Licence Pro SIL

Systèmes Internet et Intranet pour l'entreprise

Chapitre II : Bases de l'administration système et méthodes d'installation des applications

Département IEM / UB

Eric.Leclercq@u-bourgogne.fr

Bureau G212 Aile des Sciences de l'Ingénieur

Mise-à-jour : septembre 2011

Plan du chapitre

Les rôles de l'administrateur

Les principaux rôles de l'administrateur sont :

- maintenir le bon fonctionnement du parc matériel (serveur et ou postes de travail)
- planifier l'évolution de son parc et choisir de nouveaux équipements
- résoudre les incidents de fonctionnement
- gérer les utilisateurs (création, expiration, limitations)
- installer et mettre à jour les nouveaux logiciels
- gérer et maintenir les systèmes de fichiers
- surveiller la sécurité du système et du réseau
- optimiser les paramétrages des machines
- guider et conseiller les utilisateurs et les décideurs
- administrer le réseau local et l'accès au réseau public
- participer aux développements

Les rôles de l'administrateur

La complexité du rôle d'administrateur système peut varier :

- Un poste individuel : peut être délégué aujourd'hui à son propriétaire avec des procédure de remise en route rapide
- Un seul serveur : forte responsabilité sur la disponibilité des services
- Un parc de machines homogènes : responsable de la distribution et de l'accès aux ressources
- Un parc de machines hétérogènes : s'efforcer de rendre l'ensemble intéropérable (transparence d'accès, d'utilisation de d'accès). Attention : la complexité croît avec l'hétérogénéité
- Un site entier (machines + réseau) : rechercher un point d'équilibre entre les services offerts, la disponibilité et les coûts d'administration puis le maintenir stable sur une longue durée
- Un site entier (moyen et long terme) : ajoute la problématique de la cohabitation des anciennes et nouvelles versions d'application (assurer une pérennité)

Connaissances requises

- Connaître Unix :
  - ses concepts et son architecture (noyau, processus, file-system, devices, couches réseau)
  - les principaux outils
  - les principaux langages associés (à détailler)
  - ses forces et ses faiblesses (à détailler)
- Connaître l'environnement :
  - les équipements matériels (l'offre et les technologies)
  - l'offre logiciels
  - les besoins des utilisateurs
  - les autres administrateurs système et l'administrateur réseau
- Suivre l'évolution des technologies et des pratiques au moyen d'une activité de veille technologique (planifier des journées de travail)
- Rédiger les documentations pour l'exploitation
- Savoir chiffrer un projet et rédiger des spécifications techniques

Différentes méthodes d'installation de logiciels

On distingue généralement 4 modes d'installation différents :

- la notion de packages
- la notion de port ou portage
- installation à partir des sources
- installation des binaires
- installation selon une méthode spécifique (propriétaire)

Notions de package

- Ensemble de fichiers dédiés à une application, rassemblés dans une archive, valable pour une architecture précise (x86 32bits, SPARC, etc.)
- Utilisés dans les distribution RedHat, Suse, Mandrake (RPM) Debian (.deb) mais aussi sous Solaris (.pkg)
- Une base de données enregistre les packages installés dans le système
- Toute modification ou installation hasardeuse peut entraîner une incohérence de la base de données
- Veiller à installer uniquement des packages pour la distribution installée, et l'architecture matérielle de la machine
- Les applications n'existant pas sous la forme de package ou installées depuis le code source doivent être localisées dans une partie spécifique du système de fichiers

Gestion de l'arborescence du système

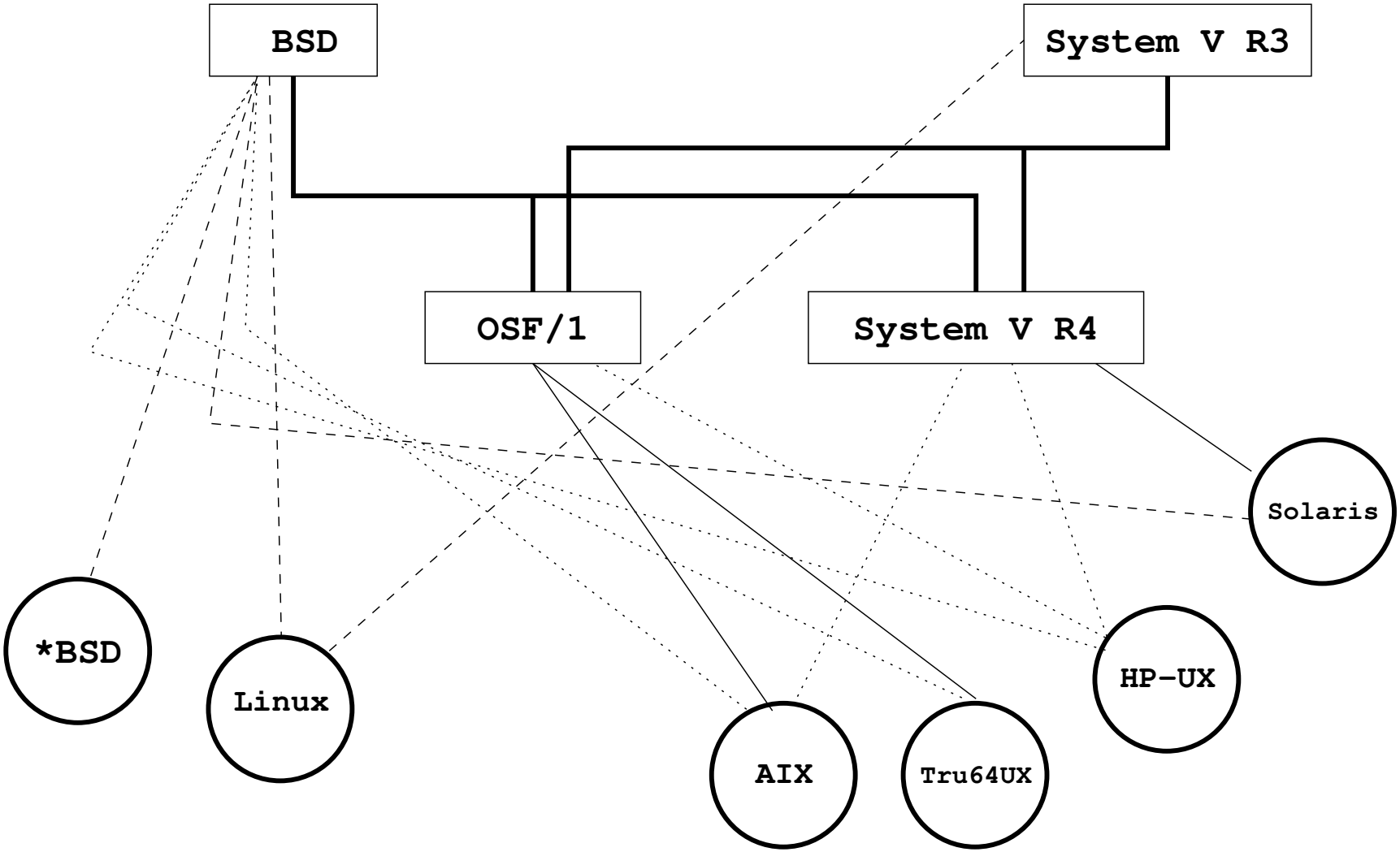
- `/etc` : répertoire essentiel à l'administration du système (fichiers de configuration, tables diverses)
- `/dev` : système de fichier propre aux périphériques (interventions via `mknod`)
- `/usr` : arborescence des utilitaires et bibliothèques du constructeur ou du distributeur
- `/usr/local` : utilitaires et bibliothèques installées à partir des sources (le plus souvent des logiciels libres)
- `/opt` : logiciels optionnels sous licences variées (ORACLE, Adobe etc.)
- `/var, /var/adm, /var/spool, /var/tmp` : données variables à durée de vie moyenne
- `/tmp` : fichiers temporaires à très courte durée de vie (effacé au reboot)
- et les autres : `/bin, /mnt, /proc`

Le répertoire /etc

Les éléments importants de ce répertoire concernent :

- Démarrage du système :
  - fichier `inittab`
  - `/etc/init.d`, `/etc/rc0.d/`, `/etc/rc1.d/`, `/etc/rc2.d/`, `/etc/rc3.d/` (SysV)
  - différents suivant la distribution et la famille d'origine
- Utilisateurs : `/etc/passwd`, `/etc/shadow`, `/etc/group`
- Système de fichiers `/etc/fstab` (BSD), `/etc/vfstab` (SysV)
- Réseau : `/etc/hosts`, `/etc/services`, `/etc/networks`

Généalogie Simplifiée



Gestion de packages (RPM)

- RedHat Package Manager (RPM)
- Lors des mises à jour, RPM traite les fichiers de configuration de façon particulière, de sorte les personnalisations sont préservées
- RPM permet aussi au développeur d'empaqueter le code source d'un logiciel et de l'insérer dans des paquetages source et binaires destinés aux utilisateurs finaux
- Objectifs des RPM :
  - **Évolutivité :** facilité des mises à jour et des corrections
  - **Fonction d'interrogation puissante :** effectuer des recherches dans la BD des packages ou seulement dans certains fichiers, retrouver aisément à quel package appartient un fichier
  - **Vérification du système :** suppression d'un fichier important pour un paquetage quelconque $\rightarrow$ vérifier l'intégrité du package

Le nommage des packages

Les packages rpm adoptent la convention de nommage
`nom-version-release.architecture.rpm`
La signification de chaque partie du nom du package :

- `nom` : représente le nom du package, en général il s'agit du nom du logiciel mais celui-ci peut aussi être décomposé en plusieurs packages. Ainsi, on trouve couramment :
  - `nom-devel` : package de développement (include C, bibliothèques) pour permettre la compilation de logiciels utilisant les ressources fournies par le packages ;
  - `libnom` : les bibliothèques du logiciel sont séparées du package principal ;
  - `nom-common` : partie du logiciel utilisable pour un serveur ou un client ;
  - `nom-server` : partie du logiciel pour un serveur ;
  - `nom-client` : partie du logiciel pour un client.
- `version` : telle qu'elle est définie par le ou les développeurs du logiciel ;

Le nommage des packages

- `release` : elle est définie par celui qui construit le package ;
- `architecture` : représente la catégorie de processeur sur laquelle le package peut être installé :
  - ppc pour powerpc (mac), sparc pour stations SUN, alpha pour PC avec processeurs alpha etc.
  - Pour les architectures Intel la dénomination est la suivante : i* = processeurs intel, + i386 (la base commune 80386) + i486, + i566 : pentium, i686 : pentium II, III, IV. Tous ces packages contiennent des programmes compilés, donc non portables entre les différents processeurs (Intel, powerpc, sparc)
  - les programmes dépendent également des librairies utilisées lors de la compilation et sont donc souvent non portables entre distributions différentes, voire sur la même distribution, entre versions différentes

Le nommage des packages

- il existe deux types d'architectures portables :
  - `noarch` : ces packages contiennent des fichiers de configuration, ou des programmes en langage interprété (shell, perl, python ...), et peuvent donc être installés partout ;
  - `src` : ces "pseudo-packages" contiennent les programmes sources, avec ce qu'il faut pour les recompiler (`makefile`, `configure`) et fabriquer un package.

Modes de fonctionnement

- RPM offre cinq modes de fonctionnement de base, exploitables via la commande `rpm` :
  1. installation
  2. désinstallation
  3. mise à jour
  4. interrogations
  5. vérifications
- L'ensemble des possibilités est donné par `rpm --help` les principales sont présentées dans les transparents suivants.

RPM : Installation

- Les fichier RPM portent généralement des noms tels que `prog-1.0-1.i386.rpm`, la version (1.0) du code, l'édition (1) màj RedHat, l'architecture (i386)
- **Installer le package :**
  `rpm -ivh prog-1.0-1.i386.rpm`
- La commande -U est généralement utilisée pour mettre à jour des paquetages, elle permet également d'en installer de nouveaux
- **Réinstaller un package :** `--replacepkgs`
  `rpm -ivh --replacepkgs prog-1.0-1.i386.rpm`

RPM : Désinstallation

- La désinstallation d'un package se fait par la commande : `rpm -e prog`
- Nous avons utilisé le nom `prog`, pas le nom du fichier d'origine `prog-1.0-1.i386.rpm`
- Une erreur de dépendance peut se produire :

```
# rpm -e prog
removing these packages would break dependencies:
prog is needed by xyz-1.0-1
```

- Pour faire en sorte que RPM ignore cette erreur et désinstalle le paquetage malgré tout, utiliser l'option `--nodeps`

RPM : Mise à jour

- La mise à jour est similaire à l'installation :
  `rpm -Uvh prog-2.0-1.i386.rpm`
- RPM effectue une mise à jour intelligente un message du type suivant peut apparaître :

  ```
  enregistrement de /etc/prog.conf /etc/prog.conf.rpmsave
  ```

  - les modifications apportées au fichier de configuration risquent de ne pas être compatibles avec le nouveau fichier de configuration du package (utilisation de suffixes `.rpmnew` ou `.rpmsave`)
  - RPM a enregistré le fichier d'origine et en a installé un nouveau
- rechercher les `.rpmnew` ou `.rpmsave` dans l'arboresence

RPM : Mise à jour

- mise à niveau vers un paquetage portant un numéro de version plus ancien, le système affiche le message suivant :

```
rpm -Uvh foo-1.0-1.i386.rpm foo
package foo-2.0-1 (which is newer) is
already installed error: foo-1.0-1.i386.rpm
  cannot be installed
```

- Pour faire en sorte que RPM effectue malgré tout la mise à niveau, utilisez `--oldpackage` dans la ligne de commande :
`rpm -Uvh --oldpackage prog-1.0-1.i386.rpm prog`

RPM : Recherches

- L'interrogation de la BD des packages installés s'effectue au moyen de `rpm -q`
- `rpm -q prog` imprime le nom, la version de prog (installé)
- On peut aussi utiliser les options suivantes avec -q pour spécifier le(s) paquetage(s) interrogés.
  - `-a` recherche tous les paquetages actuellement installés
  - `-f fic` interroge le package contenant `fic`
  - `-p nomp` interroge le package `nomp`
- Plusieurs manières de spécifier les informations à afficher :
  - `-i` affiche nom, description, version, taille, date de compilation, date d'installation, éditeur, etc.
  - `-l` affiche la liste des fichiers contenus dans le package
  - `-d` affiche la liste des fichiers de documentation
  - `-c` affiche la liste des fichiers de configuration
- Pour les options qui affichent des listes de fichiers, -v pour obtenir les listes dans un format ls -l

RPM : Vérification

- Comparer les informations sur les fichiers d'un package installé avec celles de l'original : taille, MD5, autorisations, type, propriétaire et groupe de chaque fichier : `rpm -V`
- Pour vérifier un package contenant un fichier particulier : `rpm -Vf nomfic`
- Pour vérifier tous les paquetages installés : `rpm -Va`
- Pour comparer un package installé à un fichier RPM : `rpm -Vp prog-1.0-1.i386.rpm`
- Si la vérification est correcte, elle ne fournit aucun résultat
- Si il y a des différences, le résultat est une chaîne de 8 caractères (indiquant l'échec de certains tests) :

5 : somme de contrôle MD5	S : taille de fichier
L : lien symbolique	U : utilisateur
G : groupe	M : mode (permissions et type)

RPM : Problèmes courants

- **Conflits de fichiers :** si vous tentez d'installer un package contenant un fichier déjà installé par un autre package, le système affiche :

```
# rpm -ivh truc-1.0-1.i386.rpm
foo  /usr/bin/truc conflicts with file from xyz-1.0-1
error: truc-1.0-1.i386.rpm cannot be installed
```

- Solution : utiliser l'option `--replacefiles`
- **Dépendance non résolue :** les RPM peuvent dépendre d'autres paquetages

```
# rpm -ivh prog-1.0-1.i386.rpm
failed dependencies: truc is needed by prog-1.0-1
```

  - Pour résoudre cette erreur, installez les paquetages demandés.
  - Pour forcer l'installation utiliser l'option `--nodeps`
  - À utiliser avec précaution car les màj seront plus difficiles

RPM : Questions Réponses

1. j'ai supprimé certains fichiers accidentellement comment déterminer quels sont les éléments manquants : `rpm -Va`
2. je veux voire ce que va installer le package toto.rpm
   `rpm -qlp toto.rpm`
3. je recontre un fichier inconnu, a quel package appartient il ?
   `rpm -qf /usr/sbin/fichierInconnu`
4. j'ai un probleme avec le programme truc, je ne connais pas le package : `rpm -Vf /usr/bin/truc`
5. je recherche les fichiers de documentation de ftp, où sont ils installés ?
   `rpm -qdf /usr/bin/ftp`
6. que fait le RPM wu-ftpd ?
   `rpm -qip wu-ftpd.2.6.2.i386.rpm`
7. un programme a besoin de la commande `cmd`, j'ignore le package qui la contient

Conclusion sur les packages

Pour ou contre les packages ?

+ flexibilité,
+ facilités de mises à jours,
+ interfaçage avec le réseau `yum` pour les système Fedora
- optimization,
- sécurité (à discuter)
- réactivité en cas de bugs
- tous les programmes ne sont pas livrés sous la forme de package

Les packages sous Debian

Le système apt se base sur les commandes dpkg (utilisé également sur Ubuntu).

- La distribution Debian propose un gestionnaire de package intégrant un téléchargement automatique depuis une liste de serveurs pré-définie (`/etc/apt/source.list`)
- Accessible via les commandes : `apt-get` et `apt-cache`
- Mises-à-jour : options `update`, `upgrade` ou `dist-upgrade` de la commande `apt-get`
- Installation : `apt-get install nompackage1 nompackage2`
- Déinstallation : `apt-get remove package1`, l'option `-purge` efface en plus les fichier de configuration
- Éffacer les fichier temporaires : `apt-get clean`

Les packages sous Debian

Recherche dans le système de packages

- Recherche dans la base des packages disponibles : `apt-cache search liste de mots clés`
- Description d'un package : `apt-cache show package1`

Remarque sur le gestion des bibliothèques :

- Les packages avec le mot clé `lib` sont les librairies binaires (.so, .a) utilisables par les modules binaires de applications
- Les packages avec le mot clé `dev` (et éventuellement `lib`) contiennent les en-têtes nécessaires pour compiler des programmes utilisant les librairies

Exercice : apt ne permet pas de tout gérer, retrouver les fonctionnalités du RPM RedHat avec dpkg

Les packages sous Debian (dpkg)

- La distribution Debian propose un gestionnaire de package (sur-couage de dpkg) intégrant un téléchargement automatique depuis une liste de serveurs prédéfinie
- Accessible via les commandes : `apt-get`, `apt-cache`
- Mises à jours : `apt-get update`, `apt-get -s upgrade`, `apt-get upgrade`
- Depuis 2 versions Debian propose la commande `aptitude` plus simple à utiliser
- Travail à rendre : faire une fiche de synthèse sur l'utilisation de la commande `aptitude`

Les packages sous Solaris

- Notion identique aux packages RPM Linux
- Accessible via les commandes `pkgadd`, `pkgrm` et `pkginfo`
- Il existe une commande (à installer en plus de solaris) permettant un traitement des packages de façon similaire à Debian : `pkg-get`

La notion de portage

- issus des distribution BSD
- le package est distribué sous la forme de fichiers sources
- nécessite une compilation sur la machine cible
- permet une adaptation et des optimisation très poussées
- utilisé dans les distributions BSD en Linux Gentoo

Principes

- télécharger les archives des sources depuis le site officiel ou un de ses miroirs
- décompresser l'archive dans un répertoire utilisateur, lancer la procédure de configuration, compiler
- installer les binaires dans une partie spécifique du système de fichiers
- utiliser des règles de nommage précises pour faciliter les desinstallation
- lire rigoureusement les documentations

Conventions d'installation

Modification à apporter au système de fichiers afin de permettre un installation/dé-installation des applications recompilées par l'administrateur :

- isoler les applications recompilées
- utiliser `/usr/local` pour l'installation des binaires
- utiliser un répertoire utilisateur (par exemple créer l'utilisateur source) pour recompiler les applications
- pour gérer chaque version de chaque application utiliser des liens symboliques

Illustration avec les logiciels Apache, MySQL, PHP, PostgreSQL

Installation de PostgreSQL (1)

```
tar xzfv postgresql-7.4.5.tar.gz
./configure --prefix=/usr/local/postgresql-7.4.5 \
---with-java --enable-thread-safety
make
make install
make install-all-headers
```

Vérifier que l'utilisateur postgres existe :

```
su postgres
make check
```

Attention mettre les droits x sur l'ensemble des répertoires traversés pour accéder a postgres, positionner le propriétaire des sources sur postgres. En etant root, sous `/usr/local`, créer le lien et le repertoire des database files.

```
ln -sf postgresql-7.4.5/ postgresql
mkdir /usr/local/postgres/data
chown postgres /usr/local/postgres/data
```

Installation de PostgreSQL (2)

Initialisation du dictionnaire de donnees :

```
su postgres
/usr/local/postgresql/bin/initdb -D \
        /usr/local/postgresql/data
```

résultat :

```
Success. You can now start the database server using:
    /usr/local/postgresql/bin/postmaster \
     -D /usr/local/postgresql/data
or
    /usr/local/postgresql/bin/pg_ctl \
     -D /usr/local/postgresql/data -l logfile start
```

Installation de MySQL (1)

```
tar xzfv mysql-4.0.21.tar.gz
groupadd mysql
useradd -g mysql mysql

 ./configure --prefix=/usr/local/mysql-4.0.21 \
    --with-innodb --enable-thread-safe-client
 make
 make install
 scripts/mysql_install_db
 ln -sf mysql-4.0.21/ mysql
 chown -R root /usr/local/mysql
 chgrp -R mysql /usr/local/mysql
 chown -R mysql /usr/local/mysql/var
```

Installation de MySQL (2)

Résultat :

```
To start mysqld at boot time you have to copy support-files/mysql.server
to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:
/usr/local/mysql-4.0.21/bin/mysqladmin -u root password 'new-password'
/usr/local/mysql-4.0.21/bin/mysqladmin -u root -h XYZ.u-bourgogne.fr password 'new-password'
See the manual for more instructions.

You can start the MySQL daemon with:
cd /usr/local/mysql-4.0.21 ; /usr/local/mysql-4.0.21/bin/mysqld_safe &

You can test the MySQL daemon with the benchmarks in the 'sql-bench' directory:
cd sql-bench ; perl run-all-tests

Please report any problems with the /usr/local/mysql-4.0.21/bin/mysqlbug script!

The latest information about MySQL is available on the web at
http://www.mysql.com
Support MySQL by buying support/licenses at https://order.mysql.com
```

Installation de MySQL (3)

Fixer le password de root (root mysql, pas celui du systeme)

```
/usr/local/mysql-4.0.21/bin/mysqld_safe &
/usr/local/mysql-4.0.21/bin/mysqladmin \
  -u root password 'toto'
/usr/local/mysql-4.0.21/bin/mysqladmin \
  -u root -h XYZ.u-bourgogne.fr password 'toto'
```

Installation d'apache

Vérifier l'existence de l'utilisateur et du groupe `nobody`

```
tar xzfv apache-1.3.30.tar.gz
./configure --prefix=/usr/local/apache-1.3.30 \
--enable-module=so
make
make install
```

Éditer le fichier `httpd.conf` et lancer le serveur apache

```
/usr/local/apache/bin/apachectl start
```

Ajouter des utilisateurs pour les différentes applications, les positionner dans le groupe `nobody`

Installation d'apache : configuration

```
...
# BindAddress: You can support virtual hosts with this option. This directive
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
BindAddress *
...
# Example:
# LoadModule foo_module libexec/mod_foo.so
LoadModule php4_module        libexec/libphp4.so
...
#
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
#
Port 80
...
User nobody
Group nobody

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin admin@serveur.com
...
```

Installation d'apache : configuration

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
#
# DocumentRoot "/usr/local/apache_1.3.29/htdocs"
DocumentRoot "/usr/local/www"
...
#
# This should be changed to whatever you set DocumentRoot to.
#
# <Directory "/usr/local/apache_1.3.29/htdocs">
```

Installation d'apache : configuration

```
alias /master "/home0/fdess/"
<Directory "/home0/fdess/">
    Options Indexes FollowSymLinks
    AllowOverride None
</Directory>

alias /infoiem "/home0/infoiem/www"
<Directory "/home0/infoiem/www/">
    Options Indexes FollowSymLinks
    AllowOverride None
</Directory>

alias /webcal "/home0/webcal/www"
<Directory "/home0/webcal/www/">
    Options Indexes FollowSymLinks
    AllowOverride None
</Directory>
```

Installation de PHP

```
./configure --prefix=/usr/local/php-4.3.8
  --with-mysql=/usr/local/mysql
  --with-postgres=/usr/local/postgresql
  --with-apxs=/usr/local/apache/bin/apxs
  --with-zlib --with-jpeg --with-png
  --with-gettext --with-gd2
  make
  make install
```

Éditer la config http `/usr/local/apache/conf/httpd.conf`

```
ajouter AddType application/x-httpd-php .php
(vers les autres lignes concernees par le addtype)
/usr/local/apache/bin/apachectl stop
/usr/local/apache/bin/apachectl start
```

Exemple d'infrastcuture de SI