Praktikum 10

Internationalization & Advance Feature AWK

Tujuan Pembelajaran

Mahasiswa dapat memahami dan menggunakan mode *internationalization* dan *advance feature* yang ada dalam bahasa pemrograman awk.

Dasar Teori

Internationalization dan **Localization** merupakan salah satu cara untuk menghandle perbedaan bahasa dari berbagai bahasa pemrograman. Internationalization adalah penulisan (modifikasi) program dalam satu waktu, sehingga mampu menggunakan berbagai bahasa tanpa memerlukan perubahan source code lagi. Sedangkan Localization adalah menyediakan data untuk program internationalization sehingga dapat bekerja pada bahasa tertentu. Setiap file harus berisi informasi TEXTDOMAIN yang sama agar keseluruhannya dapat berjalan. Syarat utamanya adalah pemberian nama TEXTDOMAIN yang sama pada program yang dibuat.

Percobaan 1: Internationalization with gawk

Gawk menyediakan variable TEXTDOMAIN untuk internationalization. Variabel ini menandakan text domain aplikasi. _"string", string yang diawali dengan underscore akan diterjemahkan pada saat runtime.

```
$awk 'BEGIN {
      TEXTDOMAIN = "guide"
      print _"hello, world"
      x = _"you goofed"
      printf(_"Number of users is %d\n", nusers)
}'
```

Percobaan berikut ini menunjukkan bahwa pembuatan string dinamis masih bisa diterjemahkan menggunakan “dcgettext”.

```
$awk ‘BEGIN {
      TEXTDOMAIN = “guide”
      message = nusers “user logged in”
      message = dcgettext(message, “adminprog”)
      print message
}’
```

Spesifikasi posisi terdiri dari perhitungan integer (bilangan bulat), yang menandakan argumen mana yang akan digunakan dan ‘$’. Pada percobaan ini, ‘string’ adalah argumen pertama dan ‘length(string)’ adalah argumen ke-dua.

```
$awk ‘BEGIN {
      string = “Don’t Panic”
      printf “%2$d characters line in \”%1$s\”\n”,
            string, length(string)
}’
```

Percobaan berikut ini menunjukkan bahwa spersifikasi posisi dapat digunakan dengan lebar space yang dinamis.

```
$awk ‘BEGIN {
      printf(“%*.*s\n”, 10, 20, “hello”)
      printf(“%3$*2$.*1$s\n”, 20, 10, “hello”)
}’
```

Percobaan ini menunjukkan cara meng-internationalize dan localize program awk sederhana.

```
$awk ‘BEGIN {
      TEXTDOMAIN(“.”)
      print _”Don’t panic”
```

```
      print _”The Answer Is”, 42
      print ”Pardon me, Zaphod who?”}’ > guide.po
$cp guide.po guide-mellow.po
$mkdir en_US enUS/LC_MESSAGE
```

Utilitas pesan “message” digunakan untuk melakukan konversi dari file ‘.po’ yang dapat dibaca oleh manusia ke file ‘.mo’ yang dapat dibaca oleh mesin. Default-nya, msgfmt akan membuat file dengan nama ‘message’. File tersebut harus diberi nama baru (rename) dan diletakkan pada direktori adar gawk dapat membaca file tersbeut.

```
$ msgfmt guide-mellow.po
$ mv messages.mo en_US/LC_MESSAGES/guide.mo
```

Percobaan 2: Advanced Features of gawk

Program gawk pada percobaan ini menggunakan option ‘—non-decimal-data’, sehingga diperoleh output desimal dari input yang non-desimal.

```
$echo 0123 123 0x123 | gawk –non-decimal-data ‘{
      printf “%d, %d, %d\n”, $1, $2, $3
}’
```

Program gawk pada percobaan ini tidak menggunakan option ‘—non-decimal-data’, sehingga ouput yang dihasilkan dalam format yang sama dengan input. Tanpa option tersebut gawk akan menganggap data input sebagai data numerik.

```
$ echo 0123 123 0x123 | gawk ‘{
      print $1, $2, $3
}’
```

Utilitas print menganggap data input sebagai data string, walaupun sebenarnya dapat dianggap sebagai data numerik, namun dalam hal ini dianggap sebagai data string.

```
$ echo 0123 123 0x123 | gawk '{
      print "%d, %d, %d\n", $1, $2, $3
      print $1 + 0, $2 + 0, $3 + 0
}'
```

Program pada percobaan ini menggunakan komunikasi dua arah. Bentuk komunikasi ini sangat berguna untuk dapat mengirim data ke program yang terpisah untuk memproses dan kemudian membaca hasilnya. Program ini kurang efektif karena masih berjalan dalam sebuah direktori yang tidak dapat di-share dengan pengguna lain. Operator '| &' merupakan operator yang digunakan pada saat pertama kali operasi I/O dijalankan. gawk menciptakan pipe untuk proses child yang menjalankan program lain.

```
# write the data for processing
tempfile = ("mydata." PROCINFO["pid"])
while (not done with data)
print data | ("subprogram > " tempfile)
close("subprogram > " tempfile)

# read the results, remove tempfile when done
while ((getline newdata < tempfile) > 0)
process newdata appropriately
close(tempfile)
system("rm " tempfile)

do {
      print data |& "subprogram"
      "subprogram" |& getline results
} while (data left to process)
close("subprogram")
```

Program pada percobaan ini membaca tanggal dan waktu dari TCP ‘daytime’ sistem lokal server. Program ini kemudian mencetak hasil dan mengakhiri koneksi (close). Operator ‘| &’ berfungsi untuk komunikasi dua arah. Mekanisme portal sistem operasi kemudian mengelola proses yang berkaitan dengan portal komunikasi sesuai dengan proses portal.

```
# gawk profile, created Sun Aug 13 00:00:15 2000
# BEGIN block(s)
BEGIN {
      print "First BEGIN rule"
      print "Second BEGIN rule"
}
# Rule(s)
/foo/ { # 2
      print "matched /foo/, gosh"
      for (i = 1; i <= 3; i++) {
            sing()
      }
}
{ if (/foo/) { # 2
      print "if is true"
      }
      else {
      print "else is true"
      }
}
# END block(s)
END {
      print "First END rule"
      print "Second END rule"
}

# Functions, listed alphabetically
function sing(dummy) {
      print "I gotta be me!"
}
```

Daftar Pustaka

[1] Arnold D. Robbins, *GAWK: Effective AWK Programming, A User's Guide for GNU Awk Edition 4.1*, 2014.

[2] Dale Dougherty & Arnold Robbins, *Sed and AWK*, 2000.

[3] T.P. Love, *Shell Scripts and Awk*, 2009.