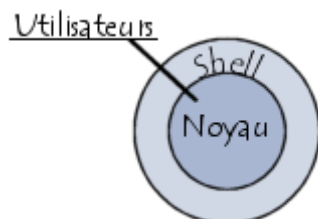


Introduction au shell

L'interpréteur de commandes est l'interface entre l'utilisateur et le système d'exploitation, d'où son nom anglais «**shell**», qui signifie «coquille».



Le shell est ainsi chargé de faire l'intermédiaire le système d'exploitation et l'utilisateur grâce aux lignes de commandes saisies par ce dernier. Son rôle consiste ainsi à lire la ligne de commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

Le shell est ainsi un fichier exécutable chargé d'interpréter les commandes, de les transmettre au système et de retourner le résultat. Il existe plusieurs shells, les plus courants étant **sh** (appelé «*Bourne shell*»), **bash** («*Bourne again shell*»), **csh** («*C Shell*»), **Tcsh** («*Tenex C shell*»), **ksh** («*Korn shell*») et **zsh** («*Zero shell*»). Leur nom correspond généralement au nom de l'exécutable.

Chaque utilisateur possède un shell par défaut, qui sera lancé à l'ouverture d'une invite de commande. Le shell par défaut est précisé dans le fichier de configuration `/etc/passwd` dans le dernier champ de la ligne correspondant à l'utilisateur. Il est possible de changer de shell dans une session en exécutant tout simplement le fichier exécutable correspondant, par exemple :

```
/bin/bash
```

Qu'est-ce que le shell Bash et pourquoi est-il si important pour Linux ?

Si vous êtes un utilisateur de [Linux](#), vous avez sans doute entendu parler du [shell](#) Bash. Mais de quoi s'agit-il exactement ? Le [shell](#) Bash également connu sous le nom de "bin bash", est une interface de ligne de commande qui vous permet d'interagir avec votre ordinateur. Grâce à lui, vous pouvez exécuter des programmes, naviguer dans des répertoires et effectuer toutes sortes d'autres tâches.

L'interpréteur de commandes Bash porte le nom de son créateur, Brian Fox. Il a été publié pour la première fois en 1989 et est devenu depuis le [shell](#) le plus populaire pour [Linux](#). En fait, il est si populaire qu'il est également disponible sur d'autres systèmes d'exploitation, tels que macOS et Windows.

Sur [Linux](#), l'interpréteur de commandes Bash peut vous sembler intimidant. Mais ne vous inquiétez pas, ce n'est pas aussi difficile que cela en a l'air. Dans cet article, nous allons vous aborder les bases du [shell](#) Bash et vous montrer comment l'utiliser ! À la fin de cet article, vous devriez avoir une meilleure compréhension de ce qu'il est et de la façon de l'utiliser.

1. Qu'est-ce que le shell Bash et que fait-il ?

Le [shell](#) Bash également connu sous le nom de "bin bash" est une interface de ligne de commande qui vous permet d'interagir avec votre ordinateur. Grâce à lui, vous pouvez exécuter des programmes, naviguer dans des répertoires et effectuer toutes sortes d'autres tâches. Le [shell](#) Bash doit son nom à son créateur, Brian Fox. Il a été publié pour la première fois en 1989 et est devenu depuis le [shell](#) le plus populaire pour [Linux](#). En fait, il est si populaire qu'il est également disponible sur d'autres systèmes d'exploitation, tels que macOS et Windows. Le [shell](#) Bash prend en charge de nombreux langages de programmation, notamment les scripts bash.

Sous [Linux](#), le [shell](#) Bash peut sembler intimidant. Mais ne vous inquiétez pas, il n'est pas aussi difficile qu'il n'y paraît. Dans cet article, nous allons couvrir les bases du [shell](#) Bash et vous montrer comment l'utiliser ! À la fin de l'article vous devriez mieux comprendre de quoi il s'agit et comment l'utiliser.

Nous vous invitons à regarder avant de commencer une vidéo, qui démystifie 3 idées reçues sur [Linux](#) !

2. Comment utiliser le shell Bash pour exécuter des commandes et des scripts sur votre ordinateur ?

La première chose que vous devez savoir est que le [shell](#) Bash (bin bash) est une interface de ligne de commande. Cela signifie que vous pouvez interagir avec votre ordinateur en tapant des commandes dans le [terminal](#). La bonne nouvelle est qu'il existe de nombreuses ressources disponibles pour vous aider à apprendre à utiliser le [shell](#) Bash

Pour exécuter une commande dans le [shell](#) Bash (ou Bourne Again [Shell](#)), utilisé dans un [shell](#) Unix, il suffit de la saisir dans le champ et d'appuyer sur la touche Entrée. Par exemple, pour savoir dans quel répertoire vous vous trouvez actuellement, vous pouvez taper la commande "pwd" et appuyer sur Entrée. Cela affichera à l'écran le répertoire de travail actuel. Le [shell](#) Bash est également compatible avec de nombreuses autres commandes [Linux](#), offrant ainsi une flexibilité et une puissance d'utilisation étendues. Il prend en charge de nombreux langages de programmation, notamment les programmes bash dans le cadre du projet GNU.

Si vous voulez exécuter un script, vous pouvez le faire en tapant "./scriptname.sh". Ceci exécutera le script dans le répertoire courant. Vous pouvez également spécifier le chemin d'accès complet au script, par exemple "/home/user/scriptname.sh".

Il existe de nombreuses autres commandes que vous pouvez utiliser dans le [shell](#) Bash. Mais il ne s'agit là que de quelques-unes des plus fondamentales.

3. Quelles sont les caractéristiques et les avantages du shell Bash par rapport à d'autres shells comme PowerShell ou tcsh ?

Voici quelques-unes des caractéristiques et des avantages du [shell](#) Bash

Il est facile à utiliser et à apprendre

Il est portable, ce qui signifie qu'il peut être utilisé sur de nombreux systèmes d'exploitation différents.

Voilà, c'est fait ! Ce ne sont là que quelques-unes des bases du [shell](#) Bash. Si vous souhaitez en savoir plus, nous vous suggérons de consulter les ressources ci-dessous. Et si vous avez des questions, n'hésitez pas à nous les poser dans la section des commentaires !

Le [shell](#) Bash (abréviation de Bourne Again [Shell](#)) est une interface de ligne de commande qui permet aux utilisateurs d'interagir avec leur ordinateur. Il porte le nom de son créateur, Brian Fox. Le [shell](#) Bash a été publié pour la première fois en 1989 et est depuis devenu le [shell](#) le plus populaire utilisé sous [Linux](#). Cependant, il est également disponible sur d'autres

systèmes d'exploitation tels que macOS et Windows. Dans cet article, nous allons couvrir les bases du [shell](#) Bash et comment l'utiliser ! Par à la fin de l'article, vous devriez mieux comprendre ce qu'il est et comment l'utiliser. Par exemple la commande echo peut être personnalisé pour afficher des messages spécifiques selon vos besoins.

- Offre de bonnes performances.
- Prend en charge de nombreux langages de programmation différents.
- Dispose d'une grande communauté d'utilisateurs et de [développeurs](#) pour l'aide en cas de problèmes.
- Est constamment mis à jour avec de nouvelles fonctionnalités et améliorations.

Le [shell](#) Bash (qui est un interpréteur de commandes) est facile à utiliser et à apprendre. Il est également portable, ce qui signifie qu'il peut être utilisé sur de nombreux systèmes d'exploitation différents. En outre, le [shell](#) Bash offre de bonnes performances et prend en charge de nombreux langages de programmation différents. Par exemple, la commande echo peut être utilisée pour afficher des messages personnalisés. le [shell](#) Bash dispose d'une grande communauté d'utilisateurs et de [développeurs](#) qui peuvent vous aider si vous rencontrez des problèmes. Enfin, le [shell](#) Bash est constamment mis à jour avec de nouvelles fonctionnalités et des améliorations.

Exemple de Script pour renommer des fichiers en ligne de commande de façon récursive

```
#!/bin/bash

# Définit le format de la date
DATE=$(date +%Y-%m-%d)

# Fonction pour renommer les dossiers
rename_folders() {
    for d in "$1"/*/ ; do
        if [ -d "$d" ]; then
            # Récursivité pour les sous-dossiers
            rename_folders "$d"
            # Renomme le dossier
            mv "$d" "${d%/_}$DATE"
        fi
    done
}

# Chemin du répertoire à traiter
DIRECTORY="/chemin/vers/le/répertoire"

# Appel de la fonction avec le répertoire cible
rename_folders "$DIRECTORY"
```

4. Comment pouvez-vous personnaliser le shell Bash pour qu'il fonctionne mieux pour vous ?

Il existe de nombreuses façons de personnaliser le [shell](#) Bash. Par exemple, vous pouvez modifier le jeu de couleurs, les raccourcis clavier ou les [applications](#) par défaut. Vous pouvez également installer des logiciels supplémentaires dans le répertoire "usr bin" pour ajouter de nouvelles fonctionnalités ou améliorer les performances. Et si vous êtes un [développeur](#), vous pouvez même créer vos propres commandes personnalisées !

Pour en savoir plus sur la manière de personnaliser le Bash:

- Comment changer le schéma de couleurs dans Bash ?
- Comment modifier les raccourcis clavier dans Bash ?
- Comment installer des logiciels supplémentaires dans Bash ?
- Comment créer des commandes personnalisées dans Bash ?

La personnalisation de l'interpréteur de commandes Bash est un excellent moyen d'améliorer son fonctionnement. En modifiant le jeu de couleurs, les raccourcis clavier ou les [applications](#) par défaut, vous pouvez rendre le [shell](#) Bash plus convivial. En outre, l'installation de logiciels supplémentaires, utilisant la commande "echo", peut ajouter de nouvelles fonctionnalités ou améliorer les performances. Et si vous êtes un [développeur](#), vous pouvez même créer vos propres commandes personnalisées !

5. Existe-t-il des alternatives au shell Bash que vous devriez envisager d'utiliser à la place ?

Le [shell](#) Bash n'est pas la seule interface de ligne de commande disponible. Il existe d'autres options telles que le [shell](#) Z (zsh) et le [shell](#) fish. Chacun de ces [shells](#) a ses propres avantages et inconvénients. Vous devriez essayer chacun d'entre eux pour voir lequel fonctionne le mieux pour vous.

En savoir plus sur les coquilles alternatives :

- Qu'est-ce que la coquille Z ?
- Qu'est-ce que la coquille de poisson ?
- Comment choisir le bon coquillage pour vous ?

Il existe quelques alternatives à la coquille Bash que vous pouvez envisager. Le [shell](#) Z (zsh) et le [shell](#) fish sont deux options populaires. Chacun d'eux

En conclusion, le [shell](#) Bash est une interface de ligne de commande puissante et polyvalente qui peut être utilisée sur de nombreux systèmes d'exploitation différents. Il est facile à utiliser et à personnaliser, et il offre de bonnes performances et le support de nombreux langages de programmation. Si vous recherchez une alternative à l'interpréteur de commandes Bash, vous devriez envisager l'interpréteur de commandes Z (zsh) ou l'interpréteur de commandes. C'est tout pour cet article ! Nous espérons qu'il vous a été utile. Si vous avez des questions ou des commentaires, n'hésitez pas à les laisser ci-dessous !

Quelle est la différence entre useradd et adduser ?

Linux est livré avec de nombreuses commandes Terminal, chacune ayant son propre objectif. Certains d'entre eux remplissent la même fonction mais procèdent de différentes manières lors de leur exécution. C'est le cas de **adduser** et **useradd**. Les deux sont utilisés pour créer un nouvel utilisateur mais suivent différentes manières de l'exécuter. Cet article est destiné à

informer le lecteur sur les principales différences entre les deux commandes, avec des exemples sur comment et quand les utiliser.

Pourquoi utiliser adduser et useradd ?

Pour expliquer pourquoi nous utilisons adduser et useradd, nous devons d'abord comprendre ce que sont les utilisateurs et les groupes sous Linux.

Le terme Utilisateur désigne un être ou une unité responsable de l'édition, de la gestion et de la manipulation de fichiers et d'opérations.

Un groupe fait référence à une collection d'utilisateurs auxquels des autorisations spéciales sont accordées. Nous pouvons dire qu'un utilisateur est analogue à un compte et qu'un groupe est une classe de comptes avec des autorisations similaires.

Les commandes adduser et useradd sont utilisées pour créer de tels utilisateurs. La principale différence est que adduser configure facilement les dossiers, répertoires et autres fonctions nécessaires des utilisateurs, tandis que useradd crée un nouvel utilisateur sans ajouter les répertoires mentionnés ci-dessus et les paramètres.

La commande adduser

La commande adduser crée un nouvel utilisateur et des informations supplémentaires sur l'utilisateur, les répertoires et un mot de passe. Selon les options de la ligne de commande et les paramètres donnés, des éléments supplémentaires peuvent être ajoutés. Sa syntaxe est donnée ci-dessous :

```
$ adduser -- options arguments
```

Par exemple:

```
$ adduser --help Displays a help window with a list of possible commands
```

```
ubuntu@ubuntu:~$ adduser --help
adduser [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID]
[--firstuid ID] [--lastuid ID] [--gecos GECOS] [--ingroup GROUP | --gid ID]
[--disabled-password] [--disabled-login] [--add_extra_groups]
[--encrypt-home] USER
    Add a normal user

adduser --system [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID]
[--gecos GECOS] [--group | --ingroup GROUP | --gid ID] [--disabled-password]
[--disabled-login] [--add_extra_groups] USER
    Add a system user

adduser --group [--gid ID] GROUP
addgroup [--gid ID] GROUP
    Add a user group

addgroup --system [--gid ID] GROUP
    Add a system group

adduser USER GROUP
    Add an existing user to an existing group

general options:
  --quiet | -q          don't give process information to stdout
  --force-badname       allow usernames which do not match the
                        NAME_REGEX[_SYSTEM] configuration variable
  --extrausers          uses extra users as the database
  --help | -h          usage message
  --version | -v        version number and copyright
  --conf | -c FILE     use FILE as configuration file

ubuntu@ubuntu:~$
```

Vous avez besoin d'autorisations spéciales pour créer un utilisateur, c'est-à-dire que vous devez être un superutilisateur. Pour cela, nous utilisons la commande sudo. Entrez en tant que root en exécutant la commande ci-dessous.

\$ sudo -i


```
root@ubuntu: ~  
ubuntu@ubuntu:~$ sudo -i  
root@ubuntu:~# adduser user1  
Adding user `user1' ...  
Adding new group `user1' (1000) ...  
Adding new user `user1' (1000) with group `user1' ...  
Creating home directory `/home/user1' ...  
Copying files from `/etc/skel' ...  
New password:  
Retype new password:  
passwd: password updated successfully  
Changing the user information for user1  
Enter the new value, or press ENTER for the default  
    Full Name []: user1  
    Room Number []: 5  
    Work Phone []: 777  
    Home Phone []: 666  
    Other []: none  
Is the information correct? [Y/n] y  
root@ubuntu:~#
```

La commande useradd

La commande useradd est utilisée pour créer un nouvel utilisateur ou pour modifier l'utilisateur existant. Contrairement à adduser, cependant, il ne crée pas de répertoires spécifiés, sauf indication contraire. Useradd crée également un groupe par défaut. La syntaxe de useradd est la suivante :

\$ useradd [options]

Par exemple:

\$ useradd --help Displays a help window with a list of possible commands


```

ubuntu@ubuntu:~$ useradd --help
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
    --badnames           do not check for bad names
    -b, --base-dir BASE_DIR  base directory for the home directory of the
                           new account
    --btrfs-subvolume-home  use BTRFS subvolume for home directory
    -c, --comment COMMENT  GECOS field of the new account
    -d, --home-dir HOME_DIR  home directory of the new account
    -D, --defaults         print or change default useradd configuration
    -e, --expiredate EXPIRE_DATE  expiration date of the new account
    -f, --inactive INACTIVE  password inactivity period of the new account
    -g, --gid GROUP        name or ID of the primary group of the new
                           account
    -G, --groups GROUPS    list of supplementary groups of the new
                           account
    -h, --help             display this help message and exit
    -k, --skel SKEL_DIR    use this alternative skeleton directory
    -K, --key KEY=VALUE    override /etc/login.defs defaults
    -l, --no-log-init      do not add the user to the lastlog and
                           faillog databases
    -m, --create-home      create the user's home directory
    -M, --no-create-home   do not create the user's home directory
    -N, --no-user-group     do not create a group with the same name as
                           the user
    -o, --non-unique       allow to create users with duplicate
                           (non-unique) UID
    -p, --password PASSWORD  encrypted password of the new account
    -r, --system           create a system account
    -R, --root CHROOT_DIR  directory to chroot into
    -P, --prefix PREFIX_DIR  prefix directory where are located the /etc/*
    -s, --shell SHELL      login shell of the new account
    -u, --uid UID          user ID of the new account

```

Pour créer un nouvel utilisateur

```
$ useradd [options] [username]
```

Useradd nécessite des options ou des drapeaux pour fonctionner correctement. Certains drapeaux couramment utilisés sont donnés ci-dessous :

- -D, –valeurs par défaut ; Crée un nouvel utilisateur avec les valeurs par défaut/définit les valeurs utilisateur existantes par défaut
- -c, –commentaire ; Utilisé pour ajouter une chaîne de texte.
- -m ; Utilisé pour créer un répertoire personnel pour le nouvel utilisateur
- -G ; Ajoute un utilisateur à des groupes supplémentaires
- -g ; Affiche le nom du groupe ou le numéro de groupe (GID)
- -h, –aide ; Affiche toutes les commandes possibles

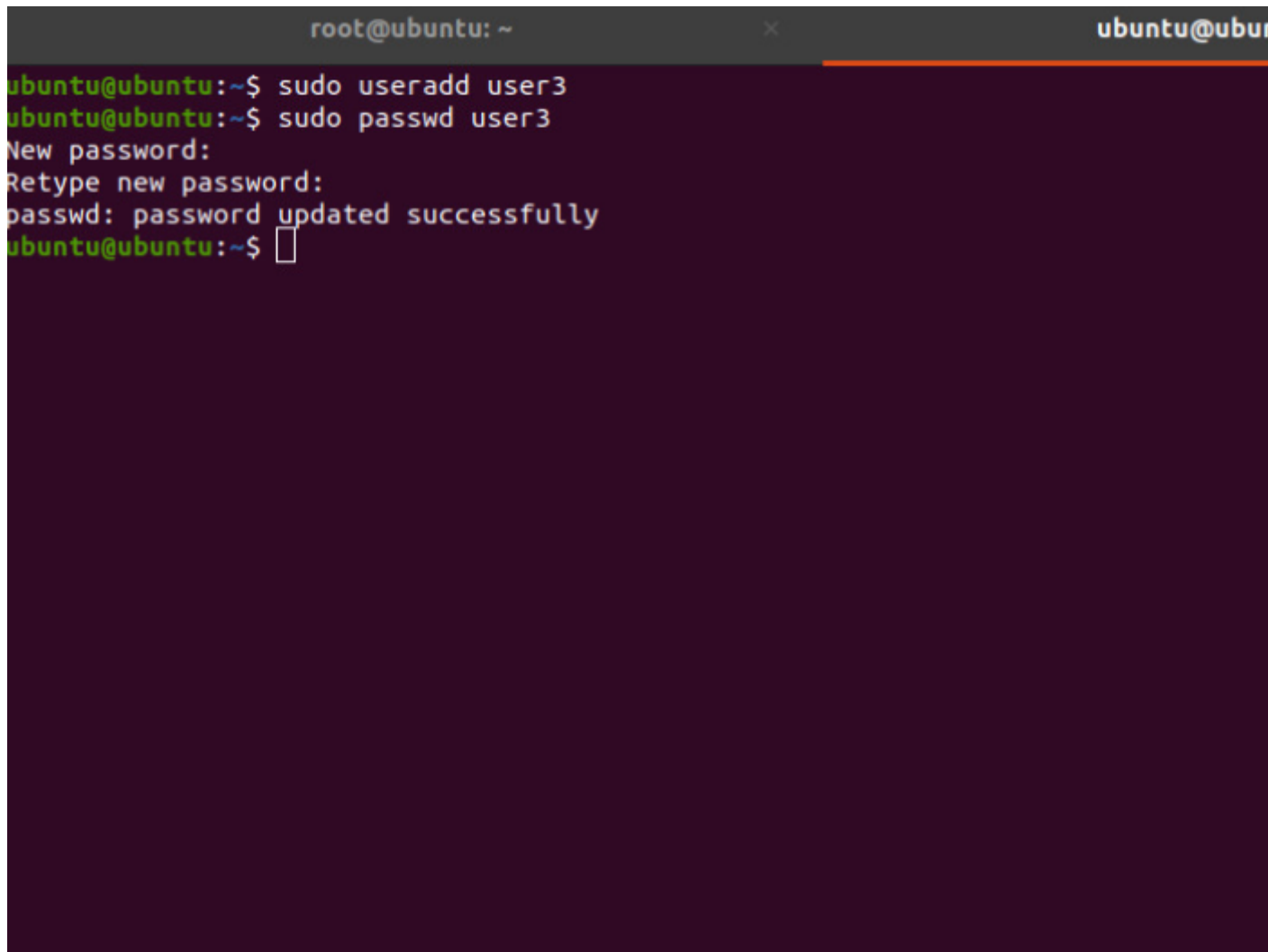
- `-e, --expire` ; Utilisé pour définir la date d'expiration de l'utilisateur (AAAA/MM/JJ)

Comme dans le cas de `adduser`, `useradd` nécessite également certaines autorisations pour créer un nouvel utilisateur. Par conséquent, nous utilisons la commande `sudo` sous la syntaxe suivante :

```
$ sudo useradd [options] [username]
```

Pour configurer un mot de passe pour le nouvel utilisateur, utilisez :

```
$ sudo passwd [username]
```

A terminal window with a dark purple background. The title bar shows 'root@ubuntu: ~' on the left and 'ubuntu@ubuntu' on the right. The terminal content shows the following commands and output:

```
ubuntu@ubuntu:~$ sudo useradd user3
ubuntu@ubuntu:~$ sudo passwd user3
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ubuntu:~$
```

Similarités entre `useradd` et `adduser`

- Les deux sont des commandes de terminal Linux
- Les deux sont utilisés pour créer de nouveaux utilisateurs

Différences entre useradd et adduser

Ce qui sépare adduser de useradd est la différence de procédure d'implémentation et d'exécution.

Useradd est une commande intégrée fournie avec toutes les distributions Linux. Adduser se présente sous la forme d'un lien symbolique ou d'un script Perl et n'est pas disponible avec certaines distributions Linux. La commande Adduser utilise useradd dans le backend.

Adduser est une commande utilitaire de haut niveau avec une syntaxe facile à comprendre. Il invite l'utilisateur à lui demander les informations nécessaires pour créer un profil complet. Lors de l'exécution, il guide l'utilisateur à travers un processus étape par étape pour s'assurer que tous les répertoires, groupes et autorisations sont définis en fonction des besoins.

Adduser configure automatiquement le répertoire utilisateur dans le dossier de départ.

D'autre part, useradd n'exécute que la commande qui lui est donnée en fonction de l'ensemble des drapeaux qui lui sont fournis, ce qui signifie qu'il créera un utilisateur sans demander d'informations supplémentaires (mots de passe, autorisations, etc.).

Cela implique de créer un utilisateur avec tous les répertoires et informations, et vous devez utiliser plusieurs drapeaux et options pour obtenir le même résultat que vous obtiendriez avec une seule commande adduser.

Useradd vs Adduser, lequel devez-vous utiliser ?

En regardant en arrière le fonctionnement des deux commandes, il est prudent de dire que adduser devrait être votre préférence lors de la création d'un nouvel utilisateur. La configuration des mots de passe, des répertoires et des groupes est plus propre et plus facile à comprendre. Dans la plupart des cas, vous devriez utiliser la commande adduser.

Cela ne signifie pas que useradd n'a aucun but. Il offre plus de flexibilité lors de la création d'un utilisateur. Par exemple, si vous devez créer un utilisateur temporaire et que vous ne souhaitez pas allouer de ressources supplémentaires aux répertoires personnels, groupes, etc., vous pouvez utiliser la commande useradd.

Useradd est également plus flexible en ce qui concerne la mise en œuvre de groupe. Vous pouvez ajouter l'utilisateur à plusieurs groupes à l'aide de l'option -G. Le même processus nécessiterait plusieurs instructions de la part de adduser.

Étant une commande utilitaire de bas niveau, useradd assurerait une portabilité maximale sur toutes les distributions Linux.

Si vous souhaitez créer des utilisateurs sans vous soucier de l'allocation des ressources, adduser est la solution. Cependant, supposons que vous cherchiez à avoir plus de contrôle sur les répertoires et les informations avec lesquels vous souhaitez travailler sans vous soucier de la portabilité. Dans ce cas, useradd est la commande qu'il vous faut.

Conclusion

`adduser` et `useradd` ont le même objectif, c'est-à-dire créer un nouvel utilisateur. L'utilisation varie en fonction des besoins de l'utilisateur. Nous espérons que ce guide vous a aidé à comprendre les différences entre les deux, affinant ainsi votre compréhension des commandes Linux essentielles.
