Administration du système

Fabien Rico

Univ. Claude Bernard Lyon 1

séance 8

Jacques Bonneville	jacques.bonneville@univ-lyon1.fr	TP
Adil Khalfa	adil.khalfa@cc.in2p3.fr	TD + TP
Léo Le Taro	leo.le-taro@inria.fr	TD + TP
Fabien Rico	fabien.rico@univ-lyon1.fr	CM+ TD + TP

Système et administration

Système d'exploitation

- Savoir comment les choses sont organisées
- Pour comprendre les problèmes
  - Problèmes mémoire
  - Interblocages
- Pour utiliser les caractéristiques du système
  - Communications entre processus
  - Multi-threading
  - Gestion des resources

Administration

- Configuration
- Installation
- Utilisateurs

- Chaque logiciel doit stocker des informations spécifiques de configuration,
- qui doivent être conservées
- Facilement modifiables
- Organisées
  - Centraliser les données de la machine
  - Distribuer celles des utilisateurs
  - Organiser par thème
  - Organiser par logiciel
  - Organiser par programmation ou entreprise
- Pour le stockage on peut utiliser :
  - Un système de fichiers
  - Une base de données
- Et les interfaces ?

Quoi ?

- Comment démarrer (`/boot/`, `c:\boot.ini`)
- Les services : logiciel annexe à démarrer (service d'authentification, serveur d'accès, gestion du matériel, reseaux...)
- Installation, librairies partagées et drivers
- Les utilisateurs, droits
- Configurations des logiciels.
- Configurations spécifiques aux utilisateurs.

Système de fichiers

Sous unix tout est fichier, les périphériques /dev/, les variables du système et leurs informations/sys/ et /proc/, la configuration aussi

- /boot/ pour le démarrage ;
- /etc/rc.d, /etc/ init .d, /etc/[x] inetd pour les services ;
- /etc/<nom du logiciel d'installation>, /etc/ld.so.conf (librairies) ;
- /etc/<nom du logiciel> par exemple /etc/X11 pour le serveur graphique ;
- fichiers sur le compte utilisateur par ex./home/rico/.subversion ;

Avantages et inconvénients

Avantages

- Objet simple déjà connu cela permet d'utiliser des techniques éprouvées (cp, rsync, diff, grep, locate...).
- Souple, on peut utiliser le format le plus adapté (langage de programmation de l'application, xml, script).
- Organisation naturelle par répertoires.

Inconvénients

- Lent (à voir)
- Souple (pas de schéma général)
- Difficile à distribuer dans un réseau (nis, ldap, nsswitch.conf)
- Pas forcement adapté :
  - Modifications concurrentes (Fichiers *.d)
  - Droits (Il faut être root)

Modifications concurrentes

Des fichiers différents peuvent avoir besoin de modifier la même configuration.

- Les serveurs doivent modifier la configuration des services
- Beaucoup de logiciels doivent modifier les configurations des utilisateurs (menu variable d'environement, logiciels)

Il est très difficile de modifier un fichiers de façon automatique (à l'installation ou à la désinstallation)

- Les fichiers les plus importants ou liés aux serveurs les plus populaires ont tendance à être coupés en plusieurs morceaux.
  - /etc/apache/http.conf → /etc/apache/conf.d/php.conf, /etc/apache/conf.d/squid.conf,...
  - /etc/modprobe.d/broadcom−wl−blacklist.conf
- Apparition de logiciels de configuration qui gèrent un ensemble de fichiers ... et ont eux même des fichiers de configurations.
  - dhclient (/etc/dhclient.conf) modifie /etc/resolv.conf
  - authconfig (red hat - /etc/sysconfig/authconfig) modifie les configuration de pam, nsswitch nis et libnsswitch −ldap.

La base de registre

- Pour améliorer l'accès aux configurations, on peut utiliser une base de données
- Avec une organisation similaire aux répertoires des systèmes de fichiers
- Chaque répertoire contient des données de différents types.
- C'est la base de registre.

Par exemple :

\HKEY_CLASSES_ROOT \.pdf	contient le type associé aux fichiers d'extention .pdf
\HKEY_CLASSES_ROOT \AcroExch.Document\	contient les icônes et logiciels associés

Stockage

- La base est stockée dans des fichiers « *backend* ».
- Mais, il faut passer par des utilitaires dédiés pour lire/modifier le contenu.
  - Gconf pour gnome (fichier xml)
  - Netinfo pour OS X (fichier de bd)
  - Regedit pour windows (fichier « ruches »)

Exemple : la base windows

- HKEY_LOCAL_MACHINE\HARDWARE la liste du matériel détecté
- HKLM\SYSTEM la configuration du système (fichier %systemroot%\system32\config\SYSTEM)
- HKLM\SAM les comptes (fichier %systemroot%\system32\config\SAM)
- HKLM\Software les logiciels (dont la sous branche classes est HK_CLASSES_ROOT)
- HK_USERS*id* Les données de l'utilisateur (fichier \Document and setting*login*\NTUSER.DAT)

Avantages et inconvénients

Avantages

- Plus rapide,
- Distribution des données souple ⇒ facile à distribuer.
- Permet les recouvrements (HKEY_CURRENT_USER contient HKEY_USERS*<id de l'utilisateur courant>*)
- Chaque configuration est un objet avec ses propres droits.
- Logiciel spécifique (sauvegarde, versions...)

Inconvénients

- Logiciel spécifique (plus difficile à gérer)
- Chaque configuration est un objet avec ses propres droits (compliqué)
- Modifications concurrentes

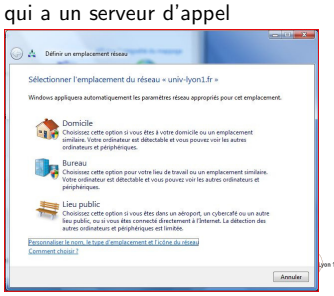
Interface

- Modifier directement les fichiers est parfois compliqué,
  - Par exemple pour utiliser l'authentification ldap sous linux il faut configurer :
  - le client ldap.
  - l'agent d'authentification qui doit l'utiliser.
  - le système qui doit remplacer les fichiers d'utilisateurs par un client.
- Pour ça il existe des utilitaires
- Pratique mais l'utilitaire :
  - Ne permet pas de tout configurer (ex : firewall).
  - Ne permet pas toujours de sauvegarder les configurations.
- Effet boite noire.

Assistants

Certains utilitaires sont basés sur des configurations toutes faites qu'on sélectionne en répondant à quelques questions. Ce sont les *Assistants*, ou *Wizards*

- C'est très simple à utiliser tout en permettant de faire des configurations très pointues
- La méthode est de choisir un scénario d'utilisation courant et de l'appliquer. Comme une « hotline » qui a un serveur d'appel automatique...
- ...Comme une hotline c'est souvent très difficile à utiliser.
  - Par exemple pour configurer un ordinateur portable au Nautibus :



Utilisateurs

- L'une des configurations les plus importantes est celle qui gère les utilisateurs
- Il faut
  - permettre l'authentification (généralement par mot de passe) ;
  - gérer les groupes ;
  - gérer les droits sur les différents composants ;
  - conserver les données (`/home/`).
- Ces données peuvent être
  - stockées localement sur la machine ;
  - centralisées sur un serveur ;
  - stockées dans un annuaire.

Localement

- Les informations sont stockées dans un fichier de la machine
  - `%systemroot%\system32\config\SAM` sous windows
  - /etc/passwd, /etc/shadow et /etc/groups sous linux
- Les fichiers ne contiennent pas directement les mots de passe mais leur empreinte numérique par une fonction de hachage.
  - Pour authentifier un utilisateur le système récupère le mot de passe en clair.
  - Il utilise la même fonction de hachage et compare les résultats.
- Ces fichiers sont critiques pour le système
  - Problème des mots de passe identiques.
  - Problème des mots de passe trop simples.

En réseau

- Dans un réseau local, il est nécessaire centraliser la gestion des utilisateurs.
- On peut modifier les utilitaires qui accèdent aux descriptions des utilisateurs pour qu'ils contactent un serveur.
  - Ex : NIS à chaque accès, le fichier correspondant est demandé au serveur.
  - Le service utilisé pour chaque fichier est géré par le « Name Service Switch » (fichier /etc/nsswitch.conf).
  - Cache local.
- On peut déléguer une partie du travail à un serveur
  - Ex : les domaines windows
  - le PDC fournit l'authentification
  - le reste est fait par des scripts
- Avec ces deux méthodes les informations centralisées sont limitées.

Annuaire

- Un annuaire est une base de données
  - optimisée pour la lecture.
  - Pouvant contenir tout type d'information
  - Avec une organisation hiérarchisée (arbre).
  - Permettant des recherches multiples.
  - Proposant un système d'authentification.
- Par exemple :
  - OpenLdap « Lightweight Directory Access Protocol ».
  - Active Directory qui utilise le protocole de nom ldap.

Ldap/AD

- Les objets sont placés dans une *structure arborescente*.
- La racine de la structure est liée au domaine DNS.
  DC=polytech,DC=upmc,DC=fr
- chaque objet a un nom unique le *distinguished name* ou *dn* faisant apparaître le chemin dans l'arbre
  - OU=comptes,DC=polytech,DC=upmc,DC=fr par exemple l'entité qui rassemble tous les comptes
  - OU=encad,OU=comptes,DC=polytech,DC=upmc,DC=fr par exemple l'entité qui rassemble tous les enseignants
  - CN=rico,OU=encad,OU=comptes, DC=polytech,DC=upmc,DC=fr mon compte
- À chaque objet on associe des données
- Le type des données et leurs positions dans l'arbre sont fixés par des *shémas* (donc identiques entre serveur, mais adaptables).
- Les droits d'accès aux données sont gérés par des *ACL* (Voir plus loin)

Gestion des droits

L'existence de différents utilisateurs sur une machine permet de gérer différents droits.

- Un utilisateur standard a le droit
  - D'utiliser les logiciels.
  - D'utiliser son espace de stockage (compte).
  - De lire les données partagées.
- Certains utilisateurs particuliers servent à
  - Limiter les droits des serveurs (util. appache).
  - Gérer des accès distants (Administration à distance).
  - Avoir des configurations particulières des droits particuliers (ex : oracle).
- Un utilisateur spécial a tous les droits *Administrateur*(windows) *root*(unix).

Gestion des droits unix

- Les droits sont les droits sur les fichiers (tout est fichier).
- Les droits de base sont :
  - **read** lecture du fichier, liste du contenu du répertoire ;
  - **write** écriture dans le fichier, ajout/suppression de fichier dans le répertoire ;
  - execute exécution du fichier, aller dans le répertoire *ou un sous répertoire*.
- Pour un fichier un utilisateur est dans l'une des classes :
  - user, u : propriétaire ;
  - group, g : groupe du propriétaire ;
  - other, o : tous les autres.
- root a toujours le droit
- Tout processus a un propriétaire égal à :
  - celui qui a lancé la commande (SetUID bit = 0) ;
  - celui à qui appartient la commande (SetUID bit = 1).

Exemple

- Pour mettre en place sa page internet personnelle, il faut que l'utilisateur *apache* ou *html* ait le droit de lire le contenu du répertoire `~/public_html/` donc :
  - `~/` doit être autorisé en exécution pour les autres.
  - `~/public_html/` doit être autorisé en exécution et lecture pour les autres.
- Les mots de passes doivent être protégés . Mais la commande password doit permettre de lire et modifier le mot de passe.
  - `/etc/shadow` est en lecture uniquement pour son propriétaire root
  - `/usr/bin/passwd` appartient à root, est autorisé en exécution pour tous avec un setUID bit = 1.
- Les droits permettent de protéger le système tout en délégant des droits étendus via certaines commandes.

ACL

Le système de droits n'est pas suffisant :

- Il n'y a pas de droits négatifs (tous sauf ...).
  - Par exemple avec apache, les accès sont basés sur allow et deny et un ordre de lecture des droits
- Seulement 3 types de personnes
  - Fastidieux, pour gérer finement les droits les utilisateurs doivent être dans de nombreux groupes
  - Quand un utilisateur crée un fichier, à quel groupe appartient-il ?
- Une solution est d'associer à chaque objet une liste de droits (ou déni de droits) accordés à des utilisateurs ou des groupes. Ce sont les *Access Control List* ou *ACL*.

ACL (suite)

- Une `ACL` est une liste d'*ACE*
- Les droits sont positifs ou négatifs
- Un `ACE` est formé :
  - d'un droit particulier (lecture, écriture, contrôle totale, changer les droits...) ;
  - d'un utilisateur ou d'un groupe ;
  - d'un objet sujet ;
  - d'un booléen `Allow` ou `Deny`.
- On doit définir un ordre de lecture
- Exemple
  - Windows (droits de base, droit sur NTFS), OS X, Linux (compat. mais peu utilisé)
  - ldap, firewall , AFS
  - Forums, blogs ...

Résolution de problème

≪ *Tout programme non trivial possède au moins un bug.* ≫
Corollaire de la loi de Murphy.

- Il est donc nécessaire de savoir trouver et corriger les problèmes.
- Les systèmes donnent beaucoup d'informations qui généralement permettent de trouver la solution.
- Mais il faut savoir où chercher :
  - *Historique des événements* (logs) du système.
  - Service en mode *debug*.
  - Utilitaires.

Historique des événements

- Comme tous programmes, les services systèmes rendent compte de leurs actions.
- Ces messages sont centralisés et rassemblés /var/log/.
  - /var/log/message pour la plupart des logs.
  - /var/log/httpd/* ou /var/log/apache/* pour le serveur web
  - Xorg.0.log pour le serveur graphique
  - ...
- Il est souvent très instructif de suivre les logs système pour voir en temps réel les effets d'une action.
  tail −f /var/log/httpd/error.log
- Tous les services ont dans leur configuration un *niveau de log* qui permet d'augmenter le nombre d'information disponible.

Message d'erreur

- Généralement, les services sont lancés en tâche de fond, dans un mode complexe (multithread/multiprocessus,...).
- Mais ils peuvent être lancer en avant-plan pour la correction de problèmes.
  - dhcp −f
  - httpd −X
  - slapd −d 3
  - ...
- Cela permet de les lancer dans un debugger, ou d'obtenir tous les message d'erreur de l'application.

Outils

- Debugger dbg, ddd, kdbg.
- Pour un script utiliser l'option −x qui affiche les commandes avant de les exécuter.
- Utilitaires d'écoute sur le réseau : tcpdump, wireshark.
- strace qui affiche les appels systèmes d'un programme.
- ltrace qui affiche les appels à une librairie et leur parametres.

Règles élémentaires

- Stopper les services de cache nscd et de sécurité firewall , selinux.
- Rechercher les options de sécurité par défaut.

Installation

- Installer un logiciel signifie
  - Copier les exécutables
  - Copier les librairies internes au logiciel
  - Installer les dépendances
  - Adapter les configurations
  - Se souvenir des modifications
- On distingue 2 types d'installations avec des utilitaires différents
  - Installation du système et des logiciels de base
  - Installation d'un logiciel particulier

Librairies

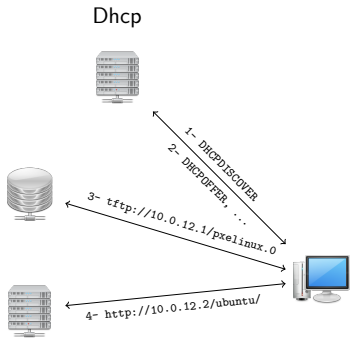
- Une librairie est un ensemble de code de fonctions.
- Ces fonctions ne sont pas écrites dans chaque exécutable ni copiées dans la mémoire de chaque processus.
- Mais partagées entre processus.
- Cela signifie qu'un logiciel qui utilise ces librairies dépend de fichier externe.
- La gestion des dépendances pose problème :
  - développement des applications :
    - gestion des fichiers d'entête .h : #include, −I ;
    - gestion des fichiers de code : −L ..., −lpthread, −lm
  - installation (dépendance)
  - mise à jour

Installation d'un système

On peut utiliser

- Un CD
  - système préconfigurés (windows, mac OS)
  - système simple à installer (linux)
- Un système de copie d'image disque (ex : `ghost`)
  - très efficace ;
  - mais très spécifique ;
  - de retour grâce à la virtualisation.
- Installation automatique
  - `RedHat`, `Fedora` : kickstart, `Debian` : preseeding
  - `Windows` : WDS Windows Deployement Service.

Principe de l'installation automatique



- démarrage par PXE ;
  - Récupération d'une adresse automatique (DHCP)
  - Récupération d'une mini image d'installation (TFTP)
- Installation automatisée ;
  - Récupération des paquets systèmes (HTTP, SMB, ...)
  - Récupération d'un script d'installation

Installation automatique

	Partage de fichier	configuration	noyau d'installation	système
Kickstart /preseeding	nfs, http, ftp	fichier	linux sur CD ou PXE	linux
WDS	smb	fichier	WindowsPE PXE	windows

- Avantages :
  - C'est une installation donc s'adapte au matériel
  - Permet différents scénarios
  - Automatique (pas d'intervention)
- Inconvénients :
  - Long
  - Uniquement le système de base
  - Ne permet pas facilement la configuration

Provisioning

Il existe des logiciels facilitant ces différentes configurations :

- configuration automatique des différents serveur (Dhcp, TFTP, http...) ;
- gestion des images système disponibles ;
- gestion de modèles de provisionnement ;
- choix à distance entre installation, démarrage normal, récupération.

Par exemple Foreman (intercafe web), WDS.

- Ils permettent la gestion de parc de serveur ou de cluster.
- Ils proposent une gestion indifférenciée de machines physiques (bare metal) ou virtuelles.
- Ils sont associés à des outils de supervision, orchestration.

Quel est la différence

Pourquoi faire la différence entre installation de logiciel et de la machine ?

- Installation sur des systèmes qui évoluent
- Les scripts d'installation automatique existent mais il n'y a pas de standard
- Installation plus simple
  - copie de fichiers
  - peu de configuration
- Entretient plus complexe
  - Mise à jour
  - Suppression
- L'éditeur du logiciel fournit un programme d'installation .
- Pour régler les problèmes on utilise un système de paquets

Les paquets

C'est une « archive » contenant :

- Les fichiers à copier sur le système.
- Les configurations.
- Un script d'installation et de désinstallation.
- Les dépendances (parfois).
- Avantages :
  - Le gestionnaire de paquets se souvient des installations
  - C'est automatisable
- Inconvénients :
  - L'éditeur ne fournit pas toujours un paquet
  - Surtout les logiciels avec leur propre système de paquets (perl, matlab, R...)
  - Dépendances

Exemple : les paquets rpm (Fedora/RedHat/Suse/Mandrake...)

- Les sources du logiciel et patch
- Des scripts
  - Compilation
  - Pre/post installation
  - Pre/post déinstallation
- Liste des fichiers installés.
- 2 types de paquets, source et binaire.
- On crée un paquet en compilant le logiciel.
- Les dépendances sont données à la main ou calculée automatiquement (moins clairement).
- Avantages/Inconvénients
  - \+ fiabilité
  - \+ portabilité
  - \- demande beaucoup de connaissance sur le logiciel à installer.

Exemple : les paquets MSI (Microsoft Installer)

- Installation du logiciel sur un ordinateur témoin
- Utilisation d'un système de photographie du disque et différence avant/après l'installation
- Même chose pour la base de registres
- Le paquet est constitué des différences.
- On peut le modifier après
- Avantage/Inconvénients :
  - \+ Demande peut de connaissances dans les cas simples.
  - \- Données inutiles dans le paquet.
  - \- Dépend du logiciel d'installation fournit par l'éditeur du logiciel.
  - \- Dépend de l'ordinateur témoin.

Gestionnaires

L'intérêt des paquets est d'automatiser l'installation.

- Utilisation de gestionnaires capables
  - D'aller chercher les paquets en local ou sur internet (*dépôts* ou *repository*).
  - De gérer des groupes de logiciel.
  - De gérer les mise-à-jour.
  - De gérer les dépendances.
- Permet de déployer des logiciels
- Permet des gérer des configurations logiciels

Lesquels

- Linux
  - yum (fedora, redhat), apt (debian,ubuntu)
  - \+ mise-à-jour et installation
  - \+ dépôts fiables
  - \- Pas vraiment adapté à la gestion de configurations logiciels
- Windows
  - Windows update (mise à jour) Stratégie de groupe AD (installation)
  - \- pas de dépôts
  - \- mise à jour
  - \+ gestion très fine des logiciels

Conclusion

- Installation
  - Recherche dans les dépôts
  - Recherche de paquets
  - Création d'un paquet
  - installation à la main
- Administration
  - Il faut comprendre de qu'on fait.
  - Il faut être capable de l'adapter.