

ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 9: LESSON 2

THE AWK UTILITY

PHOTOS AND ICONS USED IN THIS SLIDE SHOW ARE LICENSED UNDER [CC BY-SA](#)



LESSON 2 TOPICS

The `awk` Utility

- Definition / Purpose
- Usage
- Using `awk` as a Filter with Pipeline Commands
- Demonstration

Perform Week 11 Tutorial

- Investigation 2
- Review Questions (**Parts C and D**)

Work on Assignment #3

- **Section 2: sed & awk**

AWK UTILITY

Definition / Purpose

*Awk is mostly used for **pattern scanning** and **processing**.*

*It searches one or more files to see if they contain lines that **matches** with the specified patterns and then performs the associated **actions**.*

The awk command is useful for reading **database files** to produce **reports**.

Reference: <https://www.geeksforgeeks.org/awk-command-unixlinux-examples/>



AWK UTILITY



Usage

```
awk [-F] 'selection_criteria {action}' file-name
```

How it Works:

- The **awk** command reads all lines in the input file and will be exposed to the **expression** (contained within **quotes**) for processing.
- The expression (contained in quotes) represents **selection criteria**, and action to **execute** contained within braces **{ }**
- If selection criteria is **matched**, then **action** (between braces) is **executed**.
- The **-F** option can be used to specify the default field delimiter (separator) character
 - eg. `awk -F";"` (would indicate a semi-colon delimited input file)

AWK UTILITY



Usage

```
awk [-F] 'selection _criteria {action}' file-name
```

Selection Criteria:

- You can use a regular expression, enclosed within slashes, as a pattern.
 - Example: `/pattern/`
- The `~` operator tests whether a field or variable matches a regular expression.
 - Example: `$1 ~ /^[0-9]/`
- The `!~` operator tests for no match.
 - Example: `$2 !~ /line/`

AWK UTILITY



Usage

```
awk [-F] 'selection _criteria {action}' file-name
```

Selection Criteria:

- You can perform both numeric and string comparisons using relational operators ($>$, $>=$, $<$, $<=$, $==$, $!=$).
- You can combine any of the patterns using the Boolean operators $||$ (OR) and $&&$ (AND).
- You can use **built-in variables** (like **NR** or "record number" representing line number) with comparison operators.
 - Example: `NR >=1 && NR <= 5`

AWK UTILITY



Usage

```
awk [-F] 'selection _criteria {action}' file-name
```

Action (execution):

- Action to be executed is contained within braces `{ }`
- The `print` command can be used to display text (fields).
- You can use parameters like `$1`, `$2` to represent **first field**, **second field**, etc. The parameter `$0` represents all fields within a **record** (line).
- You can use **built-in variables** (like `NR` or "record number" representing line number
 - eg. `{print NR,$0}` (will print record number, then entire record)

AWK UTILITY

Example 1

```
cat data.txt
```

```
Saul Murray professor
```

```
David Ward retired
```

```
Fernades Mark professor
```

```
awk '{print}' data.txt
```

```
Saul Murray professor
```

```
David Ward retired
```

```
Fernades Mark professor
```



If no pattern is specified, awk selects **all lines** in the input



AWK UTILITY

Example 2

```
cat data.txt
```

```
Saul Murray professor
```

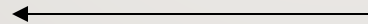
```
David Ward retired
```

```
Fernades Mark professor
```

```
awk '/^[F-Z]/ {print}' data.txt
```

```
Saul Murray professor
```

```
Fernades Mark professor
```



You can use a regular expression, enclosed within slashes, as a pattern.

In this case, the pattern is matched at the **BEGINNING** of each line (record) read from the input file.



AWK UTILITY

Example 3



```
cat data.txt
```

```
Saul Murray professor
```

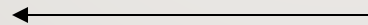
```
David Ward retired
```

```
Fernades Mark professor
```

```
awk '/^[F-Z]/' data.txt
```

```
Saul Murray professor
```

```
Fernades Mark professor
```



If no action is specified, awk copies the selected lines to standard output

AWK UTILITY



Using Variables with awk Utility

You can use parameters which represent fields within records (lines) within the expression of the awk utility.

The parameter **\$0** represents all of the fields contained in the record (line).

The parameters **\$1**, **\$2**, **\$3** ... **\$9** represent the first, second and third to the 9th fields contained within the record. Parameters greater than nine requires the value of the parameter to be placed within braces (for example: **\${10}** , **\${11}** , **\${12}** , etc.)

Unless you separate items in a print command with a **comma**, awk **catenates** them.

AWK UTILITY



Example 4

```
cat data.txt
```

```
Saul Murray professor
```

```
David Ward retired
```

```
Fernades Mark professor
```

```
awk '$1 ~ /^[F-Z]/ {print}' data.txt
```

```
Saul Murray professor
```

```
Fernades Mark professor
```

```
awk '$3 ~ /retired/ {print}' data.txt
```

```
David Ward retired
```

← The parameters **\$1, \$2, \$3 ... \$9** represent the first, second and third to the 9th fields contained within the record.

← The **~** operator tests whether a field or variable matches a regular expression

AWK UTILITY

Example 5



```
cat data.txt
```

```
Saul Murray professor
```

```
David Ward retired
```

```
Fernades Mark professor
```

```
awk '$3 !~ /retired/ {print}' data.txt
```

```
Saul Murray professor
```

```
Fernades Mark professor
```



The !~ operator tests for no match.

AWK UTILITY

Example 6

```
cat customer.dat
```

```
A100 Acme-Inc. 5400
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```

```
awk '$3 > 10000 {print}' customer.dat
```

```
R100 Rain-Ltd. 11224
```

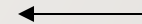
```
awk '$3 <= 6000 {print}' customer.dat
```

```
A100 Acme-Inc. 5400
```

```
T100 Toy-Inc. 3413
```



Using relational operators with the awk command.



AWK UTILITY

Example 7

```
cat customer.dat
```

```
A100 Acme-Inc. 5400
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```

```
awk '$3 >= 5000 && $3 <= 10000 {print}' customer.dat
```

```
A100 Acme-Inc. 5400
```

```
awk '$3 <= 5000 || $3 >= 10000 {print}' customer.dat
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```



← Using the && and || conditional operators with the awk command.

AWK UTILITY

Example 8

```
cat customer.dat
```

```
A100 Acme-Inc. 5400
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```

```
awk '$3 > 10000 {print $1,$2}' customer.dat
```

```
R100 Rain-Ltd.
```

```
awk '$2 ~ /Acme-Inc./ {print $3}' customer.dat
```

```
5400
```



Using parameters to specify fields with print command to display output.

AWK UTILITY

Other Variables for awk Utility

The table below show other variables that can be used with the awk command.

- **FILENAME** Name of the current input file
- **FS** Input field separator (default: SPACE or TAB)
- **NF** Number of fields in the current record
- **NR** Record number of the current record
- **OFS** Output field separator (default: SPACE)
- **ORS** Output record separator (default: NEWLINE)
- **RS** Input record separator (default: NEWLINE)



AWK UTILITY



Example

```
cat customer.dat
```

```
A100 Acme-Inc. 5400
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```

```
awk '{print NR,$0}' customer.dat
```

```
1 A100 Acme-Inc. 5400
```

```
2 R100 Rain-Ltd. 11224
```

```
3 T100 Toy-Inc. 3413
```

```
awk 'NR ==2 {print}' customer.dat
```

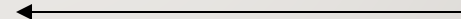
```
R100 Rain-Ltd. 11224
```

```
awk 'NR > 1 && NR < 5{print}' customer.dat
```

```
R100 Rain-Ltd. 11224
```

```
T100 Toy-Inc. 3413
```

Using **NR** (record number) variable with the awk utility



AWK UTILITY

Using awk Utility as a Filter

Although awk can be used as a streaming editor for text contained within a text file, awk can also be used as a filter using a pipeline command.

Examples

```
ls | awk '{print $1,$2}'
```



AWK UTILITY

Instructor Demonstration

Your instructor will demonstrate additional examples of using the **awk** utility.



AWK UTILITY

Getting Practice

To get practice to help perform **online Assignment #3**, perform **Week 11 Tutorial**:

- [INVESTIGATION 2: USING THE AWK UTILITY](#)
- [LINUX PRACTICE QUESTIONS](#) (Parts **C** and **D**)

Work on Assignment #3

- **Section 2: sed & awk**