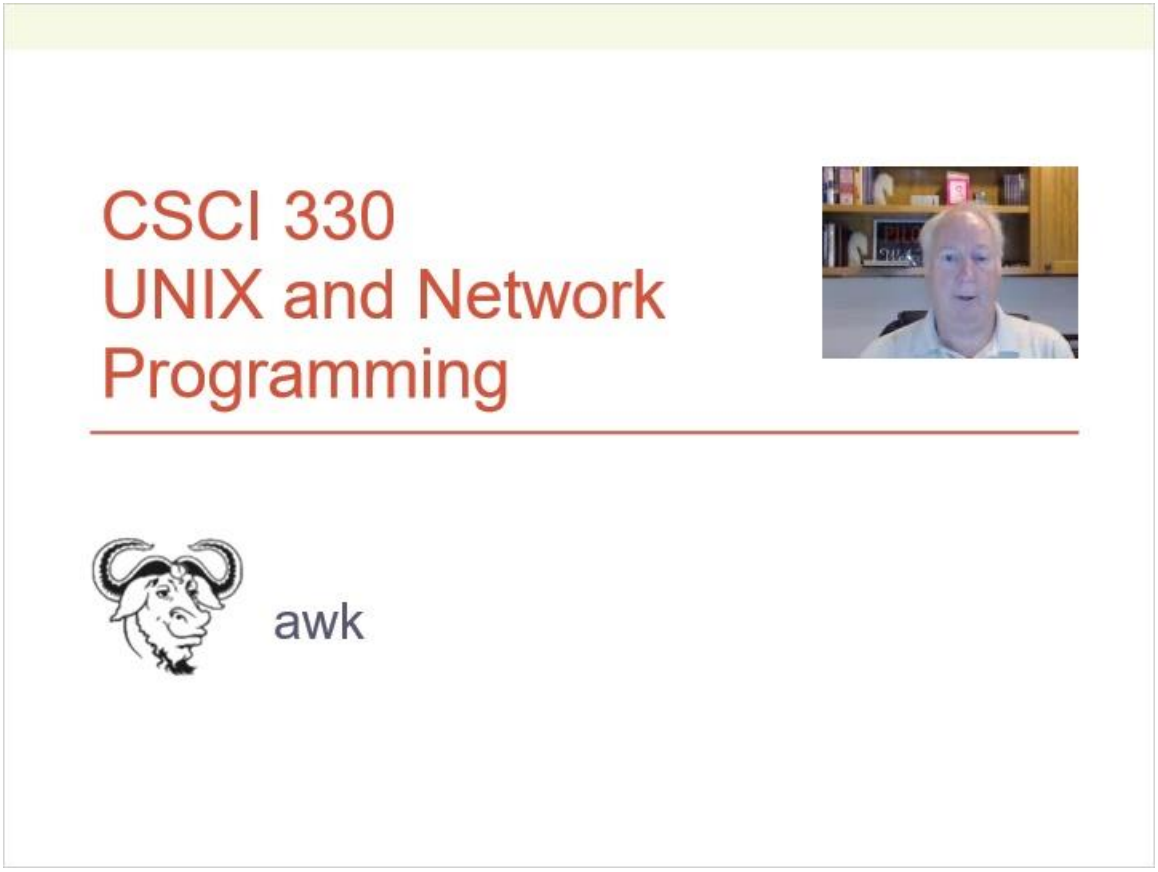
Awk

1. Awk

1.1 CSCI 330

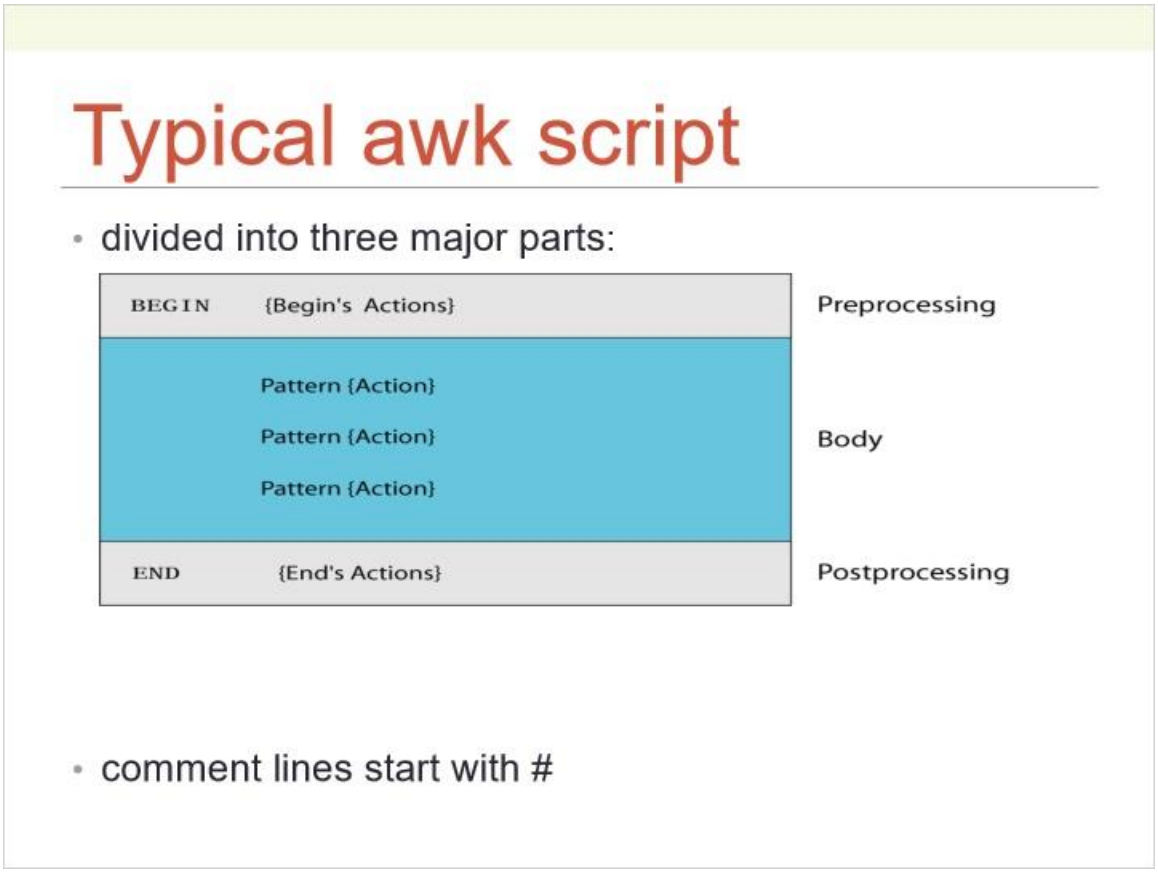


1.2 What can you do with awk?

What can you do with awk?

- awk operation:
  - scans a file line by line
  - splits each input line into fields
  - compares input line/fields to pattern
  - performs action(s) on matched lines
- Useful for:
  - transform data files
  - produce formatted reports
- Programming constructs:
  - format output lines
  - arithmetic and string operations
  - conditionals and loops

1.3 Typical awk script



1.4 awk Array

awk Array

- awk allows one-dimensional arrays
  - index can be number or string
  - elements can be string or number
- array need not be declared
  - its size
  - its element type
- array elements are created when first used
  - initialized to 0 or “”

1.5 Arrays in awk

Arrays in awk

Syntax:

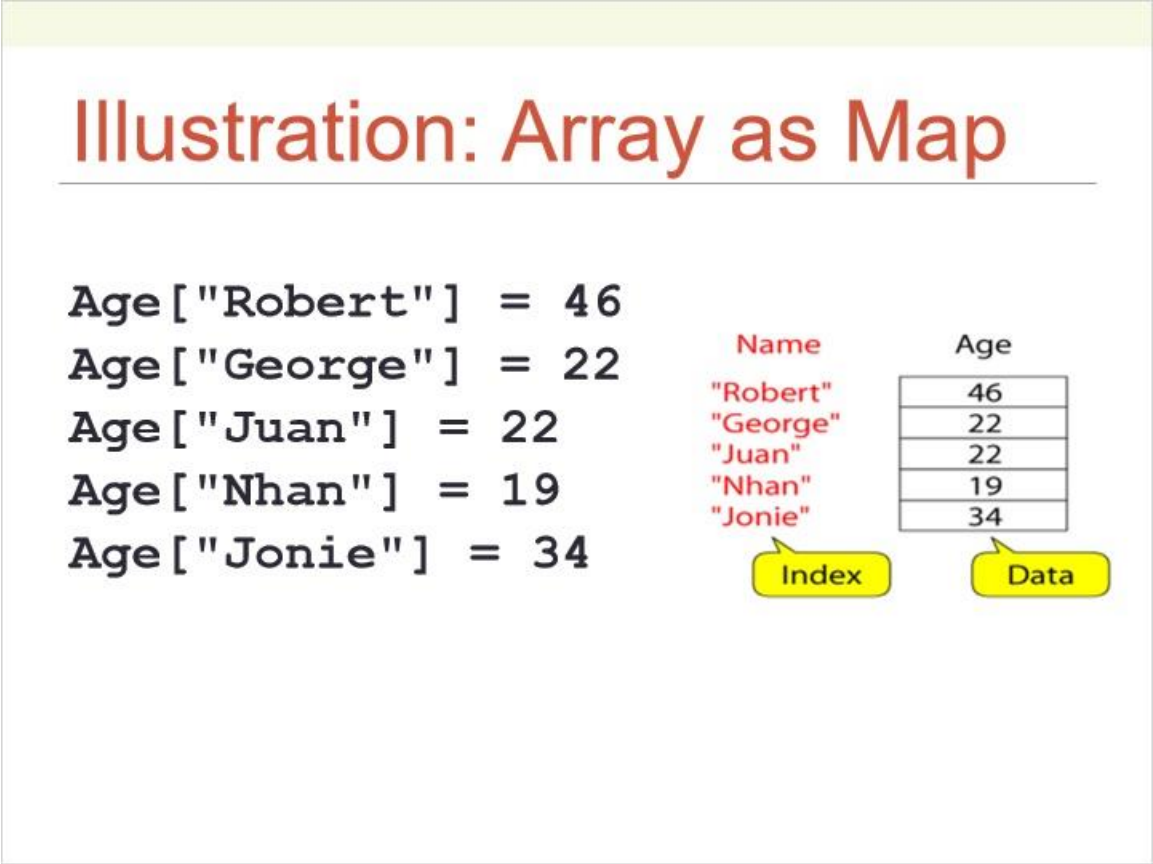
```
arrayName[index] = value
```

Examples:

```
list[1] = "some value"
list[2] = 123

list["other"] = "oh my !"
```

1.6 Illustration: Array as Map



1.7 Example: process sales data

Example: process sales data

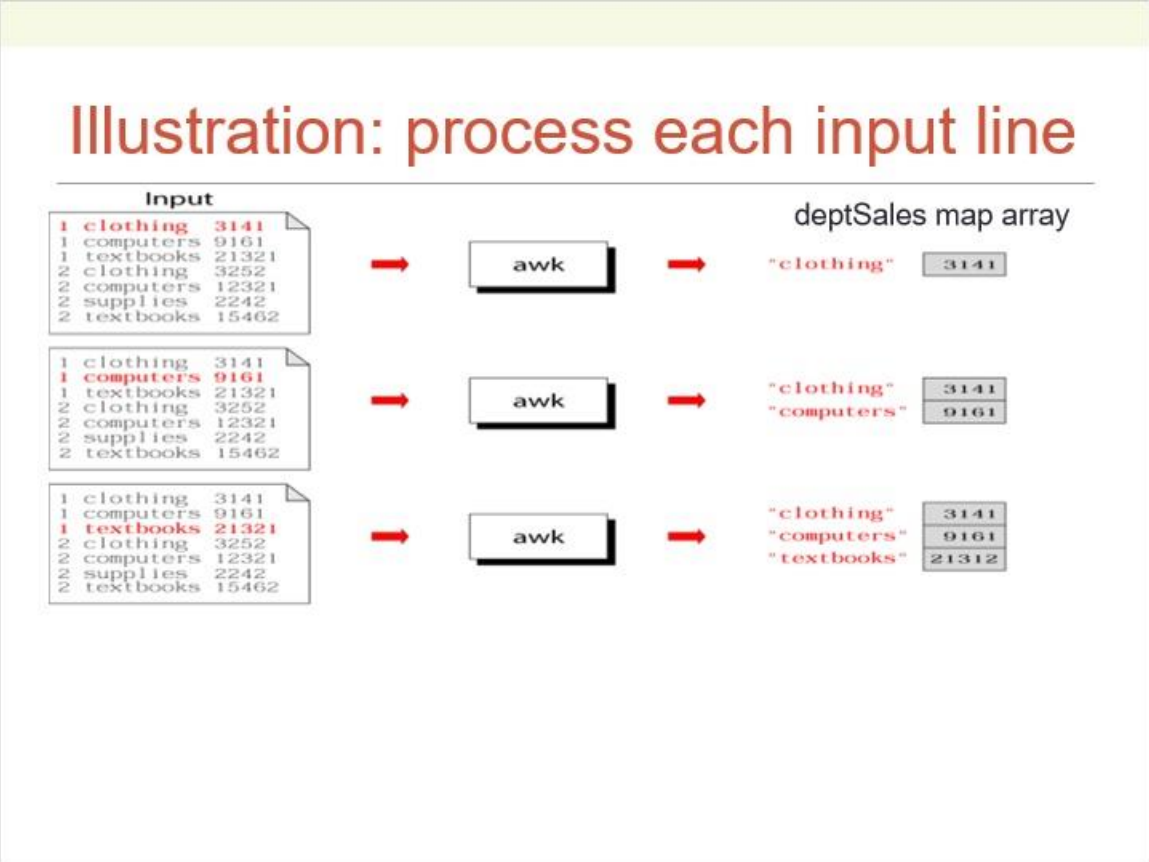
- input file:

```
1  clothing   3141
1  computers  9161
1  textbooks 21321
2  clothing   3252
2  computers 12321
2  supplies   2242
2  textbooks 15462
```

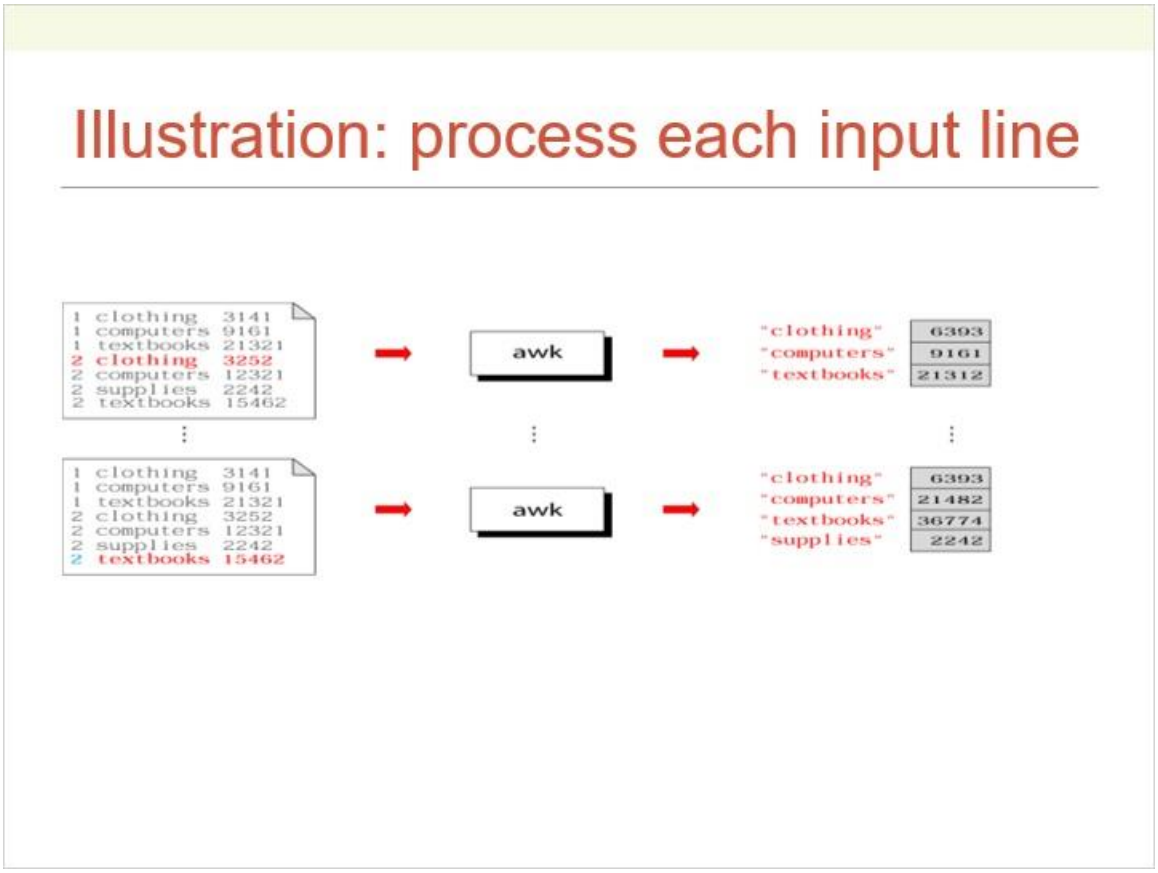
- desired output:

  summary of department sales

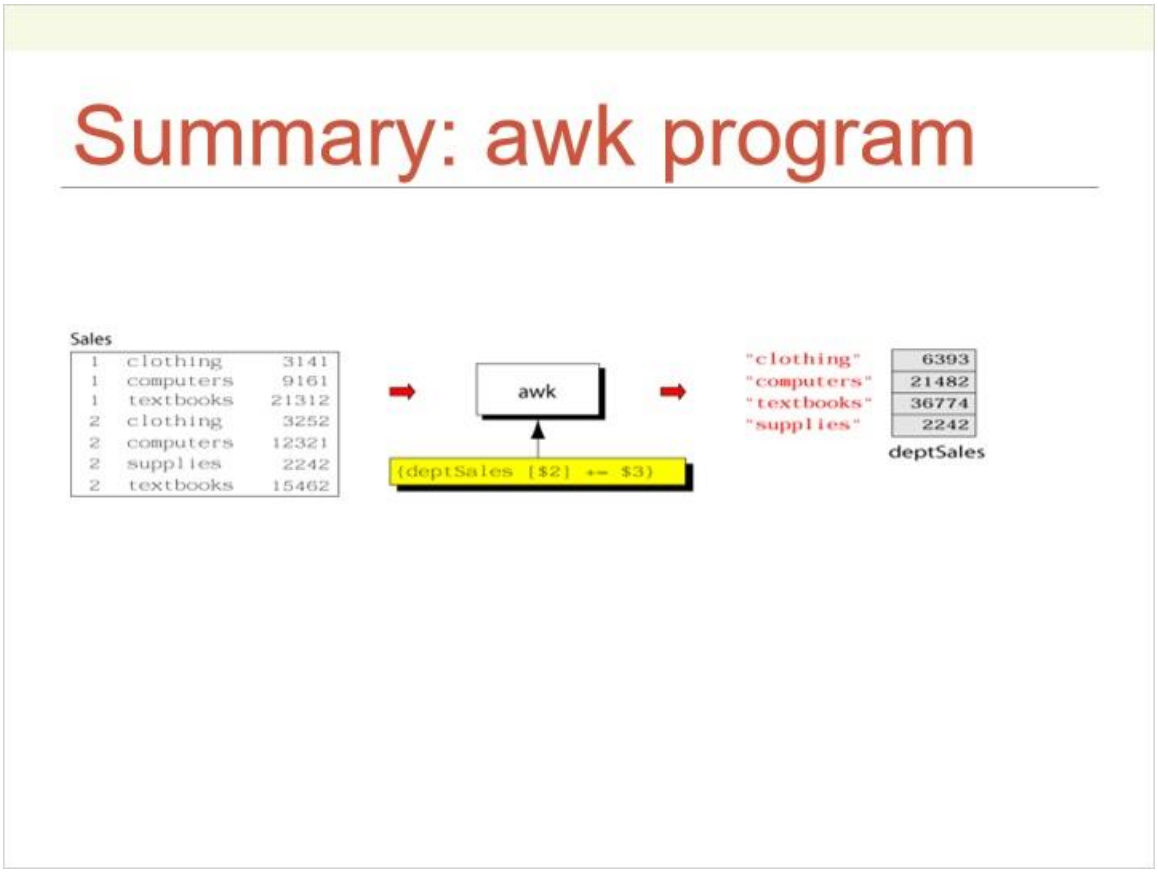
1.8 Illustration: process each input line



1.9 Illustration: process each input line



1.10 Summary: awk program



1.11 Example: complete program

Example: complete program

```
{
      deptSales[$2] += $3
}
END {
         for (x in deptSales)
                  print x,
 deptSales[x]
}
```

1.12 awk built-in functions

awk built-in functions

- arithmetic

  ex.: **sqrt, rand**

- string

  ex.: **index, length, split, substr**

  sprintf, tolower, toupper

- misc.

  ex.: **system, systime**

1.13 awk built-in split function

awk built-in split function

`split(string, array, fieldsep)`

- divides **`string`** into pieces separated by **`fieldsep`**
- stores the pieces in **`array`**
- if **`fieldsep`** is omitted, the value of FS is used

Example:

`split("26:Miller:Comedian", fields, ":")`

- sets the contents of the array **`fields`** as follows:

```
fields[1] = "26"
fields[2] = "Miller"
fields[3] = "Comedian"
```

1.14 awk control structures

awk control structures

- Conditional
  - if-else
- Repetition
  - for
    - with counter
    - with array index
  - while

also: break, continue

1.15 if Statement

if Statement

Syntax:

```
if (conditional expression)
 statement-1
else
 statement-2
```

Example:

```
if ( NR < 3 )
 print $2
else
 print $3
```

- Use compound { } for more than one statement:

```
{
    ...
    ...
}
```

1.16 if Statement for arrays

if Statement for arrays

Syntax:

```
if (value in array)
 statement-1
else
 statement-2
```

Example:

```
if ("clothing" in deptSales)
 print deptSales["clothing"]
else
 print "not found"
```

1.17 for Loop

for Loop

Syntax:

```
for (initialization; limit-test;
 update)
     statement
```

Example:

```
for (i=1; i <= 10; i++)
  print "The square of ", i, " is ",
 i*i
```

1.18 for Loop for arrays

for Loop for arrays

Syntax:

```
for (var in array)
        statement
```

Example:

```
for (x in deptSales) {
   print x
   print deptSales[x]
}
```

1.19 while Loop

while Loop

Syntax:

```
while (logical expression)
        statement
```

Example:

```
i=1
while (i <= 10) {
   print "The square of ", i, " is ", i*i
   i = i+1
}
```

1.20 loop control statements

loop control statements

- **`break`**

  exits loop

- **`continue`**

  skips rest of current iteration, continues with next iteration

1.21 Example: sensor names

Example: sensor names

```
1 Temperature
2 Rainfall
3 Snowfall
4 Windspeed
5 Winddirection
```

- also: sensor readings
- Plan: print report with average readings per sensor

1.22 Example: sensor readings

Example: sensor readings

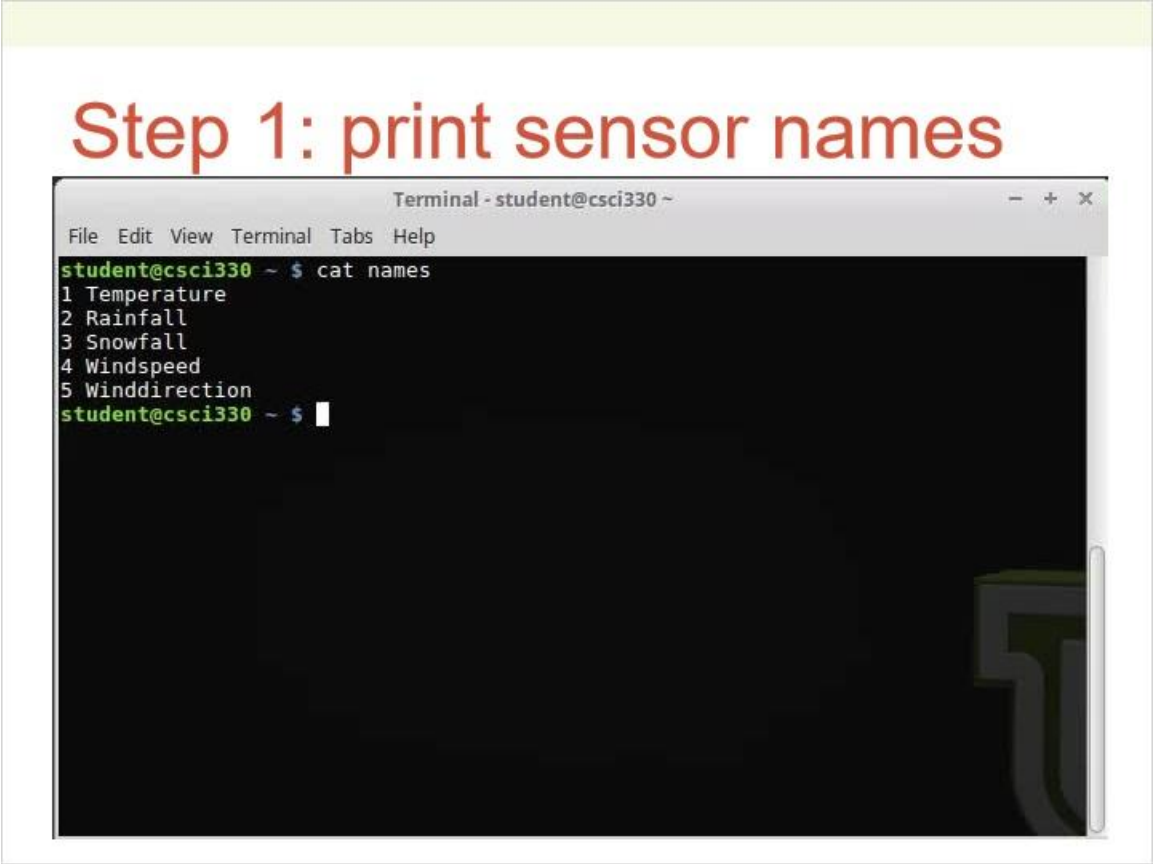
```
2012-10-01/1/68
2012-10-02/2/6
2011-10-03/3/4
2012-10-04/4/25
2012-10-05/5/120
2012-10-01/1/89
2011-10-01/4/35
2012-11-01/5/360
2012-10-01/1/45
2011-12-01/1/61
2012-10-10/1/32
```

1.23 Report: average readings

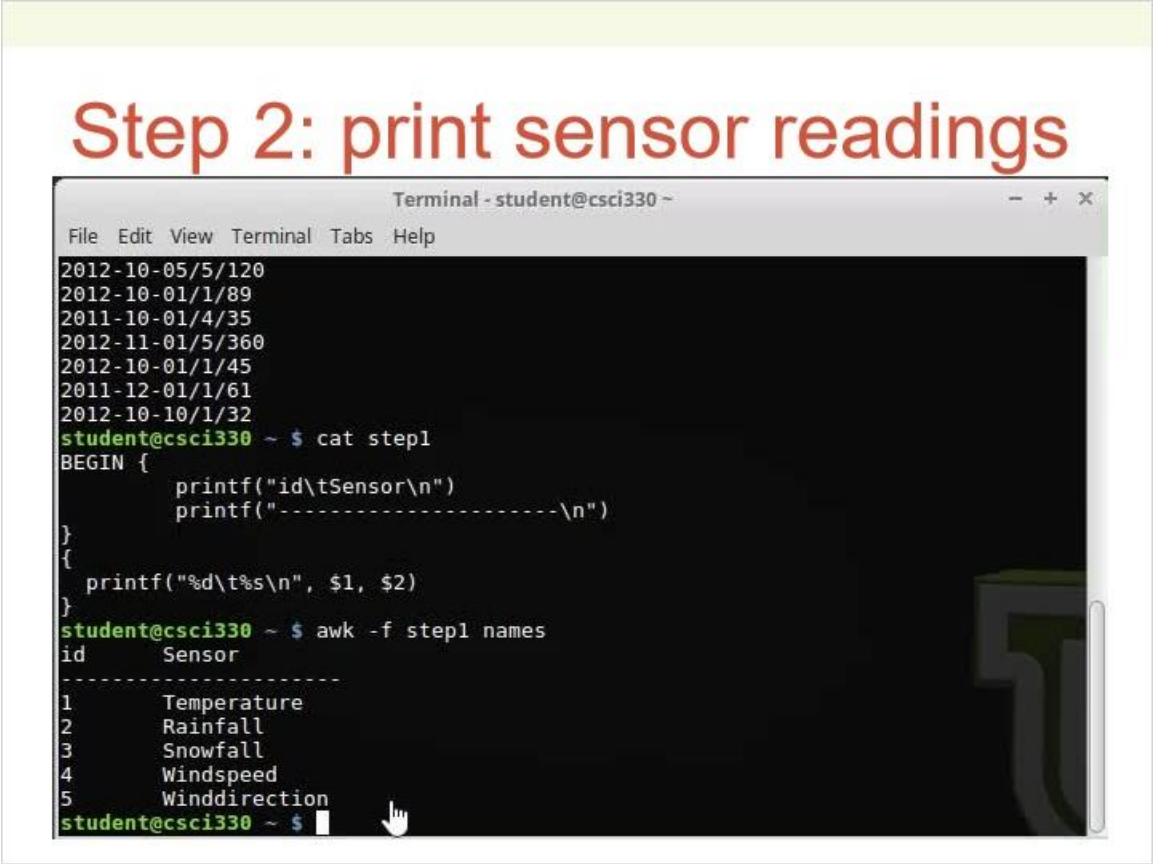
Report: average readings

Sensor	Average
Windspeed	30.00
Winddirection	240.00
Temperature	59.00
Rainfall	6.00
Snowfall	4.00

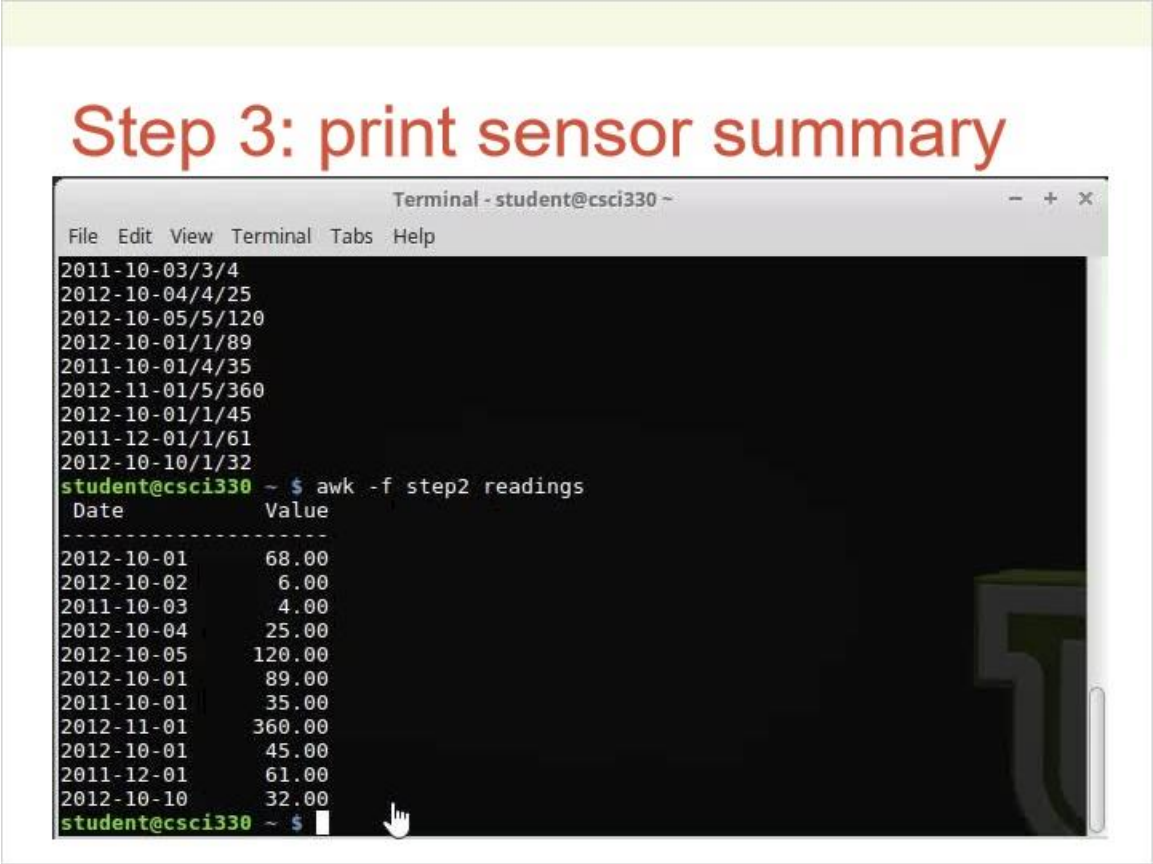
1.24 Step 1: print sensor names



1.25 Step 2: print sensor readings



1.26 Step 3: print sensor summary



1.27 Next steps: Remaining tasks

Next steps: Remaining tasks

- `awk -f sense.awk names readings`

```
          Sensor Average
----------------------
       Windspeed   30.00
   Winddirection  240.00
     Temperature   59.00
        Rainfall    6.00
        Snowfall    4.00
```

2 input files

sensor names

1.28 Next steps: Remaining tasks

Next steps: Remaining tasks

- 2 input files:
  - recognize nature of input data
    use: number of fields in record
- Sensors names:
  - substitute sensor id with sensor name
    use: array of sensor names

1.29 Example: sense.awk

Example: sense.awk

Terminal - student@csci330 ~

File Edit View Terminal Tabs Help

```
2012-10-05     120.00
2012-10-01      89.00
2011-10-01      35.00
2012-11-01     360.00
2012-10-01      45.00
2011-12-01      61.00
2012-10-10      32.00
student@csci330 ~ $ cat step3
BEGIN {   FS="/"  }
{
        sum[$2] += $3
        count[$2]++
}
END {
 for (i in sum)
   printf("%d %7.2f\n", i, sum[i]/count[i])
}
student@csci330 ~ $ awk -f step3 readings
1   59.00
2    6.00
3    4.00
4   30.00
5  240.00
student@csci330 ~ $
```

1.30 Summary

Summary

- awk
  - similar in operation to sed
    - transform input lines to output lines
  - powerful report generator