آزمایشگاه سیستم عامل

دستورکار ۸: ابزار awk

ابزار awk معمولاً برای گزارش گیری از فایل‌های مختلف مورد استفاده قرار می‌گیرد. این ابزار در Bell Labs در دهه ی ۱۹۷۰ میلادی توسعه یافت. ابزار awk قابلیت‌های بیشماری دارد که از جمله ی مهم‌ترین آن‌ها می‌توان به موارد زیر اشاره کرد:

- یک ابزار قدرتمند و یک زبان برنامه نویسی تفسیری است
- برای پردازش متن، تهیه ی گزارش و ارائه‌ی گزارش‌های اطلاعاتی به کار گرفته می‌شود.

awk می‌تواند فایل‌های داده متنی وstream ها را پردازش کند. داده‌های ورودی به رکوردها و فیلدها تقسیم می‌شوند. awk در هر بار روی یک رکورد عمل می‌کند تا زمانی که ورودی به پایان برسد. رکوردها توسط کاراکتری به نام جداکننده رکورد (record seprator) از هم جدا می‌شوند. جداکننده رکورد پیش فرض، کاراکتر خط جدید است؛ به این معنی که هر خط در داده‌های متنی یک رکورد است. یک جداکننده رکورد جدید را می‌توان با استفاده از متغیر RS تنظیم کرد.

رکوردها شامل فیلدهایی هستند که با جداکننده فیلد از هم جدا می‌شوند. به طور پیش فرض، فیلدها با یک فضای خالی شامل یک یا چندtab ، فاصله و کاراکترهای خط جدید از هم جدا می‌شوند.

فیلدهای هر رکورد با علامت دلار ($) و به دنبال آن شماره فیلد، که با ۱ شروع می‌شود، ارجاع داده می‌شوند. فیلد اول با $1، فیلد دوم با $2، و ... نشان داده می‌شود. آخرین فیلد را می‌توان با متغیر ویژه $NF نیز ارجاع داد. کل رکورد را می‌توان با $0 ارجاع داد.

```
awk ‘pattern {action}’ input-file >output-file
```

هنگامی‌که awk داده‌ها را پردازش می‌کند، اگر رکورد با الگو مطابقت داشته باشد، آن اقدام تعیین شده را روی آن رکورد انجام می‌دهد. وقتی قانون فاقد الگو باشد، تمام رکوردها (خطوط) مطابقت خواهند داشت.

اقدامات تعریف شده در awk در کاراکتر {} محصور شده و شامل چندین عبارت است که هر عبارت عملیاتی را که باید انجام شود، مشخص می‌کند. یک action می‌تواند بیش از یک عبارت داشته باشد که با خط جدید یا نقطه ویرگول (;) از هم جدا شده‌اند. اگر قانون هیچ actionای نداشته باشد، به طور پیش‌فرض کل رکورد را چاپ می‌کند.

اگر جداکننده ستون‌ها به جز space و tab باشد از F– استفاده می‌شود.

awk انواع مختلفی از عبارات را پشتیبانی می‌کند، از جمله expression ها، شرط‌ها، دستورات ورودی، خروجی و غیره. رایج‌ترین عبارات awk به شرح زیر هستند:

exit: اجرای کل برنامه را متوقف کرده و خارج می‌شود.

next: پردازش رکورد فعلی را متوقف کرده و به رکورد بعدی در داده‌های ورودی می‌رود.

print: رکوردها، فیلدها، متغیرها و متن سفارشی را چاپ می‌کند.

printf: به شما کنترل بیشتری بر فرمت خروجی می‌دهد.

هنگام نوشتن برنامه‌های awk ، تا انتهای خط هر چیزی که بعد از علامت هش (#) باشد، یک comment در نظر گرفته می‌شود. خطوط طولانی را می‌توان با استفاده از کاراکتر ادامه یعنی بک اسلش (\) به چند خط تقسیم کرد.

اجرای برنامه‌های awk

یک برنامه awk را می‌توان به روش‌های مختلفی اجرا کرد. اگر برنامه کوتاه و ساده باشد، می‌توان آن را مستقیماً به مفسر awk در خط فرمان ارسال نمود:

```
awk 'program' input-file…
```

هنگامی‌که برنامه را در خط فرمان اجرا می‌کنید، باید آن را در کاراکتر ' ' محصور کرد، بنابراین shell برنامه را تفسیر نمی‌کند.

اگر برنامه بزرگ و پیچیده است، بهتر است آن را در یک فایل قرار داده و از گزینه f- برای ارسال فایل به دستور awk استفاده نمایید:

```
awk -f program-file input-file…
```

در مثال‌های زیر، از فایلی با نام "test.txt" استفاده خواهیم کرد که شامل خطوط زیر است:

```
Name1 Milwaukee    60 22 0.732
Name2 Toronto    58 24 0.707
33Name3 Philadelphia 51 31 0.622
Name4 Boston     49 33 0.598
Name5 Indiana     48 34 0.585
```

الگوهای awk

الگوهای موجود در awk کنترل می‌کنند که آیا action مرتبط باید اجرا شود یا خیر.

awk از انواع مختلف الگوها، از جمله عبارات منظم، عبارات رابطه‌ای، عبارات محدوده‌ای و عبارات خاص پشتیبانی می‌کند.

هنگامی‌که قانون بدون الگو باشد، هر رکورد ورودی مطابقت خواهد داشت. در اینجا یک مثال از یک قانون است که شامل تنها یک action است:

```
awk '{ print $3 }' test.txt
```

برنامه، فیلد سوم هر رکورد را به صورت زیر چاپ می‌کند:

```
output:
60
58
51
49
48
```

الگوهای عبارات منظم

یک عبارت منظم یا regex الگویی است که با مجموعه‌ای از رشته‌ها مطابقت دارد. الگوهای عبارت منظم awk درون دو اسلش (//) قرار می‌گیرند:

```
/regex pattern/ { action }
```

مثال)

```
awk '/0.5/ { print $1 }' test.txt
```

```
output:
Name4
Name5
```

الگو می‌تواند هر نوع عبارت منظم پیچیده‌ای باشد. در اینجا مثالی وجود دارد که با استفاده از آن اگر رکورد با دو یا چند رقم شروع شود، اولین فیلد را چاپ می‌کند:

```
awk '/^[0-9][0-9]/ { print $1 }' test.txt
```

```
output:
33Name3
```

الگوهای عبارات رابطه‌ای

الگوهای عبارات رابطه‌ای معمولاً برای مطابقت با محتوای یک فیلد یا متغیر خاص استفاده می‌شوند.

به طور پیش فرض، الگوهای عبارات منظم با رکوردها مطابقت دارند. برای تطبیق یک regex با یک فیلد، فیلد را مشخص کرده و از "contain" یا عملگر مقایسه (~) برای الگو استفاده کنید.

به عنوان مثال، به منظور چاپ اولین فیلد هر رکورد که فیلد دوم آن حاوی "ia" است، باید به صورت زیر عمل نمایید:

```
awk '$2 ~ /ia/ { print $1 }' test.txt
```

```
Output:
33Name3
```

```
Name5
```

به منظور تطبیق فیلدهایی که دارای الگوی مشخصی نیستند از عملگر !~ استفاده نمایید:

```
awk '$2 !~ /ia/ { print $1 }' test.txt
```

```
output:
Name1
Name2
Name4
```

شما می‌توانید رشته‌ها یا اعداد را با روابطی مانند، بزرگ‌تر، کوچک‌تر، مساوی و غیره مقایسه کنید. دستور زیر، اولین فیلد از تمام رکوردهایی را که فیلد سوم آن‌ها بزرگتر از ۵۰ است، چاپ می‌کند:

```
awk '$3 > 50 { print $1 }' test.txt
```

```
Name1
Name2
33Name3
```

الگوهای محدوده(Range)

الگوهای محدوده شامل دو الگو هستند که با کاما از هم جدا می‌شوند:

```
pattern1, pattern2
```

یک دسته از رکوردها که در آن اولین رکورد با الگوی اول منطبق می‌شود و آخرین رکورد با الگوی دوم منطبق می‌شود، مطابقت خواهند داشت.

در اینجا یک مثال آورده شده است که اولین فیلد همه رکوردها از رکورد شامل "Name2" تا رکورد شامل "Name4" را چاپ می‌کند:

```
awk '/Name2/,/Name4/ { print $1 }' test.txt
```

```
output:
Name2
33Name3
Name4
```

الگوها همچنین می‌توانند عبارات رابطه‌ای باشند. دستور زیر تمام رکوردها را از رکوردی که فیلد چهارم آن برابر با ۳۲ است تا رکوردی که فیلد چهارم آن برابر با ۳۳ است چاپ می‌کند:

```
awk '$4 == 31, $4 == 33 { print $0 }' test.txt
```

```
output:
33Name3 Philadelphia 51 31 0.622
Name4Boston     49 33 0.598
```

دقت نمایید که الگوهای محدوده را نمی‌توان با سایر عبارات الگو ترکیب کرد.

الگوهای عبارات خاص

awk شامل الگوهای ویژه زیر است:

BEGIN: برای انجام اقدامات قبل از پردازش رکوردها استفاده می‌شود.

END: برای انجام اقدامات پس از پردازش رکوردها استفاده می‌شود.

الگوی BEGIN به طور کلی برای تنظیم متغیرها و الگوی END برای پردازش داده‌ها از رکوردها مانند محاسبات استفاده می‌شود.

مثال زیر "Start Processing" را چاپ می‌کند، سپس قسمت سوم هر رکورد و در نهایت "End Processing" را چاپ می‌کند:

```
awk 'BEGIN { print "Start Processing." }; { print $3 }; END { print "End Processing." }'
test.txt
```

```
ouput:
Start Processing
60
58
51
49
48
End Processing.
```

اگر یک برنامه، تنها یک الگوی BEGIN داشته باشد، actionها اجرا می‌شوند ولی ورودی پردازش نمی‌شود. اگر یک برنامه، تنها یک الگوی END داشته باشد، ورودی قبل از اجرایactionهای قانون پردازش می‌شود.

نسخه Gnu awk شامل دو الگوی خاص دیگر BEGINFILE و ENDFILE نیز هست که به شما امکان می‌دهد، هنگام پردازش فایل‌ها اقداماتی را انجام دهید.

ترکیب الگوها

awk به شما امکان می‌دهد دو یا چند الگو را با استفاده از عملگر AND منطقی (&&) و عملگر OR منطقی (||) ترکیب نمایید.

در ادامه، نمونه‌ای ارائه شده است که از عملگر && برای چاپ اولین فیلد از رکوردهایی که فیلد سوم آن‌ها بزرگ‌تر از ۵۰ و فیلد چهارم آن‌ها کوچک‌تر از ۳۰ است، استفاده می‌کند:

```
awk '$3 > 50 && $4 < 30 { print $1 }' test.txt
```

```
Name1
Name2
```

متغیرهای داخلی awk

awkچندین متغیر داخلی دارد که حاوی اطلاعات مفیدی هستند و به شما امکان می‌دهند نحوه پردازش برنامه را کنترل کنید. در ادامه به شرح برخی از متداول‌ترین متغیرهای داخلی Awk پرداخته شده است:

NF: تعداد فیلدهای رکورد را نمایش می‌دهد.

NR: تعداد رکوردهای فعلی را نشان می‌دهد.

FILENAME: نام فایل ورودی که در حال حاضر پردازش می‌شود را نمایش می‌دهد.

FS:جداکننده فیلدها را نشان می‌دهد.

RS: جداکننده رکورد را نشان می‌دهد.

OFS: جداکننده فیلد خروجی را نشان می‌دهد.

ORS: جداکننده رکورد خروجی را نشان می‌دهد.

در ادامه، مثالی وجود دارد که نحوه چاپ نام فایل و تعداد خطوط (رکوردها) را نشان می‌دهد:

```
awk 'END { print "File", FILENAME, "contains", NR, "lines." }' test.txt
```

```
output:
File test.txt contains 5 lines.
```

متغیرها در awk می‌توانند در هر خطی از برنامه تنظیم شوند. به منظور تعریف یک متغیر برای کل برنامه، آن را در یک الگوی BEGIN قرار دهید.

تغییر جداکننده فیلد و رکورد

مقدار پیش‌فرض جداکننده فیلد، تعدادی کاراکتر فاصله یا tab است. این مقدار را می‌توان با تنظیم متغیر FS تغییر داد.

به عنوان مثال، برای تنظیم جداکننده فیلد بر روی کاراکتر dot می‌توانید از دستور زیر استفاده نمایید:

```
awk 'BEGIN { FS = "." } { print $1 }' test.txt
```

```
output:
Name1 Milwaukee    60 22 0
Name2 Toronto    58 24 0
33Name3 Philadelphia 51 31 0
Name4 Boston     49 33 0
Name5 Indiana     48 34 0
```

جداکننده فیلد همچنین می‌تواند روی بیش از یک کاراکتر تنظیم شود:

```
awk 'BEGIN { FS = ".." } { print $1 }' test.txt
```

هنگام اجرای awk one-liners در خط فرمان، شما می‌توانید از گزینه F- نیز برای تغییر جداکننده فیلد استفاده کنید:

```
awk -F "." '{ print $1 }' test.txt
```

به طور پیش فرض، جداکننده رکورد، یک کاراکتر خط جدید است و با استفاده از متغیر RS می‌توان مقدار آن را تغییر داد.

در مثال زیر، جداکننده رکورد برروی مقدار dot تنظیم می‌شود:

```
awk 'BEGIN { RS = "." } { print $1 }' test.txt
```

```
output:
Name1 Milwaukee    60 22 0
732
Name2 Toronto    58 24 0
707
33Name3 Philadelphia 51 31 0
622
Name4 Boston     49 33 0
598
Name5 Indiana     48 34 0
585
```

actionهای awk

همانطور که قبلا بیان شد، actionهای Awk داخل کاراکترهای {} قرار می‌گیرند و زمانی که تطابقی با الگو یافت شود، اجرا می‌شوند. یک action می‌تواند صفر یا چند عبارت داشته باشد. ترتیب اجرای عبارات چندگانه به ترتیب ظاهر شدن آن‌ها است. این عبارات با خط جدید یا نقطه ویرگول (;) از هم جدا می‌شوند.

انواع مختلفی از عبارات action وجود دارد که در awk پشتیبانی می‌شوند:

Expression؛ مانند انتصاب متغیر، عملگرهای حسابی، عملگرهای افزایشی و کاهشی.

Control statement؛ مورد استفاده برای کنترل جریان برنامه؛ مانند if ، for، while، switch و ....

Output statment؛ مانند print و.printf

Compound statement؛ مورد استفاده برای گروه‌بندی عبارات دیگر.

Input statement؛ مورد استفاده برای کنترل پردازش ورودی.

Deletion statement؛ مورد استفاده برای حذف عناصر آرایه.

دستور print معمولاً پر استفاده‌ترین دستور awk است که یک خروجی فرمت‌دار از متن، رکوردها، فیلدها و متغیرها را چاپ می‌کند.

هنگام چاپ چندین مورد، باید آن‌ها را با کاما از هم جدا نمایید. به عنوان مثال:

```
awk '{ print $1, $3, $5 }' test.txt
```

موارد چاپ شده با فاصله از هم جدا می‌شوند:

```
output:
Name1 60 0.732
Name2 58 0.707
33Name3 51 0.622
Name4 49 0.598
Name5 48 0.585
```

اگر از کاما استفاده نکنید، بین آیتم‌ها فاصله وجود نخواهد داشت:

```
awk '{ print $1 $3 $5 }' test.txt
```

همانطور که مشاهده می‌کنید، در خروجی مقادیر چاپ شده به هم پیوسته‌اند:

```
output:
Name1600.732
Name2580.707
33Name3510.622
Name4490.598
Name5480.585
```

هنگامی‌که print بدون آرگومان استفاده می‌شود، به طور پیش فرض $0 print در نظر گرفته می‌شود و رکورد فعلی چاپ می‌شود.

برای چاپ یک متن سفارشی، باید متن را داخل کاراکترهای "" قرار دهید.

```
awk '{ print "The first field:", $1}' test.txt
```

```
output:
The first field: Name1
The first field: Name2
The first field: 33Name3
The first field: Name4
The first field: Name5
```

بدین صورت، شما همچنین می‌توانید کاراکترهای خاص مانند خط جدید را چاپ نمایید:

```
awk 'BEGIN { print "First line\nSecond line\nThird line" }'
```

```
output:
First line
Second line
Third line
```

دستور printf به شما کنترل بیشتری برروی فرمت خروجی می‌دهد. در مثال زیر، شماره خطوط به همراه خروجی درج شده است:

```
awk '{ printf "%3d. %s\n", NR, $0 }' test.txt
```

printf بعد از هر رکورد یک خط جدید ایجاد نمی‌کند، بنابراین ما از \n استفاده می‌کنیم:

```
output:
  1. Name1 Milwaukee    60 22 0.732
  2. Name2 Toronto    58 24 0.707
  3. 33Name3 Philadelphia 51 31 0.622
  4. Name4 Boston     49 33 0.598
  5. Pacers Indiana     48 34 0.585
```

دستور زیر، مجموع مقادیر ذخیره شده در فیلد سوم را در هر خط محاسبه می‌کند:

```
awk '{ sum += $3 } END { printf "%d\n", sum }' test.txt
```

```
 output:
 266
```

در ادامه، مثال دیگری وجود دارد که نحوه استفاده از عبارات و دستورات کنترل را برای چاپ مربع اعداد از ۱ تا ۵ نشان می‌دهد:

```
awk 'BEGIN { i = 1; while (i < 6) { print "Square of", i, "is", i*i; ++i } }'
```

```
Output:
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
```

درک و کار با دستورات تک خطی مانند دستور بالا سخت‌تر است؛ بنابراین هنگام نوشتن برنامه‌های طولانی‌تر، باید یک فایل برنامه جداگانه ایجاد نمایید:

```
            prg.awk

BEGIN {
```

```
  i = 1
  while (i < 6) {
    print "Square of", i, "is", i*i;
    ++i
  }
}
```

برنامه را با ارسال نام فایل به مفسر awk اجرا کنید:

```
awk -f prg.awk
```

استفاده از متغیرهای shell در برنامه‌های awk

اگر از دستور awk در اسکریپت‌های shell استفاده می‌کنید، به احتمال زیاد باید یک متغیر shell را به برنامه awk ارسال نمایید. یکی از گزینه‌ها این است که برنامه را به جای ‘ ’ با “ ” محصور کرده و متغیر را در برنامه جایگزین نمایید. با این حال، این گزینه برنامه awk شما را پیچیده‌تر می‌کند؛ زیرا باید از متغیرهای awk بگذرید.

روش پیشنهادی برای استفاده از متغیرهای shell در برنامه‌های awk، اختصاص متغیر shell به متغیر awk است. به عنوان مثال:

```
num=51
awk -v n="$num" 'BEGIN {print n}'
```

```
51
```

تمرین:

۱. فایلی بسازید که سه ستون و ۵ سطر داشته باشد و در هر ستون اعدادی تصادفی را وارد کنید.
   الف) اعداد ستون دوم را با هم جمع کند.
   ب) میانگین اعداد ستون سوم سطرهایی که مقادیر ستون اول آنها بزرگتر از ۲ است را محاسبه کنید.
۲. لیست کاربرانی که شناسه کاربری بزرگتر یا مساوی ۵۰۰ دارند و با حروف [a-k] آغاز می شوند را نمایش دهید.
۳. لیست نام فایلها و دایرکتوری ها در دایرکتوری خانگی‌تان را نمایش دهید که آخرین زمان دسترسی آن‌ها در یک ماه اخیر باشد.