Ubuntu: Installation à partir d'un système linux

Cette section explique comment installer Ubuntu à partir d'un système Unix ou Linux existant, sans utiliser le programme d'installation piloté par menu. Dans cette section, une certaine familiarité avec la saisie de commandes *nix et la navigation dans le système de fichiers est supposée. Dans cette section, $ symbolise une commande à entrer dans le système actuel de l'utilisateur, tandis que # fait référence à une commande entrée dans le chroot Ubuntu.

Une fois que le nouveau système Ubuntu a été configuré, on peut y migrer less données utilisateur existantes (le cas échéant) et continuer ainsi. Il s'agit donc d'une installation Ubuntu à «zéro temps d'arrêt». C'est également un moyen astucieux de gérer du matériel qui ne fonctionne pas correctement avec différents supports de démarrage ou d'installation.

Comme il s'agit d'une procédure essentiellement manuelle, il faudra effectuer une grande partie de la configuration de base du système, ce qui nécessitera également une meilleure connaissance de Ubuntu et de Linux en général qu'une installation régulière. On ne peut pas s'attendre à ce que cette procédure aboutisse à un système identique à un système issu d'une installation standard. Cette procédure ne donne que les étapes de base pour configurer un système. Des étapes supplémentaires d'installation et/ou de configuration peuvent être nécessaires. En général, cette méthode d'installation n'est pas recommandée aux utilisateurs occasionnels ou débutants.

Préparation du système de fichiers

Avec les outils de partitionnement *nix actuels, repartitionner le disque dur selon les besoins, en créant au moins un système de fichiers et un swap. Il faut environ 506 Mo d'espace disponible pour une installation sur console uniquement.

Ensuite, créez des systèmes de fichiers sur les partitions. Par exemple, pour créer un système de fichiers ext3 sur la partition /dev/sda6 (c'est notre exemple de partition racine):

```
# mke2fs -j /dev/sda6
```

Pour créer un système de fichiers ext2 à la place, omettre -j.

Initialiser et activer le swap (remplacez le numéro de partition par votre partition de swap Ubuntu prévue):

```
# mkswap/dev/sda5
# sync
# swapon/dev/sda5
```

Au lieu d'utiliser une partition swap dédiée, on peut omettre la configuration de la partition swap ici et plus tard, en utilisant simplement un fichier swap.

Monter la partition ainsi créée en tant que `/mnt/ubuntu` (le point d'installation, qui sera le système de fichiers racine (/) sur le nouveau système). Le nom du point de montage est strictement arbitraire, il est référencé plus tard.

```
# mkdir /mnt/ubuntu
# mount /dev/sda6/mnt/ubuntu
```

On peut monter des parties du système de fichiers (par exemple,/usr) sur des partitions distinctes, pour cela il faudra créer et monter ces répertoires manuellement avant de passer à l'étape suivante.

Installer debootstrap

L'utilitaire utilisé par le programme d'installation Ubuntu et reconnu comme le moyen officiel d'installer un système de base Ubuntu est **debootstrap**. Il utilise **wget** et **ar**, mais ne dépend sinon que de `/bin/sh` et des outils de base Unix/Linux. Installer **wget** et **ar** s'ils ne sont pas déjà sur le système actuel, puis télécharger et installer **debootstrap**. Si ces étapes sont exécutées sous Ubuntu, on peut le faire simplement en installant **debootstrap**.

Dan un système basé sur RPM (Red Hat Package Manager), on peut utiliser alien, disponible dans les référentiels Debian, pour convertir le fichier .deb en un fichier .rpm utilisable.

Ou, on peut utiliser la procédure suivante pour l'installer manuellement. Créer un dossier de travail pour extraire le .deb dans:

```
# mkdir work
# cd work
```

Le binaire **debootstrap** se trouve dans l'archive Ubuntu (sélectionner le fichier approprié pour l'architecture). Télécharger le fichier **debootstrap.deb** à partir du pool, copier le package dans le dossier `work` et extraire les fichiers de celui-ci. Il faut disposer des privilèges root pour installer les fichiers.

```
# ar -x debootstrap_0.X.X_all.deb
# cd /
# zcat /full-path-to-work/work/data.tar.gz | tar xv
```

Lancer debootstrap

debootstrap peut télécharger les fichiers nécessaires directement à partir de l'archive lorsqu'on l'exécute. On peut remplacer ports.ubuntu.com/ubuntu-ports par n'importe quel miroir d'archive Ubuntu dans l'exemple de commande ci-dessous, on utilise de préférence un miroir proche du réseau. Les miroirs sont répertoriés à l'adresse `http://wiki.ubuntu.com/Archive.`

Si un CD bionique Ubuntu monté sur /cdrom, on peut remplacer l'URL `http:` du fichier par l'URL `file://cdrom/ubuntu/`

Dans la commande **debootstrap**, remplacer **ARCH** par l'un des éléments suivants: **amd64**, **arm64**, **armhf**, **i386**, **powerpc**, **ppc64el** ou **s390**x.

```
# /usr/sbin/debootstrap --arch ARCH bionic/mnt/ubuntu
```

Configurer le système de base

On a maintenant un vrai système Ubuntu, bien que maigre, sur disque. chrooteé dedans:

```
# LANG=C.UTF-8 chroot /mnt/ubuntu /bin/bash
```

Après avoir chrooté, on peut définir un terminal pour qu'il soit compatible avec le système de base Ubuntu, par exemple:

```
# export TERM=xterm-color
```

En fonction de la valeur de TERM, il faudra peut-être installer le package **ncurses-term** pour obtenir de l'aide.

Si des avertissements se produisent comme:

```
ash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
```

Les fichiers de localisation requis doivent être générés:

```
# sudo locale-gen en_US.UTF-8
```

Configurer Apt

Debootstrap aura créé un fichier `/etc/apt/sources.list` très basique qui permettra d'installer des paquets supplémentaires. Toutefois, il est suggéré d'ajouter des sources supplémentaires, par

exemple pour les packages source et les mises à jour de sécurité:

```
deb-src http://archive.ubuntu.com/ubuntu bionic main

deb http://security.ubuntu.com/ubuntu bionic-security main
deb-src http://security.ubuntu.com/ubuntu bionic-security main
```

Exécuter apt update après avoir modifié la liste des sources.

Installer des paquets supplémentaires

Maintenant, il est nécessaire d'installer certains paquets supplémentaires, tels que **makedev** (nécessaire pour la section suivante):

```
apt install makedev
```

Lorsqu'on utilise l'architecture s390x installer le paquet obligatoire **s390-tools**:

```
apt install s390-tools
```

Créer des fichiers de périphérique

À ce stade, `/dev/` ne contient que des fichiers de périphérique très élémentaires. Pour les prochaines étapes de l'installation, des fichiers de périphérique supplémentaires peuvent être nécessaires. Il existe différentes façons de procéder. La méthode à utiliser dépend du système hôte utilisé pour l'installation, de l'intention d'utiliser un noyau modulaire ou non, et de l'intention d'utiliser **Dynamic** (par exemple, avec udev).) ou des fichiers de périphérique statiques pour le nouveau système.

Quelques unes des options disponibles sont:

- créer un ensemble par défaut à l'aide de fichiers de périphériques statiques (après chrooter)

```
# mount none /proc -t proc
# cd /dev
# MAKEDEV generic
```

ou selon l'architecture spécifique:

```
# MAKEDEV std
# cd ..
```

- créer manuellement uniquement des fichiers de périphérique spécifiques à l'aide de MAKEDEV

Sur s390x, les périphériques DASD doivent être créés comme suit:

```
# cd /dev
# MAKEDEV dasd
# cd ..
```

- lier `mount/dev` à partir du système hôte sur le `/dev` dans le système cible, par exemple:

```
mount --bind dev/dev
```

Les scripts **postinst** de certains paquetages peuvent essayer de créer des fichiers de périphérique. Cette option ne doit donc être utilisée qu'avec précaution.

- monter proc et sysfs

On peut monter les systèmes de fichiers proc et sysfs plusieurs fois et à des emplacements arbitraires, bien que `/proc` et `/sys` soient respectivement usuels.

```
# mount -t proc proc /proc
# mount -t sysfs sysfs /sys
```

La commande ls/proc doit maintenant afficher un répertoire non vide. Si cela échoue, il faudra peut-être monter proc de l'extérieur du chroot:

```
# mount -t proc proc/mnt/ubuntu/proc
```

Disques DASD et partitions

Même si lsdasd répertorie déjà les périphériques DASD:

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
==
0.0.1601   active      dasda     94:0    ECKD  4096   7043MB    1803060
0.0.260a   active      dasdb     94:4    ECKD  4096   7043MB    1803060
```

... ainsi que d'autres périphériques CCW tels que DASD, FCP ou QETH, ne peuvent pas encore être utilisés complètement et de manière persistante.

```
# lszdev --online | head -n 1 && lszdev --online | grep dasd-eckd
TYPE       ID                                              ON   PERS  NAMES
dasd-eckd  0.0.0123                                        yes  no    dasda
dasd-eckd  0.0.1234                                        yes  no    dasdb
```

Ici, DASD 1234 est celui utilisé pour debootstrap. Maintenant, activer ce DASD de manière persistante à l'aide de l'outil habituel `chzdev`:

```
# chzdev -e 1234
ECKD DASD 0.0.1234 configured
# lszdev --online 1234
TYPE       ID        ON   PERS  NAMES
dasd-eckd  0.0.1234  yes  yes   dasdb
```

Répéter les mêmes étapes pour d'autres périphériques CCW, tels que FCP, QETH ou d'autres périphériques DASD, si nécessaire.

Monter les partitions

créer /etc/fstab.

```
# vi /etc/fstab
```

Voici un exemple de fichier que l'on peut adapter :

```
# /etc/fstab: static file system information.
# hand-crafted
# file system    mount point   type    options                 dump pass
#
/dev/dasdb1      /             ext4    defaults                0    1

/swapfile        none          swap    sw                      0    0
```

Utiliser `mount -a` pour monter tous les systèmes de fichiers que spécifiés dans /etc/fstab ou, pour monter des systèmes de fichiers individuellement, utiliser:

```
# mount /path   # e.g.: mount /usr
```

Les systèmes Ubuntu actuels ont des points de montage pour les supports amovibles sous /media, mais conservent les liens symboliques de compatibilité dans /. Créez-les au besoin, par exemple:

```
# cd /media
# mkdir cdrom0
# ln -s cdrom0 cdrom
# cd /
```

```
# ln -s media/cdrom
```

La commande ls /proc doit maintenant afficher un répertoire non vide. Si cela échoue, il faudra monter proc de l'extérieur du chroot:

```
# mount -t proc proc /mnt/ubuntu/proc
```

Réglage du fuseau horaire

Le réglage de la troisième ligne du fichier `/etc/adjtime` sur “UTC” ou “LOCAL” détermine si le système interprétera l'horloge matérielle comme étant définie sur l'heure locale UTC respective. La commande suivante vous permet de définir cela.

```
# vi /etc/adjtime
```

Voici un exemple:

```
0.0 0 0.0
0
UTC
```

La commande suivante active le fuseau horaire.

```
# dpkg-reconfigure tzdata
```

Configurer le réseau

Pour configurer la mise en réseau, éditer `/etc/network/interfaces`, `/etc/resolv.conf`, `/etc/hostname` et `/etc/hosts`.

```
# vi /etc/network/interfaces
```

Voici quelques exemples simples extraits de `/usr/share/doc/ifupdown/examples`:

```
################################################## ####################
#/etc/network/interfaces - fichier de configuration pour ifup (8), ifdown
(8)
# Voir la page de manuel interfaces (5) pour plus d’informations sur les
options disponibles.
# disponible.
################################################## ####################
```

```
# Nous voulons toujours l'interface de bouclage.
#
auto lo
iface lo inet loopback

# Pour utiliser dhcp:
#
# auto eth0
# iface eth0 inet dhcp

# Exemple de configuration IP statique: (la diffusion et la passerelle sont
facultatifs)
#
# auto eth0
# iface eth0 inet static
#     address 192.168.0.42
#     network 192.168.0.0
#     netmask 255.255.255.0
#     broadcast 192.168.0.255
#     gateway 192.168.0.1
```

Entrer le serveur de noms et les directives de résolution dans `/etc/resolv.conf`:

```
# vi /etc/resolv.conf
```

Un exemple simple `/etc/resolv.conf`:

```
search hqdom.local
nameserver 10.1.1.36
nameserver 192.168.9.100
```

Entrer le nom d'hôte de votre système (2 à 63 caractères):

```
# echo UbuntuHostName > /etc/hostname
```

Exemple de fichier `/etc/hosts` de base avec support IPv6:

```
127.0.0.1 localhost
127.0.1.1 UbuntuHostName

# Les lignes suivantes sont souhaitables pour les hôtes compatibles IPv6
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Lorsqu'on a plusieurs cartes réseau, il faut organiser les noms des modules de pilotes dans le fichier `/etc/modules` dans l'ordre souhaité. Ensuite, au démarrage, chaque carte sera associée au nom d'interface (eth0, eth1, etc.) attendu.

Configurer les paramètres régionaux et le clavier

Pour configurer les paramètres régionaux de manière à utiliser une langue autre que l'anglais, installer les modules de langue appropriés et les configurer. Actuellement, l'utilisation des paramètres régionaux UTF-8 est recommandée.

```
# apt installer language-pack-de-language-pack-gnome-de
```

Pour configurer le clavier (si nécessaire):

```
# apt install console-setup
# dpkg-reconfigure keyboard-configuration
```

le clavier ne peut pas être configuré dans le chroot, mais sera configuré pour le prochain redémarrage.

Installer un noyau

Pour démarrer ce système, il faut installer un noyau Linux et un chargeur de démarrage. Identifier les noyaux pré-emballés disponibles avec:

```
# apt-cache search linux-image
```

Ensuite, installer le paquet de noyau chpoisi en utilisant son nom.

```
# apt install linux-image-arch-etc
```

(On peut également installer linux-image-generic.)

Configurer le chargeur d'amorçage

Pour que le système Ubuntu soit amorçable, configurer le chargeur de démarrage pour charger le

noyau installé depuis la nouvelle partition racine.

debootstrap n'installe pas de chargeur de démarrage, bien que l'on puisse utiliser apt dans le chroot Ubuntu pour le faire.

Étant donné que les noms de fichiers du noyau et **initrd** dans `/boot` sont versionnés, il est recommandé de les lier à des noms par défaut non versionnés, tels que:

```
# ln -s /boot/vmlinuz-4.15.0-23-generic /boot/vmlinuz
# ln -s /boot/initrd.img-4.15.0-23-generic /boot/initrd.img

# ls -la /boot/vmlinuz* /boot/initrd.img*
lrwxrwxrwx 1 root root       28 Jun 20 07:31 /boot/initrd.img -> initrd.img-
<version>-generic
-rw-r--r-- 1 root root 11245088 Jun 20 07:14 /boot/initrd.img-<version>-
generic
lrwxrwxrwx 1 root root       25 Jun 20 07:31 /boot/vmlinuz -> vmlinuz-
<version>-generic
-rw------- 1 root root  4390912 May 23 12:54 /boot/vmlinuz-<version>-generic
```

Le chargeur de démarrage «zipl» fait partie du package «s390-tools» installé précédemment. Consulter man zipl.conf pour des instructions sur la configuration du chargeur de démarrage. Créer une configuration zipl.conf à partir de rien ou la copier et modifier celle existante.

Voici un exemple de base de `/etc/zipl.conf`:

```
[defaultboot]
defaultmenu = menu

:menu
target = /boot
1 = ubuntu
default = 1

[ubuntu]
target = /boot
image = /boot/vmlinuz
ramdisk = /boot/initrd.img
parameters = root=/dev/dasdb1
```

Enfin, exécuter la commande `zipl` pour écrire la configuration sur le disque:

```
# zipl
Using config file '/etc/zipl.conf'
Building bootmap in '/boot'
Building menu 'menu'
```

```
Adding #1: IPL section 'ubuntu' (default)
Preparing boot device: dasdb (<your dasd device number>).
Done.
```

Accès à distance: Installation de SSH et configuration de l'accès

Si on peut se connecter au système via une console, on peut ignorer cette section. Si le système doit être accessible via le réseau ultérieurement, il faudra installer SSH et configurer l'accès.

```
# apt install openssh-server
```

La connexion root avec mot de passe est désactivée par défaut. Pour configurer l'accès il faut définir un mot de passe root

```
# passwd root
```

...et réactiver la connexion root avec mot de passe:

```
# vi /etc/ssh/sshd_config

PermitRootLogin yes
```

L'accès peut également être configuré en ajoutant une clé ssh au compte root:

```
# mkdir /root/.ssh
# cat << EOF> /root/.ssh/authorized_keys
ssh-rsa ....
EOF
```

Enfin, l'accès peut être configuré en ajoutant un utilisateur non root avec un mot de passe:

```
# adduser joe
# passwd joe
```

Finalisation

Comme mentionné précédemment, le système installé sera très basique. Pour rendre le système un peu plus mature, il existe une méthode simple pour installer tous les packages «standard»:

```
# apt install tasksel
```

```
# tasksel install standard
```

Bien sûr, on peut aussi simplement utiliser apt pour installer les paquets individuellement.

Après l'installation, il y aura beaucoup de packages téléchargés dans /var/cache/apt/archives /. On peutlibérer de l'espace disque en lançant:

```
# apt clean
```

Créer un utilisateur

Utiliser la commande `adduser` pour créer un nouveau compte utilisateur:

```
# adduser myusername
```

Puis entrer un nom complet et un mot de passe.

La configuration normale d'Ubuntu consiste à permettre à ce nouvel utilisateur d'administrer le système à l'aide de sudo. Pour le configurer, créer d'abord le groupe `admin` et y ajouter le nouvel utilisateur:

```
# addgroup --system admin
# adduser myusername admin
```

Utiliser la commande `visudo` pour ajouter ces lignes à la fin de `/etc/sudoers`, afin que tout utilisateur du groupe `admin` puisse administrer le système:

```
# Les membres du groupe admin peuvent obtenir les privilèges root
%admin ALL=(ALL) ALL
```