Introduction aux systèmes GNU/Linux

S19E03 inetdoc.net

UPSSITECH
ÉCOLE D'INGÉNIEURS

Philippe Latu / Université Toulouse 3

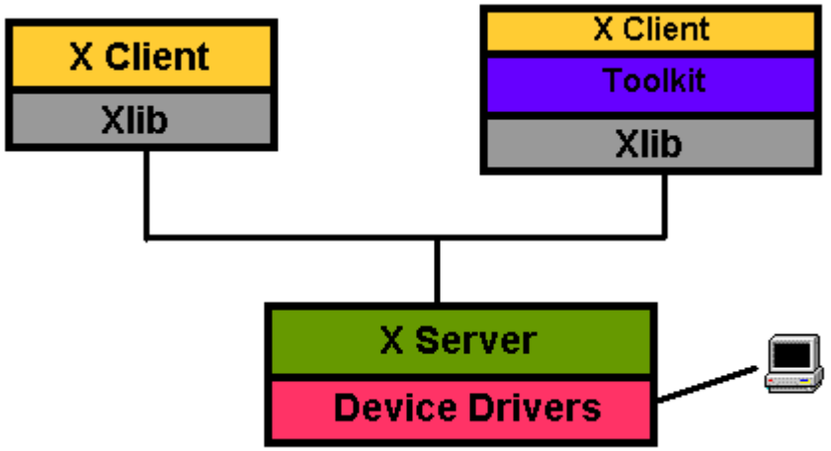
Plan séance 3

- Séance 3 – Environnements graphiques – gestion de paquets
  - Identifier les caractéristiques des serveurs graphiques
  - Utiliser l'environnement graphique KDE
  - Identifier les caractéristiques d'un gestionnaire de paquets
  - Utiliser les outils associés à l'Advanced Package Tool Debian
  - Gérer une collection de paquets

- Manipulations réalisables sur machines virtuelles
  - Duplication des jeux de paquets lors d'un clonage
  - Utilisation de différentes tâches avec tasksel ou aptitude
  - Tests sur les serveurs & environnements graphiques

Environnements graphiques

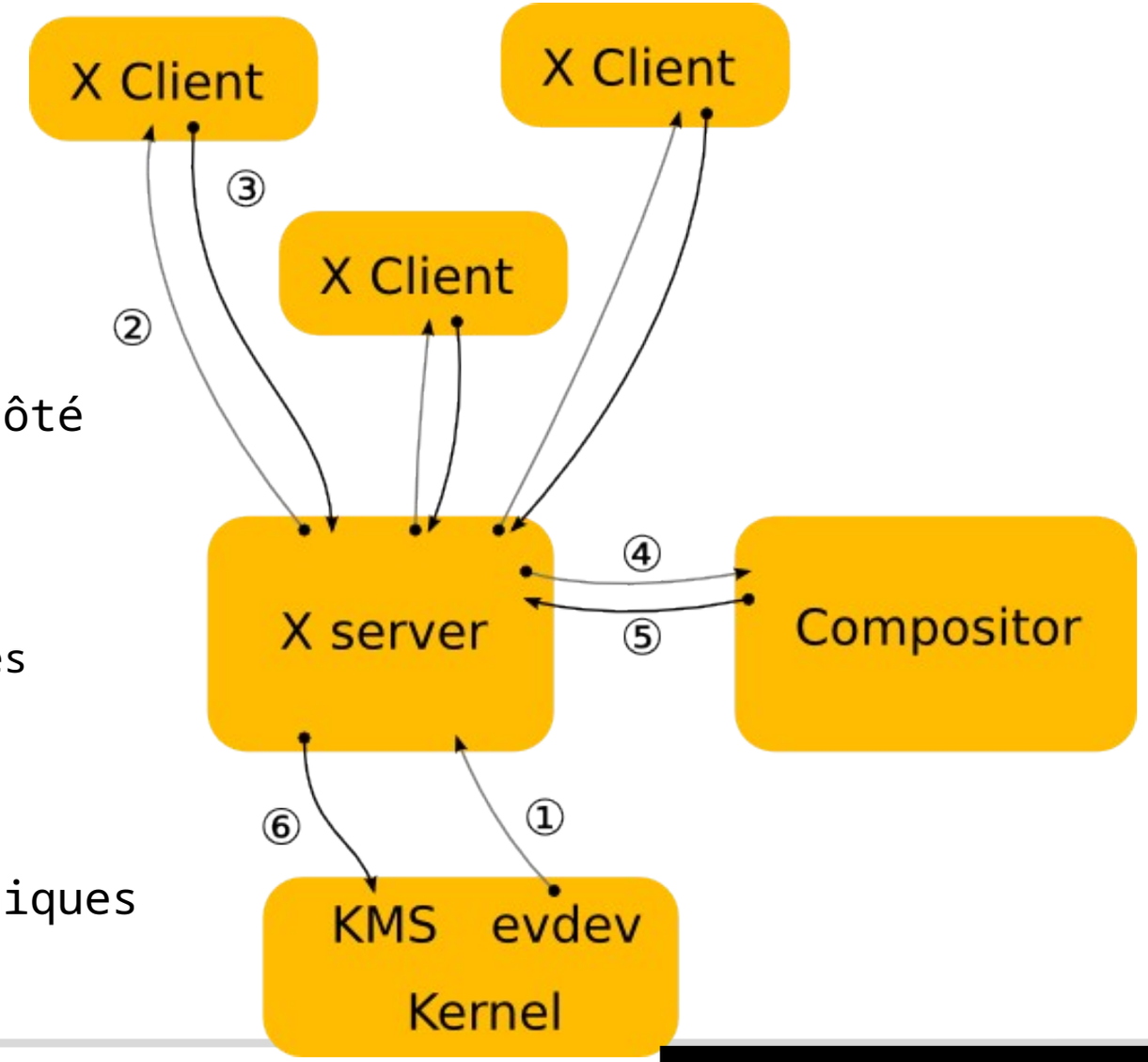
- Historique X.org
  - Consortium X-Window
    - Projet historique de développement des interfaces graphiques Unix
  - 1986 : première version diffusée par le MIT
  - 1992 : début du projet XFree86
    - http://www.xfree86.org
    - Initialement dédié aux processeurs Intel i386
    - Étendu au catalogue des processeurs du projet GNU
  - 2004 : lancement de la fondation X.Org
    - http://www.x.org
    - Architecture modulaire
  - 2008 : début du projet wayland
    - Architecture «allégée» pour les outils mobiles



X = Protocole réseau

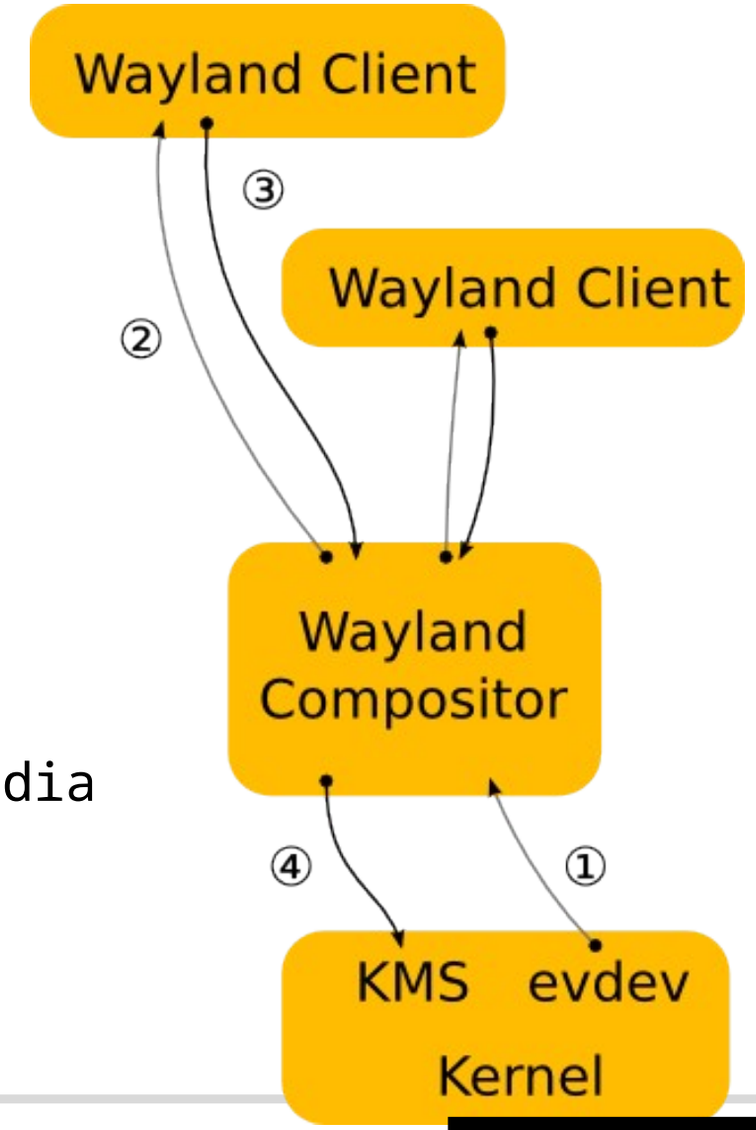
Environnements graphiques

- Architecture X.org
  - Source freedesktop.org
  - Architecture client/serveur
  - Serveur X = proxy
    - Tous les traitements ont lieu côté Compositor
  - Solution modulaire
    - ⊕ Ajout de fonctions supplémentaires
    - ⊖ Accumulation dans le temps
  - Fonctions réseau
    - Utilisation d'application graphiques à distance



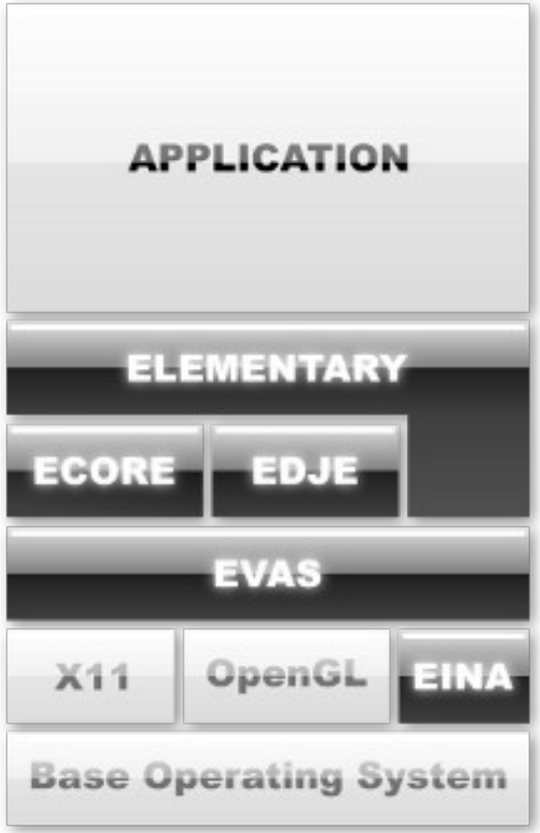
Environnements graphiques

- Architecture Wayland
  - Source freedesktop.org
  - Architecture «simplifiée»
    - Projet relativement jeune
  - Dispositifs mobiles
    - Solution adoptée par Ubuntu
    - Sponsors actifs : Qt, Intel
    - Pas encore de support matériel : NVidia



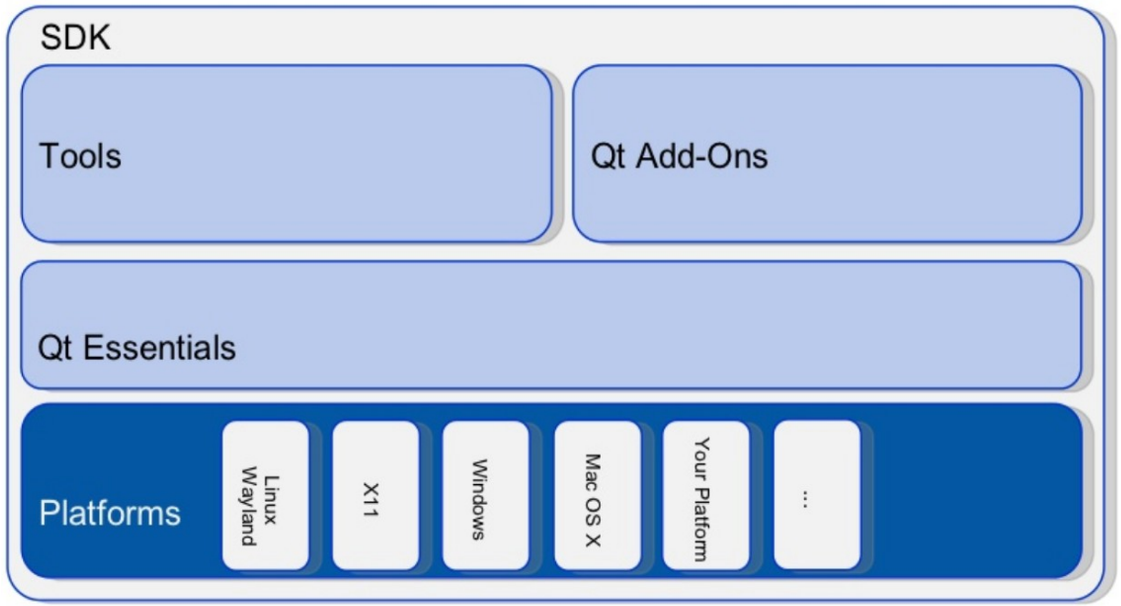
Environnements graphiques

- Processus de développement
  - Environnement graphique = chaîne de développement
  - Une architecture de base → Projets multiples
  - Modèle de développement OpenSource
    - ⊕ Introduction facile de nouvelles fonctionnalités
    - ⊖ Coordination difficile entre projets «concurrents»
  - Quelques exemples
    - Enlightenment
      - http://www.enlightenment.org/
      - Bibliothèques Enlightenment Foundation Libraries (EFL)
      - Dispositifs mobiles et faible puissance de calcul



Environnements graphiques

- Quelques exemples (suite)
  - LXDE & XFCE
    - Solutions orientées faible puissance de calcul
      - Systèmes embarqués, Live CD/DVD
      - Xubuntu : https://xubuntu.org/
  - Gnome & GTK
    - Environnement : http://www.gnome.org/
    - Bibliothèques : http://www.gtk.org/
    - RAD - glade : http://glade.gnome.org/
  - KDE & Qt
    - Environnement : http://www.kde.org/
    - Bibliothèques : http://qt.nokia.com/
    - RAD – qtcreator : http://qt-project.org/wiki/Category:Tools::QtCreator



Environnements graphiques

- Quelques exemples d'utilisation de KDE
  - Ouvrir le système d'aide
    - Rechercher les pages de manuels sur la commande 'find'
  - Ouvrir le centre d'information système
    - Consulter la liste des protocoles
    - Effectuer plusieurs tests de protocoles avec le gestionnaire de fichiers
    - Rechercher la liste des périphériques connectés sur le bus PCI
  - Ouvrir l'utilitaire de surveillance système
    - Consulter la table des processus en cours d'exécution

Gestion de paquets

- Distribution
  - **Noyau Linux + Shell(s) + collection de paquets**
- Gestionnaire de paquets
  - Construire un catalogue des paquets disponibles
  - Construire un arbre de dépendances
  - Interroger l'index
  - Consulter les propriétés
  - Télécharger & installer depuis un miroir ou dépôt
- Bibliothèques partagées
  - Fonctions logicielles partagées entre applications
  - Dépendance → relation entre applications et bibliothèques

Gestion de paquets

- Paquets **binaires**
  - Programmes compilés → exécution immédiate
  - Formats principaux : rpm et deb
  - Un paquet binaire par architecture supportée (i386|amd64)
  - Code exécutable générique par famille de processeurs
- Paquets **sources**
  - Programmes à compiler → temps d'installation plus long
  - Formats principaux : ports, emerge et deb-src
  - Code exécutable adapté au processeur cible

Gestion de paquets ou magasin d'applications ?

- Paquets
  - Intégration au niveau système
    - Taille réduite
    - Dépendances → autres paquets
  - Assurance qualité cohérente
    - Processus piloté au niveau distribution
  - Mises à jour contrôlées

- Snaps
  - Indépendant du système
    - Taille plus importante
    - Dépendances → dans chaque snap
  - Assurance qualité par snap
    - Responsable de publication
  - Mises à jour automatiques

- Fragmentation de l'écosystème versus plate-forme
  - Système d'exploitation
  - SDK développeur
  - Langage de conception
  - Magasin d'applications

4 conditions à satisfaire pour le développement d'une plate-forme autonome

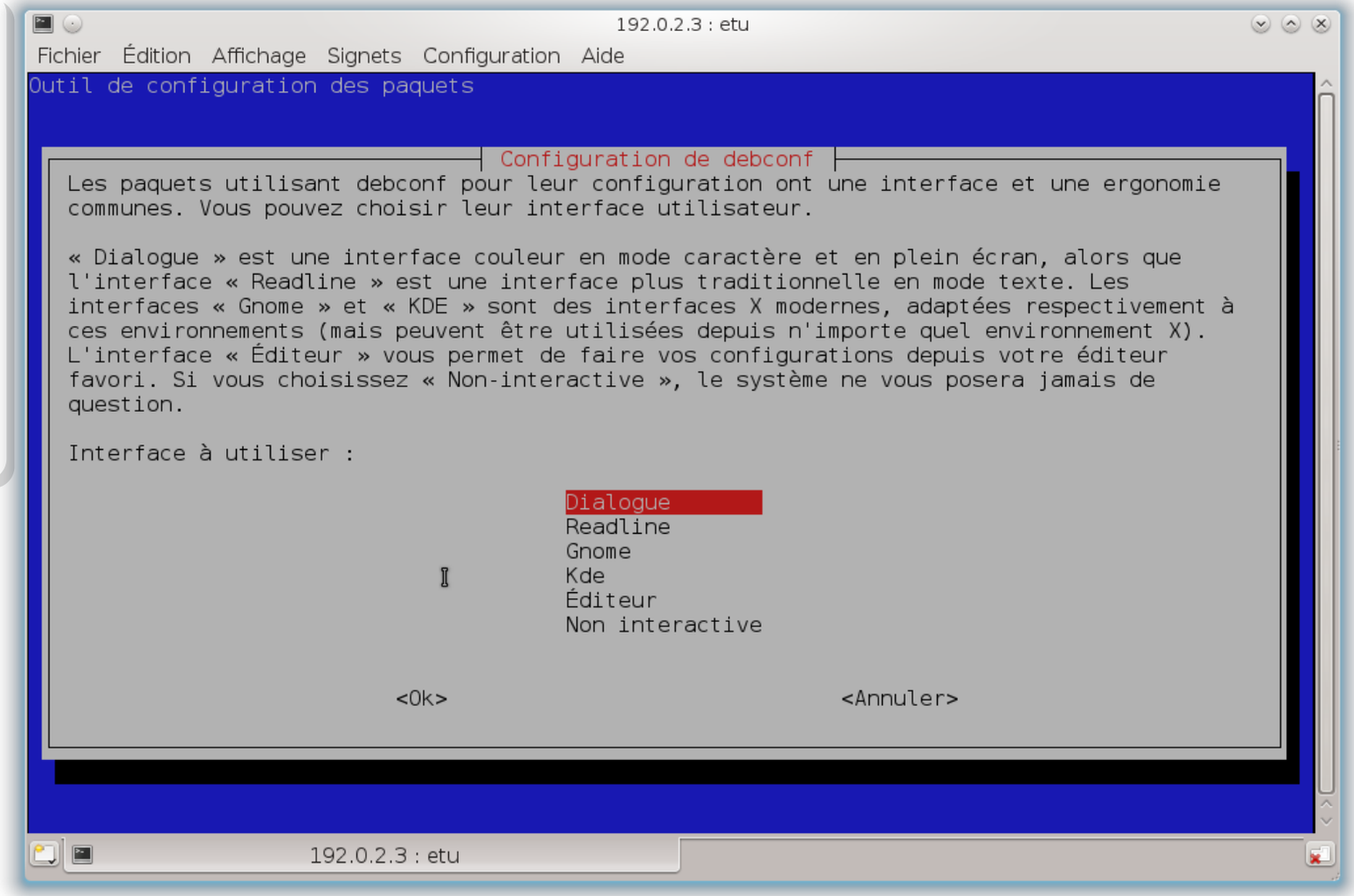
Advanced Package Tool

- APT : Advanced Package Tool
  - Gestion automatisée des dépendances depuis l'origine (1993)
  - APT → Bibliothèque C++ utilisée par différentes applications
  - Concepts progressivement appliqués aux autres systèmes
  - Mises à jour continues & incrémentales
    - Installation unique pour toute la durée de vie d'un système
    - Stratégie établie suivant les branches de la distribution
    - Configuration préservée entre les mises à jour
    - Interface de configuration standardisée → debconf

Advanced Package Tool

- Gestion de la configuration des paquets
- Plusieurs interfaces utilisateur disponibles
- Mémorisation des choix courants

```
# dpkg-reconfigure debconf
```



Advanced Package Tool

- Advanced Package Tool
  - Branches & catégories de paquets
  - 4 branches permanentes et indépendantes des versions principales
  - Branches
    - stable → paquets officiels
    - testing → paquets en attente d'intégration dans la version stable
    - unstable → paquets les plus récents en cours de test
    - experimental → paquets en cours de développement
  - Catégories
    - main → paquets conformes aux règles de définition du logiciel libre selon Debian
    - contrib → paquets de logiciels libres dépendant d'outils non libres
    - non-free → paquets avec des conditions de redistribution particulières

Advanced Package Tool

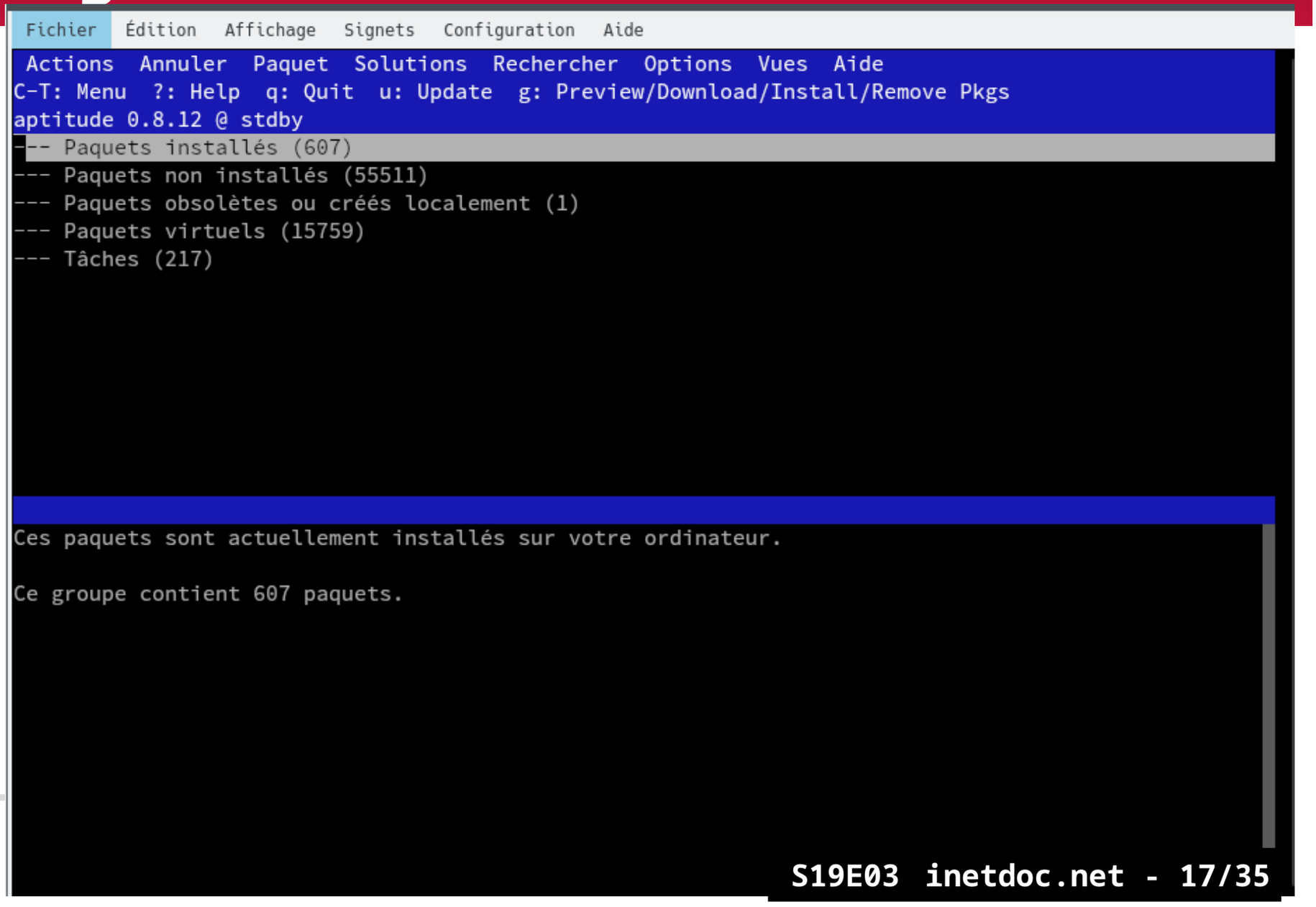
- Rythme des mises à jour suivant les branches
  - stable
    - Risque minimal → parfait pour les infrastructures critiques
    - Intégration des correctifs de sécurité sans évolution de version
    - Rythme d'évolution trop lent → http://www.backports.org/
  - testing
    - Risque moyen → convient bien pour les infrastructures de test
    - Intégration des correctifs de sécurité avec évolution de version
    - Rythme d'évolution satisfaisant pour couvrir tous les besoins
  - unstable
    - Risque élevé → «il faut assumer ses propres choix»
    - Pas de correctifs de sécurité
    - Rythme d'évolution le plus rapide

Advanced Package Tool

- Interfaces utilisateur de gestion des paquets
  - Console → aptitude
    - Utilisable dans tous les contextes
  - Ligne de commande → aptitude | apt | apt-get | apt-cache
    - Manipulations sur les paquets à partir du catalogue réseau
  - Ligne de commande → dpkg | dpkg-reconfigure | dpkg-buildpackage
    - Manipulations sur les paquets à partir du catalogue local

Advanced Package Tool

- aptitude
- 'u' → update
- 'U' → marquer les paquets à mettre à jour
- 'g' 2 fois → installer/enlever des paquets



Advanced Package Tool

- Catalogue & Arbre des dépendances
  - Fichier /etc/apt/sources.list

```
deb http://deb.debian.org/debian/ stable main contrib non-free
```

paquets binaires

adresse miroir

branche

catégories

```
deb-src http://ftp.fr.debian.org/debian/ stable main contrib non-free
```

paquets sources

```
deb http://security.debian.org/ stable/updates main
```

adresse miroir correctifs de sécurité

```
deb http://www.deb-multimedia.org stable main
```

adresse miroir non officiel

Advanced Package Tool

- Construction du catalogue
  - En mode console → `aptitude -u`
  - En ligne de commande → `aptitude update` | `apt update`

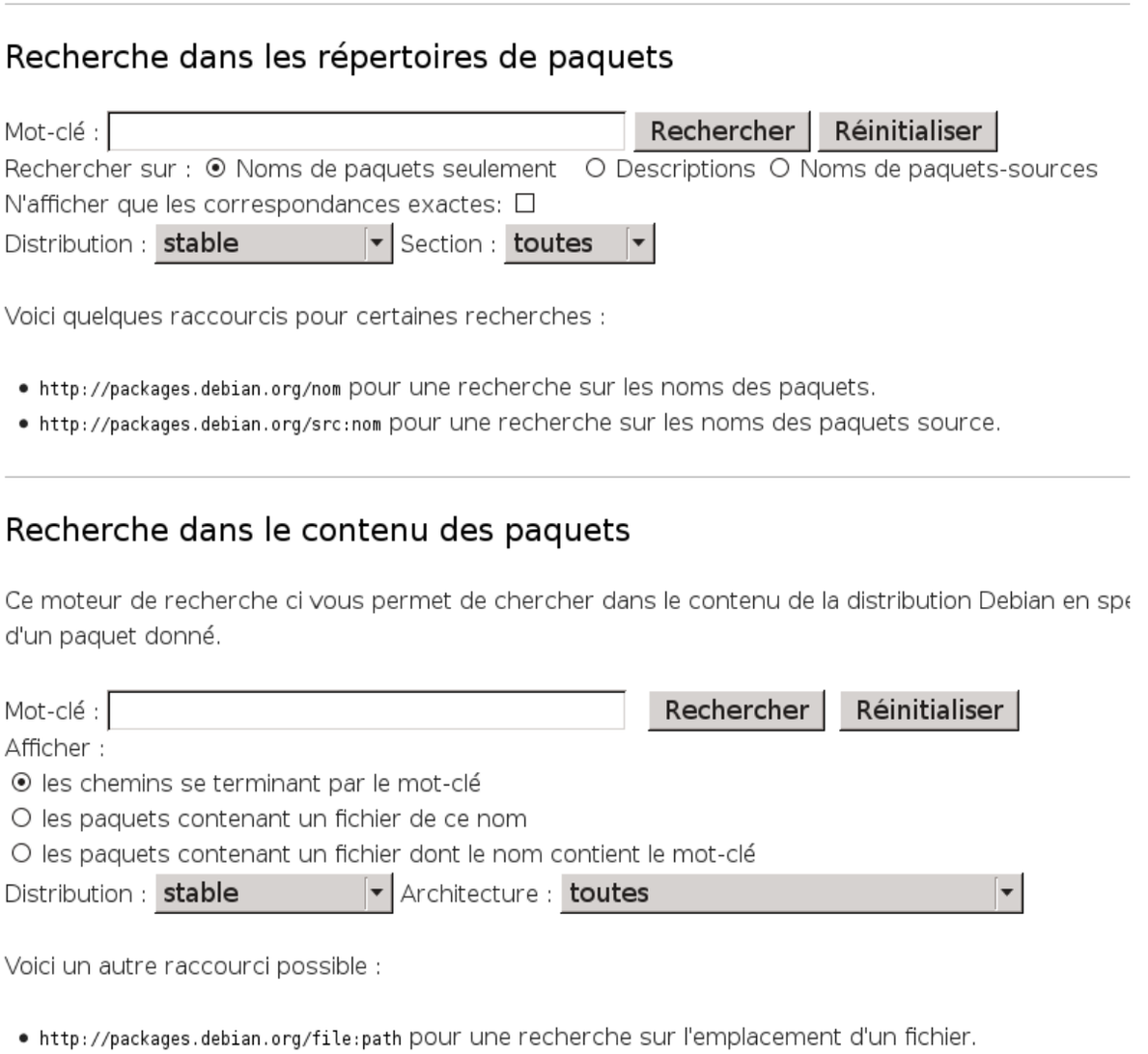
```
$ sudo apt update
Atteint :1 http://deb.debian.org/debian testing InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
```

Advanced Package Tool

- Mise à jour à l'échelle système
  - En ligne de commande → aptitude safe-upgrade | apt upgrade
    - Résolution complète des dépendances
    - Suppression de paquets installés interdite
  - En ligne de commande → aptitude full-upgrade | apt full-upgrade
    - Résolution complète des dépendances
    - Suppression de paquets installés en conflit avec un nouveau paquet possible
- Installation individuelle
  - En ligne de commande → aptitude install <paquet> | apt install
  - Proposition de solution en cas de conflit

Advanced Package Tool

- Recherches dans le catalogue des paquets
  - À partir du serveur web Debian
    - http://packages.debian.org
    - Recherche sur un nom de paquet
    - Recherche sur un nom de fichier appartenant à un paquet
    - Exemple : existe-t-il un paquet contenant le programme wireshark ?



Advanced Package Tool

- Rechercher dans le catalogue des paquets
  - En ligne de commande → `aptitude search` | `apt search`
  - Exemple : existe-t-il un paquet ayant pour nom wireshark ?

```
$ aptitude -w 80 search ^wireshark
i   wireshark                    - analyseur de trafic réseau – métapaquet
i   wireshark-common             - network traffic analyzer - common files
p   wireshark-dev                - analyseur de trafic réseau - outils de dév
...
```

Le nom débute par wireshark

- Rechercher uniquement parmi les paquets installés
  - En ligne de commande → `aptitude search ~i`

```
$ aptitude -w 80 search ~iwireshark
i   libwireshark-data                - network packet dissection library -- data
i A libwireshark12                   - network packet dissection library -- share
i   wireshark                        - lyseur de trafic réseau – métapaquet
i   wireshark-common                 - work traffic analyzer - common files
i A wireshark-qt                     - analyseur de trafic réseau– version Qt
```

Affichage sur 80 caractères

Advanced Package Tool

- Rechercher dans le catalogue local des paquets
  - En ligne de commande → `dpkg -l`
  - Exemple : existe-t-il un paquet ayant pour nom wireshark ?

```
$ dpkg -l wireshark*
Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé/W=attend-
traitement-déclenchements
|/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais)
||/ Nom              Version      Architecture Description
+++-================-============-============-=====================================
ii  wireshark        3.0.5-1      amd64        network traffic analyzer - meta-package
ii  wireshark-common 3.0.5-1      amd64        network traffic analyzer - common files
un  wi              c    <aucun             <aucune>     (aucune description n'est disponible)
un  wi              k    <aucu              une>         (aucune description n'est disponible)
ii  wireshark-qt     3.0.5              4    network traffic analyzer - Qt version
```

État du paquet

Version du paquet installé

Advanced Package Tool

- Rechercher dans le catalogue local des paquets
  - En ligne de commande → dpkg -S
  - Exemple : Quel est le paquet qui contient le programme wireshark ?

```
$ which wireshark
/usr/bin/wireshark
$ dpkg -S /usr/bin/wireshark
wireshark: /usr/bin/wireshark
```

Recherche du programme dans l'arborescence du système

Recherche dans les listes de fichiers des paquets installés

- Rechercher dans la liste des fichiers d'un paquet installé
  - En ligne de commande → dpkg -L
  - Exemple : Quel est le binaire contenu dans le paquet wireshark installé ?

```
$ dpkg -L wireshark | grep bin/
/usr/bin/wireshark
```

Advanced Package Tool

- Affichage des méta-données d'un paquet
  - En ligne de commande → aptitude show | apt show
  - Exemple : quelles sont les dépendances du paquet apache2 ?

```
$ aptitude show apache2
Paquet : apache2
Version : 2.4.41-1
État: non installé
Priorité : optionnel
Section : httpd
Responsable : Debian Apache Maintainers <debian-apache@lists.debian.org>
Architecture : amd64
Taille décompressée : 617 k
Dépend: apache2-bin (= 2.4.41-1), apache2-data (= 2.4.41-1), apache2-utils (= 2.4.41-1), lsb-base, mime-
support, perl:any, procps
Pré-dépend: dpkg (>= 1.17.14)
Recommande: ssl-cert
Suggère: apache2-doc, apache2-suexec-pristine | apache2-suexec-custom, www-browser
Est en conflit: apache2.2-bin, apache2.2-common
Casse: libapache2-mod-proxy-uwsgi (< 2.4.33)
Remplace: apache2.2-bin, apache2.2-common, libapache2-mod-proxy-uwsgi (< 2.4.33)
Fournit: httpd, httpd-cgi
```

État du paquet

Liste des dépendances

Advanced Package Tool

- Suppression d'un paquet installé
  - En ligne de commande → aptitude remove | apt remove
  - Supprime les fichiers binaires mais conserve la configuration locale
  - Exemple : Comment supprimer le paquet apache2 ?

```
# aptitude remove apache2
Les paquets suivants seront ENLEVÉS :
  apache2 apache2-bin{u} apache2-data{u} apache2-utils{u} libapr1{u} libaprutil1{u}
libaprutil1-dbd-sqlite3{u} libaprutil1-ldap{u}
0 paquets mis à jour, 0 nouvellement installés, 8 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 0 o d'archives. Après dépaquetage, 7 607 ko seront libérés.
Voulez-vous continuer ? [Y/n/?]
```

Suppression des paquets installés automatiquement

```
# aptitude search ^apache2 | egrep -e '^(p|c)'
c   apache2 - Serveur HTTP Apache
p   apache2-bin - Serveur HTTP Apache (modules et autres fichiers binaires)
p   apache2-data - Serveur HTTP Apache - fichiers communs
```

État du paquet après suppression

Advanced Package Tool

- Purge d'un paquet installé
  - En ligne de commande → aptitude purge | apt purge
  - Supprime les fichiers binaires et la configuration locale
  - Exemple : Comment purger le paquet wireshark ?

```
# aptitude purge wireshark
Les paquets suivants seront ENLEVÉS :
  wireshark{p} wireshark-qt{u}
0 paquets mis à jour, 0 nouvellement installés, 2 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 0 o d'archives. Après dépaquetage, 8 289 ko seront libérés.
```

Suppression des paquets installés automatiquement

```
# aptitude search ^wireshark
p   wireshark                 - analyseur de trafic réseau – métapaquet
i   wireshark-common          - network traffic analyzer - common files
p   wireshark-dev             - analyseur de trafic réseau - outils de développement
p   wireshark-doc             - analyseur de trafic réseau - documentation
p   wireshark-gtk             - al dummy package
p   wireshark-qt              - de trafic réseau– version Qt
```

État du paquet après suppression

Advanced Package Tool

- Suppression & purge d'un paquet individuel avec dpkg
  - Suppression en ligne de commande → `dpkg --remove`
  - Purge en ligne de commande → `dpkg --purge`

Installé & configuré

Supprimé & encore configuré

Ni installé Ni configuré

```
# dpkg -l nano | egrep '^(i|r|u)'
ii  nano           4.4-1        amd64   small, friendly text editor inspired by Pico

# dpkg --remove nano
Suppression de nano (4.4-1) …

# dpkg -l nano | egrep '^(i|r|u)'
rc  nano           4.4-1        amd64   small, friendly text editor inspired by Pico
# dpkg --purge nano
Purge des fichiers de configuration de nano (4.4-1) …

# dpkg -l nano | egrep '^(i|r|u)'
dpkg-query: aucun paquet ne correspond à nano
```

Advanced Package Tool

- Duplication du jeu de paquets d'un système à l'autre
  - Installation «optimale» → processus long
  - Duplication de la liste des paquets installés sans les configurations
  - Évolution/Migration d'une architecture à l'autre (ex. i386 → amd64)

- Sur le système **source**

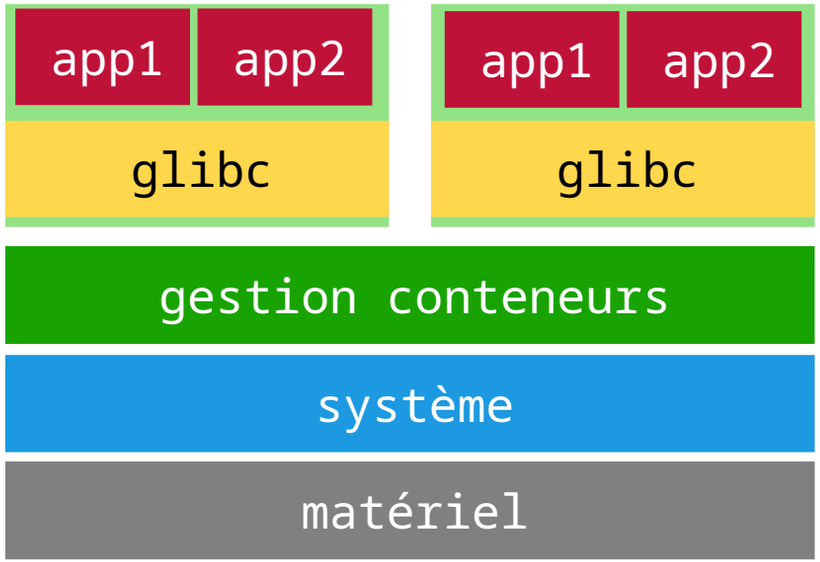
```
$ aptitude search "?installed?not(?automatic)" -F %p > package-list.txt
```

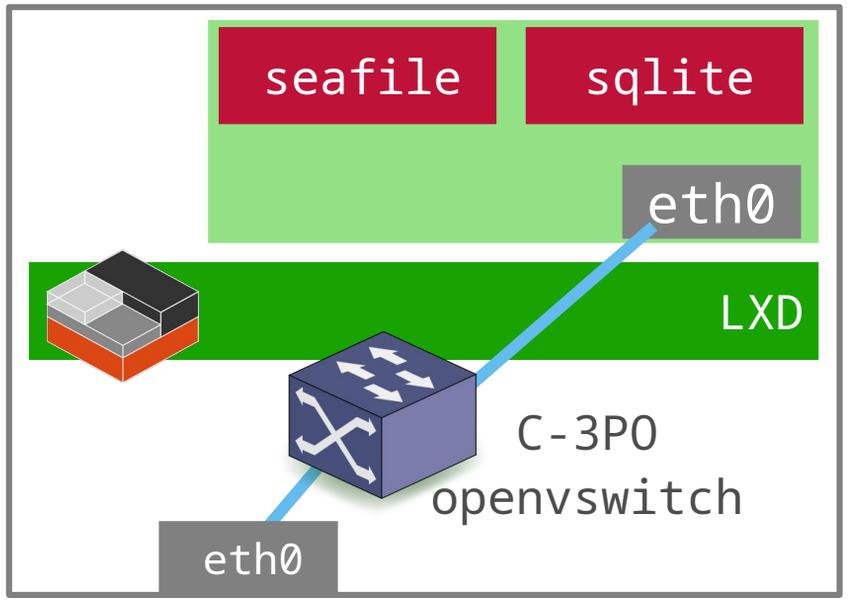
- Sur le système **cible**

```
# aptitude install $(cat package-list.txt | tr '\n' ' ')
```

Application → Serveur Seafile - 0

- Objectifs
  - Transformer le système hôte en commutateur/routeur réseau
  - Installer de gestionnaire de conteneurs LXD
  - Créer un conteneur ‘seafile’





Application → Serveur Seafile - 1

- Installation des paquets, du service snapd et de LXD
  - Installer les paquets openvswitch-switch et snapd
  - Installer de gestionnaire de conteneurs LXD
  - Attention !
    - L'utilisateur 'etu' doit appartenir au groupe système 'lxd'
    - La liste des chemins de recherche d'applications doit contenir '/snap/bin'

```
etu@vm0:~$ sudo apt install openvswitch-switch snapd
etu@vm0:~$ sudo snap install lxd
etu@vm0:~$ sudo adduser etu lxd
Déconnexion / Reconnexion

etu@vm0:~$ id | grep -o lxd
lxd
etu@vm0:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/snap/bin
```

Application → Serveur Seafile - 2

- Interconnexion réseau
  1. Désactiver la configuration système initiale

     $ sudo ifdown eth0
  2. Éditer le fichier /etc/network/interfaces
  3. Activer la nouvelle configuration

     $ sudo ifup C-3PO
  - Vérifier l'état des interfaces

    $ ip addr ls

```
etu@vm0:~$ cat /etc/network/interfaces
# This file describes the network
# interfaces available on your system
# and how to activate them.
# For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-ovs C-3PO
iface C-3PO inet dhcp
        ovs_type OVSBridge
        ovs_ports eth0

allow-C-3PO eth0
iface eth0 inet manual
        ovs_bridge C-3PO
        ovs_type OVSPort
        up ip link set dev $IFACE up
        down ip link set dev $IFACE down
```

Application → Serveur Seafile - 3

- Gestion des conteneurs
  1. Configuration initiale

     ```
     $ lxd init
     ```

  2. Éditer le profil ‘default’

     ```
     $ lxc profile edit default
           nictype : bridged
     ```

  3. Créer le conteneur ‘seafile’

```
$ lxc launch images:debian/bullseye seafile
$ lxc ls
```

```
etu@vm0:~$ lxd init --dump
config: {}
networks: []
storage_pools:
- config:
    size: 15GB
    source:
/var/snap/lxd/common/lxd/disks/default.img
  description: ""
  name: default
  driver: btrfs
profiles:
- config: {}
  description: Default LXD profile
  devices:
    eth0:
      name: eth0
      nictype: bridged
      parent: C-3PO
      type: nic
    root:
      path: /
      pool: default
      type: disk
  name: default
```

Application → Serveur Seafile - 4

- Configuration du service
  - Dans le conteneur seafile
    - Suivre les instruction du gist : «Seafile + sqlite instance on LXD lab»
      - https://gist.github.com/platu/67751fab9ae702fe29bd52cbfe9b3a01

1. Installation du serveur
2. Installation des paquets requis
3. Lancer le script ‘setup-seafile.sh’ en faisant correspondre
   - Le nom du seveur
   - L’adresse IP sur laquelle le service est en écoute
4. Lancer les deux services ‘seafile’ et ‘seahub’
5. Configurer le service mandataire (proxy) sur le système hôte
   - Uniquement si les réseaux du système hôte et du conteneur sont isolés

Bilan séance 3

- Environnements Graphiques
  - Interfaces utilisateur + chaînes de développement
  - Évolutions importantes côté dispositifs mobiles
  - Social Desktop → nextcloud.com
- Gestionnaire de paquets → Advanced Package Tool
  - **Base de l'administration système**
  - **Bibliothèques partagées entre applications**
    - Notion de dépendance
    - Gestion automatisée des relations entre applications et bibliothèques
  - **Gestion autonome des configurations**
  - **Gestion automatisée des correctifs de sécurité**