Comment générer des mots de passe chiffrer/déchiffrer aléatoires dans Linux

Dans un environnement multi-utilisateurs Linux, il est une pratique standard pour créer des logins avec un certain mot de passe par défaut.
Oui, mais on manque souvent d'imagination pour les créer, du moins j'en manque souvent.

Dans cet article, nous allons partager quelques trucs et astuces Linux intéressants pour générer des mots de passe aléatoires et aussi comment chiffrer et déchiffrer les mots de passe.

1. Générer un mot de passe aléatoire unique de 10 caractères en utilisant la commande "pwgen"

Si vous n'avez pas encore installé pwgen, utilisé `apt-get` ou `yum` pour l'obtenir. Voici le résultat :

```
test@ksh:~# pwgen 10 1
eiH1gooGha
test@ksh:~# pwgen 10 1
Taeleequ4w
test@ksh:~# pwgen 10 1
ixooB2aong
test@ksh:~# pwgen 10 1
Gi7gein3hi
test@ksh:~# pwgen 10 1
eiChohCee4
test@ksh:~# pwgen 10 1
Een3au1kub
```

Générer plusieurs mots de passe aléatoires de 50 caractère en une seule fois

```
test@ksh:~# pwgen 50
thaop5Pah0eePa1ais1PaiHie5WooWuhoo2EsaiJ1yohraid4e
wee8QueoD4cu8eetee8Aibocaeth3roe8ooquip8yeCohkepha
Thahbohhiasoa6idooquahge8Uche7che7iexiev7ahtieXoBe
Airee1joocaetieh7Riu1jueG7iewomae4ushai2jakiacuewi
jiekaefi5uqueiKephuYimipieyohphu9RaefuaLoodahkooyo
yoogau7jaiquetei8mahWaehooThoh2Mah8baeTa9ooweе1tho
ejomaiSeejeiweivu4shi6chobu2odazahaBa3yeivahn8eelo
eveipowe6iev9iquobaiv6ohthaiXosiwoixaotheu9ooxu4oo
oh2baiReWo0woo5aeciech0ooJaeJeo6thae6Veshaeg9au3ie
rohth0nujue1pho0ish7yo5ciy0kioVeet7QueeNg4ohghua2j
nuo4umueng6ijahmoo2iekeiQua1Oozahaixei1EiQu4Jah6do
Aivohk5zuiraphaiquu6Ohph0Ahng6sue0yiithee8weewei6x
XaiNg1ohR0chohqu6Wieci4Aish2Ioteeleepobaetach8Ahse
oboo8seix1pheilei6tioJac6Jatuy5voht7ioYa8ahv3Foh8g
```

```
phoo8biephae4hau7itaegoo3Ahph4acha9ubaseijae8iu6bo
zouxee9yupie1aingooquooph3queicaineo2maekahSie5ceF
vaepa1daeshohTheij8yooroome5yaiv0bair0aek4chaing7j
quiSheit4iShaiJoo4hap8ufookawoopaeyohchaejif6keiRu
cah4rohX1eiM6ingua4ChaeP0oopiepodohvahgheep0ahPhei
vii7iec6Ou6eir6iemee0tieth1looloa6ogaeH1neiy4choo9
```

2. Vous pouvez utiliser "makepasswd" pour générer des mot de passe aléatoires de longueur donnée

Assurez-vous que vous avez installé `makepasswd`.

Générer un mot de passe aléatoire de 10 caractères, ce qui est la valeur par défaut:

```
test@ksh:~# makepasswd
xEL01DUh7
test@ksh:~# makepasswd
FD5uwVU5J
test@ksh:~# makepasswd
IfPHmvyqB
test@ksh:~# makepasswd
fPu6gyRQ
test@ksh:~# makepasswd
8K3Y2bEy
test@ksh:~# makepasswd
rqb6TyqW
test@ksh:~# makepasswd
o4MEoU7jN
test@ksh:~# makepasswd
9iBryfHJad
test@ksh:~# makepasswd
8HmK8zwig
```

Générer un mot de passe aléatoire de 50 caractères :

```
test@ksh:~# makepasswd  --char 50
5SgXXNoNQV7IzGrecEETREiKLBBjibxBf6ngNYqHyYXatU4ifW
test@ksh:~# makepasswd  --char 50
e4Xnx27NtKuYqa3YHGy0tX2eLFD7Nxbd07cp5gd0f0RjH4F8BE
test@ksh:~# makepasswd  --char 50
ECGtd7FwICF1E2dj4T2rJv6hmFuqHGcAK2RjuEKfvGvP1JIYdh
test@ksh:~# makepasswd  --char 50
H2JdHSuwq2356hpEFvTw6dzmiAMumtwdbSwyqtN5VfT26xf6Mz
test@ksh:~# makepasswd  --char 50
PKTK2y7LAwtuXNIsnccofSFDYGMf0RqoKCXxiHGGNPWuCQyp24
test@ksh:~# makepasswd  --char 50
zgNIejLHJobDpDwNajS4eSSHF5T33Fm8AbLRg5LeyGL0rQ0Kss
```

Générer 7 mots de passe de 20 caractères :

```
test@ksh:~# makepasswd --char 20 --count 7
rKfARgtJznLXzYo1WyPK
49f2EVta8VuWH239UELc
KEx9ChY2XwgUPg9QuA7z
SNUWWdw2HTvujPUdRP2R
aPGmHxXBCqCREiw4pDT2
hU5m38rLWEyvedd9qiCy
IdxJM6w6BfmJUNmwVtH8
```

3. chiffrer un mot de passe en utilisant le sel

Pour ceux qui ne savent pas ce que c'est que le sel.

Le sel est une chaîne que le programme peut retrouver d'après le hash puis, connaissant cette chaine (le sel), le programme peut reconstituer un hash à partir du mot de passe en clair : si ce hash résultant est identique au hash stocké, alors le mot de passe est correct.
L'intérêt du sel, c'est justement qu'un même mot de passe peut alors avoir plusieurs hash, ce qui empêche les attaques.

Sans sel, un attaquant a juste besoin de ce qu'on appelle une “rainbow table” : c'est une table qui donne la correspondance entre tous les mots de passe possibles et tous les hash correspondants dès qu'il a un hash, il trouve le mot de passe en une fraction de seconde en vérifiant dans sa table.

Avec sel, l'attaque doit se faire par force brute, ce qui prend plus de temps (de plusieurs minutes à plusieurs années, selon la force de l'algorithme).
Le sel est une donnée aléatoire qui sert comme une entrée supplémentaire à la fonction à sens unique afin de protéger le mot de passe contre les attaques par dictionnaire.

La commande ci-dessous permet de chiffrer le mot de passe avec le sel.
La valeur de sel est prise au hasard.
Ainsi chaque fois que vous exécutez la commande ci-dessous, il va générer une sortie différente, car il accepte une valeur aléatoire du sel.

```
test@ksh:~$ mkpasswd ksh
ABpwGJKElNQlk
test@ksh:~$ mkpasswd ksh
Bfo3qxUVX2xx6
test@ksh:~$ mkpasswd ksh
BX56s0JKbvLUQ
test@ksh:~$ mkpasswd ksh
cQHvTGnktA6oU
test@ksh:~$ mkpasswd ksh
yVwOgViNsgvXw
```

Maintenant nous allons définir une valeur pour le sel.
Il va afficher le même résultat à chaque fois.

```
test@ksh:~$ mkpasswd ksh -s do
```

```
dozjkuz1nOm.s
test@ksh:~$ mkpasswd ksh -s do
dozjkuz1nOm.s
test@ksh:~$ mkpasswd ksh -s do
dozjkuz1nOm.s
test@ksh:~$ mkpasswd ksh -s do
dozjkuz1nOm.s
```

4. Chiffrer une chaîne genre “ksh-veux-dire-korn-shell” en utilisant le cryptage AES-256-CBC en utilisant le mot de passe genre “ksh” et le sel.

```
test@ksh:~$ echo ksh-veux-dire-korn-shell | openssl enc -aes-256-cbc -a -
salt -pass pass:ksh
U2FsdGVkX1+ZKDMi0XiqbAQ4Tufz1dq4q/xOwbQMqAEwsP5qWthIPgiKyvk7GMqL
```

Ici, dans l'exemple ci-dessus, la sortie de la commande `echo` est un pipeline avec la commande `openssl`, qui chiffre la chaîne en utilise un Cipher (ENC) avec l'algorithme de chiffrement AES-256-CBC et finalement avec du sel, il est chiffré en utilisant le mot de passe (ksh).

5. Déchiffrer la chaîne ci-dessus en utilisant la commande `openssl` utilisant l'algorithme de chiffrement AES-256-CBC

```
souyo@gw01:~$ echo
U2FsdGVkX1+ZKDMi0XiqbAQ4Tufz1dq4q/xOwbQMqAEwsP5qWthIPgiKyvk7GMqL | openssl
enc -aes-256-cbc -a -d -salt -pass pass:ksh
ksh-veux-dire-korn-shell
```