantara pembukaan dan penutupan kutipan. Shell tidak melakukan interpretasi pada teks kutipan, ia akan melewatkan verbatim (catatan lengkap / kata per kata) yang ada pada perintah. Dengan demikian, hal ini tidak memungkinkan untuk menggabungkan kutip tunggal (*single quote*) ke dalam text yang dikutip (*single-quoted text*).
- Tanda kutip ganda atau *double quote* (") melindungi semua karakter diantara pembukaan dan penutupan kutipan. Shell melakukan hal ini setidaknya ada variabel dan substitusi perintah pada teks kutipan.
- Karakter tertentu yang ada di dalam tanda kutip ganda akan dikeluarkan (*escape*) dari teks oleh shell. Beberapa diantaranya karakter: dolar ($), petik tunggal ('), backslash (\), dan petik ganda ("). Semuanya harus didahului dengan backslash (\) diantara kutipan ganda (") jika ingin dilewatkan pada program. Contohnya:

```
$ awk "BEGIN { print \"Don't Panic\" }"
= Don't Panic!
```

- String **null** akan dihilangkan ketika awk mengetahui akan terjadinya *non-null command-line argument*, sementara secara eksplisit objek non-null disimpan. Misalnya, untuk menentukan bahwa ***Field Separator (FS)*** harus di set sebagai null-string.

```
awk –F "" ’program’ files # correct

awk –F"" ’program’ files # wrong!
```

Untuk pemakaian quote sendiri terdapat berbagai alternatif yang dapat dipilih yang mana beberapa alternatif tersebut bisa menghasilkan output yang sama.

Cara pertama dengan menggabungkan tanda kutip tunggal dan ganda. Misalnya dengan menggabungkan tiga quoted string, yang pertama dan yang ketiga adalah kutip tunggal (‘), yang kedua kutip ganda (“). Contohnya:

```
$ awk ’BEGIN { print "Here is a single quote <’"’"’>"
}’
= Here is a single quote <’>
```

atau dengan menyederhanakan bentuk diatas. Contohnya:

```
$ awk ’BEGIN { print "Here is a single quote <’\’’>" }’
= Here is a single quote <’>
```

Cara kedua dengan menggunkan ***awk-level double quotes*** yaitu kutipan ganda (”) yang digunakan untuk keluar dari masalah penggabungan (***escape from embedded***).

```
$ awk "BEGIN { print \"Here is a single quote <’>\" }"
= Here is a single quote <’>
```

Cara ketiga dengan menggunakan ***octal escape sequence equivalents,*** *Contohnya:*

```
$ awk ’BEGIN { print "Here is a single quote <\47>" }’
= Here is a single quote <’>
$ awk ’BEGIN { print "Here is a double quote <\42>" }’
= Here is a double quote <">
```

Cara keempat dengan menggunakan ***command-line variable assignment***. *Contohnya:*

```
$ awk –v sq="’" ’BEGIN { print "Here is a single quote
<" sq ">" }’
```