eBook Gratuit

APPRENEZ grep

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#grep

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: grep

It is an unofficial and free grep ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official grep.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec grep

Remarques

grep imprime des lignes contenant une correspondance pour un motif dans les fichiers.

grep peut utiliser des expressions régulières et dispose de plusieurs options pour améliorer la qualité des résultats.

Les références

- POSIX
- Page de manuel de FreeBSD
- Page de manuel OpenBSD
- Manuel en ligne GNU grep
- Illumos page de manuel

Versions

POSIX grep

Version	Date de sortie
POSIX.2	1992-01-01
Norme IEEE 1003.1-2001	2001-12-06
Norme IEEE 1003.1, édition 2004	2004-01-01
Norme IEEE 1003.1, édition 2013	2013-04-19
IEEE Std 1003.1, édition 2016	2016-09-30

Illumos / OpenSolaris grep

Version	Date de sortie
2005-06-14	2005-06-14

Version	Date de sortie
2005-09-06	2005-09-06
2012-03-30	2012-03-30
2012-09-17	2012-09-17
2013-05-14	2013-05-14

GNU grep

Version	Date de sortie
2.0	1996-10-01
2.2	1998-04-27
2.3	1999-02-14
2.4.1	2000-03-01
2.4.2	2000-03-09
2.4	1999-12-03
2.5.1	2004-10-29
2.5.1a	2004-11-19
2.5.3	2007-08-02
2.5.4	2009-02-09
2,5	2002-03-13
2.6.1	2010-03-25
2.6.2	2010-03-29
2.6.3	2010-04-02
2.6	2010-03-23
2.7	2010-09-20
2.8	2011-05-13
2.9	2011-06-21

Version	Date de sortie
2.10	2011-11-16
2.11	2012-03-02
2.12	2012-04-23
2.13	2012-07-04
2.14	2012-08-20
2.15	2013-10-26
2.16	2014-01-01
2.17	2014-02-17
2.18	2014-02-20
2.19	2014-05-22
2.20	2014-06-03
2.21	2014-11-23
2.22	2015-11-01
2.23	2016-02-04
2.24	2016-03-10
2,25	2016-04-21
2.26	2016-10-02
2.27	2016-12-06
2.28	2017-02-06

BSD grep / FreeGrep

Version	Date de sortie
OpenBSD 3.0	2001-12-01
OpenBSD 3.4	2003-11-01
OpenBSD 3.5	2004-05-01

Version	Date de sortie
OpenBSD 3.6	2004-11-01
OpenBSD 3.7	2005-05-19
OpenBSD 3.8	2005-11-01
OpenBSD 3.9	2006-05-01
OpenBSD 4.0	2006-11-01
OpenBSD 4.1	2007-05-01
OpenBSD 4.3	2008-05-01
OpenBSD 4.8	2010-11-01
OpenBSD 5.0	2011-11-01
OpenBSD 5.3	2013-05-01
OpenBSD 5.7	2015-05-01
OpenBSD 5.8	2015-10-18
OpenBSD 5.9	2016-03-29
NetBSD 2.0	2004-12-09
NetBSD 4.0	2007-12-19
NetBSD 6.0	2012-10-17
NetBSD 7.0	2015-09-25
FreeBSD 9.0	2012-01-02
FreeBSD 10.0	2014-01-16

Examples

Utilisation de base

Lancer la commande:

```
grep sam someFile.txt
```

Lorsque `someFile.txt` contient:

```
fred 14 m foo
sam 68 m bar
christina 83 f baz
bob 22 m qux
Sam 41 m quux
```

Produira cette sortie:

```
sam 68 m bar
```

Ignorer le cas

Étant donné un `sample` fichier:

```
hello
Hello
HELLO_there
```

Un `grep` normal pour "hello" renvoie:

```
$ grep "hello" sample
hello
```

Utiliser `-i` permet d'ignorer la casse et de faire correspondre tout "hello":

```
$ grep -i "hello" sample
hello
Hello
HELLO_there
```

Faire correspondre des mots entiers

Étant donné un `sample` fichier:

```
hello world
ahello here
hello_there
```

Un `grep` normal pour "hello" renvoie:

```
$ grep hello sample
hello world
ahello here
hello_there
```

Utiliser `-w` permet de sélectionner les lignes contenant des correspondances qui forment des mots entiers:

```
$ grep -w hello sample
hello world
```

Rechercher du texte dans un répertoire donné, récursivement

Utiliser GNU grep

```
grep -r 'pattern' <directory path>
```

Pour répertorier également les numéros de ligne des correspondances, utilisez l'option `-n`

```
grep -rn 'pattern' <directory path>
```

Pour rechercher uniquement les fichiers avec un motif de globes particulier

```
grep --include='*.txt' -r 'pattern' <directory path>
```

Exclure les modèles de fichiers ou les répertoires

```
grep -R --exclude=*.log 'pattern' <directory path>
grep -R --exclude={*.log,*.class} 'pattern' <directory path>

grep -R --exclude-dir=tmp 'pattern' <directory path>
grep -R --exclude-dir={tmp,lib} 'pattern' <directory path>
```

Notes et autres options utiles

- `<directory path>` peut être ignoré si vous recherchez dans le répertoire actuel
- Les options `-R` suivent tous les liens symboliques, contrairement à `-r` qui ne suit les liens symboliques que s'ils sont sur la ligne de commande
- `-l` pour ne lister que les fichiers correspondants
- `-h` pour supprimer le préfixe du nom de fichier
- `--color=auto` pour mettre en évidence les motifs correspondants
- `-m <num>` pour spécifier le nombre maximal de correspondances pour chaque entrée de fichier

Solution de contournement POSIX pour rechercher de manière récursive

```
find <directory path> -type f -exec grep -l 'pattern' {} +
```

- Des options comme `-n` , `-l` , etc. peuvent être utilisées si nécessaire
- Si `{} +` n'est pas supporté, utilisez `{} \;` au lieu
- Voir la documentation de recherche pour plus d'aide sur la commande `find` comme par exemple comment inclure / exclure des types de fichiers, des répertoires, etc.

Imprime uniquement la partie correspondante des lignes

```
echo "Prints only the matching part of the lines" | grep -o "matching"
# prints matching
```

Contrôle du contexte Grep

Étant donné un fichier exemple appelé liste de films.

```
Troy
Gladiator
Robin Hood
King Arthur
BraveHeart
The Last Samurai
```

Le grep normal retourne

```
grep "Gladiator" movieslist
Gladiator
```

Maintenant, en utilisant grep pour imprimer les lignes ci-dessous ou ci-dessus du fichier.

Pour imprimer la ligne ci-dessous

```
grep -A 1 Gladiator movieslist
Gladiator
Robin Hood
```

Pour imprimer la ligne ci-dessus

```
grep -B 1 Gladiator movieslist
Troy
Gladiator
```

Pour imprimer les deux

```
grep -C 1 Gladiator movieslist
Troy
Gladiator
Robin Hood
```

Lire Démarrer avec grep en ligne: https://riptutorial.com/fr/grep/topic/2198/demarrer-avec-grep

Chapitre 2: Contrôle de ligne de contexte

Remarques

`-A` options `-A` , `-B` et `-C` ne sont pas disponibles dans POSIX (voir les spécifications POSIX pour `grep`).

Examples

Imprimer des lignes avant et / ou après le motif correspondant

En général, `grep` imprime uniquement les lignes correspondantes. Dans l'exemple ci-dessous, `seq 9` génère une liste de nombres de 1 à 9, un par ligne et `grep` imprime une seule ligne correspondante:

```
seq 9 | grep 5
# 5
```

L'option `-C n` (ou `--context=n` sous forme longue) `--context=n n` lignes avant et après chaque ligne correspondante, en plus de la ligne correspondante elle-même:

```
seq 9 | grep -C 2 '5'
# 3
# 4
# 5
# 6
# 7
```

Naturellement, moins de `n` lignes seront imprimées si la fin du fichier ou le début du fichier est atteint.

Si nous voulons imprimer des lignes seulement avant ou seulement après, mais pas les deux, nous pouvons utiliser `-B n` (`--before-context=n`) ou `-A n` (`--after-context=n`):

```
seq 9 | grep -B 2 '5'
# 3
# 4
# 5

seq 9 | grep -A 2 '5'
# 5
# 6
# 7
```

Notez que ces options ne sont pas disponibles dans POSIX (voir les spécifications POSIX pour `grep`).

Si les contextes de deux ou plusieurs lignes correspondantes se chevauchent, alors toutes les

lignes sont imprimées ensemble dans un grand contexte. Dans l'exemple ci-dessous, `5` fait partie du contexte des `3` et `7` :

```
seq 9 | grep -E --context=2 '3|7'
# 1
# 2
# 3
# 4
# 5
# 6
# 7
# 8
# 9
```

Cependant, si les contextes ne se chevauchent pas, ils sont imprimés avec une ligne de séparation de groupe. Par défaut, il s'agit d'un double trait d'union (`--`):

```
seq 9 | grep -E --context=2 '2|8'
# 1
# 2
# 3
# 4
# --
# 6
# 7
# 8
# 9
```

Nous pouvons définir une ligne de séparation de groupe différente en utilisant l' `--group-separator=SEP` ou supprimer complètement cette ligne en utilisant l'option `--no-group-separator` :

```
seq 9 | grep -E --context=0 --group-separator='****' '2|8'
# 2
# ****
# 8

seq 9 | grep -E --context=0 --group-separator='' '2|8'
# 2
#
# 8

seq 9 | grep -E --context=0 --no-group-separator '2|8'
# 2
# 8
```

Enfin, si nous choisissons l'option `-v` pour imprimer des lignes non correspondantes, alors le contexte est fourni autour de ces lignes:

```
seq 9 | grep -E -v '1|3|4|5|6|7|9'
# 2
# --
# 8

seq 9 | grep -E -v -C 1 '1|3|4|5|6|7|9'
# 1
```

```
# 2
# 3
# --
# 7
# 8
# 9
```

Lire Contrôle de ligne de contexte en ligne: https://riptutorial.com/fr/grep/topic/4152/controle-de-ligne-de-contexte

Chapitre 3: Différence entre grep, egrep, fgrep, pgrep.

Introduction

grep , **egrep** , **fgrep** , **rgrep** , **pgrep** - sont des commandes dans des systèmes d'exploitation de type Unix qui impriment des lignes correspondant à un modèle. Le grep recherche les *FICHIERS d'* entrée *nommés* pour les lignes contenant une correspondance avec le *PATTERN* donné. Par défaut, il imprime les lignes correspondantes. En outre, les programmes de variantes **egrep** , **fgrep** et rgrep sont identiques à **grep -E** , **grep -F** et **grep -r** respectivement. Ces variantes sont obsolètes, mais sont fournies pour compatibilité avec les versions antérieures.

Syntaxe

- `grep [OPTIONS] PATTERN [FILE...]`
- `grep [OPTIONS] [-e PATTERN]... [-f FILE]... [FILE...]`

Paramètres

symbole	Détails Expressions régulières de base (BRE)
`^`	le circonflexe est utilisé pour faire correspondre le début d'une ligne.
`$`	utilisé pour correspondre à la fin d'une ligne.
`.`	correspond à n'importe quel caractère sauf une nouvelle ligne.
`[]`	correspond à un caractère unique entre crochets. S'il y a un `^` intérieur, cela correspondrait à tout sauf aux caractères du crochet.
`\`	avant que l'un des caractères non alphanumériques les cite.
`*`	Le symbole correspond au caractère précédent ou à la sous-expression zéro, une ou plusieurs fois.
`\1`	Les références 1 à 9 correspondent au texte exact du groupe correspondant.
`\{m,n\}`	correspond aux éléments précédents au moins *m* et pas plus de *n* fois.
`|`	`foo|bar` correspond à foo ou à bar.
`\?`	court pour {0,1}
`\+`	(court pour **{1,}**) correspond au caractère ou à la sous-expression précédent au

symbole	Détails Expressions régulières de base (BRE)
	maximum 1 fois, ou au moins 1 fois respectivement.
`\n`	correspond à une nouvelle ligne, `\t` correspond à un onglet, etc.
`\w`	correspond à n'importe quel composant de mot et \ W correspond à tout caractère qui n'est pas un constituant de mot.
`\<\>`	faire correspondre la chaîne vide uniquement au début ou à la fin d'un mot
`\b`	correspond à l'un ou l'autre et **\ B** correspond où \ b ne le fait pas.
symbole	**Détails Expressions régulières étendues (ERE)**
`^`	ne correspond qu'au début
`$`	ne correspond qu'à la fin d'une ligne.
`.`	correspond à n'importe quel caractère (ou n'importe quel caractère sauf une nouvelle ligne).
`[…]`	correspond à n'importe quel caractère figurant entre crochets (jeu de caractères). Ajoutez un initial ^ et range le travail comme dans BRE (voir ci-dessus).
`(…)`	groupe syntaxique, à utiliser avec les remplacements * ou \ DIGIT.
`|`	pour l'alternance: `foo` | `bar` correspond à foo ou bar.
`*`	correspond au caractère ou à la sous-expression précédent plusieurs fois: 0, 1 ou plusieurs fois
`+`	correspond à une ou plusieurs fois le caractère précédent.
`?`	correspond aux caractères précédents 0 ou 1 fois.
`\`	La barre oblique inverse cite le caractère suivant s'il n'est pas alphanumérique.
`{m,n}`	correspond au caractère précédent ou à la sous-expression entre m et n fois (manquant dans certaines implémentations); n ou m peut être omis et `{m}` signifie exactement m

Remarques

fgrep signifie "Fixed-string Global Regular Expressions Print". **fgrep** est le même que **`grep -F`** . Cette commande est un grep plus rapide et se comporte comme grep mais ne reconnaît PAS les méta-caractères d'une expression régulière comme étant spéciaux. La recherche sera plus rapide car elle ne traite qu'une chaîne simple plutôt qu'un modèle complexe.

pgrep est l'acronyme de "Process-ID Global Regular Expressions Print". pgrep examine les processus en cours d'exécution et répertorie les ID de processus qui correspondent aux critères de sélection à la sortie standard. pgrep est pratique lorsque tout ce que vous voulez savoir est le processus id entier d'un processus.

grep	egrep (grep -E)	fgrep (grep -F)	pgrep
Expressions régulières de base (BRE)	Expressions régulières étendues (ERE)	Ne recherche que les chaînes	Processus de recherche par nom

Pour plus d'informations et de références, utilisez les liens suivants:

Quelle est la différence entre grep, egrep et fgrep? Unix & Linux StackExchange

Pourquoi mon expression régulière fonctionne-t-elle en X mais pas en Y? Unix & Linux StackExchange

Quelle est la différence entre grep, pgrep, egrep, fgrep? Superutilisateur

Examples

grep avec les expressions régulières de base

```
$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

egrep avec des expressions régulières étendues

```
$ egrep '^(0|1)+ [a-zA-Z]+$' searchfile.txt
011 AaBBS
```

fgrep sans expressions régulières

```
$ fgrep "." .bashrc
# will match lines with a dot.
```

pgrep avec le nom du processus

```
$ pgrep python
1299
```

Lire Différence entre grep, egrep, fgrep, pgrep. en ligne:
https://riptutorial.com/fr/grep/topic/8936/difference-entre-grep--egrep--fgrep--pgrep-

Chapitre 4: Expressions régulières

Examples

Expressions régulières

Le modèle de recherche peut également être une expression régulière. Fonctionnement:

```
grep '^[A-Z]' someFile.txt
```

Lorsque `someFile.txt` contient:

```
fred 14 m foo
sam 68 m bar
christina 83 f baz
bob 22 m qux
Sam 41 m quux
```

Produira la sortie:

```
Sam 41 m quux
```

puisque c'est la seule ligne de `someFile.txt` commençant par une lettre majuscule.

Regarde derrière

Étant donné le fichier suivant:

```
hello how are you
i am fine
let's go, you!
let's go, baby!
```

`grep` avec look-behind permet d'imprimer seulement certaines parties:

```
$ grep -Po "(?<=let's go, ).*" file
you!
baby!
```

Dans ce cas, il correspond à ce qui se passe après "allons-y".

Lire Expressions régulières en ligne: https://riptutorial.com/fr/grep/topic/4183/expressions-regulieres

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec grep	Batsu, Benjamin W., Community, David Pärsson, depperm, dingalapadum, fedorqui, Jerry Jeremiah, kdhp, mszymborski, Stuxnet78, Sundeep, UNagaswamy
2	Contrôle de ligne de contexte	Benjamin W., fedorqui, ghostarbeiter
3	Différence entre grep, egrep, fgrep, pgrep.	Bor
4	Expressions régulières	David Pärsson, dingalapadum, fedorqui