

UNIX 및 실습

11장 보충 – awk (2)

파이프 열기와 닫기

▶ END 패턴에서 먼저 파이프를 닫고 후속 처리

▶ 연 파이프와 동일하게 표현

```
[kgu@lily ch11_awk]$ cat awk.sc1
# awk script using pipe
BEGIN{
    printf "%-22s%s\n", "NAME", "DISTRICT"
    print "-----"
}
/west/ {count++}
{printf "%-10s %-10s %-15s\n", $3, $4, $1 | "sort -k 1"}
END{
    close ("sort -k 1")
    print "-----"
    printf "The number of sales persons in Western "
    printf "region is " count ".\n"
}
```

```
[kgu@lily ch11_awk]$ gawk -f awk.sc1 datafile
NAME                DISTRICT
-----
AM      Main      northeast
Ann     Stephens  central
Charles Main      northwest
Lewis   Dalsase   southwest
Margot  Weber     north
Patrick Hemenway  southeast
Sharon  Gray      western
Suan    Chin     southern
TB      Savage   eastern
-----
The number of sales persons in Western region is 3.
```

조건문

▶ 구문

- ▶ 패턴에서는 생략하지만, 동작에서는 if를 반드시 명시해야 함

```
if (조건식) {  
    문장; 문장;  
}
```

```
{ if (조건식) {  
    문장; 문장;  
}  
else if (조건식) {  
    문장; 문장;  
}  
else {  
    문장; 문장;  
}  
}
```

```
[kgu@lily ch11_awk]$ gawk '{if ($8 > 20) print $1 " Too High "; else print  
"Range is OK!}' datafile  
northwest Too High  
western Too High  
Range is OK!  
Range is OK!  
Range is OK!  
Range is OK!  
Range is OK!  
Range is OK!  
Range is OK!
```

반복문

▶ while

▶ 사용 예

```
[kgu@lily ch11_awk]$ gawk '{i = 1; while (i <= NF) {print NF, $i; i++; }}' datafile
```

▶ for

▶ 사용 예

```
[kgu@lily ch11_awk]$ gawk '{for (i = 1; i <= NF; i++) print NF, $i }' datafile
```

▶ 반복문 제어 : break, continue

```
[kgu@lily ch11_awk]$ cat awk.sc2
{ for ( x = 3; x <= NF; x++)
    if ( $x < 0 ) { print "Bottomed out!"; break }
    # break for loop
}
{ for ( x = 3; x <= NF; x++)
    if ($x == 0) { print "Get Next item"; continue }
}
```

배열 (1)

- ▶ Associative array : 배열첨자에 숫자나 문자열 모두 사용 가능
 - ▶ 배열 첨자를 키로, 해쉬 알고리즘 적용

```
[kgu@lily ch11_awk]$ gawk '{name[x++] = $2}; END {for(i=0; i < NR; i++) print i, name[i]}' employees
0 Jones
1 Adams
2 Chang
3 Black
```

```
[kgu@lily ch11_awk]$ cat awk.sc3
/tom/ {count["tom"]++}
/mary/ {count["mary"]++}
END {print "There are " count["tom"] " Toms and " count["mary"] " Marys in the file." }
[kgu@lily ch11_awk]$ gawk -f awk.sc3 datafile2
There are 2 Toms and 4 Marys in the file.
```

배열 (2)

▶ 특수 for 반복문

```
{ for (아이템 in 배열명) {  
    print 배열명[아이템]  
}
```

```
[kgu@lily ch11_awk]$ gawk '/^tom/{name[NR]=$1; END {for (i = 1; i <= NR; i++) {print name[i]}}' datafile2  
tom
```

```
tom
```

```
[kgu@lily ch11_awk]$ gawk '/^tom/{name[NR]=$1; END {for (i in name) {print name[i]}}' datafile2  
tom  
tom
```

```
[kgu@lily ch11_awk]$ gawk '{count[$2]++} END{for (name in count) print name, count[name]}' employees
```

배열 (3)

▶ 배열과 split 함수

split (문자열, 배열, 필드구분자)
또는
split (문자열, 배열)

```
[kgu@lily ch11_awk]$ gawk 'BEGIN { split("3/15/2013", date, "/"); print "The month is " date[1] " and the year is " date[3];}' employees  
The month is 3 and the year is 2013
```

gawk 내장함수

▶ sub, gsub

sub (정규표현식, 치환문자);
sub (정규표현식, 치환문자, 대상문자열);

```
[kgu@lily ch11_awk]$ gawk '{sub (/Mac/, "Macintosh"); print}' testfile  
[kgu@lily ch11_awk]$ gawk '{sub (/Mac/, "Macintosh", $1); print}' testfile
```

▶ index

index (문자열, 부분문자열)

```
[kgu@lily ch11_awk]$ gawk '{print index("Hello", "llo")}' testfile  
3
```

▶ substr

substr (문자열, 시작위치);
substr (문자열, 시작위치, 문자열길이);

```
[kgu@lily ch11_awk]$ gawk '{ print substr("Santa Claus", 7, 6) }' testfile  
Claus
```


gawk 내장 산술함수

- ▶ `atan2(x, y)`
- ▶ `cos(x)`
- ▶ `exp(x)`
- ▶ `int(x)`
- ▶ `log(x)`
- ▶ `rand(x)` : $0 < r < 1$ 사이 난수
- ▶ `sin(x)`
- ▶ `sqrt(x)`
- ▶ `srand(x)` : `rand()`를 위한 seed

사용자 정의 함수

▶ 특징

- ▶ 변수들은 값에 의한 호출로 함수에 전달된다.
- ▶ 배열은 참조에 의한 호출이며, 따라서 함수 안에서 내용물들을 바꿀 수 있다.
- ▶ 매개변수 목록에 들어있지 않은 변수들은 모두 전역변수로 취급된다.
- ▶ 따라서 함수에 쓰일 내부변수들은 매개변수 목록에 들어있어야 한다. (보통 마지막에 포함)

```
[kgu@lily ch11_awk]$ cat awk.sc4
function my_sort ( scores, num_elements, temp, i, j ) {
    # temp, i, j will be local
    for ( i = 2; i <= num_elements; i++) {
        for (j = i; scores[j-1] > scores[j]; j--) {
            temp = scores[j]
            scores[j] = scores[j-1]
            scores[j-1] = temp
        }
    }
}

{
    for (i = 1; i <= NF; i++)
        grades[i] = $i
    my_sort(grades, NF) # only two arguments are passed
    for (j = 1; j <= NF; j++)
        printf ("%d ", grades[j])
    printf("\n")
}
```

실습과제

(awk와 모든 유틸리티 사용)

▶ last 명령 결과

```
[kgu@lily ch11_awk]$ last
ahs6113 pts/7      220.68.175.178  Fri May 24 12:14  still logged in
ahs6113 pts/4      220.68.175.178  Fri May 24 12:09  still logged in
ahs6113 pts/6      220.68.175.178  Fri May 24 11:49  still logged in
ahs6113 pts/5      220.68.175.178  Fri May 24 11:48  still logged in
kgu     pts/2      220.68.173.204  Fri May 24 11:10  still logged in
kira43  pts/1      203.232.252.162 Fri May 24 11:10 - 12:24 (01:14)
kgu     pts/0      220.68.173.204  Fri May 24 11:07  still logged in
beome13 pts/0      203.232.252.159 Fri May 24 11:04 - 11:06 (00:01)
y2k0605 pts/0      203.232.252.153 Fri May 24 10:53 - 11:01 (00:08)
ahs6113 pts/19     203.232.252.154 Fri May 24 10:46 - 10:48 (00:01)
mrun5   pts/18     203.232.252.145 Fri May 24 10:45 - 10:52 (00:06)
ahs6113 pts/23     203.232.252.154 Fri May 24 10:39 - 10:48 (00:08)
race0   pts/22     203.232.252.153 Fri May 24 10:39 - 10:48 (00:08)
race0   pts/21     203.232.252.153 Fri May 24 10:38 - 10:48 (00:09)
missjuba pts/20     203.232.252.138 Fri May 24 10:38 - 10:42 (00:04)
beome13 pts/19     203.232.252.139 Fri May 24 10:38 - 10:44 (00:06)
...
...

wtmp begins Thu May  9 06:52:14 2013
```

1. 사용자와 로그인한 횟수 출력
2. 사용자별로 작업한 시간을 모두 합산해 시, 분, 초 형식으로 출력
 - ▶ still logged in의 경우 합산하지 않음
3. 사용자별로 작업시간이 제일 작았던 경우와 제일 길었던 경우 출력
4. 작업시간 합계를 기준으로 정렬해서 표 형식으로 출력
5. 특정 사용자의 요일별 작업시간 계산하여 출력

[실습과제]

- ▶ 실습 각 단계 화면 캡처하여 pdf 파일로 정리하여 과제 제출 (cms.mmu.ac.kr/bear)
- ▶ 제출기한 : 5월 30일 자정