Awk

Awk comparé aux outils Unix et Python Magazine Marque GNU/Linux Magazine n° Numéro 273 Mois de parution janvier 2025 Auteurs Par Blondon Stéphane

Résumé

Vous connaissez probablement déjà les outils classiques installés sur toutes les distributions GNU/Linux, et Python, langage populaire aujourd'hui. Découvrez Awk, un (vieux) langage qui pourrait vous servir lorsque les premiers sont trop limités et qu'un script Python vous semble surdimensionné.

Un script shell permet de réaliser des traitements basiques sur des lignes de texte grâce à un ensemble d'outils Unix (cut, grep, sort, wc, tr…). Cependant, lorsque les traitements deviennent plus complexes, un script shell avoue ses limites : faire des calculs est laborieux, le code est généralement peu structuré, etc. Dans ce cas, un script dans un langage plus évolué (comme Python) sera plus adapté. Cet article présente une solution intermédiaire : Awk, un langage créé par Alfred Aho, Peter Weinberger et Brian Kernighan. La version initiale date de 1977, révisée dans les années 80. Bien que toujours disponible, son heure de gloire semble passée. Il est spécialisé dans le traitement de lignes et leur transformation, ce qui le rapproche du comportement des enchaînements de filtres dans un terminal.

L'article présente Awk, en le comparant aux outils Unix classiques et à du code Python.

1. Disponibilité

En général, plusieurs interpréteurs Awk sont disponibles dans une distribution GNU/Linux (souvent gawk et mawk) et il est bien possible qu'il soit déjà installé. Tapez juste awk dans un terminal pour vérifier.

Par exemple, si mawk est installé, le début de la sortie sera :

ex001.txt

```
$ awk
Usage: mawk [Options] [Program] [file ...]

Program:
    The -f option value is the name of a file containing program text.
[...]
```

Dans le cas contraire, vous pouvez en installer un depuis les paquets de votre distribution. Ils sont nommés mawk ou gawk sous Debian et Arch Linux (mawk est dans les dépôts AUR). D'autres interpréteurs plus récents existent (nawk, goawk).

2. Utilisation

Awk a été conçu pour l'écriture de one-liners, mais il est aussi possible d'écrire un programme Awk pour l'exécuter.

L'usage en tant que one-liner est :

```
$ awk ‘le_programme’ les_fichiers_à_traiter
```

L'usage avec un programme Awk écrit dans un fichier est :

```
$ awk -f programme.awk les_fichiers_à_traiter
```

Dans le cas où le programme est écrit sur la ligne de commande, il faut l'entourer de guillemets simples, car le programme va utiliser le caractère dollar (par exemple, la variable $0). Si ce sont des guillemets doubles qui sont utilisés, le programme ne fonctionnera pas correctement, car le shell va tenter d'interpréter les variables préfixées par le dollar comme des variables du shell avant d'exécuter le programme.

Ceci est un comportement classique du shell et n’est donc pas une spécificité liée à Awk.

Pour la suite de l’article, nous allons utiliser un fichier nommé distribs.txt contenant les données suivantes :

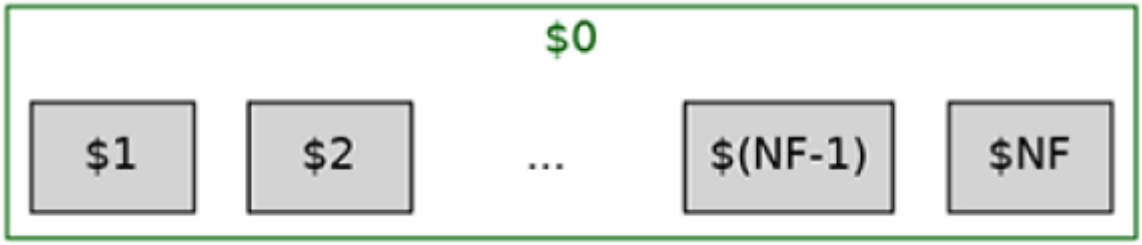
```
Raspbian linux 2012
Archlinux linux 2002
Debian linux 1993
FreeBSD bsd 1993
pfSense bsd 2004
```

2.1 Imiter cat

Par exemple, pour faire l'équivalent de cat distribs.txt :

```
$ awk '{print $0}' distribs.txt
```

Comme le montre la figure 1, $0 représente l'enregistrement total (qu'on peut assimiler ici à la ligne). $1, le contenu du premier champ, $2 le contenu du deuxième champ, etc. Le contenu du dernier champ est $NF.NF signifie Number of Fields. On peut donc obtenir astucieusement le contenu de l'avant-dernier champ avec $(NF-1).



awk compare fig 01-s Fig. 1 : Représentation des données accessibles d’un enregistrement.

La plupart des langages de programmation démarrent les listes à partir de 0 (comme C, Java, Lisp, etc.) ; Python fait aussi partie de cette catégorie. Quelques langages, comme Awk, commencent à partir de 1. $0 représentant l'enregistrement complet, le premier élément ne peut pas lui aussi commencer à 0. Parmi les autres langages commençant aussi à 1, R ou Lua sont probablement les plus connus. Julia permet de choisir n'importe quel entier, mais utilise 1 comme index par défaut.

2.2 Imiter cut

Afficher le premier et le troisième champ avec cut avec les paramètres longs :

```
$ cut --delimiter " " --fields 1,3 distribs.txt
```

Avec les paramètres courts :

```
$ cut -d " " -f 1,3 distribs.txt
```

Avec Awk :

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

Alors que cut ne fait que filtrer des champs en gardant le même ordre, Awk permet de changer facilement l'ordre des champs ou d'insérer du texte dans la ligne qui sera affichée :

```
$ awk '{print $3, "est l’année de création de", $1}' distribs.txt
```